IBM® DB2® Universal Database

응용프로그램 빌드 안내서

버전 7

SA30-0991-01

IBM® DB2® Universal Database

응용프로그램 빌드 안내서

버전 7

SA30-0991-01

이 책의 정보와 지원하는 제품을 사용하기 전에 반드시 491 페이지의 『부록E. 주의사항』을 읽으십시오.

목차

DB2 응용프로그램 개발 환영

주: 행의 왼쪽 여백에 있는 개정 막대 "|"는 해당 행의 텍스트가 DB2 Universal Database 버전 7에서 처음 발행된 이후로 추가되었거나 수정되었음을 나타냅니다. 장 제목이나 절 제목 옆에 있는 개정 막대 "|"는 해당 장이나 절에 개정된 텍스트가 있음을 나타냅니다.

이 서문에서는 DB2 응용프로그램 개발(특히, DB2 Developer's Edition 제품)을 시작하는 데 필요한 정보를 제공합니다.

『DB2 Developer's Edition』 절에서는 특정 개발 요구에 대한 설치 정보를 제공합니다. 이 정보는 사용자가 IBM DB2 Universal Developer's Edition 버전 7 또는 IBM DB2 Personal Developer's Edition 버전 7에서 DB2를 설치하는 방법을 결정하는 데 도움을 줍니다.

xiii 페이지의 『DB2 응용프로그램 개발 책』 절에서는 응용프로그램 개발에 대한 DB2 라이브러리의 주요 책에 대해 설명합니다.

xiv 페이지의 『DB2 프로그래밍 인터페이스』 절에서는 DB2 응용프로그램 개발의 중요한 프로그래밍 인터페이스 개념에 대해 설명합니다.

xxvi 페이지의 『DB2 기능』 절에서는 DB2 응용프로그램 개발에 사용 가능한 주요 기능에 대해 설명합니다.

DB2 Developer's Edition

DB2 Universal Database는 응용프로그램 개발을 위한 두 가지 제품 패키지(DB2 Personal Developer's Edition 및 DB2 Universal Developer's Edition)를 제공합니다. Personal Developer's Edition은 OS/2, Linux 및 Windows 32비트 운영 체제에서 수행되는 DB2 Universal Database 및 DB2 Connect Personal Edition 제품을 제공합니다. DB2 Universal Developer's Edition은 이러한 플랫폼과 함께 AIX, HP-UX, PTX, Silicon Graphics IRIX 및 Solaris** 운영 환경**에도 DB2 제품을 제공합니다.

이들 제품과 함께 제공되는 소프트웨어를 사용하여 하나의 운영 체제에서 수행되는 응용프로그램을 개발 및 테스트하여 동일하거나 다른 운영 체제에서 데이터베이스에 액세스할 수 있습니다. 예를 들어, Windows NT 운영 체제에서 수행되지만 AIX와 같은 UNIX 플랫폼에 있는 데이터베이스에 액세스하는 응용프로그램을 작성할 수 있습니다. Developer's Edition 제품 사용 기간 및 조건에 대한 자세한 내용은 사용권 계약을 참조하십시오.

Personal Developer's Edition에는 응용프로그램을 개발하고 테스트하는 데 필요한 모든 코드가 있는 몇 개의 CD-ROM이 들어 있습니다. 각각의 상자에는 다음과 같은 것들이 있습니다.

- OS/2, Linux 및 Windows 운영 체제용 DB2 Universal Database 제품 CD-ROM. 각 CD-ROM에는 지원되는 운영 체제를 위한 DB2 서버, 관리 클라이언트, 응용프로그램 개발 클라이언트 및 런타임 클라이언트가 포함되어 있습니다. 이들 CD-ROM은 단지 응용프로그램 테스트용으로 제공됩니다. 데이터베이스를 설치하고 사용해야 하는 경우, Universal Database 제품을 구입하여 유효한 사용권을 획득해야 합니다.
- DB2 Connect Personal Edition
- PDF 형식으로 DB2 책을 포함하는 DB2 서적 CD-ROM.
- DB2 Extenders(OS/2 및 Windows 전용)
- DB2 XML Extender(Windows 전용)
- DB2 OLAP Starter Kit(Windows 전용). OLAP Starter Kit을 설치하기 전에 DB2 서버를 설치해야 합니다.
- VisualAge for Java, Entry Edition

Universal Developer's Edition에는 다음을 포함하여 DB2에서 지원하는 모든 운영 체제에 대한 CD-ROM이 들어 있습니다.

- DB2 Universal Database Personal Edition, Workgroup Edition 및 Enterprise Edition
- DB2 Connect Personal Edition 및 DB2 Connect Enterprise Edition

- 모든 플랫폼에 대한 Administration Client. 이 클라이언트에는 제어 센터와 이벤트 분석기 같은 데이터베이스를 관리하기 위한 도구가 들어 있습니다. 또한 이 클라이언트를 사용하면 어떠한 시스템에서도 응용프로그램을 실행할 수 있습니다.
- 모든 플랫폼에 대한 Application Development Client. 이 클라이언트에는 응용프로그램 개발 도구, 샘플 프로그램 및 헤더 파일이 포함됩니다. 각 DB2 AD 클라이언트에는 응용프로그램을 개발하는 데 필요한 모든 것이 들어 있습니다. AD 클라이언트에 대한 자세한 내용은 5 페이지의 『DB2 Application Development Client 정보』를 참조하십시오.
- 모든 플랫폼에 대한 런타임 클라이언트. 임의의 시스템의 Run-Time Client에서 응용프로그램을 수행할 수 있습니다. Run-Time Client에는 DB2 제어 센터와 이벤트 분석기 같은 일부 Administration Client 기능은 들어 있지 않으므로 공간을 덜 차지합니다.
- DB2 Satellite Edition
- DB2 Extenders
- DB2 XML Extender
- DB2 OLAP Starter Kit
- Net.Data
- VisualAge for Java, Professional Edition(OS/2 및 Windows)
- Websphere Studio
- Websphere Application Server, 표준판
- Query Management Facility(시험 후 구매)

또한 두 Developer's Edition 모두에 대해 응용프로그램 개발에 유용한 다른 소프트웨어 사본을 확보할 수 있습니다. 이 소프트웨어는 수시로 달라질 수 있으며 사용을 위한 사용권 계약과 함께 제공됩니다.

설치 정보

각 DB2 제품 CD-ROM에는 CD-ROM에서 직접 볼 수 있는 HTML로 작성된 설치 정보가 들어 있습니다. 지원되는 플랫폼을 위한 *빠른 시작* 책이 제공되며 여

기에는 DB2 서버, Administration Clients, Application Development Clients 및 Run-Time Clients를 설치하는 방법이 설명되어 있습니다.

추가 정보는 *설치 및 구성 보충 설명서*에서 찾을 수 있습니다. 이 책은 Client CD-ROM에서만 볼 수 있습니다.

사용자가 보기 원하는 책은 사용 중인 언어에 대한 서브디렉토리에 있습니다. CD-ROM README.TXT 파일을 보면 CD-ROM에서 책을 찾을 수 있는 위치를 알 수 있습니다.

브라우저가 실행 중인 상태에서 서적 서브디렉토리에서 `index.htm` 파일을 누르십시오. 이 디렉토리는 *빠른 시작* 및 *설치 및 구성 보충 설명서* 책을 위한 서브디렉토리입니다.

db2i2 *OS/2용 DB2 빠른 시작*

db2ix *UNIX용 DB2 빠른 시작*

db2i6 *Windows용 DB2 빠른 시작*

db2iy *설치 및 구성 보충 설명서*

DB2 서적 CD-ROM에는 제품과 함께 제공되는 모든 DB2 책에 대한 PDF 파일이 들어 있습니다. 이들은 Adobe Acrobat Reader를 사용하여 CD-ROM에서 직접 볼 수 있습니다. 사용자의 언어로 사용할 수 있는 DB2 책은 해당 언어 서브디렉토리에 들어 있습니다. 설치 정보에 대한 자세한 내용은 *빠른 시작*과 *설치 및 구성 보충 설명서* 책을 참조하십시오. 이들 파일의 이름은 위에 나열된 문자 세트로 시작됩니다.

PDF 파일 인쇄에 대한 자세한 내용은 481 페이지의 『PDF 책 인쇄』를 참조하십시오.

책 파일 액세스 방법에 대한 자세한 내용은 CD-ROM의 README.TXT 파일을 참조하십시오.

DB2 응용프로그램 개발 책

DB2 라이브러리는 471 페이지의 『부록D. DB2 라이브러리 사용』에서 설명됩니다. 두 가지 일반 범주의 DB2 책(DB2 데이터베이스를 관리하기 위한 정보를 제공하는 책 및 DB2 응용프로그램 개발을 위한 정보를 제공하는 책)이 있습니다. 일부 책은 두 종류의 정보를 모두 제공합니다. DB2 응용프로그램 개발자로서, 두 가지 범주에서 모두 DB2 책을 참조할 수도 있습니다. 그러나 사용자의 촛점과 이 절의 촛점은 DB2 라이브러리의 주요 응용프로그램 개발 책에 관한 것입니다.

응용프로그램 프로그래밍을 위한 두 가지 주요 책이 있습니다. *CLI Guide and Reference*에서는 DB2 CLI 응용프로그램 프로그래밍에 대해 설명하며, *응용프로그램 개발 안내서*에서는 DB2 CLI 이외의 모든 다른 종류의 DB2 프로그래밍에 대해 설명합니다. 이들 책에는 모두 참조 정보가 들어 있습니다.

지금 읽고 있는 *응용프로그램 빌드 안내서* 책에는 개발 환경 설정, 응용프로그램 컴파일, 링크 및 수행에 대한 정보가 있습니다.

SQL문 및 함수의 구문에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.

DB2 라이브러리의 관리 범주로 간주되는 두 가지 책 또한 응용프로그램 프로그래밍에 대한 중요한 참조서입니다. *Administrative API Reference*는 DB2 데이터베이스 관리에 사용되는 모든 관리 기능에 대한 세부사항을 포함합니다. SQL문과 함께 또는 SQL문 대신 응용프로그램에서 DB2 API를 사용하면 편리하다는 것을 알 수 있습니다. *Command Reference*는 모든 DB2 명령에 대한 세부사항(SQL문 제외)을 포함하며, DB2 명령행 처리기(CLP)를 사용하는 방법에 대해 설명합니다.

환경 설정 또는 프로그램 개발에 대한 문제점을 해결하려면 *문제점 해결 안내서*를 참조하십시오. DB2 고객 서비스와의 상담에서 오류의 소스를 판별하고, 문제점을 해결하며, 진단 도구를 사용하는 데 도움을 줍니다. *메시지 참조서*는 DB2 오류 메시지의 전체 목록과 설명을 포함하며, 응용프로그램을 디버깅할 때 매우 유용한 참조서입니다.

DB2 라이브러리의 이들 책과 그 밖의 책들은 DB2 환경에서 사용 가능한 온라인 정보와 함께 DB2 응용프로그램을 개발하는 데 필요한 정보를 사용자에게 제공합

니다. DB2에 제한되지 않는 개발 정보에 대한 자세한 내용은 사용 중인 컴파일러, 인터프리터 또는 그 밖의 개발 도구의 벤더에서 제공하는 문서를 참조하십시오.

DB2 프로그래밍 인터페이스

몇 가지 서로 다른 프로그래밍 인터페이스를 사용하여 DB2 데이터베이스를 관리하거나 DB2 데이터베이스에 액세스할 수 있습니다. 다음을 수행할 수 있습니다.

1. DB2 API를 사용하여 데이터베이스 백업 및 복원과 같은 관리 기능 수행.
2. 응용프로그램에서 정적 및 동적 SQL문 포함.
3. 동적 SQL문을 호출하도록 응용프로그램에서 DB2 콜 레벨 인터페이스(DB2 CLI) 함수 호출 코드화.
4. JDBC(Java Database Connectivity) API를 호출하는 Java 응용프로그램 및 애플릿 개발.
5. DAO(Data Access Object) 및 RDO(Remote Data Object) 스펙을 따르는 Microsoft Visual Basic 및 Visual C++ 응용프로그램과 OLE DB 브릿지를 사용한 ADO(ActiveX Data Object) 응용프로그램 개발.
6. Net.Data, Excel, Perl, ODBC(Open Database Connectivity) 일반 사용자 도구(예: Lotus Approach, 해당 프로그래밍 언어 LotusScript)와 같은 IBM 또는 써드 파티 도구를 사용한 응용프로그램 개발.

응용프로그램이 DB2 데이터베이스에 액세스하는 방법은 개발하려는 응용프로그램 유형에 따라 다릅니다. 예를 들어, 데이터 입력 응용프로그램을 개발하려는 경우, 응용프로그램에서 정적 SQL문을 포함하도록 선택할 수 있습니다. 월드 와이드 웹에 대한 조회를 수행하는 응용프로그램을 개발하려는 경우, Net.Data, Perl 또는 Java를 선택할 수 있습니다.

Embedded SQL문 사용

SQL(Structured Query Language)은 DB2 데이터베이스의 데이터에 액세스하고 관리하는 데 사용되는 데이터베이스 인터페이스 언어입니다. 데이터 검색 또는 저장과 같은 SQL이 지원하는 타스크를 수행하여 응용프로그램에 SQL문을 내포할

수 있습니다. DB2를 사용하여 C/C++, COBOL, FORTRAN, Java(SQLJ) 및 REXX 프로그래밍 언어로 Embedded SQL 응용프로그램을 코드화할 수 있습니다.

SQL문을 포함한 응용프로그램을 호스트 프로그램이라고 합니다. 호스트 프로그램을 작성하는 데 사용하는 프로그래밍 언어를 호스트 언어라고 합니다. SQL문을 소유하거나 수용하기 때문에 이러한 방법으로 프로그램과 언어를 정의합니다.

정적 SQL문의 경우, 사용자는 컴파일 시간 이전에 SQL문 유형과 테이블 및 컬럼 이름을 알게 됩니다. 유일하게 알려지지 않은 사항은 명령문이 검색 중이거나 갱신 중인 특정 데이터 값입니다. 호스트 언어 변수에 이러한 값을 표시할 수 있습니다. 응용프로그램을 수행하기 전에 정적 SQL문을 사전 처리 컴파일하고 바인드한 다음 컴파일합니다. 정적 SQL은 통계가 많은 양을 변경하지 않는 데이터베이스에서 가장 잘 수행됩니다. 그렇지 않은 경우, 명령문은 곧 쓸모없게 됩니다.

이와 반대로, 동적 SQL문은 응용프로그램이 런타임시 빌드하고 실행합니다. 일반 사용자에게 검색할 테이블 및 컬럼의 이름과 같은 SQL문의 주요한 부분을 프롬프트하는 대화식 응용프로그램이 동적 SQL의 좋은 예입니다. 응용프로그램이 수행되는 동안 SQL문을 빌드한 다음 처리하도록 명령문을 제출합니다.

정적 SQL문, 동적 SQL문 또는 두 SQL문이 혼합된 응용프로그램을 작성할 수 있습니다.

일반적으로, 정적 SQL문은 사전 정의된 트랜잭션이 있는 고성능 응용프로그램에 가장 적합합니다. 예약 시스템은 이러한 응용프로그램의 좋은 예입니다.

일반적으로, 동적 SQL문은 런타임시 트랜잭션을 지정해야 하는, 빠르게 변화하는 데이터베이스에 대해 수행되는 응용프로그램에 적합합니다. 대화식 조회 인터페이스는 이러한 응용프로그램의 좋은 예입니다.

응용프로그램에 SQL문을 포함할 때 다음과 같은 단계를 사용하여 응용프로그램을 사전 처리 컴파일한 다음 데이터베이스에 바인드해야 합니다.

1. Embedded SQL문이 있는 프로그램을 포함하는 소스 파일을 작성하십시오.
2. 데이터베이스에 연결한 다음 각각의 소스 파일을 사전 처리 컴파일하십시오.

사전 처리 컴파일러는 각 소스 파일에 있는 SQL문을 데이터베이스 관리자에 대한 DB2 런타임 API 호출로 변환합니다. 또한 사전 처리 컴파일러는 데이터베이스에 액세스 패키지를 작성하며, 사용자가 작성하려고 지정한 경우에는 선택적으로 바인드 파일을 생성합니다.

액세스 패키지에는 응용프로그램에 있는 정적 SQL문에 대한 DB2 최적화 알고리즘에서 선택한 액세스 플랜이 들어 있습니다. 액세스 플랜에는 최적화 알고리즘이 결정한 가장 효율적인 방법으로 정적 SQL문을 실행하기 위한 데이터베이스 관리자에서 요구하는 정보가 들어 있습니다. 동적 SQL문의 경우, 최적화 알고리즘이 응용프로그램 수행시 액세스 플랜을 작성합니다.

바인드 파일에는 액세스 패키지를 작성하는 데 필요한 SQL문과 그 밖의 데이터가 들어 있습니다. 먼저 사전 처리 컴파일할 필요없이 바인드 파일을 사용하여 응용프로그램을 나중에 리바인드할 수 있습니다. 리바인딩은 현재 데이터베이스 조건에 대해 최적화된 액세스 플랜을 작성합니다. 사전 처리 컴파일된 것과 다른 데이터베이스에 액세스하려면 응용프로그램을 리바인드해야 합니다. 마지막 바인딩 이후 데이터베이스 통계가 변경된 경우, 응용프로그램을 리바인드해야 합니다.

3. 호스트 언어 컴파일러를 사용하여 수정된 소스 파일(및 SQL문이 없는 기타 파일)을 컴파일하십시오.
4. 실행 가능 프로그램을 생성하도록 오브젝트 파일을 DB2 및 호스트 언어 라이브러리와 링크하십시오.
5. 사전 처리 컴파일 동안에 수행하지 않았거나, 다른 데이터베이스에 액세스하려는 경우, 바인드 파일을 바인드하여 액세스 패키지를 작성하십시오.
6. 응용프로그램을 수행하십시오. 응용프로그램은 패키지의 액세스 플랜을 사용하여 데이터베이스에 액세스합니다.

SQLJ(Embedded SQL for Java)

DB2 SQLJ(Java Embedded SQL) 지원은 DB2 AD Client에서 제공합니다. DB2 SQLJ 지원과 함께 DB2 JDBC 지원을 사용하여 SQLJ 애플릿, 응용프로그램 및 저장 프로시듀어를 빌드하고 수행할 수 있습니다. 이들은 정적 SQL을 포함하며 DB2 데이터베이스에 바인드된 Embedded SQL문을 사용합니다.

DB2 SQLJ 지원에 관한 자세한 내용을 보려면 다음 주소의 웹 페이지를 방문하십시오.

```
http://www.ibm.com/software/data/db2/java
```

DB2 콜 레벨 인터페이스 사용

DB2 CLI는 C 및 C++ 응용프로그램이 DB2 데이터베이스에 액세스하는 데 사용할 수 있는 프로그래밍 인터페이스입니다. DB2 CLI는 Microsoft ODBC(Open Database Connectivity) 스펙 및 ISO CLI 표준에 근거합니다. DB2 CLI가 산업 표준에 근거하므로 이들 데이터베이스 인터페이스에 이미 익숙한 응용프로그램 프로그래머는 보다 짧은 학습 과정에서 이점을 취할 수도 있습니다.

DB2 CLI 사용시, 응용프로그램은 동적 SQL문을 함수 인수로 데이터베이스 관리자에 처리하도록 전달합니다. 이와 같이, DB2 CLI는 Embedded 동적 SQL의 대안입니다.

또한 CLI, ODBC 또는 JDBC 응용프로그램에서 SQL문을 정적 SQL로 수행할 수 있습니다. CLI/ODBC/JDBC 정적 프로파일링 기능을 사용하면 응용프로그램의 일반 사용자는 많은 경우에 동적 SQL의 사용을 정적 SQL로 바꿀 수 있습니다. 자세한 내용은 다음을 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/staticcli
```

UNIX에서는 `libdb2`로, OS/2 및 Windows 32비트 운영 체제에서는 `db2cli.lib`으로 응용프로그램을 링크하여 ODBC 드라이버 관리자를 사용하지 않고 ODBC 응용프로그램을 빌드하고 모든 플랫폼에서 DB2의 ODBC 드라이버를 쉽게 사용할 수 있습니다. DB2 CLI 샘플 프로그램에서 이 과정을 볼 수 있습니다. 이 샘플 프로그램은 UNIX에서는 `sqllib/samples/cli`에 있고 OS/2 및 Windows 32비트 운영 체제에서는 `%DB2PATH%\samples\cli`에 있습니다.

DB2에서 제공하는 공통 액세스 패키지를 사용하므로 DB2 CLI 응용프로그램을 사전 처리 컴파일하거나 바인드할 필요가 없습니다. 간단히 응용프로그램을 컴파일하고 링크하십시오.

그러나 DB2 CLI 또는 ODBC 응용프로그램이 DB2 데이터베이스에 액세스하려면 DB2 AD Client와 함께 제공되는 DB2 CLI 바인드 파일을 액세스할 각각의

DB2 데이터베이스에 바인드해야 합니다. 이는 첫 번째 명령문의 실행시 자동으로 발생하지만, 데이터베이스 관리자가 DB2 데이터베이스에 액세스할 각각의 플랫폼에 있는 클라이언트에서 바인드 파일을 바인드하는 것이 좋습니다. 바인드 지시사항에 대해서는 51 페이지의 『바인딩』을 참조하십시오.

예를 들어, 각각 두 개의 DB2 데이터베이스에 액세스하는 OS/2, AIX 및 Windows 95 클라이언트가 있다고 가정하십시오. 관리자는 액세스할 각각의 데이터베이스에 있는 하나의 OS/2 클라이언트에서 바인드 파일을 바인드해야 합니다. 다음으로, 관리자는 액세스할 각각의 데이터베이스에 있는 하나의 AIX 클라이언트에서 바인드 파일을 바인드해야 합니다. 마지막으로, 관리자는 하나의 Windows 95 클라이언트에서 동일한 작업을 수행해야 합니다.

DB2 CLI 대 Embedded 동적 SQL

Embedded 동적 SQL문 또는 DB2 CLI를 사용하여 동적 응용프로그램을 개발할 수 있습니다. 두 경우 모두, SQL문은 런타임시 준비되고 처리됩니다. 각각의 방법에는 아래에 나열된 고유한 이점이 있습니다.

DB2 CLI 이점

이식성 DB2 CLI 응용프로그램은 표준 함수 세트를 사용하여 SQL문을 데이터베이스에 전달합니다. DB2 CLI 응용프로그램은 수행하기 전에 컴파일 및 링크하기만 하면 됩니다. 반대로, Embedded SQL 응용프로그램은 수행하기 전에 사전 처리 컴파일하고 컴파일한 다음 데이터베이스에 바인드해야 합니다. 이 프로세스는 응용프로그램을 특정 데이터베이스에 효과적으로 연결합니다.

바인딩 안함

개별 DB2 CLI 응용프로그램을 액세스하는 각각의 데이터베이스에 바인드할 필요가 없습니다. 모든 DB2 CLI 응용프로그램에 대해 한 번만 DB2 CLI와 함께 제공된 바인드 파일을 바인드하면 됩니다. 이것은 응용프로그램을 관리하는 데 소모되는 시간을 상당히 감소시킬 수 있습니다.

확장된 페치 및 입력

DB2 CLI 함수를 사용하여 데이터베이스에 있는 복수의 행을 단일 호출

을 사용하는 배열로 검색할 수 있습니다. 또한 입력 변수 배열을 사용하여 SQL문을 여러 번 실행할 수 있습니다.

카탈로그에 대한 일관적인 인터페이스

데이터베이스 시스템에는 데이터베이스 및 사용자에 관한 정보를 갖는 카탈로그 테이블이 들어 있습니다. 이들 카탈로그의 양식은 시스템에 따라 다릅니다. DB2 CLI는 테이블, 컬럼, 외부 및 기본 키, 그리고 사용자 특권과 같은 구성요소에 관한 카탈로그 정보를 조회하기 위한 일관적인 인터페이스를 제공합니다. 이는 데이터베이스 서버 릴리스 간의 카탈로그 변경사항과 데이터베이스 서버 간의 차이점으로부터 응용프로그램을 보호합니다. 특정 서버 또는 제품 버전에 대한 카탈로그 조회를 작성할 필요가 없습니다.

확장된 데이터 변환

DB2 CLI는 SQL 및 C 데이터 유형 간의 데이터를 자동으로 변환합니다. 예를 들어, SQL 데이터 유형을 C 문자 데이터 유형으로 페치하면 문자열 표현으로 변환됩니다. 이로써 DB2 CLI를 대화식 조회 응용프로그램에 적합하게 만듭니다.

전역 데이터 영역 없음

DB2 CLI는 일반적으로 Embedded SQL 응용프로그램과 연관된 SQLDA 및 SQLCA와 같은, 응용프로그램이 제어하는 복잡한 전역 데이터 영역에 대한 필요성을 제거합니다. 대신 DB2 CLI는 필요한 데이터 구조를 자동으로 할당하고 제어하며, 응용프로그램이 참조할 핸들을 제공합니다.

저장 프로시듀어에서 결과 세트 검색

DB2 CLI 응용프로그램은 서버에 상주하는 저장 프로시듀어에서 생성한 복수 행 및 결과 세트를 검색할 수 있습니다.

화면 이동 커서

DB2 CLI는 배열 출력과 함께 사용할 수 있는 서버측 화면 이동 커서를 지원합니다. 이것은 Page Up, Page Down, Home 및 End 키를 사용하는 화면 이동 상자에 데이터베이스 정보를 표시하는 GUI 응용프로그램에서 유용합니다. 커서를 화면 이동으로 선언한 다음 결과 세트에서 하나 이

상의 행씩 앞뒤로 이동할 수 있습니다. 또한 현재 행으로부터의 오프셋, 결과 세트의 시작이나 끝 또는 이전에 북마크한 특정 행을 지정하여 행을 페치할 수 있습니다.

Embedded 동적 SQL 이점

모든 DB2 CLI 사용자는 동일한 특권을 공유합니다. Embedded SQL은 특정 사용자에게 패키지에 대한 실행 특권을 부여하여 좀더 세분화된 보안의 이점을 제공합니다.

Embedded SQL은 C 및 C++ 이외의 언어를 지원합니다. 이것은 다른 언어로 응용프로그램을 코드화하려는 경우, 이점이 될 수 있습니다.

동적 SQL은 일반적으로 정적 SQL보다 더 일관적입니다. 정적 SQL을 프로그램하는 방법을 이미 알고 있는 경우, 동적 SQL로 이동하는 것은 DB2 CLI로 이동하는 것만큼 어렵지 않습니다.

JDBC 사용

DB2의 Java 지원은 표준화된 Java 메소드를 통해 응용프로그램에 대한 데이터 액세스를 제공하는 벤더 중립 동적 SQL 인터페이스인 JDBC를 포함합니다. JDBC는 JDBC 프로그램을 사전 처리 컴파일하거나 바인드할 필요가 없다는 점에서 DB2 CLI와 유사합니다. 벤더 중립 표준으로, JDBC 응용프로그램은 증가된 이식성을 제공합니다. JDBC를 사용하여 작성된 응용프로그램은 동적 SQL만을 사용합니다.

JDBC는 인터넷에서 DB2 데이터베이스에 액세스할 때 특히 유용합니다. Java 프로그래밍 언어를 사용하여 네트워크 연결로 원격 DB2 데이터베이스의 데이터에 액세스하고 조작하는 JDBC 애플릿 및 응용프로그램을 개발할 수 있습니다. 또한 서버에 상주하는 JDBC 저장 프로시듀어를 작성하고, 데이터베이스 서버에 액세스하며, 저장 프로시듀어를 호출하는 원격 클라이언트 응용프로그램으로 정보를 리턴할 수 있습니다.

JDBC API는 CLI/ODBC API와 유사하며, Java 코드로부터 데이터베이스에 액세스하는 표준 방법을 제공합니다. Java 코드는 SQL문을 메소드 인수로 DB2 JDBC 드라이버에 전달합니다. 드라이버는 클라이언트 Java 코드의 JDBC API 호출을 처리합니다.

Java의 이식성을 사용하여 하나의 Java 가능 웹 브라우저만을 요구하는, 복수 플랫폼에 있는 클라이언트에 DB2 액세스를 제공할 수 있습니다.

Java 응용프로그램은 DB2 클라이언트에 의존하여 DB2에 연결 합니다. 다른 응용프로그램과 같이 데스크탑 또는 명령행에서 응용프로그램을 시작합니다. DB2 JDBC 드라이버는 응용프로그램에서 JDBC API 호출을 처리하며, 클라이언트 연결을 사용하여 서버에 대한 요청을 통신하고 결과를 수신합니다.

Java 애플릿은 DB2 클라이언트 연결을 요구하지 않습니다. 일반적으로, HTML(HyperText Markup Language) 웹 페이지에서 애플릿을 포함합니다.

애플릿을 수행하는 클라이언트 머신에 Java 가능 웹 브라우저 또는 애플릿 표시기만이 필요합니다. HTML 페이지를 로드하면 브라우저가 Java 애플릿을 머신으로 다운로드한 다음 Java 클래스 파일 및 DB2 JDBC 드라이버를 다운로드합니다. 애플릿이 DB2에 연결하기 위해 JDBC API를 호출하면 JDBC 드라이버가 웹 서버에 상주하는 JDBC 애플릿 서버를 통해 DB2 데이터베이스와 별도의 네트워크 연결을 설정합니다.

DB2 JDBC 지원에 대한 자세한 내용은 다음 웹 페이지를 참조하십시오.

```
http://www.ibm.com/software/data/db2/java
```

DB2 API 사용

응용프로그램이 데이터베이스 작성, 활성화, 백업 또는 복원과 같은 일부 데이터베이스 관리 타스크를 수행해야 할 수도 있습니다. DB2는 Embedded SQL 및 DB2 CLI 응용프로그램을 포함하여 응용프로그램에서 이들 타스크를 수행할 수 있도록 다양한 API를 제공합니다. 이를 사용하여 xxxiii 페이지의 『DB2 Universal Database 도구』에 설명된 DB2 서버 관리 도구를 사용하여 수행할 수 있는 것과 동일한 관리 기능을 응용프로그램에 프로그래밍할 수 있습니다.

추가로, DB2 API만을 사용하여 수행할 수 있는 특정 타스크를 수행해야 할 수도 있습니다. 예를 들어, 응용프로그램이 일반 사용자에게 표시할 수 있도록 오류 메시지 텍스트를 검색하려고 할 수도 있습니다. 메시지를 검색하려면 Get Error Message API를 사용해야 합니다.

ADO 및 RDO 사용

DAO(Data Access Object) 및 RDO(Remote Data Object) 스펙을 따르는 Microsoft Visual Basic 및 Microsoft Visual C++ 데이터베이스 응용프로그램을 작성할 수 있습니다. 또한 DB2는 Microsoft OLE DB에서 ODBC로의 브릿지를 사용하는 ADO(ActiveX Data Object) 응용프로그램을 지원합니다.

ADO를 사용하여 OLE DB 제공업체를 통해 데이터베이스 서버에 있는 데이터에 액세스하고 조작하는 응용프로그램을 작성할 수 있습니다. ADO의 기본적인 이점은 고속, 사용의 용이성 및 작은 디스크 자국(footprint)입니다.

RDO는 ODBC를 통해 원격 데이터 소스에 액세스하기 위한 정보 모델을 제공합니다. RDO는 데이터베이스에 대한 연결, 조회 및 저장 프로시듀어 실행, 결과 조작 및 서버에 대한 변경사항 확약을 쉽게 해주는 오브젝트를 제공합니다. 특히 원격 ODBC 관계형 데이터 소스에 액세스하도록 설계되었으며, 복잡한 응용프로그램 코드 없이도 ODBC를 보다 쉽게 사용할 수 있습니다.

Perl 인터페이스 사용

DB2는 DBD::DB2 드라이버를 통해 데이터 액세스를 위한 Perl DBI(Database Interface) 스펙을 지원합니다. DB2 Universal Database Perl DBI 웹 사이트의 주소는 다음과 같으며,

```
http://www.ibm.com/software/data/db2/perl/
```

최신 DBD::DB2 드라이버와 관련 정보를 포함합니다.

ODBC 일반 사용자 도구 사용

또한 Lotus Approach, Microsoft Access 및 Microsoft Visual Basic과 같은 ODBC 일반 사용자 도구를 사용하여 응용프로그램을 작성할 수도 있습니다. ODBC 도구는 고급 프로그래밍 언어를 사용하는 것보다 간단한 응용프로그램 개발 대안을 제공합니다.

Lotus Approach는 DB2 데이터에 액세스하는 두 가지 방법을 제공합니다. 그래픽 인터페이스를 사용하여 조회를 수행하고, 보고서를 개발하며, 데이터를 분석할 수 있습니다. 또는 광범위한 오브젝트, 이벤트, 메소드 및 등록 정보를 제공하는 전

기능 객체 지향 프로그래밍 언어인 LotusScript를 내장 프로그램 편집기와 함께 사용하여 응용프로그램을 개발할 수 있습니다.

웹 응용프로그램을 빌드하기 위한 도구

DB2 Universal Database는 주요 인터넷 표준을 모두 지원하므로 웹에서 사용할 수 있는 이상적인 데이터베이스입니다. 이 데이터베이스는 관계형 데이터베이스의 확장성 및 사용 가능성 특성과 결합된 복잡한 텍스트 일치 및 인터넷 검색을 이용할 수 있는 인메모리 속도를 가지고 있습니다. DB2 Universal Database는 WebSphere, Java 및 XML Extender를 지원하므로 전자 상거래 응용프로그램을 쉽게 전개할 수 있게 합니다.

DB2 Universal Developer's Edition에는 웹 가능 지원을 제공하는 몇 가지의 도구가 있습니다. VisualAge for Java는 Websphere Application Server와 DB2 Universal Database에 대해 Java 응용프로그램을 빌드, 테스트 및 전개할 수 있게 하는 IDE(Integrated Development Environment)입니다. WebSphere Studio는 웹 사이트 개발의 모든 측면을 공통 인터페이스로 가져오는 일련의 도구 제품군입니다. WebSphere Application Server Standard Edition은 전자 상거래 응용프로그램을 위한 강력한 전개 환경을 제공합니다. 그 구성요소들은 개인화된 동적 웹 내용을 빠르고 쉽게 빌드 및 전개할 수 있게 합니다.

VisualAge for Java

VisualAge for Java Professional Edition(OS/2 및 Windows에서 DB2 Universal Developer's Edition과 함께 제공됨)에는 확장 가능한 고도의 전자 상거래 응용프로그램을 보다 쉽게 작성할 수 있도록 하는 기능 및 성능상의 개선사항이 포함되어 있습니다. IBM WebSphere Application Server, WebSphere Studio 및 DB2 Universal Database와 탄탄하게 통합하여 개발 시간을 줄이고 엔터프라이즈 데이터에 대한 보다 쉽고 안전한 액세스를 제공하면서 생산성을 향상시킵니다.

VisualAge for Java의 설치 가능한 기능 옵션인 IBM Data Access JavaBeans는 응용프로그램 개발자가 쉽게 JDBC에서 사용하는 관계형 데이터베이스에 액세스할 수 있게 합니다. `com.ibm.db` 패키지에 있는 IBM Data Access JavaBeans에는 관계형 데이터베이스에 대한 액세스를 단순화하고 다음과 같은 향상된 기능을 제공하기 위한 클래스가 있습니다.

- 조회 결과 캐싱
- 결과 캐쉬를 통한 갱신
- 조회 매개변수 지원
- 메타데이터 지원

WebSphere Studio

WebSphere Studio는 웹 사이트 개발의 모든 측면을 공통 인터페이스로 가져오는 일련의 도구 제품군입니다. WebSphere Studio는 동적 대화식 웹 응용프로그램을 보다 더 쉽게 통합적으로 작성, 어셈블, 발행 및 유지보수할 수 있게 합니다. Studio는 Workbench, Page Designer, Remote Debugger 및 마법사로 구성되어 있으며, Macromedia Flash, Fireworks, Freehand 및 Director와 같은 동반되는 웹 개발 제품들의 시험판과 함께 제공됩니다. WebSphere Studio는 고급 비지니스 기능을 지원하는 대화식 웹 사이트를 작성하기 위해 필요한 모든 것을 수행할 수 있게 합니다.

WebSphere Application Server Standard Edition(DB2 Universal Developer's Edition과 함께 제공됨)은 WebSphere Studio의 구성요소입니다. 이것은 서버측 비지니스 응용프로그램의 이식성과 Java 기술의 성능 및 관리성을 결합하여 Java 기반 웹 응용프로그램을 설계하기 위해 포괄적인 플랫폼을 제공합니다. 이를 사용하면 엔터프라이즈 데이터베이스 및 트랜잭션 시스템과 강력하게 상호작용할 수 있습니다. Websphere Application Server와 같은 머신이나 다른 웹 서버에서 DB2 서버를 수행할 수 있습니다.

WebSphere Application Server Advanced Edition(DB2 Universal Developer's Edition과 함께 제공되지 않음)은 Enterprise JavaBean 응용프로그램에 대한 추가 지원을 제공합니다. DB2 Universal Database는 WebSphere Application Server Advanced Edition과 함께 제공되어 관리 서버 저장소로 사용됩니다. 이는 Sun Microsystems로부터 웹 응용프로그램을 웹 이외의 비지니스 시스템에 통합하기 위한 지원을 제공하는 EJB 스펙에 대해 빌드된 응용프로그램을 위해 서버 성능을 도입합니다.

XML Extender

XML(Extensible Markup Language)은 응용프로그램 사이의 데이터 교환을 위한 승인된 표준 기술입니다. XML 문서는 사람이 쉽게 읽을 수 있는 태그화된 문서입니다. 텍스트는 문자 데이터와 마크업 태그로 구성됩니다. 마크업 태그는 문서 작성자가 정의할 수 있습니다. DTD(Document Type Definition)는 마크업 정의와 제한조건을 선언하는 데 사용됩니다. Windows의 Personal Developer's Edition 뿐만 아니라 DB2 XML Extender(DB2 Universal Developer's Edition과 함께 제공됨)는 SQL 확장자를 사용하여 XML 데이터를 조작하도록 프로그램에 대한 메커니즘을 제공합니다.

DB2 XML Extender는 세 가지의 새로운 데이터 유형인 `XMLVARCHAR`, `XMLCLOB`, `XMLFILE`을 도입합니다. Extender는 단일 또는 복수 컬럼과 테이블 내에 있는 XML 문서를 저장, 추출 및 갱신하기 위한 UDF를 제공합니다. 검색은 전체 XML 문서에서 또는 XML 경로 언어에 대한 XPath와 XSLT(Extensible Stylesheet Language Transformation) 부속 집합을 사용하는 위치 경로를 사용하여 구조적 구성요소를 기초로 수행할 수 있습니다.

XML 문서를 컬럼 세트로 저장할 수 있도록 하기 위해 DB2 XML Extender는 XML 대 관계형 데이터베이스 맵핑을 사용하는 설계자를 돕기 위한 관리 도구를 제공합니다. DAD(Document Access Definition)는 XML 문서에 대한 구조적 및 맵핑 데이터를 유지보수하는 데 사용됩니다. DAD는 쉽게 조작하고 이해할 수 있는 XML 문서로 정의되어 저장됩니다. 새로운 저장 프로시듀어를 사용하여 문서를 작성하거나 분해할 수 있습니다.

DB2 XML Extender에 대한 자세한 내용은 다음 사이트를 참조하십시오.

```
http://www.ibm.com/software/data/db2/extenders/xmlext/index.html
```

MQSeries 지원

DB2 응용프로그램이 비동기 메시징 조작과 상호작용할 수 있도록 DB2 Universal Database와 함께 MQSeries 기능 세트가 제공됩니다. 이는 DB2가 지원하는 프로그래밍 언어로 작성된 응용프로그램에 대해 MQSeries 지원을 사용할 수 있음을 의미합니다.

기본 구성에서 MQSeries 서버는 DB2 Universal Database와 함께 데이터베이스 서버 머신에 위치합니다. MQSeries 함수는 DB2 서버에서 사용할 수 있으며 다

른 MQSeries 응용프로그램에 대한 액세스를 제공합니다. 여러 DB2 클라이언트가 데이터베이스를 통해 동시에 MQSeries 함수에 액세스할 수 있습니다. MQSeries 조작은 DB2 응용프로그램이 다른 MQSeries 응용프로그램과 비동기식으로 통신할 수 있게 합니다. 예를 들어, 새 기능은 DB2 응용프로그램이 데이터베이스 이벤트를 원격 MQSeries 응용프로그램에 발행하거나, 선택적 MQSeries Workflow 제품을 통해 작업흐름을 시작하거나, 선택적 MQSeries Integrator 제품을 사용하여 기존의 응용프로그램 패키지와 통신할 수 있도록 하는 간단한 방법을 제공합니다.

Net.Data

Net.Data를 사용하여 웹 응용프로그램을 통해 인터넷 및 인트라넷을 DB2 데이터에 액세스할 수 있습니다. 공통 게이트웨이 인터페이스(CGI) 응용프로그램보다 우수한 성능을 제공하여 웹 서버 인터페이스(API)를 개발합니다. Net.Data는 Java, REXX, Perl 및 C++와 같은 언어를 사용하는 서버측 처리와 함께 클라이언트측 처리를 지원합니다. Net.Data는 조건부 논리 및 다양한 매크로 언어를 제공합니다. 또한 수동으로 태그를 입력하는 대신 Net.Data 매크로에서 출력으로 XML 태그를 생성할 수 있도록 하는 XML 지원도 제공합니다. 생성된 출력을 형식화하고 표시하는 데 사용할 XML 양식 쉬트(XSL)도 지정할 수 있습니다. Net.Data 웹 페이지의 주소는 다음과 같습니다.

```
http://www.ibm.com/software/data/net.data/
```

DB2 기능

DB2는 응용프로그램을 보충하거나 확장하는 데 사용할 수 있는 서버에서 수행되는 다양한 기능을 제공합니다. DB2 기능을 사용할 때 동일한 타스크를 수행하기 위해 직접 코드를 작성할 필요가 없습니다. 또한 DB2는 클라이언트 응용프로그램에 코드를 모두 유지하는 대신 서버에 코드의 일부를 저장할 수 있습니다. 따라서 성능 및 유지보수 이점을 갖습니다.

데이터를 보호하고 데이터 간의 관계를 정의하는 기능이 있는 반면, 융통성 있는 고급 응용프로그램을 작성하기 위한 오브젝트 관련 기능이 있습니다. 일부 기능은

둘 이상의 방법으로 사용할 수 있습니다. 예를 들어, 제한조건을 사용하여 데이터를 보호하고 데이터 값 간의 관계를 정의할 수 있습니다. 일부 키 기능은 다음과 같습니다.

- 제한조건
- 사용자 정의 유형(UDT) 및 대형 오브젝트(LOB)
- 사용자 정의 함수(UDF)
- 트리거
- 저장 프로시듀어

DB2 기능 사용 여부를 결정하려면 다음 사항을 고려하십시오.

응용프로그램 독립성
: 처리하는 데이터와 응용프로그램이 무관하게 할 수 있습니다. 데이터베이스에서 수행되는 DB2 기능을 사용하여 응용프로그램에 영향을 미치지 않고 데이터에 관한 논리를 유지보수하고 변경할 수 있습니다. 해당 논리를 변경해야 하는 경우, 데이터에 액세스하는 각각의 응용프로그램에서가 아닌 한 곳에서만(서버에서) 변경해야 합니다.

성능
: 서버에서 응용프로그램의 일부를 저장하고 수행하여 응용프로그램을 보다 빠르게 수행할 수 있습니다. 일부 처리를 일반적으로 보다 강력한 서버 머신으로 넘겨서, 클라이언트 응용프로그램과 서버 간의 네트워크 트래픽을 감소시킬 수 있습니다.

응용프로그램 요구사항
: 응용프로그램에 다른 응용프로그램에는 없는 고유 논리가 있을 수도 있습니다. 예를 들어, 응용프로그램이 다른 응용프로그램에는 적합하지 않은 특정 순서로 데이터 입력 오류를 처리하는 경우, 이러한 상황을 처리하기 위한 코드를 직접 작성할 수 있습니다.

몇몇 응용프로그램에서 동시에 사용할 경우에는 서버에서 수행되는 DB2 기능을 사용하도록 결정할 수도 있습니다. 다른 경우에는 사용자 응용프로그램만을 사용하므로 사용자 응용프로그램의 논리를 유지하고자 할 수도 있습니다.

제한조건

데이터를 보호하고 사용자 데이터 간의 관계를 정의하기 위해 일반적으로 비지니스 규칙을 정의합니다. 이들 규칙은 테이블의 컬럼에 대해 유효한 데이터 값이나, 하나 이상의 테이블에서 컬럼이 서로 관련되는 방법을 정의합니다.

DB2는 데이터베이스 시스템을 사용하여 이러한 규칙을 시행하기 위한 방법으로 제한조건을 제공합니다. 비지니스 규칙을 시행하는 데이터베이스 시스템을 사용하는 경우, 이를 시행하기 위한 코드를 응용프로그램에 작성할 필요가 없습니다. 그러나 비지니스 규칙이 하나의 응용프로그램에만 적용되는 경우, 전역 데이터베이스 제한조건을 사용하는 대신 응용프로그램에서 코딩해야 합니다.

DB2는 다음과 같은 종류의 제한조건을 제공합니다.

1. NOT NULL 제한조건
2. UNIQUE 제한조건
3. PRIMARY KEY 제한조건
4. FOREIGN KEY 제한조건
5. CHECK 제한조건

SQL문 CREATE TABLE 및 ALTER TABLE을 사용하여 제한조건을 정의합니다.

사용자 정의 유형(UDT) 및 대형 오브젝트(LOB)

데이터베이스에 있는 모든 데이터 요소는 테이블 컬럼에 저장되며, 각각의 컬럼이 데이터 유형을 갖도록 정의됩니다. 데이터 유형은 컬럼에 둘 수 있는 값의 유형과 수행할 수 있는 조작에 대한 한계를 배치합니다. 예를 들어, 정수 유형의 컬럼은 수정된 범위 내의 숫자만을 포함할 수 있습니다. DB2는 정의된 특성 및 작동을 갖는 다음과 같은 내장 데이터 유형 세트를 포함합니다. 문자열, 숫자, 날짜 시간 값, 대형 오브젝트(LOB), 널(NULL), 그래픽 문자열, 2진 문자열 및 데이터 링크.

그러나 때때로, 내장 데이터 유형이 사용자 응용프로그램의 요구를 지원하지 않을 수도 있습니다. DB2는 사용자 응용프로그램에 필요한 별도의 데이터 유형을 정의할 수 있는 사용자 정의 유형(UDT)을 제공합니다.

UDT는 내장 데이터 유형에 근거합니다. UDT를 정의할 때 UDT에 유효한 조작을 또한 정의합니다. 예를 들어, DECIMAL 데이터 유형에 근거하여 MONEY 데이터 유형을 정의할 수도 있습니다. 그러나 MONEY 데이터 유형의 경우, 덧셈 및 뺄셈 연산만이 허용되며, 곱셈 및 나눗셈 연산은 허용되지 않습니다.

대형 오브젝트(LOB)를 사용하여 데이터베이스에 크고, 복잡한 데이터 오브젝트를 저장하고 조작할 수 있습니다(오디오, 비디오, 이미지 및 대형 문서와 같은 오브젝트).

UDT와 LOB의 조합은 사용자에게 상당한 능력을 부여합니다. 더 이상 DB2가 제공하는 내장 데이터 유형을 사용하여 비지니스 데이터를 모델화하고, 해당 데이터의 의미를 캡쳐하는 것에 제한되지 않습니다. UDT를 사용하여 고급 응용프로그램에 대한 크고, 복잡한 데이터 구조를 정의할 수 있습니다.

내장 데이터 유형 확장에 추가하여 UDT는 다음과 같은 몇 가지 그 밖의 이점을 제공합니다.

응용프로그램에서 객체 지향 프로그래밍에 대한 지원

유사한 오브젝트를 관련된 데이터 유형으로 그룹화할 수 있습니다. 이들 유형에는 이름, 내부 표현 및 특정 작동이 있습니다. UDT를 사용하여 DB2에 새로운 유형의 이름과 내부적으로 표현되는 방법을 알려줄 수 있습니다. LOB는 새로운 유형에 대한 가능한 내부 표현 중 하나이며, 크고 복잡한 데이터 구조에 가장 적합한 표현입니다.

강력한 유형 지정 및 캡슐화를 통한 데이터 무결성

강력한 유형 지정은 구별 유형으로 정의된 함수 및 조작만이 유형에 적용될 수 있도록 보장합니다. 캡슐화는 UDT의 작동이 적용될 수 있는 함수와 연산자에 의해 제한되도록 보장합니다. DB2에서 UDT의 작동은 사용자 정의 함수(UDF)의 형태로 제공되며, 광범위한 사용자 요구사항을 수용하도록 작성할 수 있습니다.

데이터베이스 관리자로 통합을 통한 수행

UDT가 내장 데이터 유형과 동일한 방법으로 내부적으로 표현되기 때문에 내장 데이터 유형과 동일한 효율적인 코드를 공유하여 내장 함수, 비교 연산자, 색인 및 그 밖의 함수를 구현합니다. 이에 대한 예외는 LOB를 사용하는 UDT로, 비교 연산자 및 색인과 함께 사용할 수 없습니다.

저장 프로시듀어

일반적으로, 응용프로그램은 네트워크를 통해 데이터베이스에 액세스합니다. 많은 데이터가 리턴되는 경우, 성능이 저하될 수 있습니다. 저장 프로시듀어는 데이터베이스 서버에서 수행됩니다. 클라이언트 응용프로그램은 네트워크를 통해 불필요한 데이터를 리턴하지 않고 데이터베이스 액세스를 수행하는 저장 프로시듀어를 호출할 수 있습니다. 클라이언트 응용프로그램이 요구한 결과만을 저장 프로시듀어가 리턴합니다.

저장 프로시듀어를 사용하는 몇 가지 이점은 다음과 같습니다.

감소된 네트워크 트래픽

SQL문을 그룹화하여 네트워크 트래픽을 감소시킬 수 있습니다. 일반적인 응용프로그램은 각각의 SQL문에 대하여 네트워크에서 두 가지 단계를 요구합니다. SQL문을 그룹화하면 네트워크에서 각 명령문의 그룹에 대해 두 가지 단계가 발생하여, 응용프로그램의 성능이 향상됩니다.

서버에만 존재하는 기능 액세스

저장 프로시듀어는 LIST DATABASE DIRECTORY 및 LIST NODE DIRECTORY와 같은 서버에서만 수행되는 명령에 액세스할 수 있습니다. 서버 머신에서는 증가된 메모리와 디스크 공간의 이점이 있을 수 있으며, 서버에 설치된 추가 소프트웨어에 액세스할 수 있습니다.

비지니스 규칙의 시행

저장 프로시듀어를 사용하여 몇몇 응용프로그램에 공통되는 비지니스 규칙을 정의할 수 있습니다. 제한조건 및 트리거 사용에 추가하여, 비지니스 규칙을 정의하는 또 다른 방법입니다.

응용프로그램은 저장 프로시듀어를 호출할 때 저장 프로시듀어에 정의된 규칙에 따라 일관적인 방법으로 데이터를 처리합니다. 규칙을 변경하는 경우, 저장 프로시듀어를 호출하는 모든 응용프로그램에서가 아닌 저장 프로시듀어에서 한 번만 변경을 수행하면 됩니다.

사용자 정의 함수(UDF)

SQL을 통해 제공되는 내장 기능이 모든 응용프로그램의 요구를 충족시키지 않을 수도 있습니다. 이러한 기능을 확장할 수 있도록, DB2는 사용자 정의 함수(UDF)

를 지원합니다. 자신의 코드를 Visual Basic, C/C++ 또는 Java로 작성하여 단일 스칼라 값 또는 테이블을 리턴하는 SQL문에서 조작을 수행할 수 있습니다.

UDF는 사용자에게 유통성을 제공합니다. 데이터베이스에서 선택 목록의 일부로 단일 스칼라 값을 리턴하거나 또는 스프레드시트와 같은 데이터베이스가 아닌 소스에서 전체 테이블을 리턴할 수 있습니다.

UDF는 응용프로그램을 표준화하는 방법을 제공합니다. 공통된 사용자 정의 함수(UDF) 세트를 구현하여 여러 응용프로그램이 동일한 방법으로 데이터를 처리하고, 따라서 일관적인 결과를 보장할 수 있습니다.

또한 사용자 정의 함수(UDF)는 응용프로그램에서 객체 지향 프로그래밍을 지원합니다. UDF는 사용자가 데이터 오브젝트에 대한 조작을 수행하는 데 사용할 수 있는 메소드를 정의할 수 있도록 추출을 제공합니다. 또한 UDF는 오브젝트의 하위 데이터에 대한 액세스를 제어하여 직접 조작 및 가능한 손상으로부터 보호할 수 있도록 캡슐화를 제공합니다.

OLE DB 테이블 함수

Microsoft OLE DB는 다양한 정보 소스에 저장된 데이터에 대한 단일 형태 액세스를 응용프로그램에 제공하는 OLE/COM 인터페이스 세트입니다. DB2 Universal Database는 사용자가 OLE DB 데이터 소스에 액세스하는 테이블 함수를 정의할 수 있게 하여 OLE DB 응용프로그램 작성을 단순화합니다. OLE DB 인터페이스를 통해 데이터를 표시하는 데이터 소스에 대한 GROUP BY, JOIN 및 UNION을 포함한 조작을 수행할 수 있습니다. 예를 들어, OLE DB 테이블 함수를 정의하여 Microsoft Access 데이터베이스 또는 Microsoft Exchange 주소록에서 테이블을 리턴하도록 OLE DB 테이블 함수를 정의한 다음 해당 OLE DB 테이블 함수의 데이터를 사용자의 DB2 데이터베이스에 있는 데이터와 이음새 없이 결합하는 보고서를 작성할 수 있습니다.

OLE DB 테이블 함수를 사용하면 OLE DB 제공업체에 대한 내장 액세스를 제공하여 응용프로그램 개발 노력을 감소시킵니다. C, Java 및 OLE 자동화 함수의 경우 개발자가 테이블 함수를 구현해야 하는 반면, OLE DB 테이블 함수의 경우 일반적인 내장 OLE DB 고객이 OLE DB 제공업체와 인터페이스하여 데이터를 검색합니다. 언어 유형 OLEDB의 테이블 함수를 등록하고, OLE DB 제공업체와

관련 행 세트를 데이터 소스로 참조하기만 하면 됩니다. OLE DB 테이블 함수를 이용하기 위해 UDF 프로그래밍을 수행할 필요가 없습니다.

OLE 자동화 UDF 및 저장 프로시듀어

OLE 자동화는 Microsoft Corporation의 OLE 2.0 아키텍처의 일부입니다. OLE 자동화를 사용하여 응용프로그램이 작성된 언어에 관계없이 OLE 자동화 오브젝트에서 등록 정보와 메소드를 표시할 수 있습니다. Lotus Notes 또는 Microsoft Exchange와 같은 그 밖의 응용프로그램은 OLE 자동화를 통해 이들 등록 정보 및 메소드의 이점을 이용하여 이들 오브젝트를 통합할 수 있습니다.

Windows 32비트 운영 체제용 DB2는 UDF 및 저장 프로시듀어를 사용하여 OLE 자동화 오브젝트에 대한 액세스를 제공합니다. OLE 자동화 오브젝트에 액세스하고 메소드를 호출하려면 오브젝트의 메소드를 UDF 또는 저장 프로시듀어로 등록해야 합니다. 그런 다음 DB2 응용프로그램이 UDF나 저장 프로시듀어를 호출하여 메소드를 호출할 수 있습니다. UDF는 스칼라 또는 테이블 함수입니다.

예를 들어, Microsoft Excel과 같은 제품을 사용하여 작성된 스프레드시트에서 데이터를 조회하는 응용프로그램을 개발할 수 있습니다. 이를 수행하려면 워크쉬트에서 데이터를 검색하여 DB2로 리턴하는 OLE 자동화 테이블을 개발합니다. 그런 다음 DB2가 데이터를 처리하고, 온라인 분석 처리(OLAP)를 수행하며, 조회 결과를 응용프로그램으로 리턴할 수 있습니다.

트리거

트리거는 지정된 테이블에 대한 삭제, 삽입 또는 갱신 조작으로 실행되는 조치 세트를 정의합니다. 이러한 SQL 조작이 실행될 때 트리거가 활성화되었다고 합니다. 트리거는 SQL 조작 이전 또는 이후 활성화될 수 있습니다. SQL문 CREATE TRIGGER를 사용하여 트리거를 정의합니다.

다음과 같은 몇 가지 방법으로 갱신 또는 삽입 조작 이전에 수행되는 트리거를 사용할 수 있습니다.

- 데이터베이스에서 실제로 갱신되거나 삽입되기 전에 값 점검 또는 수정. 사용자가 보는 방식에서 내부 데이터 형식으로 데이터를 변환해야 하는 경우 유용합니다.

- 사용자 정의 함수(UDF)에 코딩된 그 밖의 데이터베이스 이외 조작 수행.

마찬가지로, 다음과 같은 몇 가지 방법으로 갱신 또는 삽입 이후에 수행되는 트리거를 사용할 수 있습니다.

- 다른 테이블에 있는 데이터 갱신. 데이터 간의 관계를 유지보수하거나 감사 추적 정보 유지에 유용합니다.
- 테이블 또는 다른 테이블에 있는 다른 데이터 점검. 참조 무결성 제한조건이 적합하지 않거나, 테이블 점검 제한조건이 점검을 현재 테이블로만 제한할 때 데이터 무결성을 보장하는 데 유용합니다.
- 사용자 정의 함수(UDF)에 코딩된 데이터베이스 이외 조작 수행. 경보를 발행하거나 데이터베이스 외부의 정보를 갱신할 때 유용합니다.

트리거를 사용하는 몇 가지 이점은 다음과 같습니다.

보다 빠른 응용프로그램 개발

트리는 데이터베이스에 저장되며, 모든 응용프로그램에 대해 사용 가능합니다. 따라서 각각의 응용프로그램에 대해 동등한 기능을 코딩할 필요가 없습니다.

비지니스 규칙의 전역 시행

트리거는 한 번만 정의되며, 트리거가 제어하는 데이터를 사용하는 모든 응용프로그램에 의해 사용됩니다.

보다 쉬운 유지보수

트리거를 사용하는 모든 응용프로그램 대신 데이터베이스에서 한 번만 변경을 수행하면 됩니다.

DB2 Universal Database 도구

응용프로그램을 개발할 때 서로 다른 다양한 도구를 사용할 수 있습니다. DB2 Universal Database는 응용프로그램에서 SQL문을 작성하고 테스트하며, 성능을 모니터할 수 있도록 도와주는 다음과 같은 도구를 제공합니다.

주: 모든 도구가 모든 플랫폼에서 사용 가능하지는 않습니다.

제어 센터

데이터베이스 오브젝트(데이터베이스, 테이블 및 패키지와 같은)와 상호 관계를 표시하는 그래픽 인터페이스. 제어 센터를 사용하여 시스템 구성, 디렉토리 관리, 시스템 백업 및 복구, 작업 스케줄 및 미디어 관리와 같은 관리 타스크를 수행하십시오.

제어 센터는 다음과 같은 기능을 포함합니다.

명령 센터
대화식 창에 DB2 명령과 SQL문을 입력하고 결과 창에서 실행 결과를 보는 데 사용됩니다. 결과 화면을 이동하거나 출력을 파일에 저장할 수 있습니다.

스크립트 센터
저장한 다음 나중에 호출할 수 있는 스크립트를 작성하는 데 사용됩니다. 이들 스크립트에는 DB2 명령, SQL문 또는 운영 체제 명령도 포함될 수 있습니다. 부재시 수행하도록 스크립트를 스케줄할 수 있습니다. 이들 작업은 한 번씩 수행하거나 반복되는 스케줄로 수행되도록 설정할 수 있습니다. 반복되는 스케줄은 백업과 같은 타스크에 특히 유용합니다.

저널 다음과 같은 유형의 정보를 보는 데 사용됩니다. 실행 보류 중이거나, 실행 중인 작업 또는 실행이 완료된 작업에 대해 얻을 수 있는 모든 정보, 복구 실행 기록 로그, 경보 로그 및 메시지 로그. 또한 저널을 사용하여 부재시 수행되는 작업의 결과를 검토할 수 있습니다.

경보 센터
잠재적 문제점을 조기에 알려주기 위해 시스템을 모니터하거나, 문제점을 정정하는 조치를 자동화하는 데 사용됩니다.

도구 설정
제어 센터, 경보 센터 및 복제에 대한 설정을 변경하는 데 사용됩니다.

성능 모니터

제어 센터에 대한 설치 가능 옵션인 성능 모니터는 DB2 시스템에 대한 포괄적인 성능 데이터 콜렉션, 뷰, 보고, 분석 및 경보 기능을 제공하는 그래픽 인터페이스입니다. 성능을 조정하려면 성능 모니터를 사용하십시오.

스냅샷 또는 이벤트를 모니터하도록 선택할 수 있습니다. 스냅샷 모니터를 사용하여 지정된 간격으로 특정 시점의 정보를 캡쳐할 수 있습니다. 이벤트 모니터는 연결과 같이 이벤트 지속 기간 동안 성능 정보를 기록할 수 있게 해줍니다.

Visual Explain

제어 센터, Visual Explain에 대한 설치 가능 옵션은 SQL문에 대해 최적화 알고리즘이 선택한 액세스 플랜 뷰를 포함하여, SQL문을 분석하고 조정할 수 있게 해주는 그래픽 인터페이스입니다.

Stored Procedure Builder

주: 이 절에서는 Stored Procedure Builder 온라인 도움말과 *응용프로그램 개발 안내서*의 "IBM DB2 Stored Procedure Builder" 장의 정보를 보충합니다.

DB2 Stored Procedure Builder는 DB2 저장 프로시듀어의 빠른 개발을 지원하는 GUI 기반 도구입니다. 워크스테이션에서 OS/390에 이르기까지 DB2 계열에 대한 단일 개발 환경을 제공합니다. Windows 32비트 운영 체제에서 이 도구는 Microsoft Visual Studio, Microsoft Visual Basic 및 IBM VisualAge for Java와 같이 자주 사용되는 응용프로그램 개발 도구에서 시작하거나, IBM DB2 Universal Database 프로그램 그룹에서 별도의 응용프로그램으로서 시작할 수 있습니다. 이것은 또한 명령행에서 `db2spb` 명령을 실행하여 지원된 모든 플랫폼을 시작할 수 있습니다.

환경 구성

AIX, Solaris 또는 Windows 시스템에서 Stored Procedure Builder를 사용하여 SQL 프로시듀어를 빌드하려면 먼저 SQL 프로시듀어에 대해 DB2 환경을 구성해야 합니다. 117 페이지의 『제5장 SQL 프로시듀어 빌드』에 있는 플랫폼에 대한 설정 지침을 따르십시오.

AIX, Solaris 또는 Windows 시스템에서 Stored Procedure Builder를 사용하여 Java 저장 프로시듀어를 빌드하려면 먼저 Java에 대해 DB2 환경을 구성해야 합니다. 77 페이지의 『환경 설정』에 있는 플랫폼에 대한 설정 지침을 따르십시오.

Solaris에서 Stored Procedure Builder를 사용하여 빌드된 Java 저장 프로시듀어의 경우, 환경 변수 JAVA_HOME을 Java가 설치된 경로(즉, /bin 및 /lib 디렉토리를 포함하는 디렉토리)로 설정해야 합니다. 또한 DB2와 함께 제공된 것과 다른 다른 JDK 및 JRE를 설치할 경우 AIX에서 이를 수행해야 합니다.

Stored Procedure Builder는 Windows NT 및 Windows 2000 플랫폼 전용 Java 1.2를 사용한 Java 저장 프로시듀어 빌드를 지원합니다. Stored Procedure Builder는 아랍어와 히브리어와 같이 Java 1.2에서 양방향 지원을 사용하는 양방향 언어를 지원합니다.

Stored Procedure Builder를 사용하여 Java 저장 프로시듀어를 내보내려면, 다음을 수행하십시오.

1. 저장 프로시듀어 폴더를 마우스 오른쪽 단추로 누른 다음 Java 저장 프로시듀어 내보내기를 눌러 Java 저장 프로시듀어 내보내기 창을 여십시오.
2. 내보낼 저장 프로시듀어를 선택한 다음 "선택된 저장 프로시듀어" 컬럼으로 이동하십시오.
3. 원하는 옵션을 선택한 다음 확인을 누르십시오.

OS/390에서 저장 프로시듀어 빌드

DB2 Stored Procedure Builder는 OS/390용 DB2 버전 6 이상에서의 Java 저장 프로시듀어 개발을 지원하며, OS/390용 DB2 버전 7 서버에서의 SQL 프로시듀어 빌드를 지원합니다. 새로운 저장 프로시듀어를 작성하거나 기존 저장 프로시듀어를 변경할 수 있습니다.

WLM(Workload Manager) 응용프로그램 환경에서 수행할 저장 프로시듀어를 OS/390에서 빌드하고 나면 DB2 Stored Procedure Builder는 자동으로 WLM 주소 공간을 새로 고칩니다.

DB2 Stored Procedure Builder에 대해 DB2 OS/390 버전 5 이상에 연결한 경우, 마법사를 사용하여 저장 프로시듀어를 삽입하고 WLM 환경 변수를 전혀 표시하지 않으면, 생성된 코드에 NO WLM ENVIRONMENT 텍스트가 포함됩니

다. 이 코드 행으로 저장 프로시듀어는 예상대로 SPAS 주소 공간에서 수행됩니다. 이러한 수정으로 DB2 Stored Procedure Builder 버전 6 이상에 있었던 문제점이 해결됩니다.

수정 후에 생성된 코드는 다음과 같이 표시됩니다.

```
CREATE PROCEDURE SYSPROC.Proc2 (  )
    RESULT SETS 1
    LANGUAGE SQL
    MODIFIES SQL DATA
    COLLID TEST
    NO WLM ENVIRONMENT
    ASUTIME NO LIMIT
    RUN OPTIONS 'NOTEST(ALL,*,,VADTCPIP&9.112.14.91:*)'
-----------------------------------------------------------------
-- SQL Stored Procedure
-----------------------------------------------------------------
P1: BEGIN
    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR
        SELECT * FROM SYSIBM.SYSPROCEDURES;

    -- Cursor left open for client application
    OPEN cursor1;

END P1
```

OS/390용 DB2 Stored Procedure Builder에 대한 자세한 내용은 다음 웹 사이트를 참조하십시오.

```
http://www-4.ibm.com/software/data/db2/os390/spb/exciting
```

빌드 옵션 설정

AIX, Solaris 및 Windows 플랫폼에서 DB2 Stored Procedure Builder를 사용하면 모든 SQL 프로시듀어에 대해 빌드 옵션을 설정할 수 있습니다. 이 빌드 옵션에는 다음과 같은 컴파일러 및 사전 처리 컴파일러 DB2 레지스트리 변수가 포함됩니다.

- DB2_SQLROUTINE_PREPOPTS
- DB2_SQLROUTINE_COMPILER_PATH
- DB2_SQLROUTINE_COMPILE_COMMAND
- DB2_SQLROUTINE_KEEP_FILES

db2set 명령을 사용하여 이러한 레지스트리 변수를 설정할 수 있다고 해도 Stored Procedure Builder를 사용하면 실제로 명령을 발행하거나 변경사항이 적용되도록 중지한 다음 재시작하기 위해 데이터베이스 서버에 액세스해야 하는 필요성이 없어집니다.

SQL 저장 프로시듀어 빌드 옵션 창을 열려면 프로젝트 보기에서 데이터베이스 연결을 마우스 오른쪽 단추로 누른 다음 SQL 저장 프로시듀어 빌드 옵션을 누르십시오. 이들 옵션의 설정에 대한 자세한 내용은 DB2 저장 프로시듀어 온라인 도움말을 참조하십시오.

MQSeries용 테이블 UDF 및 OLE DB

DB2 Stored Procedure Builder는 MQSeries 및 OLE DB에 대한 테이블 UDF 작성을 돕는 마법사를 제공합니다. OLE DB 테이블 UDF 작성 마법사를 사용하여 OLE DB 데이터 제공업체에 액세스할 수 있습니다. 마법사는 OLE 테이블 UDF 및 선택적 테이블 보기를 작성합니다. MQSeries 테이블 UDF 작성 마법사를 사용하면 옵션 테이블 보기로 테이블 UDF를 작성하여 MQSeries 메시지에 액세스하고 표 형식의 데이터를 분석할 수 있습니다.

저장 프로시듀어 디버깅

AIX, Solaris 및 Windows에서 SQL 프로시듀어 디버깅은 직접 DB2 Stored Procedure Builder에 통합됩니다. 분리되지 않은(신뢰되는) SQL 프로시듀어를 디버깅할 때 KEEPDARI 데이터베이스 관리자 구성 옵션을 YES 또는 NO로 설정할 수 있습니다. 그러나 분리(신뢰되지 않는) SQL 프로시듀어를 디버깅할 때는 YES(기본값)로 설정해야 합니다. 통합 디버거에 대한 자세한 내용은 Stored Procedure Builder 도움말을 참조하십시오.

AIX, Solaris 및 Windows 플랫폼에서 Java 및 C 저장 프로시듀어에 대해 원격 디버깅 성능을 사용하려면 IBM Distributed Debugger를 설치해야 합니다. IBM Distributed Debugger는 Visual Age for Java Professional Edition CD에 있습니다. 디버거 클라이언트는 Windows 플랫폼에서만 수행됩니다. AIX, Solaris 및 Windows는 지원되는 서버 플랫폼입니다. AIX, Solaris 및 Windows에 대해 지역 및 원격 SQL 저장 프로시듀어를 디버그하려면 Stored Procedure Builder 내장 SQL 디버그 성능을 사용하십시오. OS/390 플랫폼에서 SQL 프로시듀어를 디

버그하려면 OS/390 R1 제품용 IBM C/C++ Productivity Tools가 있어야 합니다. OS/390 R1용 IBM C/C++ Productivity Tools에 대한 자세한 내용은 다음 웹 사이트를 참조하십시오.

```
http://www.ibm.com/software/ad/c390/pt/
```

알려진 문제점, 제한사항 및 일시적인 해결책

- SQL 프로시듀어는 현재 Windows 98에서 지원되지 않습니다.
- Java 저장 프로시듀어의 경우, JAR ID, 클래스 이름 및 메소드 이름에 ASCII 이외 문자를 포함할 수 없습니다.
- AS/400에서는 다음 V4R4 PTF를 OS/400 V4R4에 적용해야 합니다.

  - SF59674
  - SF59878

- 문자 부속 유형이 FOR MIXED DATA 또는 FOR SBCS DATA인 저장 프로시듀어 매개변수는 저장 프로시듀어가 데이터베이스에서 복원될 때 편집기 분할창에서 소스 코드에 표시되지 않습니다.
- 현재, 데이터베이스에서 Java 소스 코드를 검색할 때 문제점이 있습니다. 검색 시, 코드의 주석이 축소되어 표시됩니다. 이는 ASCII 이외 코드 페이지에서 작업 중이거나 클라이언트 및 서버가 서로 다른 코드 페이지에 있는 DB2 Stored Procedure Builder 사용자에게 영향을 줍니다.
- 대만어 로케일을 사용하여 JDK 1.1.8이나 JRE 1.1.8을 사용할 경우에 문제점이 있습니다. Stored Procedure Builder 프로그램의 그래픽 측면(메뉴, 편집기 텍스트, 메시지 등을 포함하여)이 적절하게 표시되지 않습니다. 해결책은 다음 디렉토리 중 하나 또는 둘 다에 나타나는 `font.properties.zh_TW` 파일을 변경하는 것입니다.

  ```
  sqllib/java/jdk/lib
  sqllib/java/jre/lib
  ```

다음 부분을

```
monospaced.0=\u7d30\u660e\u9ad4,CHINESEBIG5_CHARSET,NEED_CONVERTED
```

다음과 같이 변경하십시오.

```
monospaced.0=Courier New,ANSI_CHARSET
```

제1장 소개

이 책은 DB2 응용프로그램을 개발하기 위한 환경을 설정하는 데 필요한 정보와, 해당 환경에서 이러한 응용프로그램을 컴파일, 링크 및 수행하는 데 필요한 단계별 지침을 제공합니다. 또한 다음 플랫폼에서 DB2 Universal Database용 DB2 Application Development(DB2 AD) Client 버전 7을 사용하여 응용프로그램을 빌드하는 방법에 대해 설명합니다.

- AIX
- HP-UX
- Linux
- OS/2
- PTX
- Silicon Graphics IRIX
- Solaris 운영 체제
- Windows 32비트 운영 체제

주:

1. NUMA-Q용 DB2는 PTX 운영 체제를 지원합니다.

2. Windows 32비트 운영 체제에는 Windows NT, Windows 95, Windows 98 및 Windows 2000이 포함됩니다. 이 책에서 Windows 32비트 운영 체제를 언급하는 경우, SNA(Systems Network Architecture) 지원, REXX 지원을 제외한 위의 모든 운영 체제를 의미합니다. 이들은 Windows NT 및 Windows 2000에서만 지원됩니다.

응용프로그램을 개발하기 위해 다음과 같은 프로그래밍 인터페이스를 사용할 수 있습니다.

DB2 API

DB2 데이터베이스를 관리하기 위한 관리 기능을 제공합니다.

DB2 콜 레벨 인터페이스(DB2 CLI)

X/Open CLI 스펙에 따라 호출 가능한 SQL 인터페이스이며, Microsoft 사의 ODBC(Open Database Connectivity) 인터페이스와 호환 가능합니다.

Embedded SQL

런타임 함수 호출로 변환하려면 사전 처리 컴파일해야 하는 프로그램에서 직접 코딩된 SQL문을 사용합니다.

SQLJ(Embedded SQL for Java)

사전 처리 컴파일되어 런타임 함수 호출로 사용자 정의되며, 데이터베이스 관리자에 대한 인터페이스를 제공하는 생성된 프로파일에서 SQL문을 사용합니다.

JDBC(Java Database Connectivity)

Java용 동적 SQL API입니다. JDBC API는 지원되는 플랫폼에서 사용 가능한 JDK에 포함됩니다.

각각의 서로 다른 프로그래밍 인터페이스에 대한 자세한 내용은 다음을 참조하십시오.

- *응용프로그램 개발 안내서*. Embedded SQL, Java용 Embedded SQL 및 JDBC(Java Database Connectivity)를 사용하여 DB2 계열 서버에 액세스하는 응용프로그램을 코딩하고 설계하는 방법에 대해 설명합니다. 또한 사용자 정의 함수(UDF)에 대해서도 설명합니다.

- *CLI Guide and Reference*. DB2 콜 레벨 인터페이스 및 ODBC를 사용하는 응용프로그램을 코딩하고 설계하는 방법에 대해 설명합니다.
- *Administrative API Reference*. DB2 API를 사용하는 응용프로그램을 코딩하고 설계하는 방법에 대해 설명합니다.

다음의 책은 자세한 제품 설치 및 설정과 같은 관련 정보를 찾는 데 유용합니다.

- *OS/2용 DB2 빠른 시작*에서는 OS/2 서버 및 클라이언트 워크스테이션에서 데이터베이스 관리자 및 DB2 Application Development Client를 설치하는 방법에 대해 설명합니다.
- *UNIX용 DB2 빠른 시작*에서는 UNIX 서버 및 클라이언트 워크스테이션에서 데이터베이스 관리자 및 DB2 Application Development Client를 설치하는 방법에 대해 설명합니다.
- *Windows용 DB2 빠른 시작*에서는 Windows 32비트 운영 체제용 서버 및 클라이언트 워크스테이션에 데이터베이스 관리자 및 DB2 Application Development Client를 설치하는 방법에 대해 설명합니다.
- *Command Reference*에서는 DB2 명령행 처리기(CLP)와 모든 DB2 명령을 사용하는 방법에 대해 설명합니다.
- *문제점 해결 안내서*에서는 DB2 클라이언트 및 서버를 포함하여 응용프로그램 개발 문제점과 데이터베이스 관리 및 연결성 관련 타스크 문제점을 해결할 수 있도록 도와줍니다.

DB2 문서 라이브러리의 전체 목록에 대한 자세한 내용은 471 페이지의 『부록D. DB2 라이브러리 사용』을 참조하십시오.

주: 이 책의 예는 어떠한 종류의 보증 없이 "현상태대로" 제공됩니다. IBM이 아닌 사용자가 품질, 성능 및 결함 복구에 있어서 전반적인 위험을 감수해야 합니다.

이 책의 사용자

DB2 Universal Database 버전 7에 대해 현재 지원되는 플랫폼 중 하나에서 프로그램을 개발하려는 경우, 이 책을 사용해야 합니다. 이 책에서는 프로그램이 DB2 API를 사용하여 DB2 데이터베이스를 관리하고, DB2 CLI, Embedded SQL, SQLJ 및 JDBC를 사용하여 DB2 데이터베이스에 액세스하는 방법에 대해 설명합니다.

이 책을 사용하려면 사용 중인 플랫폼에서 지원되는 프로그래밍 언어를 하나 이상 알아야 합니다. 이들 언어가 8 페이지의 『플랫폼별로 지원되는 소프트웨어』에 나열됩니다.

이 책의 사용법

이 책은 응용프로그램을 개발하는 데 필요한 정보에 쉽게 액세스할 수 있도록 고안되었습니다. 각 장들은 다음과 같이 묶여 있습니다.

- 1-3장: 각 장에는 모든 플랫폼에 대한 일반, 소개 정보가 들어 있습니다.
- 4-5장: 각 장에는 모든 플랫폼에 대한 특정 프로그래밍 정보가 들어 있습니다.
- 6-13장: 각 장에는 하나의 플랫폼에만 고유하게 적용되는 프로그래밍 정보가 들어 있습니다.

모든 응용프로그램 개발자는 먼저 앞의 세 장을 읽은 다음 사용 중인 운영 체제와 프로그래밍 언어에 따라 필요한 특정 프로그래밍 정보를 포함하는 "응용프로그램 빌드" 장을 읽어야 합니다.

부록은 다양한 주제에 관한 중요한 추가 정보를 제공합니다.

강조표시 규칙

이 책에서는 다음과 같은 규칙을 사용합니다.

이탤릭체

다음 중 하나를 표시합니다.

- 새로운 용어에 대한 소개
- 사용자가 제공하는 이름 또는 값
- 다른 정보 소스에 대한 참조

- 일반적인 강조

대문자 다음 중 하나를 표시합니다.

- 데이터베이스 관리자 데이터 유형
- 필드 이름
- 키워드
- SQL문

예제 텍스트

다음 중 하나를 표시합니다.

- 코딩 예 및 코드 일부
- 명령
- 시스템에서 표시하는 것과 유사한 출력 예
- 특정 데이터 값 예
- 시스템 메시지 예
- 파일 및 디렉토리 이름
- 사용자가 입력하도록 지시된 정보

굵은체 항목을 강조합니다.

DB2 Application Development Client 정보

주: Application Development Client는 이전 버전의 DB2에서는 DB2 Software Development Kit(DB2 SDK) Client로 알려져 있습니다.

DB2 Application Development(DB2 AD) Client는 DRDA(Distributed Relational Database Architecture)를 구현하는 응용프로그램 서버와 DB2 서버에 액세스하는 응용프로그램을 개발하는 데 필요한 도구와 환경을 제공합니다.

DB2 AD Client가 설치된 상태에서 DB2 응용프로그램을 빌드하고 수행할 수 있습니다. 또한 다음 DB2 클라이언트에서도 DB2 응용프로그램을 수행할 수 있습니다.

- DB2 Run-Time Client
- DB2 Administration Client

프로그래밍 환경 설정에 대한 자세한 내용은 39 페이지의 『제2장 설정』을 참조하십시오.

이 책에서 설명하는 플랫폼에 대한 DB2 AD Client는 다음과 같습니다.

- **C/C++, COBOL 및 Fortran용 사전 처리 컴파일러**(이 언어가 해당 플랫폼에서 지원되도록 제공합니다. 자세한 내용은 8 페이지의 『플랫폼별로 지원되는 소프트웨어』를 참조하십시오.)
- 프로그래밍 라이브러리, 포함 파일 및 코드 샘플을 포함하는 **Embedded SQL 응용프로그램 지원**.
- ODBC로 쉽게 포팅되고 ODBC SDK로 컴파일되는 응용프로그램을 개발하기 위한 프로그래밍 라이브러리, 포함 파일 및 코드 샘플을 포함하는 **DB2 콜 레벨 인터페이스(DB2 CLI) 응용프로그램 지원**. ODBC SDK는 Windows 32비트 운영 체제의 경우 Microsoft에서, 그 밖의 지원되는 플랫폼의 경우 다른 여러 벤더에서 구할 수 있습니다. Windows 32비트 운영 체제의 경우, DB2 클라이언트에는 Microsoft ODBC Software Developer's Kit로 개발된 응용프로그램을 지원하는 ODBC 드라이버가 들어 있습니다. 그 밖의 모든 플랫폼의 경우, DB2 클라이언트에는 존재하는 경우 해당 플랫폼에 대한 ODBC SDK로 개발할 수 있는 응용프로그램을 지원하는 선택적으로 설치된 ODBC 드라이버가 들어 있습니다. Windows 32비트 운영 체제용 DB2 클라이언트에만 ODBC 드라이버 관리자가 들어 있습니다.
- Java 응용프로그램과 애플릿을 개발하기 위한 DB2 JDBC(Java Database Connectivity) 지원과, Java Embedded SQL 응용프로그램과 애플릿을 개발하기 위한 DB2 SQLJ(DB2 Embedded SQL for Java) 지원을 포함하는 **DB2 Java 지원**.
- AIX용 DB2와 Windows 32비트 운영 체제용 DB2와 함께 설치되고 OS/2용 DB2와 함께 제공되는 IBM의 **JDK 1.1.8 및 JRE 1.1.8**.
- Microsoft Visual Basic 및 Microsoft Visual C++에서 구현되는 코드 샘플을 포함하는 Windows 32비트 운영 체제의 **ADO(ActiveX Data Objects) 및 OLE 자동화 UDF와 저장 프로시듀어**. 또한 Microsoft Visual Basic에서 구현된 RDO(Remote Data Objects)를 사용하는 코드 샘플.
- Windows 32비트 운영 체제의 **OLE DB 테이블 함수**.

- AIX, Solaris 및 Windows 32비트 운영 체제에서 사용할 수 있는 **DB2 SPB(Stored Procedure Builder)**. 이것은 DB2 저장 프로시듀어의 빠른 개발을 지원하는 GUI 기반 도구입니다. 워크스테이션에서 OS/390에 이르기까지 DB2 계열에 대한 단일 개발 환경을 제공합니다. Windows에서 이 도구는 Microsoft Visual Studio, Microsoft Visual Basic 및 IBM VisualAge for Java와 같이 자주 사용되는 응용프로그램 개발 도구로 시작하거나 IBM DB2 Universal Database 프로그램 그룹에서 별도의 응용프로그램으로 시작할 수 있습니다. AIX 및 Solaris에서 이 도구는 `db2spb` 명령을 사용하여 시작할 수 있습니다.
- SQL문의 프로토타입을 생성하거나 데이터베이스에 대해 임시 조회를 수행하기 위한 명령 센터 또는 명령행 처리기(CLP)를 통한 **대화식 SQL**.
- 다른 응용프로그램 개발 도구가 제품 내에서 직접 DB2에 대한 사전 처리 컴파일러 지원을 구현할 수 있도록 하는 **문서화된 API 세트**. 예를 들어, AIX 및 OS/2에서 IBM COBOL은 이 인터페이스를 사용합니다. 사전 처리 컴파일러 서비스 API 세트에 대한 정보는 anonymous FTP 사이트 ftp://ftp.software.ibm.com에서 구할 수 있습니다. prepapi.psbin이라는 PostScript 파일이 /ps/products/db2/info 디렉토리에 있습니다. 이 파일은 2진 형식입니다. 해당 전자 포럼에 대한 액세스 권한이 없고 이 문서의 사본을 구하려는 경우에는 서비스 정보 전단에 기술된 대로 IBM 서비스에 주문할 수 있습니다.
- 응용프로그램에서 ISO/ANSI SQL92 Entry Level 표준을 따르지 않거나, OS/390용 DB2에서 지원하지 않는 Embedded SQL문을 식별하는 **SQL92 및 MVS Conformance Flagger**. 워크스테이션에서 개발된 응용프로그램을 다른 플랫폼으로 이주하려는 경우, 플래거는 구문 비호환성을 표시하여 시간을 절약합니다. PRECOMPILE PROGRAM 명령에서 SQLFLAG 옵션에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

지원되는 서버

DB2 AD Client를 사용하여 특정 플랫폼에서 수행되는 응용프로그램을 개발할 수 있습니다. 그러나 응용프로그램은 다음과 같은 플랫폼 서버에서 원격 데이터베이스에 액세스할 수 있습니다.

- AIX용 DB2

- HP-UX용 DB2
- Linux용 DB2
- OS/2용 DB2
- NUMA-Q용 DB2
- SCO UnixWare 7용 DB2
- Solaris용 DB2
- Windows NT용 DB2
- DRDA(Distributed Relational Database Architecture) 호환 응용프로그램 서버는 다음과 같습니다.
  - OS/390용 DB2
  - AS/400용 DB2
  - VSE 및 VM용 DB2(이전에는 VM 및 VSE용 SQL/DS이였음)
  - IBM 이외의 데이터베이스 벤더의 DRDA 호환 응용프로그램 서버

주:

1. NUMA-Q용 DB2는 PTX 운영 체제를 지원합니다.
2. SCO UnixWare 7용 DB2는 DB2 버전 5.2에서만 사용할 수 있습니다.

플랫폼별로 지원되는 소프트웨어

이 절에서는 이 책에 설명된 플랫폼에 대한 DB2가 지원하는 컴파일러와 관련 소프트웨어를 나열합니다. 컴파일러 정보는 사용자가 나열된 컴파일러 중 하나에 빌드할 수 있는 사전 처리 컴파일러 지원이 아닌, 해당 플랫폼에 대해 DB2 사전 처리 컴파일러를 사용되는 것으로 가정합니다. 지원하는 통신 제품에 관한 내용은 운영 체제의 *빠른 시작* 책을 참조하십시오.

최신 DB2 컴파일러 정보 및 관련 소프트웨어 갱신사항을 보려면 다음의 DB2 응용프로그램 개발 웹 페이지를 방문하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

주:

1. **DB2 릴리스 정보**에는 지원되는 플랫폼에 대한 갱신된 컴파일러 및 운영 체제 정보가 들어 있습니다. 이 릴리스 정보는 제품 CD-ROM의 다음 경로에서 플

랫 텍스트 및 HTML 형식으로 사용할 수 있습니다. 여기서 <language_directory>는 사용 중인 언어에 대한 디렉토리이고 index.htm은 기본 HTML 파일입니다.

플랫 텍스트 파일:

doc/<language_directory>/release.txt(UNIX)

Doc\<language_directory>\release.txt(OS/2 및 Windows)

HTML 파일:

doc/<language_directory>/db2ir/index.htm(UNIX)

Doc\<language_directory>\db2ir\index.htm(OS/2 및 Windows)

2. **Fortran 및 REXX.** DB2는 DB2 Universal Database 버전 5.2에서 이들 언어에 대한 지원 레벨을 초과하는 Fortran 및 REXX 기능을 향상시키지 않습니다.
3. **Fortran.** Fortran 샘플 프로그램은 DB2 버전 7에서는 지원되지 않습니다.
4. **HP-UX**. HP-UX 버전 10 이하에서 HP-UX 버전 11로 DB2를 이주할 경우, DB2 프로그램은 HP-UX 버전 11에 있는 DB2에서 다시 사전 처리 컴파일된 다음(Embedded SQL을 포함하는 경우) 다시 컴파일되어야 합니다. 모든 DB2 응용프로그램, 저장 프로시듀어, 사용자 정의 함수(UDF) 및 User Exit 프로그램을 포함합니다. 또한 HP-UX 버전 11에서 컴파일된 DB2 프로그램은 HP-UX 버전 10 이하에서는 수행되지 않을 수 있습니다. HP-UX 버전 10에서 컴파일되고 수행되는 DB2 프로그램은 HP-UX 버전 11 서버에 원격으로 연결할 수 있습니다.
5. **Micro Focus COBOL**. DB2 버전 2.1.1 이하에서 사전 처리 컴파일되고 Micro Focus COBOL에서 컴파일된 기존 응용프로그램은 현재 DB2 버전에서 다시 사전 처리 컴파일된 다음 Micro Focus COBOL에서 다시 컴파일되어야 합니다. IBM 사전 처리 컴파일러의 초기 버전에서 빌드된 이들 응용프로그램을 다시 사전 처리 컴파일하지 않으면 비정상 종료가 발생할 경우에 데이터베이스가 손상될 가능성이 있습니다.

6. **Perl**. 인쇄시 Perl DBI(Perl 데이터베이스 인터페이스)용 DB2 UDB 드라이버(DBD::DB2) 릴리스 0.75는 AIX, HP-UX, Linux, Solaris 및 Windows NT에서 사용할 수 있습니다. 최신 드라이버는 다음 사이트에서 다운로드할 수 있습니다.

   http://www.ibm.com/software/data/db2/perl

7. **REXX**. Windows NT/95용 IBM 오브젝트 REXX는 더 이상 DB2와 함께 제공되지 않습니다. 오브젝트 REXX를 구하려면 다음 사이트를 방문하십시오.

   http://www.ibm.com/software/ad/obj-rexx/

8. **PHP**. 이제는 웹 기반 응용프로그램에서 DB2에 액세스하기 위한 방법으로 PHP를 사용할 수 있습니다. PHP는 서버측, HTML-embedded, 플랫폼 간 스크립팅 언어입니다. 이는 사용자 레벨 PHP가 ODBC 호출을 사용하여 DB2와 통신하는 Unified-ODBC 액세스 메소드를 사용한 DB2 액세스를 지원합니다. 표준 ODBC와는 달리, Unified-ODBC 메소드를 사용하면 ODBC 계층을 통하지 않고 직접 DB2 CLI 계층과 통신하게 됩니다. DB2에서의 PHP 사용에 대한 자세한 내용은 DB2 지원 사이트를 참조하십시오.

   www.ibm.com/software/data/db2/udb/winos2unix/support

AIX

AIX용 DB2는 다음과 같은 운영 체제를 지원합니다.

AIX/6000

버전 4.2.1(32비트 전용)

버전 4.3.3 이상

PowerPC상의 AIX 5

주: 달리 명시되지 않으면 이 버전들은 32비트 및 64비트용입니다.

AIX용 DB2는 다음과 같은 프로그래밍 언어와 컴파일러를 지원합니다.

C AIX용 IBM C 버전 3.6.6(32비트 전용)

AIX용 IBM C 버전 3.6.6.3

AIX용 IBM C 버전 4.4

AIX용 IBM C 버전 5.0

주: 달리 명시되지 않으면 이 버전들은 32비트 및 64비트용입니다.

C++ AIX용 IBM C Set++ 버전 3.6.6(32비트 전용)

AIX용 IBM C Set++ 버전 3.6.6.3

IBM VisualAge C++ 버전 4.0

IBM VisualAge C++ 버전 5.0

주: 달리 명시되지 않으면 이 버전들은 32비트 및 64비트용입니다.

COBOL

AIX용 IBM COBOL Set 버전 1.1

AIX 4.2.1용 Micro Focus COBOL 버전 4.0.20(PRN 12.03 이상)

AIX 4.2.1용 Micro Focus COBOL 버전 4.1.10(PRN 13.04 이상)

AIX 4.3.3 이상용 Micro Focus Server Express 1.0.00(PRN 13.08 이상)

Fortran

AIX용 IBM XL Fortran 버전 4.1(32비트 전용) 및 5.1.0(32비트 및 64비트용)

Java AIX 4.2.1 및 AIX 4.3.3 이상의 경우: IBM의 AIX용 JDK 버전 1.1.8 및 JRE 버전 1.1.8(DB2와 함께 설치). JDK 빌드 날짜는 1999년 11월 24일이나 이후여야 합니다.

AIX 4.3.3 이상의 경우: IBM의 AIX용 JDK 버전 1.2.2 및 JRE 버전 1.2.2. JDK 빌드 날짜는 2000년 4월 17일이나 이후여야 합니다.

Perl Perl 5.004_04, DBI 0.93(DB2 UDB 드라이버 DBD::DB2에 대한 내용은 위의 주 참조)

REXX

IBM AIX REXX/6000 AISPO 제품 번호: 5764-057

AIX용 IBM 오브젝트 REXX 버전 1.1

REXXSAA 4.00

주: REXX 지원은 32비트용입니다.

HP-UX

HP-UX용 DB2는 다음 운영 체제를 지원합니다.

HP-UX

버전 11 및 11i

HP-UX용 DB2는 다음과 같은 프로그래밍 언어와 컴파일러를 지원합니다.

C HP C 컴파일러 버전 A.11.00.03

C++ HP aC++ 버전 A.03.25

COBOL

Micro Focus COBOL 버전 4.1

Fortran

HP Fortran/9000 버전 10.0

HP-UX F77 B.11.00.01

Java Hewlett-Packard의 Java용 HP-UX Developer's Kit 릴리스 1.1.8

Hewlett-Packard의 Java용 HP-UX Developer's Kit 릴리스 1.2.2

Perl Perl 5.004_04, DBI 0.93(DB2 UDB 드라이버 DBD::DB2에 대한 내용은 위의 주 참조)

Linux

Intel x86을 위한 Linux용 DB2(32비트 아키텍처)는 다음과 같은 운영 체제 환경을 지원합니다.

Linux kernel 버전 2.2.12 이상, glibc 버전 2.1.2. 이상, libstdc++ 버전 2.9.0, rpm(설치시 필요) 및 pdksh 패키지

S/390의 Linux용 DB2는 다음과 같은 운영 체제 환경을 지원합니다.

Linux kernel 버전 2.2.16 이상, glibc 2.1.3 이상, libstdc++ 버전 6.1, rpm(설치시 필요), pdksh 패키지 및 다음 중 하나: S/390용 SuSE Linux v7.0 또는 zSeries 및 S/390용 Turbolinux Server 6

Linux용 DB2는 다음과 같은 프로그래밍 언어와 컴파일러를 지원합니다.

C GNU/Linux gcc 버전 egcs-2.91.66(egcs-1.1.2 릴리스)

C++ GNU/Linux g++ 버전 egcs-2.91.66(egcs-1.1.2 릴리스)

Java Linux용 IBM Developer Kit 및 런타임 환경 버전 1.1.8(JDK 빌드 날짜는 1999년 11월 24일이나 이후여야 합니다.)

Linux용 IBM Developer Kit 및 런타임 환경 버전 1.2.2

Linux용 IBM Developer Kit 및 런타임 환경 버전 1.3

Perl Perl 5.004_04, DBI 0.93(DB2 UDB 드라이버 DBD::DB2에 대한 내용은 위의 주 참조)

OS/2

OS/2용 DB2는 다음과 같은 운영 체제를 지원합니다.

OS/2 WARP 3.0, WARP 4.0 및 WARP 4.5

OS/2용 DB2는 다음과 같은 프로그래밍 언어를 지원합니다.

C/C++

OS/2용 IBM VisualAge C++ 버전 3.6.5 및 4.0

주: 다음 사이트에서 VisualAge 컴파일러 버전에 대해 사용 가능한 최신 FixPak을 다운로드하십시오.

```
http://www.ibm.com/software/ad/vacpp/service/csd.html
```

VisualAge C++ 컴파일러에 대한 향후 서비스 지원 제한사항에 대해서는 다음 사이트에 있는 뉴스 절을 참조하십시오.

```
http://www.ibm.com/software/ad/vacpp
```

COBOL

OS/2용 IBM VisualAge COBOL 버전 2.0

Micro Focus COBOL 버전 4.0.20

FORTRAN

WATCOM FORTRAN 77 32 버전 10.5

Java IBM의 JDK 버전 1.1.8 및 OS/2용 JRE 버전 1.1.8. JDK 빌드 날짜는 1999년 11월 24일이나 이후여야 합니다.

REXX

IBM Procedures Language 2/REXX(OS/2의 일부로 지원됨)

PTX

NUMA-Q용 DB2는 다음과 같은 운영 체제를 지원합니다.

PTX 버전 4.5.1 이상

NUMA-Q용 DB2는 다음과 같은 프로그래밍 언어와 컴파일러를 지원합니다.

C ptx/C 버전 4.5

C++ ptx/C++ 버전 5.2

Java ptx/JSE 버전 3.0

Silicon Graphics IRIX

Silicon Graphics IRIX용 DB2는 다음과 같은 운영 체제를 지원합니다.

Silicon Graphics IRIX

버전 6.2 이상

Silicon Graphics IRIX용 DB2는 다음과 같은 프로그래밍 언어와 컴파일러를 지원합니다.

C MIPSpro C Compiler 7.2

C++ MIPSpro C++ 7.2

Fortran

MIPSpro Fortran-77 7.2

Java Silicon Graphics, Inc.의 Java2 Software Development Kit 버전 1.2.2(JDK 1.2.2)

주: Silicon Graphics IRIX 지원은 32비트 전용입니다.

Solaris

Solaris용 DB2는 다음과 같은 운영 체제를 지원합니다.

Solaris

버전 2.6(32비트 전용)

Solaris 7 및 Solaris 8(32비트 및 64비트)

Solaris용 DB2는 다음과 같은 프로그래밍 언어와 컴파일러를 지원합니다.

C Forte/WorkShop C 버전 4.2(32비트 전용), 5.0, 6 및 6.1(32비트 및 64비트)

주: 이들 컴파일러 버전은 ″SPARCompiler″를 호출하는 데 사용됩니다.

C++ Forte/WorkShop C++ 버전 4.2(32비트 전용), 5.0, 6 및 6.1(32비트 및 64비트)

주: 이들 컴파일러 버전은 ″SPARCompiler″를 호출하는 데 사용됩니다.

COBOL

Micro Focus COBOL Server Express 버전 1.0.00

Fortran

SPARCompiler Fortran 버전 4.2 및 5.0

Java Sun Microsystems의 Solaris용 JDK 버전 1.1.8 및 1.2.2

Perl Perl 5.004_04, DBI 0.93(DB2 UDB 드라이버 DBD::DB2에 대한 내용은 위의 주 참조)

Windows 32비트 운영 체제

Windows 32비트 운영 체제용 DB2는 다음을 지원합니다.

Microsoft Windows NT

서비스 팩 4 이상이 있는 버전 4.0

Microsoft Windows Millennium Edition

Microsoft Windows 2000

Microsoft Windows 98

Microsoft Windows 95

버전 4.00.950 이상

Windows 32비트 운영 체제용 DB2는 다음과 같은 프로그래밍 언어를 지원합니다.

Basic Microsoft Visual Basic 버전 4.2 및 버전 5.0(이 언어에 대해 제공되는 DB2 사전 처리 컴파일러는 없음)

C/C++

Microsoft Visual C++ 버전 5.0 및 6.0

Windows용 IBM VisualAge C++ 버전 3.6.5 및 4.0

주: 다음 사이트에서 VisualAge 컴파일러 버전에 대해 사용 가능한 최신 FixPak을 다운로드하십시오.

```
http://www.ibm.com/software/ad/vacpp/service/csd.html
```

VisualAge C++ 컴파일러에 대한 향후 서비스 지원 제한사항에 대해서는 다음 사이트에 있는 뉴스 절을 참조하십시오.

```
http://www.ibm.com/software/ad/vacpp
```

COBOL

Micro Focus COBOL 버전 4.0.20

Micro Focus COBOL Net Express 버전 3.0

IBM VisualAge COBOL 버전 2.0

REXX

Windows NT/95용 IBM 오브젝트 REXX 버전 1.1(위의 주 참조)

Java IBM의 Win32용 JDK 버전 1.1.8 및 JRE 버전 1.1.8(선택적으로 DB2와 함께 설치). JDK 빌드 날짜는 1999년 11월 24일이나 이후여야 합니다.

Sun Microsystems의 Win32용 JDK 1.2.2

IBM의 Win32용 JDK 1.2.2 JDK 빌드 날짜는 2000년 4월 17일이나 이후여야 합니다.

Java용 Microsoft Software Developer's Kit 버전 3.1

Perl Perl 5.004_04, DBI 0.93(Windows NT에서 사용 가능. DB2 UDB 드라이버 DBD::DB2에 대한 내용은 위의 주 참조)

샘플 프로그램

주:

1. 이 절에서는 DB2에서 지원하는 모든 플랫폼의 프로그래밍 언어에 대한 샘플 프로그램에 대해 설명합니다. 모든 샘플 프로그램이 모든 플랫폼 또는 지원되는 프로그래밍 언어로 포팅되지는 않았습니다.
2. DB2 샘플 프로그램은 어떠한 종류의 보증 없이 "현상태대로" 제공됩니다. IBM이 아닌 사용자가 품질, 성능 및 결함 복구에 있어서 전반적인 위험을 감수해야 합니다.

DB2 응용프로그램 개발(DB2 AD) 클라이언트와 함께 제공되는 샘플 프로그램. 샘플 프로그램을 템플리트로 사용하여 사용자의 응용프로그램을 작성할 수 있습니다.

샘플 프로그램 파일 확장자는 지원되는 각 언어에 대해 다르며 각 언어 내에서 Embedded SQL 및 비 Embedded SQL 프로그램에 대해서도 각기 다릅니다. 또한 파일 확장자는 한 언어 내에서 프로그램 그룹에 대해서도 다를 수 있습니다. 이러한 다른 샘플 파일 확장자는 다음 표에서 범주화됩니다.

언어에 의한 샘플 파일 확장자
19 페이지의 표1.

프로그램 그룹에 의한 샘플 파일 확장자
19 페이지의 표2.

다음 표에서는 유형별로 샘플 프로그램에 대해 설명합니다.

비 Embedded SQL이 있는 DB2 API 샘플 프로그램
22 페이지의 표3.

DB2 API Embedded SQL 샘플 프로그램
25 페이지의 표4.

DB2 API가 없는 Embedded SQL 샘플 프로그램

27 페이지의 표5.

사용자 정의 함수(UDF) 샘플 프로그램

29 페이지의 표6

DB2 CLI 샘플 프로그램

29 페이지의 표7.

Java JDBC 샘플 프로그램

30 페이지의 표8.

Java SQLJ 샘플 프로그램

31 페이지의 표9.

SQL 프로시듀어 샘플 프로그램

32 페이지의 표10.

ADO(ActiveX Data Object), RDO(Remote Data Object) 및 Microsoft Transaction Server 샘플 프로그램

34 페이지의 표11.

OLE(Object Linking and Embedding) 자동 샘플 프로그램

35 페이지의 표12.

OLE DB(Object Linking and Embedding Database) 테이블 함수

36 페이지의 표13.

명령행 처리기(CLP) 샘플 프로그램

36 페이지의 표14.

Log Management User Exit 프로그램

37 페이지의 표15.

주:

1. 25 페이지의 표4는 DB2 API 및 Embedded SQL문이 둘 다 있는 프로그램을 포함합니다. 모든 DB2 API 샘플 프로그램에 대한 자세한 내용은 22 페이지의 표3 및 25 페이지의 표4를 참조하십시오. 모든 Embedded SQL 샘플 프로그램(Java SQLJ 제외)에 대한 자세한 내용은 25 페이지의 표4 및 27 페이지의 표5를 참조하십시오.

2. 29 페이지의 표6 UDF 샘플 프로그램은 DB2 CLI UDF 프로그램을 포함하지 않습니다. 자세한 내용은 29 페이지의 표7을 참조하십시오.

표 1. 언어에 의한 샘플 파일 확장자

언어	디렉토리	**Embedded SQL** 프로그램	비 **Embedded SQL** 프로그램
C	samples/c samples/cli(CLI 프로그램)	.sqc	.c
C++	samples/cpp	.sqC(UNIX) .sqx(Windows & OS/2)	.C(UNIX) .cxx(Windows & OS/2)
COBOL	samples/cobol samples/cobol_mf	.sqb	.cbl
JAVA	samples/java	.sqlj	.java
REXX	samples/rexx	.cmd	.cmd

표 2. 프로그램 그룹에 의한 샘플 파일 확장자

샘플 그룹	디렉토리	파일 확장
ADO, RDO, MTS	samples\ADO\VB(Visual Basic) samples\ADO\VC(Visual C++) samples\RDO samples\MTS	.bas .frm .vbp(Visual Basic) .cpp .dsp .dsw(Visual C++)
CLP	samples/clp	.db2
OLE	samples\ole\msvb(Visual Basic) samples\ole\msvc(Visual C++)	.bas .vbp(Visual Basic) .cpp(Visual C++)
OLE DB	samples\oledb	.db2
SQL 프로시듀어	samples/sqlproc	.db2 .c .sqc(클라이언트 응용프로그램)
User Exit	samples/c	.cad(OS/2) .cadsm(UNIX & Windows) .cdisk(UNIX & Windows) .ctape(UNIX) .cxbsa(UNIX & Windows)

주:

디렉토리 분리문자

UNIX에서는 /입니다. OS/2 및 Windows 플랫폼에서는 \입니다. 테

이블에서 디렉토리가 Windows 및/또는 OS/2에서만 사용 가능하지 않은 않으면 UNIX 분리문자가 사용됩니다.

파일 확장자

단일 확장자만이 존재하는 테이블에서 샘플에 대해 제공됩니다.

Embedded SQL 프로그램

프로그램이 수행될 때 Embedded SQL문이 해석되는 REXX Embedded SQL 프로그램을 제외하고, 사전 처리 컴파일을 요구합니다.

IBM COBOL 샘플

`cobol` 서브디렉토리에서 AIX, OS/2 및 Windows 32비트 운영 체제에 대해서만 제공됩니다.

Micro Focus Cobol 샘플

`cobol_mf` 서브디렉토리에서 AIX, HP-UX, OS/2, Solaris 운영 체제 및 Windows 32비트 운영 체제에 대해서만 제공됩니다.

Java 샘플

JDBC(Java Database Connectivity) 애플릿, 응용프로그램 및 저장 프로시듀어, SQLJ(Embedded SQL for Java) 애플릿, 응용프로그램, 저장 프로시듀어 및 Java UDF 등입니다. Java 샘플은 지원되는 모든 DB2 플랫폼에서 사용 가능합니다.

REXX 샘플

AIX, OS/2 및 Windows NT 운영 체제에 대해서만 제공됩니다.

CLP 샘플

SQL문을 실행하는 명령행 처리기 스크립트입니다.

OLE 샘플

Windows 32비트 운영 체제에 대해서만 제공되는 Microsoft Visual Basic 및 Microsoft Visual C++의 OLE입니다.

ADO, RDO 및 MTS 샘플

Windows 32비트 운영 체제에 대해서만 제공되는 Microsoft Visual

Basic 및 Microsoft Visual C++의 ActiveX Data Objects 샘플 및 Microsoft Visual Basic의 Remote Data Objects 및 Microsoft Transaction Server 샘플입니다.

User Exit 샘플

데이터베이스 로그 파일을 아카이브하고 검색하는 데 사용되는 Log Management User Exit 프로그램입니다. C 언어 프로그램으로 `.c` 확장자를 사용하여 파일의 이름을 바꾸고 컴파일해야 합니다.

DB2가 설치된 디렉토리의 `samples` 서브디렉토리에서 샘플 프로그램을 찾을 수 있습니다. 각각의 지원되는 언어에 대한 서브디렉토리가 있습니다. 다음 예에서는 각각의 지원되는 플랫폼에서 C 또는 C++로 작성된 샘플을 찾는 방법을 보여줍니다.

- **UNIX 플랫폼에서**

  데이터베이스 인스턴스 디렉토리 아래 `sqllib/samples/c`에서 Embedded SQL 및 DB2 API 프로그램에 대한 C 소스 코드를 찾을 수 있습니다. DB2 CLI 프로그램에 대한 C 소스 코드는 `sqllib/samples/cli`에 있습니다. 샘플 테이블에서 프로그램에 대한 자세한 내용은 DB2 인스턴스 아래 해당하는 `samples` 서브디렉토리에 있는 `README` 파일을 참조하십시오. `README` 파일에는 이 책에 나열되지 않은 추가 샘플이 들어 있습니다.

- **OS/2 및 Windows 32비트 운영 체제에서**

  DB2 설치 디렉토리 아래 `%DB2PATH%\samples\c`에서 Embedded SQL 및 DB2 API 프로그램에 대한 C 소스 코드를 찾을 수 있습니다. DB2 CLI 프로그램에 대한 C 소스 코드는 `%DB2PATH%\samples\cli`에 있습니다. `%DB2PATH%` 변수가 DB2가 설치된 위치를 결정합니다. DB2가 설치된 드라이브에 따라 `%DB2PATH%`는 *drive*`:\sqllib`를 가리킵니다. 샘플 테이블에서 샘플 프로그램에 대한 자세한 내용은 해당하는 `%DB2PATH%\samples` 서브디렉토리의 `README` 파일을 참조하십시오. `README` 파일에는 이 책에 나열되지 않은 추가 샘플이 들어 있습니다.

샘플 프로그램 디렉토리는 대부분의 플랫폼에서 일반적으로 읽기 전용입니다. 샘플 프로그램을 변경하거나 빌드하기 전에 작업 디렉토리로 복사하십시오.

DB2 API 비 Embedded SQL 샘플

표 3. 비 Embedded SQL이 있는 DB2 API 샘플 프로그램

샘플 프로그램	포함된 API
backrest	• sqlbftcq - Fetch Tablespace Container Query • sqlbstsc - Set Tablespace Containers • sqlfudb - Update Database Configuration • sqlubkp - Backup Database • sqluroll - Rollforward Database • sqlurst - Restore Database
checkerr	• sqlaintp - Get Error Message • sqlogstt - Get SQLSTATE Message
cli_info	• sqleqryi - Query Client Information • sqleseti - Set Client Information
client	• sqleqryc - Query Client • sqlesetc - Set Client
d_dbconf	• sqleatin - Attach • sqledtin - Detach • sqlfddb - Get Database Configuration Defaults
d_dbmcon	• sqleatin - Attach • sqledtin - Detach • sqlfdsys - Get Database Manager Configuration Defaults
db_udcs	• sqleatin - Attach • sqlecrea - Create Database • sqledrpd - Drop Database
db2mon	• sqleatin - Attach • sqlmon - Get/Update Monitor Switches • sqlmonss - Get Snapshot • sqlmonsz - Estimate Size Required for sqlmonss() Output Buffer • sqlmrset - Reset Monitor

표 3. 비 Embedded SQL이 있는 DB2 API 샘플 프로그램 (계속)

샘플 프로그램	포함된 **API**
dbcat	• sqlecadb - Catalog Database • sqledcls - Close Database Directory Scan • sqledgne - Get Next Database Directory Entry • sqledosd - Open Database Directory Scan • sqleuncd - Uncatalog Database
dbcmt	• sqledcgd - Change Database Comment • sqledcls - Close Database Directory Scan • sqledgne - Get Next Database Directory Entry • sqledosd - Open Database Directory Scan • sqleisig - Install Signal Handler
dbconf	• sqleatin - Attach • sqlecrea - Create Database • sqledrpd - Drop Database • sqlfrdb - Reset Database Configuration • sqlfudb - Update Database Configuration • sqlfxdb - Get Database Configuration
dbinst	• sqleatcp - Attach and Change Password • sqleatin - Attach • sqledtin - Detach • sqlegins - Get Instance
dbmconf	• sqleatin - Attach • sqledtin - Detach • sqlfrsys - Reset Database Manager Configuration • sqlfusys - Update Database Manager Configuration • sqlfxsys - Get Database Manager Configuration
dbsnap	• sqleatin - Attach • sqlmonss - Get Snapshot
dbstart	• sqlepstart - Start Database Manager

표 3. 비 Embedded SQL이 있는 DB2 API 샘플 프로그램 (계속)

샘플 프로그램	포함된 **API**
dbstop	• sqlefrce - Force Application • sqlepstp - Stop Database Manager
dcscat	• sqlegdad - Catalog DCS Database • sqlegdcl - Close DCS Directory Scan • sqlegdel - Uncatalog DCS Database • sqlegdge - Get DCS Directory Entry for Database • sqlegdgt - Get DCS Directory Entries • sqlegdsc - Open DCS Directory Scan
dmscont	• sqleatin - Attach • sqlecrea - Create Database • sqledrpd - Drop Database
ebcdicdb	• sqleatin - Attach • sqlecrea - Create Database • sqledrpd - Drop Database
migrate	• sqlemgdb - Migrate Database
monreset	• sqleatin - Attach • sqlmrset - Reset Monitor
monsz	• sqleatin - Attach • sqlmonss - Get Snapshot • sqlmonsz - Estimate Size Required for sqlmonss() Output Buffer
nodecat	• sqlectnd - Catalog Node • sqlencls - Close Node Directory Scan • sqlengne - Get Next Node Directory Entry • sqlenops - Open Node Directory Scan • sqleuncn - Uncatalog Node
restart	• sqlerstd - Restart Database
setact	• sqlesact - Set Accounting String
setrundg	• sqlesdeg - Set Runtime Degree

표 3. 비 Embedded SQL이 있는 DB2 API 샘플 프로그램 (계속)

샘플 프로그램	포함된 API
sws	• sqleatin - Attach • sqlmon - Get/Update Monitor Switches
utilapi	• sqlaintp - Get Error Message • sqlogstt - Get SQLSTATE Message

DB2 API Embedded SQL 샘플

표 4. DB2 API Embedded SQL 샘플 프로그램

샘플 프로그램	포함된 API
asynrlog	• sqlurlog - Asynchronous Read Log
autocfg	• db2AutoConfig -- Autoconfig • db2AutoConfigMemory -- Autoconfig Free Memory • sqlfudb -- Update Database Configuration • sqlfusys -- Update Database Manager Configuration • sqlesetc -- Set Client • sqlaintp -- SQLCA Message
dbauth	• sqluadau - Get Authorizations
dbstat	• sqlureot - Reorganize Table • sqlustat - Runstats
expsamp	• sqluexpr - Export • sqluimpr - Import
impexp	• sqluexpr - Export • sqluimpr - Import
loadqry	• db2LoadQuery - Load Query
makeapi	• sqlabndx - Bind • sqlaprep - Precompile Program • sqlepstp - Stop Database Manager • sqlepstr - Start Database Manager

표 4. DB2 API Embedded SQL 샘플 프로그램 (계속)

샘플 프로그램	포함된 **API**
rebind	• sqlarbnd - Rebind
rechist	• sqlubkp - Backup Database • sqluhcls - Close Recovery History File Scan • sqluhgne - Get Next Recovery History File Entry • sqluhops - Open Recovery History File Scan • sqluhprn - Prune Recovery History File • sqluhupd - Update Recovery History File
tabscont	• sqlbctcq - Close Tablespace Container Query • sqlbftcq - Fetch Tablespace Container Query • sqlbotcq - Open Tablespace Container Query • sqlbtcq - Tablespace Container Query • sqlefmem - Free Memory
tabspace	• sqlbctsq - Close Tablespace Query • sqlbftpq - Fetch Tablespace Query • sqlbgtss - Get Tablespace Statistics • sqlbmtsq - Tablespace Query • sqlbotsq - Open Tablespace Query • sqlbstpq - Single Tablespace Query • sqlefmem - Free Memory
tload	• sqluexpr - Export • sqluload - Load • sqluvqdp - Quiesce Tablespaces for Table

표 4. DB2 API Embedded SQL 샘플 프로그램 (계속)

샘플 프로그램	포함된 **API**
tspace	• sqlbctcq - Close Tablespace Container Query • sqlbctsq - Close Tablespace Query • sqlbftcq - Fetch Tablespace Container Query • sqlbftpq - Fetch Tablespace Query • sqlbgtss - Get Tablespace Statistics • sqlbmtsq - Tablespace Query • sqlbotcq - Open Tablespace Container Query • sqlbotsq - Open Tablespace Query • sqlbstpq - Single Tablespace Query • sqlbstsc - Set Tablespace Containers • sqlbtcq - Tablespace Container Query • sqlefmem - Free Memory
utilemb	• sqlaintp - Get Error Message • sqlogstt - Get SQLSTATE Message

DB2 API가 없는 Embedded SQL 샘플

표 5. DB2 API가 없는 Embedded SQL 샘플 프로그램

샘플 프로그램 이름	프로그램 설명
adhoc	SQL 명령을 대화식으로 처리하기 위한 동적 SQL 및 SQLDA 구조를 보여줍니다. SQL 명령은 사용자가 입력하며, SQL 명령에 대응하는 출력이 리턴됩니다.
advsql	CASE, CAST 및 스칼라 전체 선택과 같은 고급 SQL 표현식의 사용을 보여줍니다.
blobfile	샘플 데이터베이스에서 2진 대형 오브젝트(BLOB) 값을 읽고 파일에 배치하여 2진 대형 오브젝트(BLOB)의 조작을 보여줍니다. 이 파일의 내용을 외부 표시기를 사용하여 표시할 수 있습니다.
columns	동적 SQL을 사용하여 처리되는 커서의 사용을 보여줍니다. 이 프로그램은 원하는 스키마 이름 아래에 SYSCAT.COLUMNS의 결과 세트를 나열합니다.
cursor	정적 SQL을 사용하는 커서 사용을 보여줍니다.
delet	데이터베이스에서 항목을 삭제하기 위한 정적 SQL을 보여줍니다.
dynamic	동적 SQL을 사용하는 커서의 사용을 보여줍니다.
joinsql	고급 SQL 조인 표현식의 사용을 보여줍니다.

표 5. DB2 API가 없는 Embedded SQL 샘플 프로그램 (계속)

샘플 프로그램 이름	프로그램 설명
largevol	파티션된 환경에서 병렬 조회 처리와, 결과 세트 병합을 자동화하기 위한 NFS 파일 시스템 사용을 보여줍니다. AIX에서만 사용 가능합니다.
lobeval	LOB 위치 지정자의 사용을 보여주며 실제 LOB 데이터의 평가를 지연시킵니다.
lobfile	LOB 파일 핸들의 사용을 보여줍니다.
lobloc	LOB 위치 지정자의 사용을 보여줍니다.
lobval	LOB의 사용을 보여줍니다.
openftch	정적 SQL을 사용하는 행의 페치, 갱신 및 삭제를 보여줍니다.
recursql	고급 SQL 순환 조회의 사용을 보여줍니다.
sampudf	테이블 항목을 수정하기 위해 구현된 사용자 정의 유형(UDT) 및 사용자 정의 함수(UDF)를 보여줍니다. 이 프로그램에 선언되는 모든 UDF는 전래 UDF입니다.
spclient	spserver 공유 라이브러리에 있는 저장 프로시듀어를 호출하는 클라이언트 응용프로그램.
spcreate.db2	spserver 프로그램에서 작성한 저장 프로시듀어를 등록하기 위한 CREATE PROCEDURE 문을 포함하는 CLP 스크립트.
spdrop.db2	spserver 프로그램에서 작성한 저장 프로시듀어를 등록 해제하는 데 필요한 DROP PROCEDURE문을 포함하는 CLP 스크립트.
spserver	저장 프로시듀어를 보여주는 서버 프로그램. 클라이언트 프로그램은 spclient입니다.
static	정보를 검색하기 위한 정적 SQL을 보여줍니다.
tabsql	고급 SQL 테이블 표현식의 사용을 보여줍니다.
tbdefine	테이블 작성 및 삭제를 예를 보여줍니다.
thdsrver	스레드 작성 및 관리를 위한 POSIX 스레드 API의 사용을 보여줍니다. 프로그램이 문맥 풀을 유지보수합니다. generate_work 함수는 main에서 실행되며, 작업자 스레드가 실행하는 동적 SQL문을 작성합니다. 문맥이 사용 가능하게 되면 스레드가 작성되고 지정된 작업을 수행하도록 디스패치됩니다. 생성된 작업은 sample 데이터베이스의 STAFF 또는 EMPLOYEE 테이블에서 항목을 삭제하기 위한 명령문으로 구성됩니다. 이 프로그램은 UNIX 플랫폼에서만 사용 가능합니다.
trigsql	고급 SQL 트리거 및 제한조건의 사용을 보여줍니다.
udfcli	sample 데이터베이스의 테이블에 액세스하기 위해 udfsrv 프로그램에서 작성하고, 서버에 저장된 사용자 정의 함수(UDF) 호출을 보여줍니다.
updat	데이터베이스를 갱신하기 위한 정적 SQL을 보여줍니다.
varinp	매개변수 표시문자를 사용하는 Embedded 동적 SQL문 호출에 대한 변수 입력을 보여줍니다.

사용자 정의 함수(UDF) 샘플

표 6. 사용자 정의 함수(UDF) 샘플 프로그램

샘플 프로그램 이름	프로그램 설명
DB2Udf.java	정수 나눗셈, 문자 대형 오브젝트(CLOB) 조작 및 Java 인스턴스 변수의 사용을 포함한 몇 가지 타스크를 보여주는 Java UDF.
udfsrv.c	샘플 데이터베이스 테이블에 액세스하기 위해 사용자 정의 함수(UDF) ScalarUDF를 사용하여 라이브러리를 작성합니다.
UDFsrv.java	Java 사용자 정의 함수(UDF) 사용을 보여줍니다.

DB2 CLI 샘플

표 7. DB2 Universal Database에 있는 샘플 CLI 프로그램

샘플 프로그램 이름	프로그램 설명
공통 유틸리티 파일	
utilcli.c	CLI 샘플에서 사용된 유틸리티 함수.
utilapi.c	DB2 API를 호출하는 유틸리티 함수.
응용프로그램 레벨 - DB2 및 CLI의 응용프로그램 레벨을 다루는 샘플.	
apinfo.c	응용프로그램 레벨 정보를 가져오고 설정하는 방법.
aphndls.c	핸들을 할당하고 해제하는 방법.
apsqlca.c	SQLCA 데이터로 작업하는 방법.
설치 이미지 레벨 - DB2 및 CLI의 설치 이미지 레벨을 다루는 샘플.	
ilinfo.c	CLI 드라이버 버전과 같은 설치 레벨 정보를 가져오고 설정하는 방법.
인스턴스 레벨 - DB2 및 CLI의 인스턴스 레벨을 다루는 샘플.	
ininfo.c	인스턴스 레벨 정보를 가져오고 설정하는 방법.
데이터베이스 레벨 - DB2에서 데이터베이스 오브젝트를 다루는 샘플.	
dbconn.c	데이터베이스에 연결하고 연결을 해제하는 방법.
dbinfo.c	데이터베이스 레벨에서 정보를 가져오고 설정하는 방법.
dbmconn.c	DB2 API를 사용하여 두 번째 데이터베이스를 작성하고 삭제하는 다중 데이터베이스에 연결하고 연결을 해제하는 방법.
dbmuse.c	DB2 API를 사용하여 두 번째 데이터베이스를 작성하고 삭제하는 다중 데이터베이스에서 트랜잭션을 수행하는 방법.
dbnative.c	데이터 소스 특정 형식에 대한 ODBC escape 절을 포함하는 명령문을 해석하는 방법.
dbuse.c	데이터베이스 오브젝트로 작업하는 방법.
dbusemx.sqc	Embedded SQL과 함께 단일 데이터베이스를 사용하는 방법.
테이블 레벨 - DB2에서 테이블 오브젝트를 다루는 샘플.	

표 7. DB2 Universal Database에 있는 샘플 CLI 프로그램 (계속)

샘플 프로그램 이름	프로그램 설명
tbconstr.c	테이블 제한조건으로 작업하는 방법.
tbconstr.c	테이블을 작성, 변경 및 삭제하는 방법.
tbinfo.c	테이블 레벨의 정보를 가져오고 설정하는 방법.
tbmod.c	테이블의 정보를 수정하는 방법.
tbread.c	테이블의 정보를 읽는 방법.
데이터 유형 레벨 - 데이터 유형을 다루는 샘플.	
dtinfo.c	데이터 유형에 대한 정보를 가져오는 방법.
dtlob.c	LOB 데이터를 읽고 쓰는 방법.
dtudt.c	사용자 정의 구별 유형을 작성, 사용 및 삭제하는 방법.
UDF 레벨 - 사용자 정의 함수(UDF)를 보여주는 샘플.	
udfcli.c	udfsrv.c의 사용자 정의 함수(UDF)를 호출하는 클라이언트 응용프로그램.
udfsrv.c	udfcli.c 샘플에서 호출하는 사용자 정의 함수(UDF) ScalarUDF.
저장 프로시듀어 레벨 - CLI의 저장 프로시듀어를 보여주는 샘플.	
spcreate.db2	CREATE PROCEDURE문을 발행하기 위한 CLP 스크립트.
spdrop.db2	카탈로그에서 저장 프로시듀어를 삭제하기 위한 CLP 스크립트.
spclient.c	spserver.c에 선언된 서버 함수를 호출하는 데 사용되는 클라이언트 프로그램.
spserver.c	서버에서 빌드되어 수행된 저장 프로시듀어 함수.
spcall.c	저장 프로시듀어를 호출하기 위한 프로그램.

주: samples/cli 디렉토리에 있는 그 밖의 파일은 다음과 같습니다.

- README - 모든 예 파일을 나열합니다.
- makefile - 모든 파일에 대한 Makefile
- 응용프로그램 및 저장 프로시듀어에 대한 빌드 파일

Java 샘플

표 8. JDBC(Java Database Connectivity) 샘플 프로그램

샘플 프로그램 이름	프로그램 설명
DB2Appl.java	사용자 특권 호출을 사용하여 샘플 데이터베이스를 조회하는 JDBC 응용프로그램.
DB2Applt.java	JDBC 애플릿 드라이버를 사용하여 데이터베이스를 조회하는 JDBC 애플릿. DB2Applt.html에 지정된 사용자 이름, 암호, 서버 및 포트 번호 매개변수를 사용합니다.
DB2Applt.html	애플릿 샘플 프로그램 DB2Applt를 포함하는 HTML 파일. 서버 및 사용자 정보를 사용하여 사용자 정의해야 합니다.
DB2UdCli.java	Java 사용자 정의 함수(UDF) DB2Udf를 호출하는 Java 클라이언트 응용프로그램.

표 8. JDBC(Java Database Connectivity) 샘플 프로그램 (계속)

샘플 프로그램 이름	프로그램 설명
Dynamic.java	동적 SQL을 사용하는 커서를 보여줍니다.
MRSPcli.java	서버 프로그램 MRSPsrv를 호출하는 클라이언트 프로그램입니다. 프로그램은 Java 저장 프로시듀어에서 리턴하는 복수의 결과 세트를 보여줍니다.
MRSPsrv.java	클라이언트 프로그램 MRSPcli가 호출하는 서버 프로그램입니다. 프로그램은 Java 저장 프로시듀어에서 리턴하는 복수의 결과 세트를 보여줍니다.
Outcli.java	SQLJ 저장 프로시듀어 Outsrv를 호출하는 Java 클라이언트 응용프로그램.
PluginEx.java	새로운 메뉴 항목 및 도구 모음 버튼을 DB2 웹 제어 센터에 추가하는 Java 프로그램.
Spclient.java	Spserver 저장 프로시듀어 클래스의 PARAMETER STYLE JAVA 저장 프로시듀어를 호출하는 JDBC 클라이언트 응용프로그램.
Spcreate.db2	Spserver 클래스에 포함된 메소드를 저장 프로시듀어로 등록하기 위한 CREATE PROCEDURE문을 포함하는 CLP 스크립트.
Spdrop.db2	Spserver 클래스에 포함된 저장 프로시듀어를 등록 해제하는 데 필요한 DROP PROCEDURE문을 포함하는 CLP 스크립트.
Spserver.java	PARAMETER STYLE JAVA 저장 프로시듀어를 보여주는 JDBC 프로그램. 클라이언트 프로그램은 Spclient.java입니다.
UDFcli.java	Java 사용자 정의 함수(UDF) 라이브러리 UDFsrv의 함수를 호출하는 JDBC 클라이언트 응용프로그램.
UseThrds.java	SQL문을 비동기로 수행하는 스레드를 사용하는 방법을 보여줍니다(CLI 샘플 async.c의 JDBC 버전).
V5SpCli.java	DB2GENERAL 저장 프로시듀어 V5Stp.java를 호출하는 Java 클라이언트 응용프로그램.
V5Stp.java	서버에서 EMPLOYEE 테이블을 갱신하고 새로운 급여 및 지불 정보를 클라이언트로 리턴하는 DB2GENERAL 저장 프로시듀어를 보여줍니다. 클라이언트 프로그램은 V5SpCli.java입니다.
Varinp.java	매개변수 표시문자를 사용하는 Embedded 동적 SQL문 호출에 대한 변수 입력을 보여줍니다.

표 9. SQL(Embedded SQL for Java) 샘플 프로그램

샘플 프로그램 이름	프로그램 설명
App.sqlj	정적 SQL을 사용하여 샘플 데이터베이스의 EMPLOYEE 테이블에서 데이터를 검색 및 갱신합니다.
Applt.sqlj	JDBC 애플릿 드라이버를 사용하여 데이터베이스를 조회하는 애플릿. Applt.html에 지정된 사용자 이름, 암호, 서버 및 포트 번호 매개변수를 사용합니다.
Applt.html	애플릿 샘플 프로그램 Applt를 포함하는 HTML 파일. 서버 및 사용자 정보를 사용하여 사용자 정의해야 합니다.
Cursor.sqlj	정적 SQL을 사용하는 반복자를 보여줍니다.
OpF_Curs.sqlj	Openftch 프로그램에 대한 클래스 파일.

표 9. SQL(Embedded SQL for Java) 샘플 프로그램 (계속)

샘플 프로그램 이름	프로그램 설명
Openftch.sqlj	정적 SQL을 사용하는 행의 페치, 갱신 및 삭제를 보여줍니다.
Outsrv.sqlj	SQLDA 구조를 사용하는 저장 프로시듀어를 보여줍니다. SQLDA에 sample 데이터베이스의 STAFF 테이블에 있는 직원의 급여 중간 값을 입력합니다. 데이터베이스 처리(급여 중간 값 찾기) 후 저장 프로시듀어는 입력된 SQLDA와 SQLCA 상태를 JDBC 클라이언트 응용프로그램 Outcli로 리턴합니다.
Stclient.sqlj	SQLJ 저장 프로시듀어 프로그램 Stserver가 작성한 PARAMETER STYLE JAVA 저장 프로시듀어를 호출하는 SQLJ 클라이언트 응용프로그램.
Stcreate.db2	Stserver에 포함된 메소드를 저장 프로시듀어로 등록하기 위한 CREATE PROCEDURE문을 포함하는 CLP 스크립트.
Stdrop.db2	Stserver 클래스에 포함된 저장 프로시듀어를 등록 해제하는 데 필요한 DROP PROCEDURE 문을 포함하는 CLP 스크립트.
Stserver.sqlj	PARAMETER STYLE JAVA 저장 프로시듀어를 보여주는 SQLJ 프로그램. 클라이언트 프로그램은 Stclient.sqlj입니다.
Static.sqlj	정적 SQL을 사용하여 정보를 검색합니다.
Stp.sqlj	서버의 EMPLOYEE 테이블을 갱신하고 새로운 급여 및 지불 정보를 JDBC 클라이언트 프로그램 StpCli로 리턴하는 저장 프로시듀어.
UDFclie.sqlj	Java 사용자 정의 함수(UDF) 라이브러리 UDFsrv에서 함수를 호출하는 클라이언트 응용프로그램.
Updat.sqlj	정적 SQL을 사용하여 데이터베이스를 갱신합니다.

SQL 프로시듀어 샘플

표 10. SQL 프로시듀어 샘플 프로그램

샘플 프로그램 이름	프로그램 설명
basecase.db2	UPDATE_SALARY 프로시듀어는 ″sample″ 데이터베이스의 ″staff″ 테이블에 있는 ″empno″ IN 매개변수에 의해 식별되는 직원의 급여를 인상합니다. 이 프로시듀어는 ″rating″ IN 매개변수를 사용하는 CASE문에 따라 이러한 인상을 결정합니다.
basecase.sqc	UPDATE_SALARY 프로시듀어를 호출합니다.
baseif.db2	UPDATE_SALARY_IF 프로시듀어는 ″sample″ 데이터베이스 ″staff″ 테이블에 있는 ″empno″ IN 매개변수에 의해 식별되는 직원의 급여를 인상합니다. 이 프로시듀어는 ″rating″ IN 매개변수를 사용하는 IF문에 따라 이러한 인상을 결정합니다.
baseif.sqc	UPDATE_SALARY_IF 프로시듀어를 호출합니다.
dynamic.db2	CREATE_DEPT_TABLE 프로시듀어는 동적 DDL을 사용하여 새 테이블을 작성합니다. 테이블의 이름은 프로시듀어에 대한 IN 매개변수의 값에 기반합니다.
dynamic.sqc	CREATE_DEPT_TABLE 프로시듀어를 호출합니다.

표 10. SQL 프로시듀어 샘플 프로그램 (계속)

샘플 프로그램 이름	프로그램 설명
iterate.db2	ITERATOR 프로시듀어는 FETCH loop를 사용하여 ″department″ 테이블에서 데이터를 검색합니다. ″deptno″ 컬럼의 값이 ‘D11’이 아니면 수정된 데이터가 ″department″ 테이블에 삽입됩니다. ″deptno″ 컬럼의 값이 ‘D11’이면 ITERATE문은 제어의 흐름을 LOOP문의 시작 부분으로 전달합니다.
iterate.sqc	ITERATOR 프로시듀어를 호출합니다.
leave.db2	LEAVE_LOOP 프로시듀어는 ″not_found″ 조건 핸들러가 LEAVE문을 호출하기 전에 LOOP문에서 수행된 FETCH 연산의 수를 계산합니다. LEAVE문은 제어의 흐름이 loop를 나가서 저장 프로시듀어를 완료하게 만듭니다.
leave.sqc	LEAVE_LOOP 프로시듀어를 호출합니다.
loop.db2	LOOP_UNTIL_SPACE 프로시듀어는 커서가 컬럼 ″midnight″에 대해 공백(‘’) 값이 있는 행을 검색할 때까지 LOOP문에서 수행된 FETCH 연산의 수를 계산합니다. 이 LOOP문은 제어의 흐름이 loop를 나가서 저장 프로시듀어를 완료하게 만듭니다.
loop.sqc	LOOP_UNTIL_SPACE 프로시듀어를 호출합니다.
nestcase.db2	BUMP_SALARY 프로시듀어는 중첩 CASE문을 사용하여 ″sample″ 데이터베이스의 ″staff″ 테이블에서 dept IN 매개변수에 의해 식별되는 부서의 직원 급여를 인상합니다.
nestcase.sqc	BUMP_SALARY 프로시듀어를 호출합니다.
nestif.db2	BUMP_SALARY_IF 프로시듀어는 중첩 IF문을 사용하여 ″sample″ 데이터베이스의 ″staff″ 테이블에서 dept IN 매개변수에 의해 식별되는 부서의 직원 급여를 인상합니다.
nestif.sqc	BUMP_SALARY_IF 프로시듀어를 호출합니다.
repeat.db2	REPEAT_STMT 프로시듀어는 커서가 더 이상의 행을 검색할 수 없게 될 때까지 반복 문에서 수행된 FETCH 연산의 수를 계산합니다. 이 조건 핸들러는 제어의 흐름이 반복 loop를 나가서 저장 프로시듀어를 완료하게 만듭니다.
repeat.sqc	REPEAT_STMT 프로시듀어를 호출합니다.
rsultset.c	MEDIAN_RESULT_SET 프로시듀어를 호출하고 급여 중간 값을 표시한 다음 SQL 프로시듀어에서 생성한 결과 세트를 표시합니다. 이 클라이언트는 결과 세트를 승인할 수 있는 CLI API를 사용하여 작성되었습니다.
rsultset.db2	MEDIAN_RESULT_SET 프로시듀어는 ″sample″ 데이터베이스 ″staff″ 테이블에서 ″dept″ IN 매개변수에 의해 식별되는 부서의 직원 급여 중간 값을 구합니다. 중간 값은 급여 OUT 매개변수에 지정되고 ″rsultset″ 클라이언트로 리턴됩니다. 그러면 프로시듀어는 WITH RETURN 커서를 열어 중간 값보다 더 많은 급여를 받은 직원의 결과 세트를 리턴합니다. 프로시듀어는 클라이언트로 결과 세트를 리턴합니다.
spserver.db2	이 CLP 스크립트의 SQL 프로시듀어는 기본 오류 처리, 중첩된 저장 프로시듀어 호출 및 클라이언트 응용프로그램 또는 호출 응용프로그램으로 결과 세트 리턴 등의 과정을 보여줍니다. CLI 샘플 디렉토리의 ″spcall″ 응용프로그램을 사용하여 이 프로시듀어를 호출할 수 있습니다. 또한 C 및 CPP 샘플 디렉토리의 ″spclient″ 응용프로그램을 사용하여 결과 세트를 리턴하지 않는 프로시듀어를 호출할 수 있습니다.

표 10. SQL 프로시듀어 샘플 프로그램 (계속)

샘플 프로그램 이름	프로그램 설명
whiles.db2	DEPT_MEDIAN 프로시듀어는 ″sample″ 데이터베이스의 ″staff″ 테이블에서 ″dept″ IN 매개변수에 의해 식별되는 부서의 직원 급여 중간 값을 구합니다. 중간 값은 급여 OUT 매개변수에 지정되고 ″whiles″ 클라이언트로 리턴됩니다. 그러면 whiles 클라이언트는 급여 중간 값을 인쇄합니다.
whiles.sqc	DEPT_MEDIAN 프로시듀어를 호출합니다.

ADO, RDO 및 MTS 샘플

표 11. ADO, RDO 및 MTS 샘플 프로그램

샘플 프로그램 이름	프로그램 설명
Bank.vbp	고객 계정에 관한 트랜잭션을 수행하는 기능을 사용하여 은행 지점에 대한 데이터를 작성하고 유지보수하는 RDO 프로그램. 응용프로그램이 데이터를 저장하는 데 필요한 테이블을 작성하는 DDL을 포함하기 때문에 프로그램은 사용자가 지정한 모든 데이터베이스를 사용할 수 있습니다.
Blob.vbp	이 ADO 프로그램은 BLOB 데이터 검색을 보여줍니다. sample 데이터베이스의 emp_photo 테이블에서 그림을 검색하고 표시합니다. 또한 프로그램은 emp_photo 테이블의 이미지를 지역 파일의 이미지로 대체할 수 있습니다.
BLOBAccess.dsw	이 샘플은 Microsoft Visual C++를 사용하는 ADO/Blob 액세스 강조표시를 보여줍니다. Visual Basic 샘플 Blob.vbp와 유사합니다. BLOB 샘플에는 다음과 같은 두 가지 주요 기능이 있습니다. 1. Sample 데이터베이스에서 BLOB를 읽어서 화면에 표시합니다. 2. 파일에서 BLOB를 읽어서 데이터베이스에 삽입합니다(가져오기).
Connect.vbp	이 ADO 프로그램은 연결 오브젝트를 작성하고 sample 데이터베이스에 대한 연결을 설정합니다. 완료되면 프로그램이 연결 해제되고 나갑니다.
Commit.vbp	이 응용프로그램은 ADO의 자동 확약/수동 확약 기능의 사용을 보여줍니다. 프로그램이 sample 데이터베이스의 EMPLOYEE 테이블에서 직원 번호와 이름을 조회합니다. 사용자는 자동 확약 또는 수동 확약 모드로 데이터베이스에 연결하는 옵션을 갖습니다. 자동 확약 모드에서는 사용자가 레코드에 관하여 수행하는 모든 변경사항이 데이터베이스에서 자동으로 갱신됩니다. 수동 확약 모드에서는 변경을 수행하기 전에 사용자가 트랜잭션을 시작해야 합니다. 트랜잭션 시작 이후 수행된 모든 변경사항을 구간 복원을 수행하여 실행 취소할 수 있습니다. 트랜잭션을 확약하여 변경사항을 영구적으로 저장할 수 있습니다. 프로그램을 나가면 변경사항이 자동으로 구간 복원됩니다.

표 11. ADO, RDO 및 MTS 샘플 프로그램 (계속)

샘플 프로그램 이름	프로그램 설명
db2com.vbp	이 Visual Basic 프로젝트는 Microsoft Transaction Server를 사용한 데이터베이스 갱신을 보여줍니다. 클라이언트 프로그램 db2mts.vbp가 사용하는 서버 DLL을 작성하며, 다음과 같은 네 가지 클래스 모듈을 갖습니다. • UpdateNumberColumn.cls • UpdateRow.cls • UpdateStringColumn.cls • VerifyUpdate.cls 이 프로그램의 경우 임시 테이블 DB2MTS가 sample 데이터베이스에 작성됩니다.
db2mts.vbp	Microsoft Transaction Server를 사용하여 db2com.vbp에서 작성된 서버 DLL을 호출하는 클라이언트 프로그램에 대한 Visual Basic 프로젝트입니다.
Select-Update.vbp	이 ADO 프로그램은 Connect.vbp와 동일한 함수를 수행하지만, 또한 GUI 인터페이스를 제공합니다. 이 인터페이스를 사용하여 사용자는 sample 데이터베이스의 ORG 테이블에 저장된 데이터를 보기, 갱신 및 삭제할 수 있습니다.
Sample.vbp	이 Visual Basic 프로젝트는 ADO를 통해 Keyset 커서를 사용하여 sample 데이터베이스에 있는 모든 데이터에 대한 그래픽 사용자 인터페이스를 제공합니다.
VarCHAR.dsp	ADO를 사용하여 VarChar 데이터에 텍스트 필드로 액세스하는 Visual C++ 프로그램. 사용자가 sample 데이터베이스의 ORG 테이블에 있는 데이터를 보고 갱신할 수 있도록 그래픽 사용자 인터페이스를 제공합니다.

OLE(Object Linking and Embedding) 샘플

표 12. OLE(Object Linking and Embedding) 샘플 프로그램

샘플 프로그램 이름	프로그램 설명
sales	Microsoft Excel 영업 스프레드시트에 관한 지불 조회를 보여줍니다(Visual Basic에서 구현됨).
names	Lotus Notes 주소록을 조회합니다(Visual Basic으로 구현됨).
inbox	Microsoft Exchange 받은 편지함 전자우편 메시지를 OLE/Messaging을 통해 조회합니다 (Visual Basic에서 구현됨).
invoice	Microsoft Word 송장 문서를 전자우편 첨부물로 전송하는 OLE 자동화 사용자 정의 함수 (UDF) (Visual Basic에서 구현됨).
bcounter	인스턴스 변수를 사용하여 스크래치 패드를 보여주는 OLE 자동화 사용자 정의 함수(UDF) (Visual Basic에서 구현됨).
ccounter	카운터 OLE 자동화 사용자 정의 함수(UDF) (Visual C++로 구현됨).

표 12. OLE(Object Linking and Embedding) 샘플 프로그램 (계속)

샘플 프로그램 이름	프로그램 설명
salarysrv	sample 데이터베이스의 STAFF 테이블의 급여 중간 값을 계산하는 OLE 자동화 저장 프로시듀어(Visual Basic에서 구현됨).
salarycltvc	Visual Basic 저장 프로시듀어 salarysrv를 호출하는 Visual C++ DB2 CLI 샘플.
salarycltvb	Visual Basic 저장 프로시듀어 salarysrv를 호출하는 Visual Basic DB2 CLI 샘플.
salsvado	새로 작성된 테이블 STAFF2에서 급여 중간 값을 계산하여 출력 매개변수를 보여주고, 테이블에서 급여를 검색하여 결과 세트를 보여주는 32비트 Visual Basic 및 ADO에서 구현되는 OLE 자동화 저장 프로시듀어.
salclado	Visual Basic 저장 프로시듀어 salsvado를 호출하는 Visual Basic 클라이언트.
testcli	저장 프로시듀어 tstsrv를 호출하는 OLE 자동화 Embedded SQL 클라이언트 응용프로그램(Visual C++에서 구현됨).
tstsrv	클라이언트와 저장 프로시듀어 간의 다양한 유형 전달을 보여주는 OLE 자동화 저장 프로시듀어(Visual C++에서 구현됨).

표 13. OLE DB 테이블 함수

샘플 프로그램 이름	프로그램 설명
jet.db2	Microsoft.Jet.OLEDB.3.51 Provider
mapi.db2	MAPI용 INTERSOLV Connect OLE DB
msdaora.db2	Oracle용 Microsoft OLE DB Provider
msdasql.db2	ODBC 드라이버용 Microsoft OLE DB Provider
msidxs.db2	Microsoft OLE DB Index Server Provider
notes.db2	Notes용 INTERSOLV Connect OLE DB
sampprov.db2	Microsoft OLE DB Sample Provider
sqloledb.db2	SQL 서버용 Microsoft OLE DB Provider

명령행 처리기(CLP) 샘플

표 14. 명령행 처리기(CLP) 샘플 프로그램

샘플 파일 이름	파일 설명
const.db2	CHECK CONSTRAINT 절을 사용하여 테이블을 작성합니다.
cte.db2	공통 테이블 표현식을 보여줍니다. 이 고급 SQL문을 보여주는 동등한 샘플 프로그램은 tabsql입니다.
flt.db2	순환 조회를 보여줍니다. 이 고급 SQL문을 보여주는 동등한 샘플 프로그램은 recursql입니다.
join.db2	테이블의 외부 조인을 보여줍니다. 이 고급 SQL문을 보여주는 동등한 샘플 프로그램은 joinsql입니다.

표 14. 명령행 처리기(CLP) 샘플 프로그램 (계속)

샘플 파일 이름	파일 설명
stock.db2	트리거의 사용을 보여줍니다. 이 고급 SQL문을 보여주는 동등한 샘플 프로그램은 trigsql입니다.
testdata.db2	RAND() 및 TRANSLATE()와 같은 DB2 내장 함수를 사용하여 임의로 생성된 테스트 데이터로 테이블을 입력합니다.
thaisort.db2	이 스크립트는 특히 타이 사용자를 위한 것입니다. 타이 정렬은 정확한 정렬 순서로 데이터를 보기 위한 사후 정렬뿐 아니라, 선행 자음 및 모음의 사전 정렬/스와핑을 요구하는 음운 순서에 의한 정렬입니다. 파일은 UDF 함수 사전 정렬 및 사후 정렬을 작성하고 테이블을 작성하여 타이 정렬을 구현한 다음 테이블에 대하여 함수를 호출하여 테이블 데이터를 정렬합니다. 이 프로그램을 수행하려면 먼저 C 소스 파일 udf.c에서 사용자 정의 함수(UDF) 프로그램 udf를 빌드해야 합니다.

Log Management User Exit 샘플

표 15. Log Management User Exit 샘플 프로그램

샘플 파일 이름	파일 설명
db2uext2.cadsm	ADSM(ADSTAR DSM) API를 이용하여 데이터베이스 로그 파일을 아카이브하고 검색하는 샘플 User Exit입니다. 샘플은 시간소인 및 수신된 매개변수를 포함한 호출의 감사 추적(각 옵션에 대하여 별도 파일에 저장됨)을 제공합니다. 또한 시간소인 및 문제점 판별을 위한 오류 분리 문자열을 포함하여 오류 발생시 호출에 대한 오류 추적을 제공합니다. 이들 옵션을 사용하지 않을 수도 있습니다. 파일을 db2uext2.c로 이름을 바꾸고 C 프로그램으로 컴파일해야 합니다. UNIX 및 Windows 32비트 운영 체제에서 사용 가능합니다. OS/2 버전은 db2uexit.cad입니다. **주:** 레벨 3.1.6 이상에서 ADSM API 클라이언트를 사용하는 AIX의 응용프로그램은 단일 스레드 응용프로그램이라고 해도 xlc 또는 xlC가 아니라 xlc_r 또는 xlC_r 컴파일러 호출을 사용하여 빌드해야 합니다. 그러면 라이브러리가 스레드 안전 상태가 됩니다. 스레드 안전 상태가 아닌 라이브러리를 사용하여 컴파일한 응용프로그램이 있으면 fixtest IC21925E를 적용하거나 응용프로그램 제공업체에 문의할 수 있습니다. fixtest는 익명의 ftp 서버 index.storsys.ibm.com에서 구할 수 있습니다. 이는 ADSM API 레벨을 3.1.3으로 되돌립니다.
db2uexit.cad	db2uext2.cadsm의 OS/2 버전입니다. 파일을 db2uexit.c로 이름을 바꾸고 C 프로그램으로 컴파일해야 합니다.
db2uext2.cdisk	제공하는 특정 플랫폼의 시스템 복사 명령을 이용하는 샘플 User Exit입니다. 프로그램은 데이터베이스 로그 파일을 아카이브하고 검색하며 시간소인 및 수신된 매개변수를 포함한 호출의 감사 추적(각 옵션에 대하여 별도 파일에 저장됨)을 제공합니다. 또한 시간소인 및 문제점 판별을 위한 오류 분리 문자열을 포함하여 오류 발생시 호출에 대한 오류 추적을 제공합니다. 이들 옵션을 사용하지 않을 수도 있습니다. 파일을 db2uext2.c로 이름을 바꾸고 C 프로그램으로 컴파일해야 합니다. UNIX 및 Windows 32비트 운영 체제에서 사용 가능합니다.

표 15. Log Management User Exit 샘플 프로그램 (계속)

샘플 파일 이름	파일 설명
db2uext2.ctape	제공하는 특정 UNIX 플랫폼의 시스템 테이프 명령을 이용하는 샘플 User Exit입니다. 프로그램은 데이터베이스 로그 파일을 아카이브하고 검색합니다. 시스템 테이프 명령의 모든 제한조건은 이러한 User Exit에 대한 제한조건입니다. 샘플은 시간소인 및 수신된 매개변수를 포함한 호출의 감사 추적(각 옵션에 대하여 별도 파일에 저장됨)을 제공합니다. 또한 시간소인 및 문제점 판별을 위한 오류 분리 문자열을 포함하여 오류 발생시 호출에 대한 오류 추적을 제공합니다. 이들 옵션을 사용하지 않을 수도 있습니다. 파일을 db2uext2.c로 이름을 바꾸고 C 프로그램으로 컴파일해야 합니다. UNIX 플랫폼에서만 사용 가능합니다.
db2uext2.cxbsa	XBSA API를 이용하여 데이터베이스 로그 파일을 아카이브하고 검색하는 샘플 User Exit입니다. 이 샘플은 시간소인 및 수신된 매개변수를 포함한 호출의 감사 추적(각 옵션에 대하여 별도 파일에 저장됨)을 제공합니다. 또한 시간소인 및 문제점 판별을 위한 오류 격리 문자열을 포함하여 오류 발생시 호출에 대한 오류 추적을 제공합니다. 이들 옵션을 사용하지 않을 수도 있습니다. 파일을 db2uext2.c로 이름을 바꾸고 C 프로그램으로 컴파일해야 합니다. UNIX 및 Windows 플랫폼에서만 사용 가능합니다.

제2장 설정

DB2 응용프로그램을 빌드하고 수행하기 위한 적합한 환경을 만들려면 다음을 제대로 설정해야 합니다.

1. 컴파일러 또는 인터프리터
2. DB2(데이터베이스 관리자, DB2 AD 클라이언트 및 클라이언트 연결)
3. 운영 체제 환경
4. DB2 샘플 데이터베이스(선택적)

컴파일러 또는 인터프리터 환경 점검

DB2 프로그램을 개발하려면 8 페이지의 『플랫폼별로 지원되는 소프트웨어』에 나열된 사용자 운영 체제에 지원되는 프로그래밍 언어 중 하나에 컴파일러 또는 인터프리터를 사용해야 합니다. DB2가 아닌 응용프로그램을 먼저 빌드하여 기존 컴파일러 또는 인터프리터 환경이 제대로 설정되었는지 확인하는 것이 좋습니다. 그런 다음 문제점이 발생하는 경우, 컴파일러 또는 인터프리터와 함께 제공되는 문서를 참조하십시오.

DB2 환경 설정

DB2 환경을 설정하려면 다음이 설치되어 작동해야 합니다.

- 환경에 대한 데이터베이스 인스턴스를 사용하는 서버의 데이터베이스 관리자. 데이터베이스 인스턴스에 대한 자세한 내용은 457 페이지의 『부록A. 데이터베이스 관리자 인스턴스 정보』를 참조하십시오.
- 응용프로그램을 개발하려는 클라이언트 또는 서버 워크스테이션의 DB2 AD Client.
- 클라이언트 워크스테이션에서 개발 중인 경우, 원격 서버에 대한 연결.

데이터베이스 관리자 구성 파일 갱신

이 파일에는 응용프로그램 개발을 위한 중요한 설정이 들어 있습니다. 다음을 입력하여 이러한 설정을 변경할 수 있습니다.

```
db2 update dbm cfg using <keyword> <value>
```

또한 다음을 입력하여 설정을 볼 수 있습니다.

```
db2 get dbm cfg
```

이 명령의 사용법에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

- 저장 프로시듀어의 경우, 키워드 KEEPDARI는 기본값 yes를 가집니다. 이 키워드는 저장 프로시듀어 프로세스를 활성 상태로 유지합니다. 저장 프로시듀어를 개발하는 경우, 동일한 저장 프로시듀어 라이브러리를 여러 번 로드하여 테스트할 수 있습니다. 이 기본 설정은 라이브러리 재로드를 방해할 수 있습니다. 저장 프로시듀어를 개발하는 중에 이 키워드의 값을 no로 변경한 다음 저장 프로시듀어 최종 버전을 로드할 준비가 될 때 다시 yes로 변경하는 것이 좋습니다.
- Java의 경우 JDK11_PATH 키워드를 갱신하십시오. 자세한 내용은 77 페이지의 『환경 설정』을 참조하십시오.

설치 및 설정에 대한 자세한 내용은 운영 체제의 *빠른 시작* 책을 참조하십시오.

위와 같은 것들이 설치되어 작동하면 다음 절의 단계를 수행하여 운영 체제 환경을 설정할 수 있습니다.

- 41 페이지의 『OS/2 환경 설정』
- 42 페이지의 『UNIX 환경 설정』
- 44 페이지의 『Windows 32비트 운영 체제 환경 설정』

운영 체제 환경을 설정한 후 이 책의 샘플 프로그램이 사용하는 샘플 데이터베이스를 작성할 수 있습니다. 데이터베이스를 작성하려면, 48 페이지의 『샘플 데이터베이스 작성, 카탈로그화 및 바인딩』을 참조하십시오.

OS/2 환경 설정

대부분의 OS/2 컴파일러는 환경 변수를 사용하여 다양한 옵션을 제어합니다. CONFIG.SYS 파일에서 이들 변수를 설정하거나, 변수를 설정하기 위한 명령 파일을 작성할 수 있습니다.

CONFIG.SYS

CONFIG.SYS 파일에서 환경 변수를 설정하는 경우 이점은 일단 입력해 두면 컴퓨터를 시작(부팅)할 때마다 그대로 설정된다는 것입니다.

명령 파일

명령 파일에서 환경 변수를 설정하는 경우 이점은 보다 짧은 경로와 몇 가지 컴파일러를 사용하는 융통성을 가질 수 있다는 점입니다. 단점은 프로그래밍 세션을 시작할 때마다 명령 파일을 실행해야 한다는 점입니다.

명령 파일을 수행하여 환경 변수를 설정하는 경우, 환경 변수를 설정한 동일한 창에서 응용프로그램을 빌드해야 합니다. 다른 창에서 응용프로그램을 빌드하는 경우, 첫 번째 창에서 설정한 것과 동일한 옵션이 사용되지 않습니다.

DB2 AD Client를 설치하면 다음 명령문이 CONFIG.SYS 파일에 삽입됩니다.

```
set LIB=%DB2PATH%\lib;%LIB%
```

이 책에서 명령 파일은 이러한 명령문이 존재하는 것으로 가정합니다. DB2 AD Client를 설치한 후 CONFIG.SYS 파일을 편집하는 경우, 이 명령문이 제거되지 않았는지 확인하십시오.

또한, 다음과 같은 환경 변수가 DB2에 의해 자동으로 갱신됩니다.

- PATH(%DB2PATH%\bin 디렉토리를 포함하도록)
- LIBPATH(%DB2PATH%\dll 디렉토리를 포함하도록)

DB2가 갱신하는 Java 환경 변수에 대한 자세한 내용은 86 페이지의 『OS/2』를 참조하십시오.

추가로, 아래에 표시된 프로그래밍 언어 중 하나를 사용 중인 경우, CONFIG.SYS 파일에 적절한 명령문이 있어야 합니다.

C/C++

```
set INCLUDE=%DB2PATH%\include;%INCLUDE%
```

FORTAN

```
set FINCLUDE=%DB2PATH%\include;%FINCLUDE%
```

IBM COBOL

```
set SYSLIB=%SYSLIB%;%DB2PATH%\include\cobol_a
```

Micro Focus COBOL

```
set COBCPY=%DB2PATH%\include\cobol_mf;%COBCPY%
```

OS/2에서는 DB2PATH와 DB2INSTPROF를 제외하고 CONFIG.SYS에 정의되어야 하는 DB2 환경 변수가 없습니다. 모든 DB2 변수는 전역 레벨, 인스턴스 레벨 또는 인스턴스 노드 레벨(Parallel Edition)에서 DB2 Instance Profile Registry에 정의됩니다. 변수를 설정, 수정 및 나열하려면 db2set.exe 명령을 사용하십시오.

주: DB2INSTANCE는 DB2INSTDEF 레지스트리 변수를 설정하는 경우 필요하지 않습니다. 이것은 DB2INSTANCE가 설정되지 않은 경우 사용되는 기본 인스턴스 이름을 정의합니다.

UNIX 환경 설정

데이터베이스 관리자를 설치할 때 작성된 데이터베이스 인스턴스에 액세스할 수 있도록 환경 변수를 설정해야 합니다. *UNIX용 DB2 빠른 시작*에서는 환경 변수 설정에 관한 일반적인 정보를 제공합니다. 이 절은 데이터베이스 인스턴스에 액세스하기 위한 환경 변수 설정에 관한 특정 지침을 제공합니다.

각각의 데이터베이스 관리자 인스턴스에는 두 개의 파일 db2profile 및 db2cshrc가 있으며, 이들 파일에는 해당 인스턴스에 대한 환경 변수를 설정하는 스크립트가 들어 있습니다. 사용 중인 쉘에 따라 다음을 입력하여 스크립트를 수행하십시오.

bash 또는 Korn 쉘:

```
. $HOME/sqllib/db2profile
```

C 쉘의 경우:

```
source $HOME/sqllib/db2cshrc
```

여기서 $HOME은 인스턴스 소유자의 홈 디렉토리입니다.

편의상, 로그온할 때 자동으로 수행될 수 있도록 .profile 또는 .login 파일에 이 명령을 포함시킬 수도 있습니다.

인스턴스 작성 중에 사용자가 자신에 맞는 고유 인스턴스 환경 설정을 배치할 수 있도록 공백 파일 sqllib/userprofile 및 sqllib/usercshrc가 작성됩니다. 이 파일은 DB2 FixPak이나 향후 버전 설치에서 인스턴스 갱신(db2iupdt) 중에 수정되지 않습니다. db2profile 또는 db2cshrc 스크립트에서 새 환경 설정을 원하지 않으면 db2profile 또는 db2cshrc 스크립트 끝에서 호출된 것처럼 해당되는 "user" 스크립트를 사용하여 이를 대체할 수 있습니다. 인스턴스 이주(db2imigr) 동안, 사용자 스크립트는 계속해서 사용자의 환경 수정사항이 사용되도록 복사됩니다. 이 사용자 스크립트는 DB2 버전 7을 사용하여 시작할 경우에만 사용할 수 있습니다.

UNIX 플랫폼에 따라 DB2 인스턴스 작성시, 다음 환경 변수가 자동으로 갱신됩니다.

AIX:

- PATH(sqllib/bin을 포함한 몇 개의 DB2 디렉토리를 포함하도록)
- LIBPATH(sqllib/lib 디렉토리를 포함하도록)

HP-UX:

- PATH(sqllib/bin을 포함한 몇 개의 DB2 디렉토리를 포함하도록)
- SHLIB_PATH(sqllib/lib 디렉토리를 포함하도록)

Linux, PTX 및 Solaris:

- PATH(sqllib/bin을 포함한 몇 개의 DB2 디렉토리를 포함하도록)
- LD_LIBRARY_PATH(sqllib/lib 디렉토리를 포함하도록)

Silicon Graphics IRIX:

- PATH(sqllib/bin을 포함한 몇 개의 DB2 디렉토리를 포함하도록)

- LD_LIBRARY_PATH(sqllib/lib 디렉토리를 포함하도록:o32 오브젝트 유형 응용프로그램에 필요)
- LD_LIBRARYN32_PATH(sqllib/lib32 디렉토리를 포함하도록:n32 오브젝트 유형 응용프로그램에 필요)

DB2가 갱신하는 Java 환경 변수에 대한 자세한 내용은 77 페이지의 『환경 설정』을 참조하십시오.

Windows 32비트 운영 체제 환경 설정

Windows NT 및 Windows 2000에 DB2 AD Client를 설치할 때 설치 프로그램은 환경 변수 INCLUDE, LIB, PATH, DB2PATH 및 DB2INSTANCE를 사용하여 구성 레지스트리를 갱신합니다. 기본 인스턴스는 DB2입니다.

DB2가 갱신하는 Java 환경 변수에 대한 자세한 내용은 94 페이지의 『Windows 32비트 운영 체제』를 참조하십시오.

Windows 95, Windows 98 또는 Windows Millennium Edition에 DB2 AD Client를 설치할 때 설치 프로그램은 autoexec.bat 파일을 갱신합니다.

머신 또는 현재 로그온한 사용자에 대한 값을 설정하도록 이들 환경 변수를 대체할 수 있습니다. 이들 값을 대체하려면 다음 중 하나를 사용하십시오.

- Windows NT 제어판
- Windows 2000 제어판
- Windows 95, Windows 98 또는 Windows Millennium Edition 명령 창
- The Windows 95, Windows 98 또는 Windows Millennium Edition autoexec.bat 파일

주:

1. 이들 환경 변수를 변경할 때 주의하십시오. DB2PATH 환경 변수를 변경하지 **마십시오.**
2. 명령에서 %DB2PATH% 변수를 사용할 때는 set LIB="%DB2PATH%\lib";%LIB% 에서처럼 전체 경로를 따옴표로 묶으십시오. DB2 버전 7에서 이 변수에 대한 기본 설치 값은 \Program Files\sqllib입니다. 이 값에는 공백이 포함되며 따옴표를 사용하지 않으면 오류가 발생할 수 있습니다.

Windows 32비트 운영 체제에 있는 대부분의 프로그램을 수행하기 위해 이들 환경 변수를 갱신할 수 있습니다. 추가로, DB2 응용프로그램을 수행하려면 다음과 같은 특정 단계를 수행해야 합니다.

- C 또는 C++ 프로그램을 빌드 중이면 INCLUDE 환경 변수에 첫 번째 디렉토리로 %DB2PATH%\INCLUDE가 들어 있는지 확인해야 합니다.

  예를 들어 Microsoft Visual C++ 컴파일러 환경 설정 파일 Vc\bin\vcvars32.bat는 다음 명령을 포함합니다.

  ```
  set INCLUDE=%MSVCDir%\INCLUDE;%MSVCDir%\...\ATL\INCLUDE;%INCLUDE%
  ```

  이 파일을 DB2에서 사용하려면 먼저 %DB2PATH%\INCLUDE 경로를 설정하는 %INCLUDE%를 다음과 같이 목록 끝에서 처음으로 이동하십시오.

  ```
  set INCLUDE=%INCLUDE%;%MSVCDir%\INCLUDE;%MSVCDir%\...\ATL\INCLUDE;
  ```

- Micro Focus COBOL 프로그램을 빌드 중이면 %DB2PATH%\INCLUDE\cobol_mf를 가리키도록 COBCPY 환경 변수를 설정하십시오.
- IBM COBOL 프로그램을 빌드 중이면 %DB2PATH%\INCLUDE\cobol_a를 가리키도록 SYSLIB 환경 변수를 설정하십시오.
- 다음과 같은 명령을 사용하여 LIB 환경 변수가 %DB2PATH%\lib를 가리키는지 확인하십시오.

  ```
  set LIB="%DB2PATH%\lib";%LIB%
  ```

- DB2COMM 환경 변수가 원격 데이터베이스의 서버에 설정되었는지 확인하십시오.
- 보안 서비스가 서버 인증의 경우 SERVER에서, 클라이언트 인증 레벨에 따라 CLIENT에서 시작되었는지 확인하십시오. 보안 서비스를 시작하려면 NET START DB2NTSECSERVER 명령을 사용하십시오.

주:

1. 모든 DB2 환경 변수를 사용자 환경에 정의하거나 레지스트리 변수로 설정할 수 있습니다. 레지스트리 변수에 대한 자세한 내용은 *관리 안내서*를 참조하십시오. db2set 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.
2. DB2INSTANCE는 사용자 환경 레벨에서만 정의해야 합니다.
   DB2INSTANCE가 설정되지 않은 경우에 사용할 기본 인스턴스 이름을 정의하는 DB2INSTDEF 레지스트리 변수를 사용하는 경우 필요하지 않습니다.

3. Windows NT 또는 Windows 2000 환경에서 데이터베이스 관리자는 Windows NT 서비스 또는 Windows 2000 서비스로 구현되므로 서비스가 정상적으로 시작된 경우 다른 문제점이 발생할 수는 있지만 오류나 경고를 리턴하지 않습니다. 이것은 db2start 또는 NET START 명령을 수행할 때, 통신 서브시스템 시작에 실패한 경우 경고를 리턴하지 않음을 의미합니다. 따라서 사용자가 항상 Windows NT 또는 Windows 2000 이벤트 로그 또는 DB2DIAG.LOG에서 이들 명령을 수행하는 동안 발생할 수 있는 오류를 조사해야 합니다.

서버에서 통신 사용

이 절에서는 DB2 Universal Database 서버에 연결하는 방법에 대해 설명합니다.

sample 데이터베이스 설치, 카탈로그화 및 바인딩을 시작하기 전에 서버가 작동하는지, 카탈로그화할 프로토콜을 지원하도록 구성되었는지 확인해야 합니다. 서버에서 다음을 수행하십시오.

1. db2comm 환경 변수가 설정되었는지 확인하십시오. 예를 들어, TCP/IP가 사용되면 다음을 입력하십시오.

   ```
   db2set DB2COMM=tcpip
   ```

   또한 TCP/IP 지원용 프로토콜이 구성되어 있는지 확인하십시오.

   서비스 파일에 TCP/IP 설정 추가에 관한 지침은 플랫폼의 *빠른 시작* 책을 참조하십시오.

2. 다음 명령을 입력하여 데이터베이스 인스턴스를 시작하십시오.

   ```
   db2start
   ```

클라이언트로부터 샘플 데이터베이스와 유틸리티를 바인딩하는 작업이 수행되어야 합니다. 자세한 내용은 51 페이지의 『바인딩』을 참조하십시오.

Windows NT 및 Windows 2000

Windows NT 또는 Windows 2000용 DB2 생산 시스템에서 데이터베이스 인스턴스를 서비스로서 시작해야 합니다. 그 단계는 다음과 같습니다.

- 통신 프로토콜을 사용 중인 경우, db2comm 환경 변수가 Windows NT 또는 Windows 2000 제어판의 시스템 환경 변수 섹션에 설정되었는지 확인하십시오.

- 보안 서비스를 시작하십시오. 보안 서비스는 자동으로 수행되거나(아래 주 참조), 다음 명령을 사용하여 수동으로 시작할 수 있습니다.

  ```
  NET START DB2NTSECSERVER
  ```

- 다음을 입력하여 인스턴스를 시작하십시오.

  ```
  db2start
  ```

보안 서비스 자동 시작. 일반적으로 보안 서비스를 자동으로 시작하도록 설정하는 유일한 경우는 워크스테이션이 클라이언트 인증에 대해 구성된 서버에 연결된 DB2 클라이언트로 작용하는 경우입니다. 보안 서비스가 자동으로 시작되게 하려면 다음을 수행하십시오.

Windows NT

1. "시작" 버튼을 누르십시오.
2. "설정"을 누르십시오.
3. "제어판"을 누르십시오.
4. 제어판에서 "서비스"를 누르십시오.
5. 서비스 창에서 "DB2 보안 서버"를 강조표시하십시오.
6. 설정이 "시작됨" 및 "자동"으로 나열되지 않으면 "시작"을 누르십시오.
7. "자동"을 누르십시오.
8. "확인"을 누르십시오.
9. 설정이 효력을 발생하려면 머신을 재부트하십시오.

Windows 2000

1. "시작" 버튼을 누르십시오.
2. "설정"을 누르십시오.
3. "제어판"을 누르십시오.
4. "관리 도구"를 누르십시오.
5. "서비스"를 누르십시오.
6. 서비스 창에서 "DB2 보안 서버"를 강조표시하십시오.
7. 설정이 "시작됨" 및 "자동"으로 나열되지 않으면 상위 메뉴에서 "조치"를 누르십시오.

8. "등록 정보"를 누르십시오.
9. "일반" 탭에 있는지 확인하십시오.
10. '시작 유형' 드롭 다운 메뉴에서 "자동"을 선택하십시오.
11. "확인"을 누르십시오.
12. 설정이 효력을 발생하려면 머신을 재부트하십시오.

샘플 데이터베이스 작성, 카탈로그화 및 바인딩

DB2와 함께 제공된 샘플 프로그램을 사용하려면 서버 워크스테이션에 sample 데이터베이스를 작성해야 합니다. sample 데이터베이스의 내용에 대해서는 *SQL 참조서*를 참조하십시오.

원격 클라이언트를 사용하여 서버의 sample 데이터베이스에 액세스할 경우, 클라이언트 워크스테이션에서 sample 데이터베이스를 카탈로그화해야 합니다.

또한 원격 클라이언트를 사용하여 다른 DB2 버전을 수행 중이거나 다른 운영 체제에서 수행되는 서버의 sample 데이터베이스에 액세스할 경우, DB2 CLI를 포함하여 데이터베이스 유틸리티를 sample 데이터베이스에 바인드해야 합니다.

작성

sample 데이터베이스를 작성하려면 SYSADM 권한이 있어야 합니다. SYSADM 권한에 대한 자세한 내용은 운영 체제의 *빠른 시작* 책을 참조하십시오.

데이터베이스를 작성하려면 서버에서 다음을 수행하십시오.

1. db2sampl(sample 데이터베이스를 작성하는 프로그램)의 위치가 경로에 있는지 확인하십시오. db2profile 또는 db2cshrc 파일이 경로에 db2sampl을 삽입하므로, 변경하지 않으면 경로에 남아 있습니다.
   - UNIX 서버에서 db2sampl은 다음 디렉토리에 있습니다.

     ```
     $HOME/sqllib/bin
     ```

     여기서 $HOME은 DB2 인스턴스 소유자의 홈 디렉토리입니다.
   - OS/2 및 Windows에서 db2sampl은 다음 디렉토리에 있습니다.

     ```
     %DB2PATH%\bin
     ```

여기서 %DB2PATH%는 DB2가 설치된 위치입니다.

2. DB2INSTANCE 환경 변수가 sample 데이터베이스를 작성하려는 인스턴스의 이름으로 설정되었는지 확인하십시오. 설정되지 않은 경우, 다음과 같은 명령을 사용하여 설정할 수 있습니다.

   - UNIX 플랫폼에서

     다음을 입력하여 bash 또는 Korn 쉘에 대해 이를 수행할 수 있습니다.

     ```
     DB2INSTANCE=instance_name
     export DB2INSTANCE
     ```

     C 쉘의 경우 다음을 입력하십시오.

     ```
     setenv DB2INSTANCE instance_name
     ```

   - OS/2 및 Windows에서 다음을 입력하십시오.

     ```
     set DB2INSTANCE=instance_name
     ```

   여기서 *instance_name*은 데이터베이스 인스턴스의 이름입니다.

3. db2sampl 다음에 샘플 데이터베이스를 작성하려는 위치를 입력하여 sample 데이터베이스를 작성하십시오. UNIX 플랫폼에서는 *path*이며 다음을 입력합니다.

   ```
   db2sampl path
   ```

   OS/2 및 Windows에서 *drive*이며 다음을 입력합니다.

   ```
   db2sampl drive
   ```

   경로나 드라이브를 지정하지 않는 경우, 설치 프로그램은 데이터베이스 관리자 구성 파일의 DFTDBPATH 매개변수가 지정한 기본 경로 또는 드라이브에 샘플 테이블을 설치합니다. 구성 파일에 대한 자세한 내용은 *관리 안내서*를 참조하십시오.

   데이터베이스에 대한 인증 유형은 데이터베이스가 작성된 인스턴스와 동일합니다. 데이터베이스 인스턴스를 작성할 때 인증 지정에 대한 자세한 내용은 *빠른 시작* 책을 참조하십시오.

호스트 또는 AS/400 서버에서 작성

OS/390용 DB2 또는 AS/400 서버와 같은 호스트 서버에 대해 샘플 프로그램을 수행하려는 경우, *SQL 참조서*에 설명된 샘플 테이블을 포함하는 데이터베이스를 작성해야 합니다. 샘플 프로그램 expsamp를 참조할 수도 있습니다. 이 샘플 프로그램은 STAFF 및 ORG 테이블을 사용하여 DB2 Connect 데이터베이스로/로부터 테이블 및 테이블 데이터를 가져오고 내보내는 데 API를 사용하는 방법을 보여줍니다.

주: 워크스테이션에서의 DB2와 호스트 시스템에서의 DB2 사이에는 일부 SQL 구문과 DB2 명령에 있어서 차이점이 있습니다. OS/390용 DB2나 AS/400용 DB2에서 데이터베이스에 액세스 중이면 프로그램이 이러한 데이터베이스 시스템에서 지원되는 SQL문과 사전 처리 컴파일/바인드 옵션을 사용하는지 확인하십시오.

데이터베이스를 작성하려면 다음을 수행하십시오.

1. db2sampl을 사용하여 DB2 서버 인스턴스에서 sample 데이터베이스를 작성하십시오.
2. sample 데이터베이스에 연결하십시오.
3. 샘플 테이블을 파일로 내보내십시오.
4. DB2 Connect 데이터베이스에 연결하십시오.
5. 샘플 테이블을 작성하십시오.
6. 샘플 테이블을 가져오십시오.

파일 내보내기 및 가져오기에 대한 자세한 내용은 *데이터 이동 유틸리티 안내 및 참조서*를 참조하십시오. 데이터베이스에 대한 연결 및 테이블 작성에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.

카탈로그화

원격 클라이언트에서 서버에 있는 sample 데이터베이스에 액세스하려면 클라이언트 워크스테이션에 있는 sample 데이터베이스를 카탈로그화해야 합니다.

작성시 카탈로그화되기 때문에 서버 워크스테이션에 있는 sample 데이터베이스를 카탈로그화할 필요가 없습니다.

카탈로그화는 클라이언트 응용프로그램이 액세스하려는 데이터베이스의 이름을 사용하여 클라이언트 워크스테이션에 있는 데이터베이스 디렉토리를 갱신합니다. 클라이언트 요청을 처리할 때 데이터베이스 관리자는 카탈로그화된 이름을 사용하여 데이터베이스를 찾고 데이터베이스에 연결합니다.

*빠른 시작*에서는 데이터베이스 카탈로그화에 관한 일반적인 정보를 제공합니다. 이 절에서는 sample 데이터베이스를 카탈로그화하기 위한 특정 지침을 제공합니다.

원격 클라이언트 워크스테이션에서 샘플 데이터베이스를 카탈로그화하려면 다음과 같이 입력하십시오.

```
db2 catalog database sample as sample at node nodename
```

여기서 *nodename*은 서버 노드의 이름입니다.

*빠른 시작*에서는 통신 프로토콜 설정의 일부로 노드를 카탈로그화하는 방법에 대해 설명합니다. 데이터베이스에 연결하기 전에 원격 노드를 또한 카탈로그화해야 합니다.

바인딩

다른 버전의 DB2를 수행하거나 다른 운영 체제에서 수행되는 원격 클라이언트에서 서버에 있는 sample 데이터베이스에 액세스하려면 DB2 CLI를 포함하여 데이터베이스 유틸리티를 sample 데이터베이스에 바인드해야 합니다.

바인딩은 응용프로그램을 실행할 때 데이터베이스에 액세스하기 위해 데이터베이스 관리자에 필요한 패키지를 작성합니다. 사전 처리 컴파일 중에 작성된 바인드 파일에 대한 BIND 명령을 지정하여 명시적으로 바인딩을 수행할 수 있습니다.

*Command Reference*에서는 데이터베이스 유틸리티 바인딩에 대한 일반 정보를 제공합니다. 이 절에서는 데이터베이스 유틸리티를 sample 데이터베이스에 바인드하기 위한 특정 지침을 제공합니다.

사용 중인 클라이언트 워크스테이션의 플랫폼에 따라 서로 다른 데이터베이스 유틸리티를 바인드합니다.

OS/2 클라이언트 워크스테이션에서

1. 다음을 입력하여 sample 데이터베이스에 연결하십시오.

```
db2 connect to sample user userid using password
```

여기서 *userid* 및 *password*는 sample 데이터베이스가 있는 인스턴스의 사용자 ID와 암호입니다.

이 명령을 사용하면 DB2가 유틸리티를 자동으로 데이터베이스에 바인드하므로, 사용자가 명시적으로 바인드할 필요가 없습니다.

2. 명령행 처리기를 나간 다음 바인드 메시지 파일 bind.msg를 점검하여 정상적으로 바인드되었는지 검증하십시오.

UNIX 클라이언트 워크스테이션에서

1. 다음을 입력하여 sample 데이터베이스에 연결하십시오.

```
db2 connect to sample user userid using password
```

여기서 *userid* 및 *password*는 sample 데이터베이스가 있는 인스턴스의 사용자 ID와 암호입니다.

2. 다음을 입력하여 유틸리티를 데이터베이스에 바인드하십시오.

```
db2 bind BNDPATH/@db2ubind.lst blocking all sqlerror continue \
messages bind.msg grant public
db2 bind BNDPATH/@db2cli.lst blocking all sqlerror continue \
messages cli.msg grant public
```

여기서 *BNDPATH*는 $HOME/sqllib/bnd와 같은 바인드 파일이 있는 경로이며, $HOME은 DB2 인스턴스 소유자의 홈 디렉토리입니다.

3. 바인드 메시지 파일 bind.msg 및 cli.msg를 점검하여 정상적으로 바인드되었는지 확인하십시오.

Windows 32비트 운영 체제를 수행하는 클라이언트 워크스테이션에서

1. 시작 메뉴에서 프로그램을 선택하십시오.
2. 프로그램 메뉴에서 IBM DB2를 선택하십시오.
3. IBM DB2 메뉴에서 DB2 명령 창을 선택하십시오.

   명령 창이 표시됩니다.
4. 다음을 입력하여 sample 데이터베이스에 연결하십시오.

```
db2 connect to sample user userid using password
```

여기서 *userid* 및 *password*는 sample 데이터베이스가 있는 인스턴스의 사용자 ID와 암호입니다.

5. 다음을 입력하여 유틸리티를 데이터베이스에 바인드하십시오.

```
db2 bind "%DB2PATH%\bnd\@db2ubind.lst" blocking all
sqlerror continue messages bind.msg
```

여기서 %DB2PATH%는 DB2가 설치된 경로입니다.

6. 명령 창을 나간 다음 바인드 메시지 파일 bind.msg를 점검하여 정상적으로 바인드되었는지 검증하십시오.

모든 플랫폼

DRDA 호환 응용프로그램 서버에서 sample 데이터베이스를 작성한 경우, db2ubind.lst 대신 다음 .lst 파일 중 하나를 지정하십시오.

ddcsmvs.lst
OS/390용 DB2

ddcsvm.lst
VM용 DB2

ddcsvse.lst
VSE용 DB2

ddcs400.lst
AS/400용 DB2

*빠른 시작*에서는 데이터베이스 유틸리티 바인딩에 대한 일반 정보를 제공합니다.

다음 이동 위치

일단 환경을 설정하면 DB2 응용프로그램을 빌드할 준비가 되었습니다. 다음 장에서는 샘플 프로그램에 대해 설명하고 이를 컴파일, 링크 및 수행하는 방법을 보여줍니다. 먼저 55 페이지의 『제3장 DB2 응용프로그램 빌드에 대한 일반 정보』를 읽은 다음 빌드 중인 응용프로그램에 대해 수행할 특정 장을 읽으십시오.

Java를 사용하여 프로그래밍할 경우, 75 페이지의 『제4장 Java 애플릿 및 응용프로그램 빌드』를 참조하십시오.

SQL 프로시듀어를 사용하여 프로그래밍할 경우, 117 페이지의 『제5장 SQL 프로시듀어 빌드』를 참조하십시오.

DB2 APIs, DB2 CLI 및 Embedded SQL을 사용하여 프로그래밍할 경우, 플랫폼에 대한 '응용프로그램 빌드' 장을 참조하십시오.

자세한 내용은 다음의 책을 참조하십시오.

- Embedded SQL, JDBC 및 SQLJ를 사용하는 응용프로그램과 사용자 정의 함수(UDF)의 경우 *응용프로그램 개발 안내서*.
- DB2 CLI 또는 ODBC를 사용하는 응용프로그램의 경우 *CLI Guide and Reference*.
- DB2 API 응용프로그램의 경우 *Administrative API Reference*.

제3장 DB2 응용프로그램 빌드에 대한 일반 정보

이 장의 내용은 둘 이상의 운영 체제에 적용됩니다. 항목 대부분이 거의 모든 DB2 지원 플랫폼에 적용됩니다.

최신 DB2 응용프로그램 개발 갱신사항에 대한 자세한 내용은 다음 웹 사이트를 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

DB2 프로그램 빌드 및 수행에 대한 일반 사항

1. 응용프로그램 환경:
   - OS/2: CONFIG.SYS 파일이 아닌 명령 파일에 의해 환경 변수를 설정하는 경우, 동일한 창에 응용프로그램을 빌드해야 합니다.
   - UNIX: 환경 변수가 설정되는 쉘에서 DB2 응용프로그램을 빌드하고 수행해야 합니다. 사용 중인 쉘에 따라 `db2profile` 또는 `db2cshrc`를 수행하여 이 작업을 할 수 있습니다.
   - Windows 32비트 운영 체제: DB2 명령 창에서 응용프로그램을 빌드해야 합니다. 자세한 내용은 39 페이지의 『제2장 설정』을 참조하십시오.
2. Embedded SQL을 포함하는 DB2 프로그램을 빌드하거나, DB2 프로그램을 수행하려면 서버에 있는 데이터베이스 관리자를 시작해야 합니다. 데이터베이스 관리자를 시작하려면 SYSADM(시스템 관리) 권한이 필요합니다. SYSADM 권한에 대한 자세한 내용은 *빠른 시작*을 참조하십시오.

   서버에서 다음 명령을 입력하여 데이터베이스 관리자를 시작하십시오(이미 수행하고 있지 않은 경우).

```
db2start
```

3. 생산용 응용프로그램을 빌드할 때는 실행 파일에 빌드된 DB2 런타임 경로가 응용프로그램을 개발하고 있는 지역 DB2 인스턴스의 경로가 아니라 설치 경로여야 합니다. 이 책은 개발 환경에서 응용프로그램을 빌드하는 방법에 대해 설명하기 위해 작성되었으므로 UNIX의 sqllib/include 및 sqllib/lib 인스턴스 사본과, OS/2 및 Windows 32비트 운영 체제의 %DB2PATH%\include 및 %DB2PATH%\lib 인스턴스 사본을 제공합니다.
4. 샘플 프로그램을 변경하거나 빌드하기 전에 UNIX의 sqllib/samples에서, 또는 OS/2나 Windows 32비트 운영 체제 %DB2PATH%\samples에서 사용할 언어의 샘플을 사용자의 작업 디렉토리로 복사하는 것이 좋습니다. 이로써 나중에 참조해야 할 경우 원본 샘플을 보존할 수 있습니다.

빌드 파일, Makefile 및 오류 체크 유틸리티

DB2에서는 프로그램 개발에 필요한 빌드 도구를 제공합니다. 이들 도구는 제공된 샘플 프로그램을 빌드하기 쉽게 해주며, 광범위한 DB2 기능을 보여줍니다. 또한 도구를 사용하여 사용자의 데이터베이스 프로그램을 빌드할 수 있습니다. 각각의 지원되는 컴파일러에 대해 DB2는 빌드 파일, makefile 및 오류 체크 유틸리티를 제공하며, 샘플 프로그램과 함께 샘플 디렉토리에서 사용 가능합니다. 이 절에서는 이들 도구의 사용 방법에 대해 설명합니다.

빌드 파일

각각의 다음 장에서는 지원되는 플랫폼 컴파일러에서 프로그램을 빌드하기 위한 컴파일 및 링크 명령을 포함하는 파일을 사용합니다. 이들 파일을 OS/2에서는 명령 파일이라고 하고, UNIX에서는 스크립트 파일이라고 하며, Windows 32비트 운영 체제에서는 배치 파일이라고 합니다. 여기서는 일반적으로 빌드 파일이라고 합니다.

DB2가 각 언어에 대해 빌드 파일을 제공하며, 빌드한 프로그램 유형이 사용 가능한 지원되는 플랫폼에서 각 언어에 대한 샘플 프로그램과 동일한 디렉토리에 제공됩니다. 57 페이지의 표16에서는 Windows 32비트 운영 체제의 C++를 제외하고 지원되는 모든 플랫폼의 모든 언어에 대한 빌드 파일을 설명합니다. 파일 이름 확

장자는 제외시켰습니다. OS/2의 경우 확장자는 .cmd이고 Windows 32비트 운영 체제의 경우 확장자는 .bat입니다. UNIX 플랫폼에 대한 확장자는 없습니다.

Windows 32비트 운영 체제의 경우, 지원되는 두 가지 C++ 컴파일러인 Microsoft Visual C++와 IBM VisualAge C++가 있습니다. 이 컴파일러 각각에 대해 bldclisp를 제외한 각 빌드 파일 이름의 "bld" 다음에 "m"이나 "v"가 삽입되어 bldmclis 또는 bldvclis가 됩니다. 이러한 빌드 파일은 표17에 나열되어 있습니다. .bat 확장자는 제외시켰습니다.

표 16. DB2 빌드 파일

빌드 파일	빌드된 프로그램 유형
bldsqlj	Java Embedded SQLJ 응용프로그램.
bldsqljs	Java Embedded SQLJ 저장 프로시듀어.
bldcli	Embedded SQL이 있거나 없는 DB2 CLI 응용프로그램.
bldapi	데이터베이스를 작성 및 삭제하도록 DB2 API를 포함하는 utilapi 유틸리티 파일의 링크가 필요하며 Embedded SQL이 있거나 없는 DB2 CLI 응용프로그램.
bldclisp	DB2 CLI 저장 프로시듀어(비 Embedded SQL).
bldapp	Embedded SQL이 있거나 없는 응용프로그램.
bldsrv	Embedded SQL 저장 프로시듀어.
bldudf	사용자 정의 함수(UDF).
bldmt	멀티스레드 Embedded SQL 응용프로그램(지원되는 UNIX 플랫폼의 C/C++에서만 사용 가능함).
bldevm	이벤트 모니터 샘플 프로그램 evm(AIX, OS/2 및 Windows 32비트 운영 체제에서만 사용 가능함).

표 17. Windows 32비트 운영 체제용 C/C++ 빌드 파일

빌드 파일	빌드된 프로그램 유형
bldmcli	Embedded SQL이 있거나 없는 Microsoft Visual C++ DB2 CLI 응용프로그램.
bldvcli	Embedded SQL이 있거나 없는 VisualAge C++ DB2 CLI 응용프로그램.
bldmapi	데이터베이스를 작성 및 삭제하도록 DB2 API를 포함하는 utilapi 유틸리티 파일의 링크가 필요하며 Embedded SQL이 있거나 없는 Microsoft Visual C++ DB2 CLI 응용프로그램.

표 17. Windows 32비트 운영 체제용 C/C++ 빌드 파일 (계속)

빌드 파일	빌드된 프로그램 유형
bldvapi	데이터베이스를 작성 및 삭제하도록 DB2 API를 포함하는 utilapi 유틸리티 파일의 링크가 필요하며 Embedded SQL이 있거나 없는 VisualAge C++ DB2 CLI 응용프로그램.
bldmclis	Microsoft Visual C++ DB2 CLI 저장 프로시듀어(비 Embedded SQL).
bldvclis	VisualAge C++ DB2 CLI 저장 프로시듀어(비 Embedded SQL).
bldmapp	Embedded SQL이 있거나 없는 Microsoft Visual C++ 응용프로그램.
bldvapp	Embedded SQL이 있거나 없는 VisualAge C++ 응용프로그램.
bldmsrv	Microsoft Visual C++ Embedded SQL 저장 프로시듀어.
bldvsrv	VisualAge C++ Embedded SQL 저장 프로시듀어.
bldmudf	Microsoft Visual C++ 사용자 정의 함수(UDF).
bldvudf	VisualAge C++ 사용자 정의 함수(UDF).
bldmevm	Microsoft Visual C++ 컴파일러가 있는 이벤트 모니터 샘플 프로그램 evm
bldvevm	VisualAge C++ 컴파일러가 있는 이벤트 모니터 샘플 프로그램 evm

빌드 파일은 지원되는 컴파일러에서 다양한 종류의 프로그램을 빌드하기 위해 DB2에서 권장하는 컴파일 및 링크 옵션을 매우 분명하게 보여주기 때문에 이 책에서 설명합니다. 일반적으로 여러 가지 그 밖의 컴파일 및 링크 옵션이 사용 가능하며, 사용자는 자유롭게 이들을 실습할 수 있습니다. 제공되는 모든 컴파일 및 링크 옵션에 대해서는 컴파일러 문서를 참조하십시오. 샘플 프로그램 빌드 외에도, 개발자는 또한 빌드 파일을 사용하여 고유한 프로그램을 빌드할 수도 있습니다. 사용자가 프로그래밍 개발을 보조하기 위해 수정할 수 있는 템플리트로 샘플 프로그램을 사용할 수 있습니다.

편리하게도, 빌드 파일은 컴파일러가 허용하는 파일 이름으로 소스 파일을 빌드하도록 설계되어 있습니다. 프로그램 이름이 파일에 하드 코딩된 makefile과는 다릅니다. 빌드 파일은 UNIX에서 $1 변수 또는 OS/2 및 Windows 32비트 운영 체제에서 %1 변수를 사용하여 프로그램 이름을 내부적으로 대체합니다. 그 밖의 유사하게 이름 지정된 변수가 그 밖의 필수 인수를 대체합니다. 빌드 파일은 DB2 API, DB2 CLI, Embedded SQL, 저장 프로시듀어 또는 사용자 정의 함수(UDF)

와 같은 특정 종류의 프로그램 빌드에 적합하기 때문에 빠르고 쉬운 실습을 허용합니다. 설계된 특정 종류의 프로그램을 컴파일러가 지원하는 경우, 각 유형의 빌드 파일이 제공됩니다.

빌드 파일이 생성하는 오브젝트와 실행 파일은 소스 파일이 수정되지 않은 경우에도 프로그램이 빌드될 때마다 자동으로 덮어씁니다. makefile과는 다릅니다. 이것은 개발자가 이전의 오브젝트와 실행 파일을 삭제하지 않고 기존 프로그램을 재빌드하거나 소스를 수정할 수 있음을 의미합니다.

빌드 파일은 샘플 데이터베이스에 대한 기본 설정을 포함합니다. 사용자가 다른 데이터베이스에 액세스 중인 경우, 기본값을 덮어쓰도록 다른 매개변수를 제공할 수 있습니다. 다른 데이터베이스를 일관적으로 사용할 경우, 빌드 파일 내에서 sample을 바꾸어 해당 데이터베이스 이름을 하드 코딩할 수도 있습니다.

Embedded SQL 프로그램에 사용되는 빌드 파일은 Embedded SQL 프로그램에 대한 사전 처리 컴파일 및 바인드 단계를 포함하는 다른 파일 embprep를 호출합니다. 이 단계에서 Embedded SQL 프로그램이 빌드되는 위치에 따라 선택적 매개변수 사용자 ID와 암호를 요구할 수 있습니다.

데이터베이스가 있는 서버 인스턴스에 개발자가 프로그램을 빌드하는 경우, 사용자 ID와 암호가 둘 다에 공통되기 때문에 제공할 필요가 없습니다. 한편, 개발자가 서버 데이터베이스에 원격으로 액세스하는 클라이언트 머신과 같은 다른 인스턴스에 있는 경우, 이들 매개변수를 지정해야 합니다. embprep 파일은 makefile에 의해서도 사용됩니다. 이 파일에 대한 자세한 내용은 60 페이지의 『makefile』을 참조하십시오.

마지막으로, 빌드 파일은 개발자의 편의에 따라 수정할 수 있습니다. 빌드 파일에서 데이터베이스 이름을 변경하는 것 외에도(위에서 설명한 대로), 개발자는 파일 내의 다른 매개변수를 쉽게 하드 코딩하고, 컴파일 및 링크 옵션을 변경하거나 또는 기본 DB2 인스턴스 경로를 변경할 수 있습니다. 단순하고 직설적이며 특정한 빌드 파일의 특성을 사용하여 빌드 파일을 사용자의 요구에 맞게 쉬운 타스크로 조정할 수 있게 해줍니다.

makefile

지원되는 컴파일러에 대한 각각의 샘플 디렉토리는 제공된 샘플 프로그램을 빌드하도록 makefile을 포함합니다. makefile은 컴파일러에 제공된 대부분의 DB2 샘플 프로그램을 빌드합니다. 각각의 샘플 프로그램 컴파일에 사용되는 여러 공통 요소에 대한 변수를 사용합니다. makefile의 구문과 명령의 출력은 제공되는 빌드 파일과 몇 가지 중요한 관점에서 서로 다릅니다. make 명령은 사용하기 간단하며 강력합니다.

make <program_name>
: 지정된 프로그램을 컴파일하고 링크합니다.

make all
: makefile에 나열된 모든 프로그램을 컴파일하고 링크합니다.

make clean
: makefile에 나열된 모든 프로그램에 대해 오브젝트 파일과 같은 모든 중간 파일을 삭제합니다.

make cleanall
: makefile에 나열된 모든 프로그램에 대해 모든 중간 파일과 실행 파일을 삭제합니다.

빌드 파일과 달리, makefile은 파일에 나열된 프로그램에 대한 기존 중간 파일 및 실행 파일을 덮어쓰지 않습니다. `make all` 명령이 이러한 파일을 무시하기 때문에 다른 파일에 이미 실행 파일이 있는 경우 `make all`을 사용하면 일부 파일에 대한 실행 파일을 작성하는 것이 보다 빠릅니다. 그러나 `make clean` 및 `make cleanall` 명령은 필요하지 않는 기존 오브젝트와 실행 파일을 제거할 수도 있습니다.

makefile은 프로그램 개발에 사용할 수 있습니다. 빌드 파일보다 사용하기가 불편하지만, make 명령의 기능과 편리성을 원하는 경우, 고려할 수 있습니다. 기존 makefile에 새로운 프로그램을 포함시키려면 유사한 종류의 기존 샘플 프로그램을 템플리트로 사용하여 프로그램 항목에 대한 구문을 코딩할 수 있습니다. DB2 API, DB2 CLI 및 Embedded SQL 프로그램과 저장 프로시듀어 및 UDF에 대한 구문이 서로 다름을 유의하십시오.

제공된 makefile을 사용하는 방법의 예는 다음과 같습니다. AIX의 samples/cli 디렉토리에 있는 이 makefile은 IBM C 컴파일러를 사용하여 AIX에 제공된 모든 DB2 CLI 샘플을 빌드합니다. 이 예는 클라이언트 응용프로그램 spclient가 호출할 수 있는 공유 라이브러리로 저장 프로시듀어 프로그램 spserver를 빌드하는 방법을 보여줍니다.

makefile을 사용하기 전에 파일에 포함된 다음 변수를 편집해야 할 수도 있습니다.

DB 사용 중인 데이터베이스. 기본값으로 sample을 설정하십시오.

UID 사용 중인 사용자 ID. 기본값으로 값이 설정되지 않습니다.

PWD UID 사용자 ID에 대한 암호. 기본값으로 값이 설정되지 않습니다.

AIX에서 DB2 CLI makefile은 다음과 같이 DB2PATH, CC, COPY, ERASE, CFLAGS 및 LIBS 변수를 정의합니다.

```
# Set DB2PATH to where DB2 will be accessed.
# The default is the instance path.
DB2PATH=$(HOME)/sqllib
CC = cc
COPY = cp
ERASE = rm -f
# The required compiler flags
CFLAGS= -I$(DB2PATH)/include
# The required libraries
LIBS= -L$(DB2PATH)/lib -Wl,-rpath,$(DB2PATH)/lib -ldb2
```

makefile은 저장 프로시듀어, spserver를 컴파일하고 함수 서브디렉토리로 공유 라이브러리를 복사할 때 이들 변수를 사용합니다.

```
spserver : utilcli.o
        $(CC) -o spserver spserver.c utilcli.o $(CFLAGS) $(LIBS) \
        -H512 -T512 -bE:spserver.exp -e outlanguage
        $(ERASE) $(DB2PATH)/function/spserver
        $(COPY) spserver $(DB2PATH)/function/spserver
```

Embedded SQL 프로그램의 경우, makefile은 이러한 Embedded SQL 프로그램에 대한 사전 처리 컴파일 및 바인드 단계를 포함하는 embprep 파일을 호출합니다. 이 파일을 각각의 Embedded SQL 프로그램에 대해 호출되는 별도의 파일로 만들면 makefile 본문에서 이 단계들을 반복할 필요가 없습니다. 이 파일은 사용

자 ID, 암호 및 데이터베이스 이름에 대한 매개변수를 사용하여 서버의 데이터베이스에 연결합니다. makefile은 embprep를 호출할 때 여기에 이 값들을 전달합니다.

데이터베이스 변수 DB는 기본값으로 sample 데이터베이스에 하드 코딩되며, 다른 데이터베이스를 사용할 경우 사용자가 변경할 수 있습니다. 사용자 ID와 암호 변수 UID 및 PWD는 기본값으로 어떤 값으로도 설정되지 않습니다. 사용자가 서버 데이터베이스와 동일한 인스턴스에서 이미 작업 중일 경우, 이러한 선택적 매개변수를 사용할 필요가 없습니다. 그러나 사용자가 원격으로 클라이언트 머신에서 서버로 연결한 다음 UID 및 PWD 변수에 적절한 값을 제공하여 makefile을 수정할 수 있는 때와 같이 위의 경우에 해당하지 않는 경우, embprep 사전 처리 컴파일 및 바인드 파일에 이러한 선택적 매개변수가 자동으로 전달됩니다. 다음은 Embedded SQL 응용프로그램 updat를 빌드하기 위해 Windows NT의 Micro Focus COBOL makefile이 호출하는 embprep 파일의 예입니다.

```
updat.cbl : updat.sqb
        embprep updat $(DB) $(UID) $(PWD)
updat.obj : updat.cbl
        $(CC) updat.cbl;
updat : updat.obj checkerr.obj
        $(LINK) updat.obj checkerr.obj $(LIBS)
```

오류 체크 유틸리티

DB2 AD 클라이언트는 몇 가지 유틸리티 파일을 제공합니다. 이 파일들은 오류 체크 및 오류 정보를 인쇄하도록 함수를 포함하고 있습니다. 예외로 데이터베이스를 작성하고 삭제하기 위해 DB2 API를 호출하는 CLI 유틸리티 파일 utilapi.c가 있습니다. 유틸리티 파일은 샘플 디렉토리의 각 언어에 제공됩니다. 이러한 오류 체크 유틸리티 파일은 응용프로그램에서 사용될 때 유용한 오류 정보를 제공하고 보다 쉽게 DB2 프로그램을 디버그할 수 있게 도와 줍니다. 대부분의 오류 체크 유틸리티는 DB2 API GET SQLSTATE MESSAGE 및 GETERROR MESSAGE를 사용하여 프로그램 실행 중에 발생한 문제점과 관련된 SQLSTATE 및 SQLCA 정보를 얻습니다. DB2 CLI 유틸리티 파일 utilcli.c는 이러한 DB2 API를 사용하지 않고 대신 동일한 DB2 CLI문을 사용합니다. 모든 오류 체크 유틸리티에서 개발자가 문제점을 빠르게 이해할 수 있도록 설명적인 오류 메시지를 인쇄합니다.

저장 프로시듀어와 사용자 정의 함수(UDF)와 같은 일부 DB2 프로그램은 이러한 유틸리티를 사용할 필요가 없습니다. 예외가 발생하는 경우에는 SQLException 오브젝트가 삭제되기 때문에 이러한 유틸리티는 Java에도 필요하지 않습니다.

서로 다른 프로그래밍 언어에 대해 DB2 지원 컴파일러가 사용하는 오류 체크 유틸리티 파일은 다음과 같습니다.

checkerr.cbl
: COBOL 프로그램의 경우

utilcli.c
: CLI 프로그램의 경우

utilapi.c
: C 비 Embedded SQL 프로그램의 경우

utilemb.sqc
: C Embedded SQL 프로그램의 경우

utilapi.C
: C++ 비 Embedded SQL 프로그램의 경우

utilemb.sqC
: C++ Embedded SQL 프로그램의 경우

유틸리티 함수를 사용하려면 먼저 유틸리티 파일을 컴파일한 다음 목표 프로그램의 실행 파일을 작성하는 동안 링크된 오브젝트 파일을 컴파일해야 합니다. makefile 및 samples 디렉토리의 빌드 파일은 둘 다 오류 체크 유틸리티가 필요한 프로그램에 대해 이를 수행합니다.

다음 예는 DB2 프로그램에서 오류 체크 유틸리티를 사용하는 방법을 보여줍니다. utilemb.h 헤더 파일은 SqlInfoPrint() 및 TransRollback() 함수를 대체하는 EMB_SQL_CHECK 매크로를 정의합니다.

```
#define EMB_SQL_CHECK( MSG_STR )          \
   if( SqlInfoPrint( MSG_STR, &sqlca, __LINE__, __FILE__) != 0 ) TransRollback( );
```

SqlInfoPrint()는 SQLCODE 플래그를 점검합니다. 이 함수는 이 플래그에서 지정하는 특정 오류과 관련되어 사용할 수 있는 정보를 출력합니다. 또한 소스 코드에서 오류가 발생한 위치를 가리킵니다. TransRollback()을 사용하면 유틸리

티 파일이 오류가 발생한 트랜잭션을 안전하게 구간 복원할 수 있습니다. 이 함수에는 데이터베이스에 연결하고 구간 복원을 실행하기 위해 Embedded SQL문이 필요합니다. 다음은 C++ 프로그램 cursor가 매크로를 사용하여 유틸리티 함수를 호출하고 SqlInfoPrint() 함수의 MSG_STR 매개변수에 값 "DECLARE CURSOR"를 제공하는 방법의 예입니다.

```
Cursor::Fetch () {
   EXEC SQL DECLARE c1 CURSOR FOR
            SELECT name, dept FROM staff WHERE job='Mgr'
            FOR UPDATE OF job;
   EMB_SQL_CHECK("DECLARE CURSOR") ;
```

EMB_SQL_CHECK 매크로는 DECLARE문이 실패하는 경우 트랜잭션이 안전하게 구간 복원되고 해당 오류 메시지가 출력될 수 있게 합니다.

개발자는 DB2 프로그램을 작성할 때 이러한 오류 체크 유틸리티를 사용하고 빌드하는 것이 좋습니다.

Java 애플릿 및 응용프로그램

Java 애플릿 및 응용프로그램을 빌드하려면 모든 플랫폼에서 동일한 단계를 수행해야 하며, 이러한 정보에 대한 장이 75 페이지의 『제4장 Java 애플릿 및 응용프로그램 빌드』입니다. 각각의 플랫폼에 필요한 특정 설정 정보가 있으며 이 장에서 별도의 절로 제공됩니다. 설정 정보는 39 페이지의 『제2장 설정』에 제공된 DB2 설정 정보를 보충합니다.

Java 장에서는 JDBC 드라이버를 사용하는 JDBC 프로그램과, JDBC 드라이버에 추가하여 Java 지원을 위한 Embedded SQL을 사용하는 SQLJ 프로그램을 빌드하는 방법에 대해 설명합니다. JDBC 및 SQLJ 애플릿, 응용프로그램과 저장 프로시듀어 빌드 방법에 대해 설명합니다. 또한 JDBC 또는 SQLJ문을 포함하지 않는 Java 사용자 정의 함수(UDF)를 빌드하는 방법에 대해 설명합니다.

samples 디렉토리에는 Java makefile 및 SQLJ 프로그램에 대한 빌드 파일이 들어 있습니다. JDBC 프로그램은 명령행에서 빌드하기 쉽기 때문에 이에 대한 빌드 파일이 제공되지 않습니다.

DB2 API 응용프로그램

DB2 AD 클라이언트에는 DB2 API를 호출하는 샘플 프로그램이 들어 있습니다. 이 책의 뒷부분에 있는 "응용프로그램 빌드" 장에서는 해당 플랫폼에 대해 DB2 AD 클라이언트가 제공하는 DB2 응용프로그램 빌드 파일을 사용하여 지원되는 컴파일러에 대한 샘플 프로그램을 빌드하는 방법에 대해 설명합니다. 또한 제공되는 makefile을 사용할 수도 있습니다. makefile과 빌드 파일은 둘 다 사용할 수 있는 컴파일러 옵션을 보여줍니다. 이들 옵션은 각 플랫폼의 지원되는 컴파일러에 대해 해당하는 장에서 정의됩니다. 환경에 대한 옵션을 수정해야 할 수도 있습니다.

다음 샘플 프로그램이 이 책에서 사용되며 지원되는 프로그래밍 언어를 사용하여 DB2 API 응용프로그램을 빌드하고 수행하기 위한 단계를 보여줍니다. 수행하는 단계가 환경에 따라 다를 수도 있습니다.

클라이언트
: 다음 API의 사용을 보여줍니다. SET CLIENT 및 QUERY CLIENT

모든 DB2 API 샘플 프로그램에 대한 자세한 내용은 17 페이지의 『샘플 프로그램 테이블』을 참조하십시오.

지원되는 DB2 API 샘플 프로그램에 대한 소스 파일은 sqllib/samples(UNIX) 및 %DB2PATH%\samples(OS/2 및 Windows 32비트 운영 체제)의 해당하는 프로그래밍 언어 서브디렉토리에 있습니다.

샘플 프로그램을 빌드한 후 사용자의 응용프로그램을 작성하기 위한 템플리트로 사용할 수 있습니다. 제공된 makefile 또는 빌드 파일을 사용하여 DB2 API 프로그램을 빌드할 수 있습니다.

주: API 응용프로그램을 작성할 때 모든 API 입력 구조는 명시적으로 설정되지 않은 구조 요소가 0으로 초기화되도록 0으로 memset되어야 합니다. 이것은 하위 레벨의 API 버전에 대해 코딩된 응용프로그램을 재컴파일할 때 중요합니다. 재컴파일할 때 새로운 구조 정의가 사용되지만 모든 요소가 초기화되지 않을 수도 있습니다. memset을 호출하면 이들 요소가 초기화됩니다. 다음은 입력 데이터 구조 pLoadInStruct, memset이 0으로 설정된 형태를 보여주는 예입니다.

```
memset( pLoadInStruct, 0, sizeof(pLoadInStruct) );
```

DB2 CLI 응용프로그램

DB2 AD 클라이언트에는 DB2 콜 레벨 인터페이스(DB2 CLI) 함수 호출을 사용하는 샘플 프로그램이 제공됩니다. 샘플을 연구하여 응용프로그램에서 이들 함수 호출을 사용하여 DB2 데이터베이스에 액세스하는 방법을 알 수 있습니다.

ODBC SDK(DB2에 포함되지 않음)를 사용하여 응용프로그램을 재컴파일하고 ODBC 드라이버 관리자가 응용프로그램 플랫폼에서 사용 가능하도록 제공된 ODBC에서 ODBC를 준수하는 DB2 CLI 응용프로그램이 작동되도록 포팅할 수 있습니다.

샘플 프로그램, 빌드 파일 및 makefile이 UNIX의 sqllib/samples/cli 디렉토리 또는 OS/2 및 Windows 32비트 운영 체제의 %DB2PATH%\samples\cli에 포함되어 있습니다. 빌드 파일과 makefile에서 사용자의 환경에 맞게 컴파일러 옵션을 수정해야 할 수도 있습니다.

다음은 이 책에서 사용되는 샘플 프로그램으로 DB2 CLI 응용프로그램을 빌드하고 수행하기 위한 단계를 보여줍니다(수행할 단계는 환경에 따라 다를 수도 있음).

tbinfo

테이블 레벨에서 정보를 가져오고 설정하는 방법을 보여줍니다.

dbusemx

Embedded SQL과 함께 단일 데이터베이스를 사용하는 방법을 보여줍니다.

dbmconn

다중 데이터베이스에 연결 및 연결 해제하는 방법을 보여줍니다.

spclient

클라이언트/서버 예의 클라이언트 프로그램이며 서버 프로그램은 spserver입니다.

spserver

클라이언트/서버 예의 서버 프로그램이며 클라이언트 프로그램은 spclient입니다.

udfcli

사용자 정의 함수(UDF) 프로그램 udfsrv에서 작성한 함수를 사용합니다.

모든 DB2 CLI 샘플 프로그램에 대한 자세한 내용은 29 페이지의 표7을 참조하십시오. *CLI Guide and Reference*에서는 DB2 CLI를 사용하는 샘플의 작동 방식에 대해 설명합니다.

Embedded SQL 응용프로그램

주: SQLJ(Java용 Embedded SQL)는 여기에서 설명하지 않으며, 75 페이지의 『제 4장 Java 애플릿 및 응용프로그램 빌드』에서 자세히 설명합니다.

DB2 AD 클라이언트에는 SQL문을 포함하는 샘플 프로그램이 들어 있습니다. 이 책의 뒷부분에 있는 "응용프로그램 빌드" 장에서는 해당 플랫폼에 대해 DB2 AD 클라이언트가 제공하는 빌드 파일을 사용하여 지원되는 컴파일러에 대한 샘플 프로그램을 빌드하는 방법에 대해 설명합니다. 또한 제공되는 makefile을 사용할 수도 있습니다. makefile과 빌드 파일은 둘 다 사용할 수 있는 컴파일러 옵션을 보여줍니다. 이들 옵션은 각 플랫폼의 지원되는 컴파일러에 대해 해당하는 장에서 정의됩니다. 환경에 대한 옵션을 수정해야 할 수도 있습니다.

Embedded SQL을 포함하는 샘플 프로그램을 빌드하기 위해 빌드 파일을 수행할 때, 빌드 파일은 다음과 같은 단계를 실행합니다.

- 데이터베이스에 연결합니다.
- 소스 파일을 사전 처리 컴파일합니다.
- 바인드 파일을 데이터베이스에 바인드합니다.
- 데이터베이스에서 연결 해제합니다.
- 소스 파일을 컴파일하고 링크합니다.

다음은 이 책에서 사용되는 샘플 프로그램으로 지원되는 프로그래밍 언어를 사용하여 Embedded SQL 응용프로그램을 빌드하고 실행하기 위한 단계를 보여줍니다. 수행할 단계는 환경에 따라 다를 수도 있습니다.

updat 정적 SQL을 사용하여 데이터베이스를 갱신합니다.

다음 샘플은 C 및 C++를 사용하는 저장 프로시듀어 및 사용자 정의 함수(UDF)를 위한 Embedded SQL 클라이언트 응용프로그램을 보여줍니다.

spclient
: 저장 프로시듀어 호출을 보여주는 클라이언트 프로그램이며 서버 프로그램은 spserver입니다.

spserver
: 저장 프로시듀어를 보여주는 서버 프로그램이며 클라이언트 프로그램은 spclient입니다.

udfcli
: 사용자 정의 함수(UDF) 라이브러리 udfsrv에서 ScalarUDF 함수를 사용합니다.

다음 샘플은 COBOL을 사용하는 저장 프로시듀어 및 사용자 정의 함수(UDF)를 보여줍니다.

outcli
: 저장 프로시듀어 호출을 보여주는 클라이언트 프로그램이며 서버 프로그램은 outsrv입니다.

outsrv
: 저장 프로시듀어를 보여주는 서버 프로그램이며 클라이언트 프로그램은 outcli입니다.

calludf
: 사용자 정의 함수(UDF) 라이브러리 udf의 함수를 호출합니다.

모든 Embedded SQL 샘플 프로그램에 대한 자세한 내용은 17 페이지의 『샘플 프로그램 테이블』을 참조하십시오.

지원되는 이들 샘플 프로그램에 대한 소스 파일은 UNIX에서는 sqllib/samples, 그리고 OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples의 해당하는 프로그래밍 언어 서브디렉토리에 있습니다.

샘플 프로그램을 빌드한 후 사용자의 응용프로그램을 작성하기 위한 템플리트로 사용할 수 있습니다. 샘플 프로그램을 사용자의 SQL문으로 수정하여 이를 수행할 수 있습니다. 제공된 makefile 또는 빌드 파일을 사용하여 프로그램을 빌드할 수 있습니다.

17 페이지의 『샘플 프로그램』은 모든 샘플 프로그램을 나열합니다. *응용프로그램 개발 안내서*에서는 Embedded SQL을 포함하는 샘플이 작동하는 방법에 대해 설명합니다.

저장 프로시듀어

서버에서 저장 프로시듀어가 빌드되어 저장됩니다. 클라이언트 응용프로그램에는 이 프로시듀어에 원격으로 액세스할 수 있습니다. 그런 다음 저장 프로시듀어 함수는 서버 데이터베이스에서 지역적으로 처리를 수행하고 결과를 클라이언트로 다시 전송합니다. 이렇게 하면 네트워크 교통량이 줄어들고 전반적인 성능이 향상됩니다.

저장 프로시듀어는 분리(fenced) 또는 분리되지 않은(unfenced) 상태로 수행될 수 있습니다. 분리되지 않은 저장 프로시듀어는 데이터베이스 관리자와 동일한 주소 공간에서 수행되고, 데이터베이스 관리자와 분리된 주소 공간에서 수행되는 분리 저장 프로시듀어와 비교할 때, 결과적으로 성능이 향상됩니다. 분리되지 않은 저장 프로시듀어를 사용하면 사용자 코드가 실수로 데이터베이스 제어 구조를 손상시킬 수 있는 위험성이 있습니다. 따라서 성능을 최대한으로 할 때에 비분리 저장 프로시듀어만을 수행해야 합니다. 이러한 프로그램이 분리되지 않은 상태로 수행되기 전에 테스트되었는지 확인하십시오. 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.

이 책에 있는 저장 프로시듀어는 경로 sqllib/function에 있는 서버에 저장되어 있습니다. 호출된 프로시듀어가 공유 라이브러리 이름과 일치하는 DB2DARI 매개변수 스타일 저장 프로시듀어의 경우 이 위치는 저장 프로시듀어가 분리되었음을 나타냅니다. 이 저장 프로시듀어의 유형이 분리되지 않도록 하려면 이것을 sqllib/function/unfenced 디렉토리로 이동해야 합니다. 다른 모든 DB2 저장 프로시듀어 유형의 경우 호출 프로그램에서 해당 프로시듀어가 CREATE FUNCTION문과 분리 또는 분리되지 않았는지 나타냅니다. DB2 저장 프로시듀

어의 다른 유형을 작성하고 사용하는 것에 대해서는 *응용프로그램 개발 안내서*에 있는 "저장 프로시듀어" 장을 참조하십시오.

다음 샘플 프로그램은 SQL 프로시듀어를 사용하여 서버에서 저장 프로시듀어 라이브러리를 빌드하고 저장하기 위한 단계를 보여줍니다.

spserver.db2

서버에서 공유 라이브러리를 작성하는 데 사용되는 SQL 프로시듀어가 들어 있는 CLP 스크립트입니다. 이 스크립트는 CLP `call` 명령이나 C, C++ 및 CLI 디렉토리에 있는 `spclient` 응용프로그램에 의해 호출될 수 있습니다.

다음 샘플 프로그램은 C 및 C++를 사용하여 서버에서 저장 프로시듀어 라이브러리를 빌드하고 저장하는 단계를 보여줍니다.

spserver

클라이언트/서버 예의 서버 프로그램이며 클라이언트 프로그램은 `spclient` 입니다.

다음 샘플 프로그램은 Java를 사용하여 서버에서 저장 프로시듀어 라이브러리를 빌드하고 저장하기 위한 단계를 보여줍니다.

Spserver

클라이언트/서버 예의 서버 프로그램이며 클라이언트 프로그램은 `Spclient` 입니다.

다음 샘플 프로그램은 COBOL을 사용하여 서버에서 저장 프로시듀어 라이브러리를 빌드하고 저장하기 위한 단계를 보여줍니다.

outsrv

클라이언트/서버 예의 서버 프로그램이며 클라이언트 프로그램은 `outcli` 입니다.

사용자 정의 함수(UDF)

사용자 정의 함수(UDF)를 사용하여 사용자의 요구사항에 맞는 SQL의 확장자를 작성할 수 있습니다. 저장 프로시듀어와 같이, 사용자 정의 함수(UDF)는 클라이언트 응용프로그램이 액세스할 서버에 저장됩니다. UDF는 Embedded SQL문을 포함하지 않습니다.

다음 샘플 프로그램이 이 책에서 사용되며 서버에서 UDF 라이브러리를 빌드하고 저장하기 위한 단계를 보여줍니다.

udfsrv

사용자 정의 함수(UDF)의 라이브러리를 작성합니다. 이것은 C 및 C++에 대해서만 사용할 수 있습니다. 클라이언트 응용프로그램 `udfcli`는 이 함수를 호출합니다.

udf 사용자 정의 함수(UDF)의 라이브러리를 작성합니다. 이것은 COBOL에 대해서만 사용할 수 있습니다. 클라이언트 응용프로그램 `calludf`는 이 함수를 호출합니다.

UDFsrv

사용자 정의 함수(UDF)의 라이브러리를 작성합니다. 이것은 Java에 대해서만 사용할 수 있습니다. `UDFcli` 및 `UDFclie`는 각각 이러한 함수를 호출하는 JDBC 및 SQLJ 클라이언트 응용프로그램입니다.

멀티스레드 응용프로그램

DB2는 지원되는 UNIX 플랫폼에서 C 및 C++ 멀티스레드 응용프로그램을 지원합니다. 이 응용프로그램을 사용하여 몇 개의 동시 프로세스로 몇 개의 동시 실행 스레드를 작동할 수 있습니다. 이를 통해 폴링 스킴에 의존하지 않고도 비동기 이벤트를 처리하고 이벤트 중심 응용프로그램을 작성할 수 있습니다. 다음 샘플 프로그램은 DB2 멀티스레드 응용프로그램의 빌드 방법을 보여줍니다.

thdsrver

스레드 작성 및 관리를 보여줍니다.

UDF 및 저장 프로시듀어에 대한 C++ 고려사항

C++에서는 함수 이름이 겹칠 수 있습니다. 다음과 같이 서로 다른 인수를 갖는 경우 동일한 이름을 가진 두 함수가 공존할 수 있습니다.

```
int func( int i )
```

및

```
int func( char c )
```

C++ 컴파일러는 기본값으로 함수 이름을 '유형에 의해 구분'하거나 또는 '혼용'합니다. 이전의 두 예에서 func__Fi 및 func__Fc와 같이, 인수 유형 이름을 함수 이름에 추가하여 함수를 판별하는 것을 의미합니다. 혼용된 이름은 각각의 플랫폼에서 서로 다르며, 따라서 명시적으로 혼용된 이름을 사용하는 코드는 이동할 수 없습니다.

OS/2 및 Windows 32비트 운영 체제에서 유형 구분 함수 이름은 .obj(오브젝트) 파일에서 판별할 수 있습니다.

OS/2 및 Windows에서 VisualAge C++ 컴파일러를 사용하면 다음과 같이 cppfilt 명령을 사용하여 .obj(오브젝트) 파일에서 유형 구분 함수 이름을 판별할 수 있습니다.

```
cppfilt -b /p myprog.obj
```

여기서 myprog.obj는 프로그램 오브젝트 파일입니다.

Windows에서 Microsoft Visual C++ 컴파일러를 사용하면 다음과 같이 dumpbin 명령을 사용하여 .obj(오브젝트) 파일로부터 유형 구분 함수 이름을 판별할 수 있습니다.

```
dumpbin /symbols myprog.obj
```

여기서 myprog.obj는 프로그램 오브젝트 파일입니다.

UNIX 플랫폼에서는 .o(오브젝트) 파일에서, 또는 nm 명령을 사용하여 공유 라이브러리에서 유형 구분 함수 이름을 판별할 수 있습니다. 이 명령은 많은 양의 출력을 생성할 수 있으며, 따라서 다음과 같이 해당하는 행을 찾기 위해 grep로 출력을 파이프하도록 제안합니다.

```
nm myprog.o | grep myfunc
```

여기서 myprog.o는 프로그램 오브젝트 파일이며, myfunc는 프로그램 소스 파일에 있는 함수입니다.

이 명령에 의해 생성된 출력에는 모두 혼용된 함수 이름이 있는 하나의 행이 포함되어 있습니다. 예를 들어, UNIX에서는 이 행이 다음과 유사하게 나타납니다.

```
myfunc__FPlT1PsT3PcN35|     3792|unamex|          | ...
```

일단 위의 명령 중 하나로부터 혼용된 함수 이름을 구하면 이 이름을 해당 명령에서 사용할 수 있습니다. 이 책에서는 위의 UNIX 예에서 구한 혼용된 함수 이름을 사용하여 아래에 다음과 같은 예를 제공합니다. OS/2 또는 Windows에서 구한 혼용된 함수 이름을 동일한 방식으로 사용할 수 있습니다.

CREATE FUNCTION을 사용하여 UDF를 등록할 때 EXTERNAL NAME 절은 혼용된 함수 이름을 지정해야 합니다.

```
CREATE FUNCTION myfunco(...) RETURNS...
       ...
       EXTERNAL NAME '/whatever/path/myprog!myfunc__FPlT1PsT3PcN35'
       ...
```

마찬가지로, 저장 프로시듀어를 호출할 때 CALL 함수 또한 혼용된 함수 이름을 지정해야 합니다.

```
CALL 'myprog!myfunc__FPlT1PsT3PcN35' ( ... )
```

저장 프로시듀어 또는 UDF 라이브러리가 겹치는 C++ 함수 이름을 포함하지 않는 경우, 컴파일러가 함수 이름을 유형으로 구분하지 않도록 extern "C"를 사용하는 옵션이 있습니다. (DB2가 취하는 이름과 매개변수에 따라 호출할 라이브러리 함수를 판별하기 때문에, UDF에 제공하는 SQL 함수 이름을 항상 겹쳐서 사용할 수 있습니다.)

```
#include <string.h>
#include <stdlib.h>
#include "sqludf.h"

/*------------------------------------------------------------------------*/
/* function fold: output = input string is folded at point indicated      */
/*                         by the second argument.                        */
/*         inputs: CLOB,                  input string                    */
/*                 LONG                   position to fold on             */
/*         output: CLOB                   folded string                   */
/*------------------------------------------------------------------------*/
extern "C" void fold(
      SQLUDF_CLOB       *in1,              /* input CLOB to fold       */
   ...
   ...
}
/* end of UDF: fold */

/*------------------------------------------------------------------------*/
/* function find_vowel:                                                   */
/*             returns the position of the first vowel.                   */
/*             returns error if no vowel.                                 */
/*             defined as NOT NULL CALL                                   */
/*         inputs: VARCHAR(500)                                           */
/*         output: INTEGER                                                */
/*------------------------------------------------------------------------*/
extern "C" void findvwl(
      SQLUDF_VARCHAR    *in,              /* input smallint           */
   ...
   ...
}
/* end of UDF: findvwl */
```

이 예에서 UDF fold 및 findvwl은 컴파일러가 유형에 의해 구분하지 않으며, 일반적인 이름을 사용하여 CREATE FUNCTION문에 등록해야 합니다. 마찬가지로, C++ 저장 프로시듀어가 extern "C"로 코딩된 경우, 구분되지 않은 함수 이름을 CALL문에서 사용합니다.

제4장 Java 애플릿 및 응용프로그램 빌드

사용자 플랫폼에 적합한 JDK를 사용하여 DB2 데이터베이스에 액세스하도록 Java 프로그램을 개발할 수 있습니다. JDK는 Java용 동적 SQL API인 JDBC(Java Database Connectivity)를 포함합니다.

DB2 JDBC 지원은 DB2 클라이언트와 서버에서 Java 지원 옵션의 일부로 제공됩니다. 이러한 지원을 사용하여 JDBC 응용프로그램 및 애플릿을 빌드하고 수행할 수 있습니다. 이들은 동적 SQL만을 포함하며 Java 호출 인터페이스를 사용하여 SQL문을 DB2에 전달합니다.

DB2 SQLJ(Java Embedded SQL) 지원은 DB2 AD Client의 일부로 제공됩니다. DB2 SQLJ 지원과 함께 DB2 JDBC 지원을 사용하여, SQLJ 애플릿 및 응용프로그램을 빌드하고 수행할 수 있습니다. 이들은 정적 SQL을 포함하며 DB2 데이터베이스에 바인드된 Embedded SQL문을 사용합니다.

DB2 AD 클라이언트가 제공하는 SQLJ 지원은 다음과 같습니다.

- SQLJ 프로그램에 있는 Embedded SQL문을 Java 소스 명령문으로 바꾸며 SQLJ 프로그램에 있는 SQL 조작에 관한 정보를 포함하는 직렬화된 프로파일을 생성하는 DB2 SQLJ 변환기 `sqlj`.
- 직렬화된 프로파일에 저장된 SQL문을 사전 처리 컴파일하고, 런타임 함수 호출에 사용자 정의하며, DB2 데이터베이스에 패키지를 생성하는 DB2 SQLJ 프로파일 조정자 `db2profc`. `db2profc` 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.
- 프로파일의 DB2 사용자 정의 버전의 내용을 일반 텍스트로 인쇄하는 DB2 SQLJ 프로파일 프린터 `db2profp`.

DB2 Java 응용프로그램을 실행하려면 원시 스레드 지원을 제공하는 JVM을 설치하고 호출해야 합니다. 원시 스레드를 사용하여 Java 응용프로그램을 실행하기 위해 명령에서 `-native` 옵션을 사용할 수 있습니다. 예를 들어, Java 샘플 응용프로그램 `App.class`를 실행하기 위해 다음 명령을 사용할 수 있습니다.

```
java -native App
```

원시 스레드를 일부 JVM에 대한 기본 스레드 지원으로 지정할 수도 있습니다. 이 장의 정보는 원시 스레드 지원을 기본 지원으로 가정합니다. 시스템에서 원시 스레드를 기본 스레드로 지정하는 방법을 보려면 JVM 문서를 참조하십시오.

DB2 Java 애플릿을 실행하기 위해 원시 스레드 및 녹색 스레드 지원을 제공하는 JVM을 호출할 수 있습니다.

Java에서 DB2 프로그래밍에 대한 자세한 내용은 *응용프로그램 개발 안내서*의 "Java 프로그래밍" 장을 참조하십시오.

최신의 갱신된 DB2 Java에 대한 자세한 내용은 다음 웹 페이지를 참조하십시오.

```
http://www.ibm.com/software/data/db2/java
```

환경 설정

SQLJ 프로그램을 빌드하도록 지원되는 플랫폼에서 IBM JDK 1.1.8을 사용 중이면 JDK 빌드 날짜가 1999년 11월 24일(또는 이후)이어야 합니다. 그렇지 않으면 컴파일 동안 JNI 패닉 오류가 발생할 수 있습니다.

SQLJ 프로그램을 빌드하도로고 지원되는 플랫폼에서 IBM JDK 1.2을 사용 중이면 JDK 빌드 날짜가 20000년 4월 17일(또는 이후)이어야 합니다. 그렇지 않으면 컴파일 동안 올바르지 않은 Java 유형 오류가 발생할 수 있습니다.

JDBC 환경을 테스트하기 위해 `sqllib\samples\java`(Windows)나 `sqllib/samples/java`(UNIX)에서 샘플 파일 `db2JDBCVersion.java`를 사용할 수 있습니다. `db2JDBCVersion` 프로그램은 현재 사용 중인 DB2 JDBC 드라이버 버전과 JDBC 환경이 이에 알맞게 설정되어 있는지 여부를 점검합니다. 이 프로그램은 OS/2에서 사용 불가능합니다.

AIX

DB2 JDBC 지원을 사용하여 AIX에 Java 응용프로그램을 빌드하려면 사용자의 개발 머신에서 다음을 설치하고 구성해야 합니다.

1. 다음 중 하나를 뜻합니다.
   - AIX 4.2.1 및 AIX 4.3.3 이상의 경우: IBM의 AIX용 JDK 버전 1.1.8 및 JRE 버전 1.1.8(DB2와 함께 설치). JDK 빌드 날짜는 1999년 11월 24일이나 이후여야 합니다.
   - AIX 4.3.3 이상의 경우: IBM의 AIX용 JDK 버전 1.2 및 JRE 버전 1.2. JDK 빌드 날짜는 2000년 4월 17일이나 이후여야 합니다.

   (`http://www.ibm.com/software/data/db2/java` 사이트를 참조하십시오.)
2. AIX용 DB2 Universal Database 버전 7 클라이언트 및 서버에서 제공되는 DB2 Java 지원.

DB2 Java 저장 프로시듀어 또는 UDF를 수행하려면 해당 머신에서 JDK가 설치된 경로를 포함하도록 서버에서 DB2 데이터베이스 관리자 구성을 갱신해야 합니다. 서버 명령행에서 다음을 입력하여 이를 수행할 수 있습니다.

```
db2 update dbm cfg using JDK11_PATH /home/db2inst/jdk11
```

여기서 /home/db2inst/jdk11은 JDK가 설치되는 경로입니다.

Java 저장 프로시듀어 및 UDF와 함께 JDBC 2.0 드라이버를 사용하려면 다음을 입력하십시오.

```
db2 update dbm cfg using JDK11_PATH /home/db2inst/jdk12
```

여기서 /home/db2inst/jdk12는 JDK 1.2가 설치된 경로입니다.

주: JDK11_PATH는 사실 임의 레벨의 JDK 위치를 표시하는 데 사용됩니다. JDK가 1.1인지 아니면 1.2인지의 여부는 DB2 레지스트리 변수 DB2_USE_JDK12의 설정에 의해 제어됩니다.

서버에서 다음 명령을 입력하여 JDK11_PATH 필드에 대한 정확한 값을 검증하기 위해 DB2 데이터베이스 관리자 구성을 점검할 수 있습니다.

```
db2 get dbm cfg
```

보기 쉽도록 출력을 파일로 경로 재지정할 수 있습니다. JDK11_PATH 필드는 출력 시작 부분에 표시됩니다. 이들 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

DB2 JDBC 지원을 사용하여 AIX에서 JDBC 및 SQLJ 프로그램을 수행하기 위해 AIX Java 환경을 갱신하는 명령이 데이터베이스 관리자 파일 db2profile 및 db2cshrc에 포함됩니다. DB2 인스턴스가 작성되면 CLASSPATH에 다음이 포함되도록 .profile 및 .cshrc가 수정됩니다.

- "." (현재 디렉토리)
- 파일 sqllib/java/db2java.zip

SQLJ 프로그램을 빌드하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/sqlj.zip
```

SQLJ 프로그램을 수행하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/runtime.zip
```

Java 응용프로그램과 JDBC 2.0 드라이버 사용

JDBC 1.2 드라이버는 JDBC 1.2 드라이버가 사용 가능한 모든 운영 체제에서 아직도 기본 드라이버로 사용됩니다. JDBC 2.0의 새 기능을 이용하려면 JDK 1.2 지원을 설치해야 합니다. JDBC 2.0의 새 기능을 이용하는 응용프로그램을 실행하기 전에 sqllib/java12 디렉토리에서 usejdbc2 명령을 발행하여 환경을 설정해야 합니다. 응용프로그램이 항상 JDBC 2.0 드라이버를 사용하도록 하려면 다음을 수행하도록 하십시오.

- bash 또는 korn 쉘의 경우, .profile과 같은 로그인 프로파일이나 .bashrc 또는 .kshrc와 같은 쉘 초기화 스크립트에 다음 행을 추가하십시오.

  ```
  . sqllib/java12/usejdbc2
  ```

- C 쉘의 경우, 다음 행을 .cshrc 스크립트에 추가하십시오.

  ```
  source sqllib/java12/usejdbc2.csh
  ```

db2profile을 수행하려면 이 명령이 해당 명령 다음에 위치하는지 확인하십시오. usejdbc2 및 usejdbc2.csh 스크립트가 db2profile 환경 설정을 사용하기 때문입니다.

JDBC 1.2 드라이버로 다시 전환하려면 sqllib/java12 디렉토리에서 다음 명령을 실행하십시오.

- bash 또는 korn 쉘의 경우:

  ```
  . usejdbc1
  ```

- C 쉘의 경우:

  ```
  source usejdbc1.csh
  ```

Java 저장 프로시듀어 및 UDF와 JDBC 2.0 드라이버 사용

Java 저장 프로시듀어 및 UDF와 함께 JDBC 2.0 드라이버를 사용하려면 인스턴스의 분리(fenced) 사용자 ID에 대해 환경을 설정해야 합니다. 기본 분리(fenced) 사용자 ID는 db2fenc1입니다. 분리 사용자 ID에 대해 환경을 설정하려면 CLP에서 다음 명령을 발행하십시오.

```
db2set DB2_USE_JDK12=1
```

JDBC 1.2 드라이버 지원으로 다시 전환하려면 CLP에서 다음 명령을 발행하십시오.

```
db2set DB2_USE_JDK12=
```

HP-UX

DB2 JDBC 지원을 사용하여 HP-UX에 Java 응용프로그램을 빌드하려면 사용자의 개발 머신에서 다음을 설치하고 구성해야 합니다.

1. 다음 중 하나를 뜻합니다.
   - Hewlett-Packard의 Java용 HP-UX Developer's Kit 릴리스 1.1.8
   - Hewlett-Packard의 Java용 HP-UX Developer's Kit 릴리스 1.2.2

   (http://www.ibm.com/software/data/db2/java 사이트를 참조하십시오.)
2. HP-UX용 DB2 Universal Database 버전 7 클라이언트 및 서버에서 제공되는 DB2 Java 지원.

주: Java용 HP-UX Developer's Kit 릴리스 1.2.2에는 Java UDF 및 저장 프로시듀어가 필요합니다.

DB2 Java 저장 프로시듀어 또는 UDF를 수행하려면 해당 머신에서 JDK가 설치된 경로를 포함하도록 서버에서 DB2 데이터베이스 관리자 구성을 갱신해야 합니다. 서버 명령행에서 다음을 입력하여 이를 수행할 수 있습니다.

```
db2 update dbm cfg using JDK11_PATH /home/db2inst/jdk12
```

여기서 /home/db2inst/jdk12는 JDK 1.2.2가 설치된 경로입니다.

주: JDK11_PATH는 사실 임의 레벨의 JDK 위치를 표시하기 데 사용됩니다. JDK가 1.1인지 아니면 1.2인지의 여부는 DB2 레지스트리 변수 DB2_USE_JDK12의 설정에 의해 제어됩니다.

서버에서 다음 명령을 입력하여 JDK11_PATH 필드에 대한 정확한 값을 검증하기 위해 DB2 데이터베이스 관리자 구성을 점검할 수 있습니다.

```
db2 get dbm cfg
```

보기 쉽도록 출력을 파일로 경로 재지정할 수 있습니다. JDK11_PATH 필드는 출력의 시작 부분에 표시됩니다. 이들 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

Java2 저장 프로시듀어를 수행하려면 공유 라이브러리 경로가 다음과 유사한지 확인하십시오.

```
export SHLIB_PATH=$JAVADIR/jre/lib/PA_RISC:$JAVADIR/
jre/lib/PA_RISC/classic:$HOME/sqllib/lib:/usr/lib:$SHLIB_PATH
```

여기서 $JAVADIR은 Java2 SDK의 위치입니다.

주: HotSpot JVM을 사용 중이며 "HotSpot 가상 시스템 오류"를 수신할 경우, HotSpot JVM을 사용하지 않는 것이 좋습니다. 다음의 두 가지 방법 중 하나로 이를 수행할 수 있습니다.

1. sqllib/bin/db2profc 파일을 편집하고 다음 행을

   ```
   java COM.ibm.db2.sqlj.DB2SQLJInstaller "$@"
   ```

   다음과 같이 변경하십시오.

   ```
   java -classic COM.ibm.db2.sqlj.DB2SQLJInstaller "$@"
   ```

2. $JAVADIR/bin/.java_wrapper의 103행을 편집하여 "vmtype=hotspot"을 "vmtype=classic"을 읽어 기본 JVM으로 HotSpot을 사용하지 않도록 하십시오. (이는 DB2가 아닌 Java를 사용하는 모든 응용프로그램에 영향을 줍니다.) 이렇게 할 경우, 사용자가 HotSpot JVM을 사용하지 않으려고 하면 Java에 대한 인수로 "-hotspot"를 지정해야 합니다(예: "java -hotspot <program_name>").

DB2 JDBC 지원을 사용하여 HP-UX에서 JDBC 및 SQLJ 프로그램을 수행하기 위해 HP-UX Java 환경을 갱신하는 명령이 데이터베이스 관리자 파일 db2profile 및 db2cshrc에 포함됩니다. DB2 인스턴스가 작성되면 .profile 및/또는 .cshrc가 다음과 같이 수정됩니다.

1. THREADS_FLAG가 "원시(native)"로 설정됩니다.
2. CLASSPATH에는 다음 사항들이 포함됩니다.
   - "." (현재 디렉토리)
   - 파일 sqllib/java/db2java.zip

SQLJ 프로그램을 빌드하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/sqlj.zip
```

SQLJ 프로그램을 수행하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/runtime.zip
```

Java 응용프로그램과 JDBC 2.0 드라이버 사용

JDBC 1.2 드라이버는 JDBC 1.2 드라이버가 사용 가능한 모든 운영 체제에서 아직도 기본 드라이버로 사용됩니다. JDBC 2.0의 새 기능을 이용하려면 JDK 1.2 지원을 설치해야 합니다. JDBC 2.0의 새 기능을 이용하는 응용프로그램을 실행하기 전에 sqllib/java12 디렉토리에서 usejdbc2 명령을 발행하여 환경을 설정해야 합니다. 응용프로그램이 항상 JDBC 2.0 드라이버를 사용하도록 하려면 다음을 수행하도록 하십시오.

- bash 또는 korn 쉘의 경우, .profile과 같은 로그인 프로파일이나 .bashrc 또는 .kshrc와 같은 쉘 초기화 스크립트에 다음 행을 추가하십시오.

  ```
  . sqllib/java12/usejdbc2
  ```

- C 쉘의 경우, 다음 행을 .cshrc 스크립트에 추가하십시오.

  ```
  source sqllib/java12/usejdbc2.csh
  ```

db2profile을 수행하려면 이 명령이 해당 명령 다음에 위치하는지 확인하십시오. usejdbc2 및 usejdbc2.csh 스크립트가 db2profile 환경 설정을 사용하기 때문입니다.

JDBC 1.2 드라이버로 다시 전환하려면 sqllib/java12 디렉토리에서 다음 명령을 실행하십시오.

- bash 또는 korn 쉘의 경우:

  ```
  . usejdbc1
  ```

- C 쉘의 경우:

  ```
  source usejdbc1.csh
  ```

Java 저장 프로시듀어 및 UDF와 JDBC 2.0 드라이버 사용

Java 저장 프로시듀어 및 UDF와 함께 JDBC 2.0 드라이버를 사용하려면 인스턴스의 분리(fenced) 사용자 ID에 대해 환경을 설정해야 합니다. 기본 분리(fenced) 사용자 ID는 db2fenc1입니다. 분리 사용자 ID에 대해 환경을 설정하려면 CLP에서 다음 명령을 발행하십시오.

```
db2set DB2_USE_JDK12=1
```

JDBC 1.2 드라이버 지원으로 다시 전환하려면 CLP에서 다음 명령을 발행하십시오.

```
db2set DB2_USE_JDK12=
```

Linux

DB2 JDBC 지원을 사용하여 Linux에 Java 응용프로그램을 빌드하려면 사용자의 개발 머신에서 다음을 설치하고 구성해야 합니다.

1. 다음 중 하나를 뜻합니다.
   - Linux용 IBM Developer Kit 및 런타임 환경 버전 1.1.8(JDK 빌드 날짜는 1999년 11월 24일이나 이후여야 합니다.)
   - Linux용 IBM Developer Kit 및 런타임 환경 버전 1.2
   - Linux용 IBM Developer Kit 및 런타임 환경 버전 1.3

   (http://www.ibm.com/software/data/db2/java 사이트를 참조하십시오.)
2. Linux용 DB2 Universal Database 버전 7 클라이언트 및 서버에서 제공되는 DB2 Java 지원.

DB2 Java 저장 프로시듀어 또는 UDF를 수행하려면 해당 머신에서 JDK가 설치된 경로를 포함하도록 서버에서 DB2 데이터베이스 관리자 구성을 갱신해야 합니다. 서버 명령행에서 다음 입력하여 이를 수행할 수 있습니다.

```
db2 update dbm cfg using JDK11_PATH /usr/local/jdk118
```

여기서 /usr/local/jdk118은 JDK가 설치된 경로입니다.

Java 저장 프로시듀어 및 UDF와 함께 JDBC 2.0 드라이버를 사용하려면 다음을 입력하십시오.

```
db2 update dbm cfg using JDK11_PATH /home/db2inst/jdk12
```

여기서 /home/db2inst/jdk12는 JDK 1.2가 설치된 경로입니다.

주: JDK11_PATH는 사실 임의 레벨의 JDK 위치를 표시하는 데 사용됩니다. JDK가 1.1인지 아니면 1.2인지의 여부는 DB2 레지스트리 변수 DB2_USE_JDK12의 설정에 의해 제어됩니다.

서버에서 다음 명령을 입력하여 JDK11_PATH 필드에 대한 정확한 값을 검증하기 위해 DB2 데이터베이스 관리자 구성을 점검할 수 있습니다.

```
db2 get dbm cfg
```

보기 쉽도록 출력을 파일로 경로 재지정할 수 있습니다. JDK11_PATH 필드는 출력의 시작 부분에 표시됩니다. 이들 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

Java 저장 프로시듀어나 사용자 정의 함수(UDF)를 수행하려면 Linux 런타임 링커가 특정 Java 공유 라이브러리에 액세스할 수 있어야 합니다. Java 공유 라이브러리의 위치를 /etc/ld.so.conf에 추가하거나 /usr/lib 디렉토리에서 라이브러리에 대한 기호 링크를 작성할 수 있습니다.

Java 공유 라이브러리 위치를 /etc/ld.so.conf에 추가하려면 루트로 다음 명령을 수행하여 런타임 링커 캐쉬를 새로 고쳐야 합니다.

```
bash# ldconfig
```

/usr/lib에서 라이브러리에 대한 기호 링크를 작성할 경우, 링크할 라이브러리 목록이 Java용 IBM Developer Kit 버전마다 다릅니다.

Java용 IBM Developer Kit 버전 1.1.8의 경우, 다음을 가리키는 기호 링크가 필요합니다.

```
libjava.so
libjitc.so
libmath.so
libzip.so
```

Java용 IBM Developer Kit 버전 1.2 또는 1.3의 경우, 다음을 가리키는 기호 링크가 필요합니다.

```
libjava.so
libjvm.so
libhpi.so
```

DB2 JDBC 지원을 사용하여 Linux에서 JDBC 및 SQLJ 프로그램을 수행하기 위해 Linux Java 환경을 갱신하는 명령이 데이터베이스 관리자 db2profile 및 db2cshrc에 포함됩니다. DB2 인스턴스가 작성되면 .bashrc, .profile 및/또는 .cshrc가 다음과 같이 수정됩니다.

1. THREADS_FLAG가 "원시(native)"로 설정됩니다.
2. CLASSPATH에는 다음 사항들이 포함됩니다.
   - "." (현재 디렉토리)
   - 파일 sqllib/java/db2java.zip

SQLJ 프로그램을 빌드하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/sqlj.zip
```

SQLJ 프로그램을 수행하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/runtime.zip
```

Java 응용프로그램과 JDBC 2.0 드라이버 사용

JDBC 1.2 드라이버는 JDBC 1.2 드라이버가 사용 가능한 모든 운영 체제에서 아직도 기본 드라이버로 사용됩니다. JDBC 2.0의 새 기능을 이용하려면 JDK 1.2 지원을 설치해야 합니다. JDBC 2.0의 새 기능을 이용하는 응용프로그램을 실행하기 전에 sqllib/java12 디렉토리에서 usejdbc2 명령을 발행하여 환경을 설정해야 합니다. 응용프로그램이 항상 JDBC 2.0 드라이버를 사용하도록 하려면 다음을 수행하도록 하십시오.

- bash 또는 korn 쉘의 경우, .profile과 같은 로그인 프로파일이나 .bashrc 또는 .kshrc와 같은 쉘 초기화 스크립트에 다음 행을 추가하십시오.

  ```
  . sqllib/java12/usejdbc2
  ```

- C 쉘의 경우, 다음 행을 .cshrc 스크립트에 추가하십시오.

  ```
  source sqllib/java12/usejdbc2.csh
  ```

db2profile을 수행하려면 이 명령이 해당 명령 다음에 위치하는지 확인하십시오. usejdbc2 및 usejdbc2.csh 스크립트가 db2profile 환경 설정을 사용하기 때문입니다.

JDBC 1.2 드라이버로 다시 전환하려면 sqllib/java12 디렉토리에서 다음 명령을 실행하십시오.

- bash 또는 korn 쉘의 경우:

```
. usejdbc1
```

- C 쉘의 경우:

```
source usejdbc1.csh
```

Java 저장 프로시듀어 및 UDF와 JDBC 2.0 드라이버 사용

Java 저장 프로시듀어 및 UDF와 함께 JDBC 2.0 드라이버를 사용하려면 인스턴스의 분리(fenced) 사용자 ID에 대해 환경을 설정해야 합니다. 기본 분리(fenced) 사용자 ID는 db2fenc1입니다. 분리 사용자 ID에 대해 환경을 설정하려면 CLP에서 다음 명령을 발행하십시오.

```
db2set DB2_USE_JDK12=1
```

JDBC 1.2 드라이버 지원으로 다시 전환하려면 CLP에서 다음 명령을 발행하십시오.

```
db2set DB2_USE_JDK12=
```

OS/2

DB2 JDBC 지원을 사용하여 OS/2에 Java 응용프로그램을 빌드하려면 사용자의 개발 머신에서 다음을 설치하고 구성해야 합니다.

1. IBM의 JDK 버전 1.1.8 및 OS/2용 JRE 버전 1.1.8. JDK 빌드 날짜는 1999년 11월 24일이나 이후여야 합니다(http://www.ibm.com/software/data/db2/java 사이트 참조).

   주: 일부 메시지는 09/99 이전에 발표된 JDK 1.1.8 버전을 수행 중인 OS/2에서 표시되지 않습니다. 최신 JDK 버전 1.1.8을 가지고 있는지 확인하십시오.

2. OS/2용 DB2 Universal Database 버전 7 클라이언트 및 서버에서 제공되는 DB2 Java 지원.

.java와 같은 네 자 이상의 확장자를 갖는 긴 파일 이름을 허용하려면 HPFS 드라이브에 JDK를 설치해야 합니다. 또한 Java 작업 디렉토리가 HPFS 드라이브에 있어야 합니다. OS/2 서버에서 Java 저장 프로시듀어 또는 UDF를 수행하려면 저장 프로시듀어 또는 UDF .class 파일을 sqllib\function 디렉토리에 배치할 수 있도록 서버의 HPFS 드라이브에 DB2를 설치해야 합니다.

DB2 Java 저장 프로시듀어 또는 UDF를 수행하려면 해당 머신에서 JDK가 설치된 경로를 포함하도록 서버에서 DB2 데이터베이스 관리자 구성을 갱신해야 합니다. 서버 명령행에서 다음을 입력하여 이를 수행할 수 있습니다.

```
db2 update dbm cfg using JDK11_PATH c:\jdk11
```

여기서 c:\jdk11은 JDK가 설치된 경로입니다.

서버에서 다음 명령을 입력하여 JDK11_PATH 필드에 대한 정확한 값을 검증하기 위해 DB2 데이터베이스 관리자 구성을 점검할 수 있습니다.

```
db2 get dbm cfg
```

보기 쉽도록 출력을 파일로 경로 재지정할 수 있습니다. JDK11_PATH 필드는 출력의 시작 부분에 표시됩니다. 이들 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

DB2 JDBC 지원을 사용하여 OS/2에서 JDBC 및 SQLJ 프로그램을 수행하기 위해 DB2를 설치할 때 다음을 포함하도록 CLASSPATH 환경 변수가 자동으로 갱신됩니다.

- "." (현재 디렉토리)
- 파일 sqllib\java\db2java.zip

SQLJ 프로그램을 빌드하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib\java\sqlj.zip
```

SQLJ 프로그램을 수행하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib\java\runtime.zip
```

PTX

DB2 JDBC 지원을 사용하여 PTX에 Java 응용프로그램을 빌드하려면 사용자의 개발 머신에서 다음을 설치하고 구성해야 합니다.

1. Sun Microsystem JDK 1.2와 동급인 ptx/JSE 버전 3.0 (http://www.ibm.com/software/data/db2/java 사이트 참조).
2. NUMA-Q용 DB2 Universal Database 버전 7에서 제공되는 DB2 Java 지원.

DB2 Java 저장 프로시듀어 또는 UDF를 수행하려면 해당 머신에서 ptx/JSE가 설치된 경로를 포함하도록 서버에서 DB2 데이터베이스 관리자 구성을 갱신해야 합니다. 서버 명령행에서 다음을 입력하여 이를 수행할 수 있습니다.

```
db2 update dbm cfg using JDK11_PATH /opt/jse3.0
```

여기서 /opt/jse3.0은 ptx/JSE가 설치된 경로입니다.

서버에서 다음 명령을 입력하여 JDK11_PATH 필드에 대한 정확한 값을 검증하기 위해 DB2 데이터베이스 관리자 구성을 점검할 수 있습니다.

```
db2 get dbm cfg
```

보기 쉽도록 출력을 파일로 경로 재지정할 수 있습니다. JDK11_PATH 필드는 출력 시작 부분에 표시됩니다. 이들 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

DB2 JDBC 지원을 사용하여 JDBC 및 SQLJ 프로그램을 실행하기 위해 PTX Java 환경을 갱신하는 명령이 데이터베이스 관리자 파일 db2profile 및 db2cshrc에 포함됩니다. DB2 인스턴스가 작성되면 CLASSPATH에 다음이 포함되도록 .profile 및 .cshrc가 수정됩니다.

- ″.″ (현재 디렉토리)
- 파일 sqllib/java/db2java.zip

SQLJ 프로그램을 빌드하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/sqlj.zip
```

SQLJ 프로그램을 수행하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/runtime.zip
```

Silicon Graphics IRIX

DB2 JDBC 지원을 사용하여 Silicon Graphics IRIX에 Java 응용프로그램을 빌드하려면 사용자의 개발 머신에서 다음을 설치하고 구성해야 합니다.

1. Silicon Graphics, Inc.의 Java2 Software Development Kit 버전 1.2(JDK 1.2) (http://www.ibm.com/software/data/db2/java 사이트를 참조하십시오.)
2. Silicon Graphics IRIX용 DB2 Universal Database 버전 7 클라이언트에서 제공되는 DB2 Java 지원.

주: Silicon Graphics IRIX의 SQLJ 응용프로그램은 o32 오브젝트 유형을 사용해야만 빌드할 수 있습니다. Java 기본 오브젝트 유형을 o32로 변경하려면 다음 명령을 사용하여 Korn 쉘로 환경 변수 SGI_ABI를 설정하거나

```
export SGI_ABI=-o32
```

다음 명령을 사용하여 C 쉘로 환경 변수 SGI_ABI를 설정하십시오.

```
setenv SGI_ABI -o32
```

o32 오브젝트 유형으로 SQLJ 응용프로그램을 빌드 중이고 JDK 1.2에서 Java JIT 컴파일러를 사용 중이면 SQLJ 변환기가 세그먼트화 결함으로 실패할 경우에 다음 명령으로 JIT 컴파일러를 꺼야 합니다.

```
export JAVA_COMPILER=NONE
```

Silicon Graphics IRIX에 Java SQLJ 프로그램을 빌드하려면 JDK 1.2가 필요합니다.

Silicon Graphics IRIX용 DB2는 클라이언트 전용입니다. DB2 응용프로그램 및 애플릿을 수행하고 DB2 Embedded SQL 응용프로그램 및 애플릿을 빌드하려면 클라이언트 머신에서 서버 머신의 DB2 데이터베이스에 액세스해야 합니다. 이 서버 머신에서는 다른 운영 체제를 수행 중입니다. 클라이언트에서 서버로의 통신 구성에 대한 자세한 내용은 *UNIX용 DB2 빠른 시작*을 참조하십시오.

또한 다른 운영 체제에서 수행하는 원격 클라이언트에서 서버의 데이터베이스에 액세스한 이후에는 DB2 CLI를 포함하여 데이터베이스 유틸리티를 데이터베이스에 바인드해야 합니다. 자세한 내용은 51 페이지의 『바인딩』을 참조하십시오.

DB2 Java 저장 프로시듀어 또는 UDF를 수행하려면 해당 머신에서 JDK가 설치된 경로를 포함하도록 서버에서 DB2 데이터베이스 관리자 구성을 갱신해야 합니다. 서버 명령행에서 다음을 입력하여 이를 수행할 수 있습니다.

```
db2 update dbm cfg using JDK11_PATH /home/db2inst/jdk11
```

여기서 `/home/db2inst/jdk11`은 JDK가 설치된 경로입니다.

서버에서 다음 명령을 입력하여 `JDK11_PATH` 필드에 대한 정확한 값을 검증하기 위해 DB2 데이터베이스 관리자 구성을 점검할 수 있습니다.

```
db2 get dbm cfg
```

보기 쉽도록 출력을 파일로 경로 재지정할 수 있습니다. `JDK11_PATH` 필드는 출력의 시작 부분에 표시됩니다. 이들 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

DB2 JDBC 지원을 사용하여 Silicon Graphics IRIX에서 JDBC 및 SQLJ 프로그램을 수행하기 위해 Silicon Graphics IRIX Java 환경을 갱신하는 명령이 데이터베이스 관리자 파일 `db2profile` 및 `db2cshrc`에 포함됩니다. DB2 인스턴스가 작성되면 CLASSPATH에 다음이 포함되도록 `.profile` 및 `.cshrc`가 수정됩니다.

- "." (현재 디렉토리)
- 파일 `sqllib/java/db2java.zip`

SQLJ 프로그램을 빌드하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/sqlj.zip
```

SQLJ 프로그램을 수행하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/runtime.zip
```

주: Silicon Graphics IRIX에서 연결 문맥 close() 메소드가 트랩을 일으킬 수도 있습니다. 일시적인 해결책으로, 연결 문맥을 가비지 콜렉션 중에 자동으로 닫히게 두십시오.

Solaris

DB2 JDBC 지원을 사용하여 Solaris 작동 환경에 Java 응용프로그램을 빌드하려면 사용자의 개발 머신에서 다음을 설치하고 구성해야 합니다.

1. Sun Microsystems의 Solaris용 JDK 버전 1.1.8 또는 1.2(http://www.ibm.com/software/data/db2/java 사이트 참조).
2. Solaris용 DB2 Universal Database 버전 7 클라이언트 및 서버에서 제공되는 DB2 Java 지원.

DB2 Java 저장 프로시듀어 또는 UDF를 수행하려면 해당 머신에서 JDK가 설치된 경로를 포함하도록 서버에서 DB2 데이터베이스 관리자 구성을 갱신해야 합니다. 서버 명령행에서 다음을 입력하여 이를 수행할 수 있습니다.

```
db2 update dbm cfg using JDK11_PATH /usr/java
```

여기서 /usr/java는 JDK가 설치된 경로입니다.

Java 저장 프로시듀어 및 UDF와 함께 JDBC 2.0 드라이버를 사용하려면 다음을 입력하십시오.

```
db2 update dbm cfg using JDK11_PATH /home/db2inst/jdk12
```

여기서 /home/db2inst/jdk12는 JDK 1.2가 설치된 경로입니다.

주: JDK11_PATH는 사실 임의 레벨의 JDK 위치를 표시하는 데 사용됩니다. JDK가 1.1인지 아니면 1.2인지의 여부는 DB2 레지스트리 변수 DB2_USE_JDK12의 설정에 의해 제어됩니다.

서버에서 다음 명령을 입력하여 JDK11_PATH 필드에 대한 정확한 값을 검증하기 위해 DB2 데이터베이스 관리자 구성을 점검할 수 있습니다.

```
db2 get dbm cfg
```

보기 쉽도록 출력을 파일로 경로 재지정할 수 있습니다. JDK11_PATH 필드는 출력의 시작 부분에 표시됩니다. 이들 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

주: Solaris 운영 환경에서 일부 JVM 구현은 "setuid" 환경에서 수행하는 프로그램에서 잘 작동하지 않습니다. 분리(fenced) 저장 프로시듀어 및 사용자 정의 함수(UDF) 프로세스 db2dari 및 db2udf는 보통 보안을 위해 "setuid nobody"로 설치됩니다. 이는 분리 Java 사용자 정의 함수(UDF) 및 저장 프로시듀어가 시작하는 데 실패하여 SQLCODE -4300을 야기할 수 있음을 의미합니다.

일시적인 해결책은 sqllib/adm에서 db2dari 및 db2udf에 대한 사용권한을 변경하는 것입니다. 다음 명령을 입력하면 JVM 버전이 분리 UDF와 저장 프로시듀어에 대해 작동하게 됩니다.

```
chmod 0555 db2dari db2udf
```

Java 인터프리터에 대한 서버측 동적 로드에 추가 문제점이 있습니다. jdk11_path 구성 매개변수가 정확히 설정되어 있을지라도 Java 인터프리터를 포함하는 공유 라이브러리 libjava.so의 로드에 실패할 수도 있습니다. SQLCODE -4300 또는 SQLCODE -4301 오류 다음에 db2diag.log 파일의 메시지가 이를 표시할 수 있습니다. 이러한 문제점으로 "setuid" 프로그램에서 dlopen 함수를 사용하는 공유 라이브러리의 런타임 로드에 대해 Solaris 보안 제한사항이 발생합니다. 일시적인 해결책으로, 다음과 유사한 명령을 사용하여 /usr/lib에 JDK 공유 라이브러리에 필요한 모든 기호 링크를 작성할 수 있습니다(머신에서 Java가 설치된 위치에 따라).

```
ln -s /usr/java/lib/*.so /usr/lib
```

DB2 JDBC 지원을 사용하여 Solaris 운영 환경에서 JDBC 및 SQLJ 프로그램을 수행하기 위해 Solaris Java 환경을 갱신하는 명령이 데이터베이스 관리자 파일

db2profile 및 db2cshrc에 포함됩니다. DB2 인스턴스가 작성되면 .profile 및/또는 .cshrc가 다음과 같이 수정됩니다.

1. THREADS_FLAG가 "원시(native)"로 설정됩니다.
2. CLASSPATH에는 다음 사항들이 포함됩니다.
   - "." (현재 디렉토리)
   - 파일 sqllib/java/db2java.zip

SQLJ 프로그램을 빌드하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/sqlj.zip
```

SQLJ 프로그램을 수행하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib/java/runtime.zip
```

Java 응용프로그램과 JDBC 2.0 드라이버 사용

JDBC 1.2 드라이버는 JDBC 1.2 드라이버가 사용 가능한 모든 운영 체제에서 아직도 기본 드라이버로 사용됩니다. JDBC 2.0의 새 기능을 이용하려면 JDK 1.2 지원을 설치해야 합니다. JDBC 2.0의 새 기능을 이용하는 응용프로그램을 실행하기 전에 sqllib/java12 디렉토리에서 usejdbc2 명령을 발행하여 환경을 설정해야 합니다. 응용프로그램이 항상 JDBC 2.0 드라이버를 사용하도록 하려면 다음을 수행하도록 하십시오.

- bash 또는 korn 쉘의 경우, .profile과 같은 로그인 프로파일이나 .bashrc 또는 .kshrc와 같은 쉘 초기화 스크립트에 다음 행을 추가하십시오.

  ```
  . sqllib/java12/usejdbc2
  ```

- C 쉘의 경우, 다음 행을 .cshrc 스크립트에 추가하십시오.

  ```
  source sqllib/java12/usejdbc2.csh
  ```

db2profile을 수행하려면 이 명령이 해당 명령 다음에 위치하는지 확인하십시오. usejdbc2 및 usejdbc2.csh 스크립트가 db2profile 환경 설정을 사용하기 때문입니다.

JDBC 1.2 드라이버로 다시 전환하려면 sqllib/java12 디렉토리에서 다음 명령을 실행하십시오.

- bash 또는 korn 쉘의 경우:

```
. usejdbc1
```

- C 쉘의 경우:

```
source usejdbc1.csh
```

Java 저장 프로시듀어 및 UDF와 JDBC 2.0 드라이버 사용

Java 저장 프로시듀어 및 UDF와 함께 JDBC 2.0 드라이버를 사용하려면 인스턴스의 분리(fenced) 사용자 ID에 대해 환경을 설정해야 합니다. 기본 분리(fenced) 사용자 ID는 db2fenc1입니다. 분리 사용자 ID에 대해 환경을 설정하려면 CLP에서 다음 명령을 발행하십시오.

```
db2set DB2_USE_JDK12=1
```

JDBC 1.2 드라이버 지원으로 다시 전환하려면 CLP에서 다음 명령을 발행하십시오.

```
db2set DB2_USE_JDK12=
```

Windows 32비트 운영 체제

DB2 JDBC 지원을 사용하여 Windows 32비트 플랫폼에 Java 응용프로그램을 빌드하려면 사용자의 개발 머신에서 다음을 설치하고 구성해야 합니다.

1. 다음 중 하나를 뜻합니다.
   - IBM의 Win32용 JDK 버전 1.1.8 및 JRE 버전 1.1.8(선택적으로 DB2와 함께 설치). JDK 빌드 날짜는 1999년 11월 24일이나 이후여야 합니다.
   - Sun Microsystems의 Win32용 JDK 1.2
   - IBM의 Win32용 JDK 1.2. JDK 빌드 날짜는 2000년 4월 17일이나 이후여야 합니다.
   - Java용 Microsoft Software Developer's Kit 버전 3.1

   (http://www.ibm.com/software/data/db2/java 사이트를 참조하십시오.)
2. Windows 32비트 운영 체제용 DB2 Universal Database 버전 7 클라이언트 및 서버에서 제공되는 DB2 Java 지원.

DB2 Java 저장 프로시듀어 또는 UDF를 수행하려면 해당 머신에서 JDK가 설치된 경로를 포함하도록 서버에서 DB2 데이터베이스 관리자 구성을 갱신해야 합니다. 서버 명령행에서 다음을 입력하여 이를 수행할 수 있습니다.

```
db2 update dbm cfg using JDK11_PATH c:\jdk11
```

여기서 c:\jdk11은 JDK가 설치된 경로입니다.

Java 저장 프로시듀어 및 UDF와 함께 JDBC 2.0 드라이버를 사용하려면 다음을 입력하십시오.

```
db2 update dbm cfg using JDK11_PATH c:\jdk12
```

여기서 c:\jdk12는 JDK 1.2가 설치된 경로입니다.

주: JDK11_PATH는 사실 임의 레벨의 JDK 위치를 표시하는 데 사용됩니다. JDK가 1.1인지 아니면 1.2인지의 여부는 DB2 레지스트리 변수 DB2_USE_JDK12의 설정에 의해 제어됩니다.

주: JDK가 설치된 경로에 하나 이상의 공간이 있는 디렉토리 이름이 들어 있으면 경로를 작은따옴표로 묶으십시오. 예를 들면, 다음과 같습니다.

```
db2 update dbm cfg using JDK11_PATH 'c:\Program Files\jdk11'
```

서버에서 다음 명령을 입력하여 JDK11_PATH 필드에 대한 정확한 값을 검증하기 위해 DB2 데이터베이스 관리자 구성을 점검할 수 있습니다.

```
db2 get dbm cfg
```

보기 쉽도록 출력을 파일로 경로 재지정할 수 있습니다. JDK11_PATH 필드는 출력의 시작 부분에 표시됩니다. 이들 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

DB2 JDBC 지원을 사용하여 지원되는 Windows 플랫폼에서 JDBC 및 SQLJ 프로그램을 수행하기 위해 DB2를 설치할 때 다음을 포함하도록 CLASSPATH 환경 변수가 자동으로 갱신됩니다.

- "." (현재 디렉토리)
- 파일 sqllib\java\db2java.zip

SQLJ 프로그램을 빌드하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib\java\sqlj.zip
```

SQLJ 프로그램을 수행하기 위해 다음 파일을 포함하도록 CLASSPATH 또한 갱신됩니다.

```
sqllib\java\runtime.zip
```

DB2 SQLJ에 사용할 JDK를 지정하기 위해 DB2는 Windows 32비트 운영 체제에 환경 변수 DB2JVIEW를 설치합니다. 모든 DB2 SQLJ 명령에 적용됩니다 (db2profc, db2profp, profdb, profp 및 sqlj).

기본 설정 ″DB2JVIEW=0″을 사용하거나 DB2JVIEW를 설정하지 않은 경우, Sun JDK가 사용됩니다. 즉, ″profp″를 호출하는 경우, ″java sqlj.runtime.profile.util.ProfilePrinter″로 수행됩니다. ″DB2JVIEW=1″인 경우, Java용 Microsoft SDK가 사용됩니다. 즉, ″profp″를 호출하는 경우, ″jview sqlj.runtime.profile.util.ProfilePrinter″로 수행됩니다.

Java 응용프로그램과 JDBC 2.0 드라이버 사용

JDBC 1.2 드라이버는 Windows 32비트를 포함한 JDBC 1.2 드라이버가 사용 가능한 모든 운영 체제에서 아직도 기본 드라이버로 사용됩니다. JDBC 2.0의 새 기능을 이용하려면 플랫폼에 대한 JDBC 2.0 드라이버 및 JDK 1.2 지원을 모두 설치해야 합니다. Windows 32비트 운영 체제용 JDBC 2.0 드라이버를 설치하려면 `sqllib\java12` 디렉토리에서 `usejdbc2` 명령을 입력하십시오. 이 명령을 실행하면 다음 타스크가 수행됩니다.

- 1.22 드라이버 파일에 대한 `sqllib\java11` 디렉토리 작성
- JDBC 1.2 드라이버 파일을 `sqllib\java11` 디렉토리로 백업
- JDBC 2.0 드라이버 파일을 `sqllib\java12` 디렉토리에서 적절한 디렉토리로 복사

JDBC 1.2 드라이버로 다시 전환하려면 `sqllib\java12` 디렉토리에서 `usejdbc1` 배치 파일을 실행하십시오.

usejdbc1 또는 usejdbc2를 실행하기 전에 다음 서비스를 중지하십시오.

- DB2 JDBC 애플릿 서버
- DB2 JDBC 애플릿 서버 - 제어 센터
- IBM WS AdminServer(IBM WebSphere Application Server에서만 사용 가능)

JDBC 드라이버 버전을 정상적으로 설정하려면 모든 파일이 오류 없이 sqllib\java 및 sqllib\bin 디렉토리에 복사되는지 확인해야 합니다. usejdbc1 또는 usejdbc2 명령을 발행한 후에 다음 메시지 중 하나가 발생할 경우,

```
"액세스가 거부됩니다."
```

또는

```
"다른 프로세스에서 파일을 사용하고 있으므로 프로세스가 그 파일에
액세스할 수 없습니다."
```

이는 위의 서비스 중 하나 이상이 계속 수행되고 있기 때문일 것입니다. Windows 서비스 창으로 이동하여 서비스가 시작되었으면 이를 중지한 다음 usejdbc1 또는 usejdbc2 명령을 다시 수행하도록 하십시오.

계속 오류가 발생할 경우, db2stop force 명령을 발행하여 재시도하십시오. 이러한 시도가 작동되지 않으면 시스템을 재부트하여 위에 언급된 프로세스가 중지되도록 한 다음 usejdbc1 또는 usejdbc2 명령을 다시 수행하십시오.

Java 저장 프로시듀어 및 UDF와 JDBC 2.0 드라이버 사용

Java 저장 프로시듀어 및 UDF와 함께 JDBC 2.0 드라이버를 사용하려면 다음 단계를 수행하여 환경을 설정해야 합니다.

1. sqllib\java12 디렉토리에서 다음 명령을 발행하십시오.

   ```
   usejdbc2
   ```

2. CLP에서 다음 명령을 발행하십시오.

   ```
   db2set DB2_USE_JDK12=1
   ```

Java UDF 및 저장 프로시듀어에 대한 JDBC 1.2 드라이버 지원으로 다시 전환하려면 다음 단계를 수행하십시오.

1. sqllib\java12 디렉토리에서 다음 명령을 발행하십시오.

   ```
   usejdbc1
   ```

2. CLP에서 다음 명령을 발행하십시오.

   ```
   db2set DB2_USE_JDK12=
   ```

IBM JDK 환경 설정을 위한 배치 파일 예

IBM Java Development Kit을 사용하여 Java 환경을 설정하려면 다음 명령을 배치 파일에 넣으면 됩니다. 사용자의 특정 환경에 맞게 필요한 모든 경로 변경을 수행하였는지 확인하십시오. 지원되는 다른 JDK에 대해 유사한 명령을 사용할 수 있습니다.

다음은 IBM JDK 1.1.8 환경을 설정하기 위한 배치 파일 예의 명령입니다.

```
set JDKPATH=D:\JAVA\IBMjdk118
set PATH=%JDKPATH%\bin;%PATH%
set CLASSPATH=.;%JDKPATH%\lib\classes.zip;%CLASSPATH%
db2 update dbm cfg using JDK11_PATH %JDKPATH%
db2set DB2_USE_JDK12=0
db2 terminate
db2stop
db2start
```

다음은 IBM JDK 1.2 환경을 설정하기 위한 배치 파일 예의 명령입니다. set CLASSPATH 명령은 단지 JDK 1.1.8이 설정된 경우에 이를 설정 해제하기 위한 것입니다. 이전에 JDK 1.1.8을 설정하지 않았으면 이 명령은 효과가 없습니다.

```
set JDKPATH=D:\JAVA\IBMjdk122
set PATH=%JDKPATH%\bin;%PATH%
set CLASSPATH=%CLASSPATH:D:\JAVA\IBMjdk118\lib\classes.zip;=%
db2 update dbm cfg using JDK11_PATH %JDKPATH%
db2set DB2_USE_JDK12=1
db2 terminate
db2stop
db2start
```

Java 샘플 프로그램

DB2는 동적 SQL만을 사용하는 JDBC 프로그램과 정적 SQL을 사용하는 SQLJ 프로그램의 빌드 및 수행을 보여주기 위해 다음 절에서 사용되는 샘플 프로그램을 제공합니다. UNIX 플랫폼에서 Java 샘플은 sqllib/samples/java에 있습니다. OS/2 및 Windows 32비트 운영 체제에서 샘플은 sqllib\samples\java에 있습니다. 또한 샘플 디렉토리는 README, makefile 및 빌드 파일을 포함합니다.

Java makefile은 제공된 모든 샘플 프로그램을 빌드합니다. 호환 가능한 make 실행 가능 프로그램이 필요하며, 일반적으로 JDK에서 제공하지 않습니다. 자세한 내용은 makefile 텍스트의 시작 부분에 있는 주석을 참조하십시오. 일부 Java makefile 명령은 다른 언어와 다릅니다. make clean 명령은 .class 파일과 같이 Java 컴파일러에서 생성한 모든 파일을 제거합니다. make cleanall 명령은 SQLJ 변환기에서 생성한 모든 파일과 함께 이들 파일을 제거합니다.

JDBC 프로그램은 상대적으로 명령행에서 빌드하기 쉽기 때문에 이에 대한 빌드 파일이 제공되지 않습니다. 다음과 같은 두 가지 SQLJ 빌드 파일이 제공됩니다. bldsqlj는 SQLJ 애플릿 및 응용프로그램을 빌드합니다. bldsqljs는 SQLJ 저장 프로시듀어를 빌드합니다. 이들 빌드 파일을 103 페이지의『SQLJ 프로그램』절에서 설명합니다.

OS/2

OS/2에서는 작업 디렉토리가 HPFS 드라이브에 있어야 합니다. 설치시 DB2 설치 프로그램이 대상 드라이브가 FAT인 것으로 검출하면 두 개의 파일 javasamp.exe와 README 파일을 sqllib\samples\java 디렉토리에 위치시킵니다. Java 샘플 프로그램을 사용하려면 javasamp.exe를 HPFS 작업 디렉토리로 이동한 다음 이 실행 가능 프로그램을 수행하십시오. 그러면 Java 샘플 프로그램이 디렉토리로 추출됩니다. 설치 프로그램이 대상 드라이브가 HPFS인 것으로 검출하면 설치 프로세스 중에 사용하여 Java 샘플 프로그램을 sqllib\samples\java 디렉토리로 추출합니다.

JDBC 프로그램

애플릿

DB2Applt는 DB2 데이터베이스에 액세스하기 위해 JDBC 애플릿(또는 ″net″) 드라이버를 사용하는 동적 SQL Java 애플릿을 보여줍니다.

명령행에서 입력한 명령으로 애플릿을 빌드하고 수행하려면 다음을 수행하십시오.

1. 웹 서버가 DB2 머신(서버 또는 클라이언트)에 설치되고 수행 중인지 확인하십시오.
2. 파일의 지침에 따라 DB2Applt.html 파일을 수정하십시오.
3. DB2Applt.html에 지정된 TCP/IP 포트에서 JDBC 애플릿 서버를 시작하십시오. 예를 들어, DB2Applt.html에서 param name=port value='6789'를 지정한 경우, 다음을 입력합니다.

   ```
   db2jstrt 6789
   ```

   주: 연결 문자열에서 JDBC 포트 번호가 권장 기본값인 ″6789″인지 확인하십시오. 번호가 다른 포트 번호와 충돌하지 않는 것이 확실하면 이를 변경하십시오. 데이터베이스 포트 번호 ″50000″은 사용하지 마십시오.
4. 다음 명령을 사용하여 DB2Applt.class 파일을 생성하도록 DB2Applt.java를 컴파일하십시오.

   ```
   javac DB2Applt.java
   ```

5. 웹 브라우저에서 작업 디렉토리에 액세스할 수 있는지 확인하십시오. 그렇지 않은 경우, DB2Applt.class 및 DB2Applt.html을 액세스 가능한 디렉토리로 복사하십시오.
6. OS/2 또는 Windows 32비트 운영 체제의 sqllib\java\db2java.zip 파일이나 UNIX의 sqllib/java/db2java.zip 파일을 DB2Applt.class 및 DB2Applt.html과 동일한 디렉토리로 복사하십시오.
7. 클라이언트 머신에서 웹 브라우저를 시작하고(JDK 1.1을 지원해야 함) DB2Applt.html을 로드하십시오.

(1), (5) 및 (7)단계의 또 다른 방법으로, 클라이언트 머신의 작업 디렉토리에서 다음 명령을 입력하여 JDK에 제공되는 애플릿 표시기를 사용할 수 있습니다.

```
appletviewer DB2Applt.html
```

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

응용프로그램

DB2Appl은 DB2 데이터베이스에 액세스하기 위해 JDBC 응용프로그램(또는 "app") 드라이버를 사용하는 동적 SQL 응용프로그램을 보여줍니다.

명령행에서 입력한 명령으로 애플릿을 빌드하고 수행하려면 다음을 수행하십시오.

1. 다음 명령을 사용하여 DB2Appl.class 파일을 생성하도록 DB2Appl.java를 컴파일하십시오.

   ```
   javac DB2Appl.java
   ```

2. 다음 명령을 사용하여 응용프로그램에서 Java 인터프리터를 수행하십시오.

   ```
   java DB2Appl
   ```

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

저장 프로시듀어용 클라이언트 응용프로그램

Spclient는 JDBC 응용프로그램 드라이버를 사용하여 Java 저장 프로시듀어 클래스 Spserver를 호출하는 클라이언트 응용프로그램입니다. 이 클라이언트 응용프로그램을 빌드하고 수행하기 전에 서버에서 저장 프로시듀어 클래스를 빌드하십시오. 자세한 내용은 102 페이지의 『저장 프로시듀어』를 참조하십시오.

명령행에서 입력한 명령으로 이 클라이언트 프로그램을 빌드하고 수행하려면 다음을 수행하십시오.

1. 다음 명령을 사용하여 Spclient.java를 파일을 생성하도록 Spclient.class를 컴파일하십시오.

   ```
   javac Spclient.java
   ```

2. 다음 명령을 사용하여 클라이언트 프로그램에서 Java 인터프리터를 수행하십시오.

   ```
   java Spclient
   ```

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

사용자 정의 함수(UDF)용 클라이언트 응용프로그램

UDFcli는 JDBC 응용프로그램 드라이버를 사용하여 사용자 정의 함수(UDF) 서버 프로그램 UDFsrv에서 구현된 사용자 정의 함수(UDF)를 호출하는 클라이언트 프로그램입니다. 이 클라이언트 응용프로그램을 빌드하고 수행하기 전에, 서버에서 사용자 정의 함수(UDF) 프로그램 UDFsrv를 빌드하십시오. 자세한 내용은 113 페이지의 『사용자 정의 함수(UDF)』를 참조하십시오.

명령행에서 입력한 명령으로 이 클라이언트 프로그램을 빌드하고 수행하려면 다음을 수행하십시오.

1. 다음 명령을 사용하여 DB2SpCli.class 파일을 생성하도록 DB2SpCli.java를 컴파일하십시오.

   ```
   javac UDFcli.java
   ```

2. 다음 명령을 사용하여 클라이언트 프로그램에서 Java 인터프리터를 수행하십시오.

   ```
   java UDFcli
   ```

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

저장 프로시듀어

Spserver는 JDBC 응용프로그램 드라이버를 사용하는 동적 SQL PARAMETER STYLE JAVA 저장 프로시듀어를 보여줍니다. 저장 프로시듀어는 서버에 컴파일되고 저장됩니다. 클라이언트 응용프로그램이 호출하면 저장 프로시듀어는 서버 데이터베이스에 액세스하여 클라이언트 응용프로그램에 정보를 리턴합니다.

명령행에서 입력한 명령으로 서버에서 이 프로그램을 빌드하고 수행하려면 다음을 수행하십시오.

1. 다음 명령을 사용하여 Spserver.java 파일을 생성하도록 Spserver.class를 컴파일하십시오.

   ```
   javac Spserver.java
   ```

2. OS/2 및 Windows 32비트 운영 체제에서는 sqllib\function 디렉토리로, UNIX에서는 sqllib/function 디렉토리로 Spserver.class를 복사하십시오.

3. 다음에는 서버에서 Spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

   ```
   db2 connect to sample
   ```

   저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

   ```
   db2 -td@ -vf Spdrop.db2
   ```

   그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

   ```
   db2 -td@ -vf Spcreate.db2
   ```

4. 데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 "실행"으로 설정하십시오.
5. Spclient 클라이언트 응용프로그램을 컴파일하고 수행하여 저장 프로시듀어 클래스에 액세스하십시오. 101 페이지의 『저장 프로시듀어용 클라이언트 응용프로그램』을 참조하십시오.

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

SQLJ 프로그램

주: UNIX, OS/2 및 Windows 32비트 운영 체제용 IBM JDK를 사용하여 SQLJ 프로그램을 빌드하고 수행하려면 운영 체제에 대해 다음 명령을 사용하여 JDK의 JIT(just-in-time) 컴파일러를 꺼야 합니다.

OS/2 및 Windows:

```
SET JAVA_COMPILER=NONE
```

UNIX:

```
export JAVA_COMPILER=NONE
```

빌드 파일 bldsqlj는 SQLJ 애플릿 또는 응용프로그램을 빌드하기 위한 명령을 포함합니다. UNIX에서 이것은 스크립트 파일입니다. OS/2에서는 명령 파일 bldsqlj.cmd이며, Windows에서는 배치 파일 bldsqlj.bat입니다. 명령 및 배

치 파일의 내용은 동일하며, 이 버전이 먼저 제시되고 UNIX 스크립트 파일이 따라 나옵니다. 뒤따라 나오는 애플릿 및 응용프로그램 빌드 절은 빌드 파일을 다시 참조합니다.

주: DB2에 제공된 SQLJ 변환기는 변환된 `.java` 파일을 `.class` 파일로 컴파일합니다. 따라서 이 절의 빌드 파일은 Java 컴파일러를 사용하지 않습니다.

다음에 나오는 OS/2 및 Windows 32비트 운영 체제용 빌드 파일에서 첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하고, 매개변수 %4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 `sample` 데이터베이스를 사용합니다.

```
@echo off
rem bldsqlj -- OS/2 and Windows 32-bit operating systems
rem Builds a Java embedded SQL (SQLJ) program.
rem Usage: bldsqlj prog_name [ db_name [ userid password ]]
if "%1" == "" goto error
rem Translate and compile the SQLJ source file
rem and bind the package to the database.
if "%2" == "" goto case1
if "%3" == "" goto case2
if "%4" == "" goto error
goto case3
:case1
   sqlj %1.sqlj
   db2profc -url=jdbc:db2:sample -prepoptions="package using %1" %1_SJProfile0
   goto continue
:case2
   sqlj     -url=jdbc:db2:%2 %1.sqlj
   db2profc -url=jdbc:db2:%2 -prepoptions="package using %1" %1_SJProfile0
   goto continue
:case3
   sqlj     -url=jdbc:db2:%2 -user=%3 -password=%4  %1.sqlj
   db2profc -url=jdbc:db2:%2 -user=%3 -password=%4 -prepoptions="package using %1"
     %1_SJProfile0
   goto continue
:continue
goto exit
:error
echo Usage: bldsqlj prog_name [ db_name [ userid password ]]
:exit
@echo on
```

bldsqlj에 대한 번환기 및 사전 처리 컴파일 옵션
sqlj SQLJ 변환기(프로그램도 컴파일함).
%1.sqlj SQLJ 소스 파일.
%1.java SQLJ 소스 파일에서 변환된 Java 파일.
db2profc Java용 DB2 프로파일 조정자.
-url jdbc:db2:sample과 같이 데이터베이스 연결을 설정하기 위한 JDBC URL을 지정합니다.
-user 사용자 ID를 지정합니다(선택적 매개변수).
-password 암호를 지정합니다(선택적 매개변수).
-prepoptions "%1을 사용하는 패키지" 문자열과 함께 데이터베이스에 대한 패키지 이름을 지정합니다. 여기서 %1은 SQLJ의 소스 파일 이름입니다.
%1_SJProfile0 프로그램의 직렬화된 프로파일을 지정합니다.

다음에 나오는 UNIX 스크립트 파일에서 첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고, 매개변수 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldsqlj script file -- UNIX platforms
# Builds a Java embedded SQL (SQLJ) sample
# Usage: bldsqlj <prog_name> [ <db_name> [ <userid> <password> ]]
# Translate and compile the SQLJ source file
# and bind the package to the database.
if (($# < 2))
then
   sqlj $1.sqlj
   db2profc -url=jdbc:db2:sample -prepoptions="package using $1" $1_SJProfile0
```

```
elif (($# < 3))
then
   sqlj     -url=jdbc:db2:$2 $1.sqlj
   db2profc -url=jdbc:db2:$2 -prepoptions="package using $1" $1_SJProfile0
else
   sqlj     -url=jdbc:db2:$2 -user=$3 -password=$4  $1.sqlj
   db2profc -url=jdbc:db2:$2 -user=$3 -password=$4 -prepoptions="package using $1"
     $1_SJProfile0
fi
```

bldsqlj에 대한 번환기 및 사전 처리 컴파일 옵션

sqlj SQLJ 변환기(프로그램도 컴파일함).

$1.sqlj
SQLJ 소스 파일.

$1.java
SQLJ 소스 파일에서 변환된 Java 파일.

db2profc
Java용 DB2 프로파일 조정자.

-url jdbc:db2:sample과 같은 데이터베이스 연결을 설정하기 위한 JDBC URL을 지정합니다.

-user 사용자 ID를 지정합니다(선택적 매개변수).

-password
암호를 지정합니다(선택적 매개변수).

-prepoptions
"$1을 사용하는 패키지" 문자열과 함께 데이터베이스에 대한 패키지 이름을 지정합니다. 여기서 $1은 SQLJ의 소스 파일 이름입니다.

$1_SJProfile0
프로그램의 직렬화된 프로파일을 지정합니다.

애플릿

Applt는 DB2 데이터베이스에 액세스하는 SQLJ 애플릿을 보여줍니다.

빌드 파일 bldsqlj를 사용하여 애플릿을 빌드하고 수행하려면 다음을 수행하십시오.

1. 웹 서버가 DB2 머신(서버 또는 클라이언트)에 설치되고 수행 중인지 확인하십시오.
2. 파일의 지침에 따라 Applt.html 파일을 수정하십시오.
3. Applt.html에 지정된 TCP/IP 포트에서 JDBC 애플릿 서버를 시작하십시오. 예를 들어, Applt.html에서 param name=port value='6789'를 지정한 경우, 다음을 입력합니다.

   ```
   db2jstrt 6789
   ```

   주: 연결 문자열에서 JDBC 포트 번호가 권장 기본값인 "6789"인지 확인하십시오. 번호가 다른 포트 번호와 충돌하지 않는 것이 확실하면 이를 변경하십시오. 데이터베이스 포트 번호 "50000"은 사용하지 마십시오.
4. 다음 명령을 사용하여 애플릿을 빌드하십시오.

   ```
   bldsqlj Applt [ <db_name> [ <userid> <password> ]]
   ```

   여기서 선택적 매개변수 <db_name>을 사용하여 기본 sample 데이터베이스 대신 다른 데이터베이스에 액세스할 수 있습니다. 원격 클라이언트 머신에서 서버에 액세스하는 경우와 같이 액세스 중인 데이터베이스가 다른 인스턴스에 있는 경우, 선택적 매개변수 <userid>와 <password>가 필요합니다.
5. 웹 브라우저에서 작업 디렉토리에 액세스할 수 있는지 확인하십시오. 그렇지 않은 경우, 다음 파일을 액세스 가능한 디렉토리로 복사하십시오.

   ```
   Applt.html                      Applt.class
   Applt_Cursor1.class             Applt_Cursor2.class
   Applt_SJProfileKeys.class       Applt_SJProfile0.ser
   ```
6. OS/2 및 Windows 32비트 운영 체제의 sqllib\java\db2java.zip 및 sqllib\java\runtime.zip 파일이나 UNIX의 sqllib/java/db2java.zip 및 sqllib/java/runtime.zip 파일을 다른 Applt 파일과 동일한 디렉토리로 복사하십시오.
7. 클라이언트 머신에서 웹 브라우저를 시작하고(JDK 1.1을 지원해야 함) Applt.html을 로드하십시오.

(1), (5) 및 (7)단계의 또 다른 방법으로, 클라이언트 머신의 작업 디렉토리에서 다음 명령을 입력하여 JDK에 제공되는 애플릿 표시기를 사용할 수 있습니다.

```
appletviewer Applt.html
```

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

응용프로그램

App는 DB2 데이터베이스에 액세스하는 SQLJ 응용프로그램을 보여줍니다.

빌드 파일 bldsqlj를 사용하여 응용프로그램을 빌드하려면 다음 명령을 입력하십시오.

```
bldsqlj App [ <db_name> [ <userid> <password> ]]
```

여기서 선택적 매개변수 <db_name>을 사용하여 기본 sample 데이터베이스 대신 다른 데이터베이스에 액세스할 수 있습니다. 원격 클라이언트 머신에서 서버에 액세스하는 경우와 같이 액세스 중인 데이터베이스가 다른 인스턴스에 있는 경우, 선택적 매개변수 <userid>와 <password>가 필요합니다.

다음 명령을 사용하여 응용프로그램에서 Java 인터프리터를 수행하십시오.

```
java App
```

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

저장 프로시듀어용 클라이언트 프로그램

Stclient는 JDBC 응용프로그램 드라이버를 사용하여 SQLJ 저장 프로시듀어 클래스 Stserver를 호출하는 클라이언트 프로그램입니다. 이 클라이언트 응용프로그램을 빌드하고 수행하기 전에 서버에서 저장 프로시듀어 클래스를 빌드하십시오. 자세한 내용은 109 페이지의 『저장 프로시듀어』를 참조하십시오.

빌드 파일 bldsqlj를 사용하여 응용프로그램을 빌드하려면 다음 명령을 입력하십시오.

```
bldsqlj Stclient [ <db_name> [ <userid> <password> ]]
```

여기서 선택적 매개변수 <db_name>을 사용하여 기본 sample 데이터베이스 대신 다른 데이터베이스에 액세스할 수 있습니다. 원격 클라이언트 머신에서 서버에 액세스하는 경우와 같이 액세스 중인 데이터베이스가 다른 인스턴스에 있는 경우, 선택적 매개변수 <userid>와 <password>가 필요합니다.

다음 명령을 사용하여 클라이언트 응용프로그램에서 Java 인터프리터를 수행하십시오.

```
java Stclient
```

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

사용자 정의 함수(UDF)용 클라이언트 프로그램

UDFclie는 JDBC 응용프로그램 드라이버를 사용하여 서버 프로그램 UDFsrv에 구현된 사용자 정의 함수(UDF)를 호출하는 클라이언트 프로그램입니다. 이 클라이언트 응용프로그램을 빌드하고 수행하기 전에 서버에서 UDFsrv 프로그램을 빌드하십시오. 자세한 내용은 113 페이지의 『사용자 정의 함수(UDF)』를 참조하십시오.

빌드 파일 bldsqlj를 사용하여 SQLJ 클라이언트 프로그램을 빌드하려면, 다음 명령을 입력하십시오.

```
bldsqlj UDFclie [ <db_name> [ <userid> <password> ]]
```

여기서 선택적 매개변수 <db_name>을 사용하여 기본 sample 데이터베이스 대신 다른 데이터베이스에 액세스할 수 있습니다. 원격 클라이언트 머신에서 서버에 액세스하는 경우와 같이 액세스 중인 데이터베이스가 다른 인스턴스에 있는 경우, 선택적 매개변수 <userid>와 <password>가 필요합니다.

다음 명령을 사용하여 클라이언트 응용프로그램에서 Java 인터프리터를 수행하십시오.

```
java UDFclie
```

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

저장 프로시듀어

빌드 파일 bldsqljs는 SQLJ 저장 프로시듀어를 빌드하기 위한 명령을 포함합니다. UNIX에서 이것은 스크립트 파일입니다. OS/2에서는 명령 파일 bldsqljs.cmd이며, Windows에서는 배치 파일 bldsqljs.bat입니다. 명령 및 배치 파일의 내용은 동일하며, 이 버전이 먼저 제시되고 UNIX 스크립트 파일이 따라 나옵니다.

다음에 나오는 OS/2 및 Windows 32비트 운영 체제용 빌드 파일에서 첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 %2는 연결하려는

데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드되어야 하므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
@echo off
rem bldsqljs -- OS/2 and Windows 32-bit operating systems
rem Builds a Java embedded SQL (SQLJ) stored procedure
rem Usage: bldsqljs prog_name [ db_name ]
if "%1" == "" goto error
rem Translate and compile the SQLJ source file
rem and bind the package to the database.
if "%2" == "" goto case1
goto case2
:case1
   sqlj %1.sqlj
   db2profc -url=jdbc:db2:sample -prepoptions="package using %1" %1_SJProfile0
   goto continue
:case2
   sqlj     -url=jdbc:db2:%2 %1.sqlj
   db2profc -url=jdbc:db2:%2 -prepoptions="package using %1" %1_SJProfile0
:continue
rem Copy the *.class and *.ser files to the 'function' directory.
copy %1*.class "%DB2PATH%\function"
copy %1*.ser "%DB2PATH%\function"
goto exit
:error
echo Usage: bldsqljs prog_name [ db_name ]
:exit
@echo on
```

bldsqljs에 대한 번환기 및 사전 처리 컴파일 옵션
sqlj SQLJ 변환기(또한 프로그램을 컴파일합니다.)
%1.sqlj SQLJ 소스 파일.
%1.java SQLJ 소스 파일에서 변환된 Java 파일.
db2profc Java용 DB2 프로파일 조정자.
-url jdbc:db2:sample과 같은 데이터베이스 연결을 설정하기 위한 JDBC URL을 지정합니다.
-prepoptions "%1을 사용하는 패키지" 문자열과 함께 데이터베이스에 대한 패키지 이름을 지정합니다. 여기서 %1은 SQLJ의 소스 파일 이름입니다.
%1_SJProfile0 프로그램의 직렬화된 프로파일을 지정합니다.

다음에 나오는 UNIX 스크립트 파일에서 첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드되어야 하므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldsqljs script file -- UNIX platforms
# Builds a Java embedded SQL (SQLJ) stored procedure
# Usage: bldsqljs <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Translate and compile the SQLJ source file
# and bind the package to the database.
if (($# < 2))
then
   sqlj $1.sqlj
   db2profc -url=jdbc:db2:sample -prepoptions="package using $1" $1_SJProfile0
else
   sqlj     -url=jdbc:db2:$2 $1.sqlj
```

```
   db2profc -url=jdbc:db2:$2 -prepoptions="package using $1" $1_SJProfile0
fi
# Copy the *.class and *.ser files to the 'function' directory.
rm -f $DB2PATH/function/$1*.class
rm -f $DB2PATH/function/$1*.ser
cp $1*.class $DB2PATH/function
cp $1*.ser $DB2PATH/function
```

bldsqljs에 대한 번환기 및 사전 처리 컴파일 옵션
sqlj SQLJ 변환기(또한 프로그램을 컴파일합니다.) **$1.sqlj** SQLJ 소스 파일. **$1.java** SQLJ 소스 파일에서 변환된 Java 파일. **db2profc** Java용 DB2 프로파일 조정자. **-url** jdbc:db2:sample과 같은 데이터베이스 연결을 설정하기 위한 JDBC URL을 지정합니다. **-prepoptions** "$1을 사용하는 패키지" 문자열과 함께 데이터베이스에 대한 패키지 이름을 지정합니다. 여기서 $1은 SQLJ의 소스 파일 이름입니다. **$1_SJProfile0** 프로그램의 직렬화된 프로파일을 지정합니다.

Stserver는 DB2 데이터베이스에 액세스하기 위해 JDBC 응용프로그램 드라이버를 사용하여 PARAMETER STYLE JAVA 저장 프로시듀어를 보여줍니다. 저장 프로시듀어는 서버에서 컴파일되고 저장됩니다. 클라이언트 응용프로그램이 호출하면 저장 프로시듀어는 서버 데이터베이스에 액세스하여 클라이언트 응용프로그램으로 정보를 리턴합니다.

이 저장 프로시듀어 클래스를 빌드 파일 bldsqljs로 빌드하려면 다음을 수행하십시오.

1. 다음 명령을 입력하십시오.

   ```
   bldsqljs Stserver [ <db_name> ]
   ```

여기서 선택적 매개변수 <db_name>을 사용하여 기본 sample 데이터베이스 대신 다른 데이터베이스에 액세스할 수 있습니다.

2. 다음에는 서버에서 Stcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

   ```
   db2 connect to sample
   ```

   저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

   ```
   db2 -td@ -vf Stdrop.db2
   ```

   그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

   ```
   db2 -td@ -vf Stcreate.db2
   ```

3. 데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 "실행"으로 설정하십시오.
4. Stclient 클라이언트 응용프로그램을 컴파일하고 수행하여 저장 프로시듀어를 호출하십시오. 자세한 내용은 108 페이지의 『저장 프로시듀어용 클라이언트 프로그램』을 참조하십시오.

또한 Java makefile을 사용하여 이 프로그램을 빌드할 수 있습니다.

사용자 정의 함수(UDF)

UDFsrv는 Java 사용자 정의 함수(UDF)를 보여줍니다. UDF 프로그램이 SQL문을 포함하지 않으므로 Java UDF 프로그램은 SQLJ문을 포함할 수 없습니다. UDFsrv 라이브러리가 서버에 작성되면 클라이언트 응용프로그램이 액세스할 수 있습니다. DB2는 JDBC 클라이언트 응용프로그램 UDFcli와, SQLJ 클라이언트 응용프로그램 UDFclie를 둘 다 제공합니다. UDFsrv 라이브러리에 액세스하는 데 사용할 수 있습니다.

명령행에서 입력한 명령으로 서버에서 UDF 프로그램을 빌드하고 수행하려면 다음을 수행하십시오,

1. 다음 명령을 사용하여 UDFsrv.class 파일을 생성하도록 UDFsrv.java를 컴파일하십시오.

   ```
   javac UDFsrv.java
   ```

2. OS/2 및 Windows 32비트 운영 체제에서는 sqllib\function 디렉토리로, UNIX에서는 sqllib/function 디렉토리로 UDFsrv.class를 복사하십시오.
3. UDFsrv 라이브러리에 액세스하기 위해 JDBC 또는 SQLJ 클라이언트 응용프로그램을 사용할 수 있습니다. JDBC 클라이언트 응용프로그램 UDFcli를 컴파일하고 수행하려면 자세한 내용은 102 페이지의 『사용자 정의 함수(UDF)용 클라이어트 응용프로그램』을 참조하십시오. SQLJ 클라이언트 응용프로그램 UDFclie를 컴파일하고 수행하려면 자세한 내용은 109 페이지의 『사용자 정의 함수(UDF)용 클라이언트 프로그램』을 참조하십시오.

UDFcli 및 UDFclie는 데이터베이스에서 UDFsrv에 포함된 UDF를 등록하는 데 사용하는 CREATE FUNCTION SQL문을 포함합니다. 등록되면 UDFcli 및 UDFclie 또한 UDF를 사용하는 SQL문을 포함합니다.

DB2 Java 애플릿에 대한 일반적인 사항

1. Java 애플릿이 JDBC 애플릿 서버와 같은 FixPak 레벨에서 db2java.zip 파일을 사용하는 것은 중요합니다. 정상적인 상황에서 db2java.zip은 JDBC 애플릿 서버가 수행 중인 웹 서버에서 로드됩니다. 이로써 확실하게 일치하게 됩니다. 그렇지 않고 구성에 다른 위치로부터 db2java.zip을 로드하는 Java 애플릿이 있으면 불일치가 발생할 수 있습니다. 두 파일 사이의 FixPak 레벨 일치는 연결 시간에 엄밀하게 이루어집니다. 불일치가 검출되면 연결이 거부되고 클라이언트는 다음 예외를 받습니다.
   - db2java.zip이 DB2 버전 7 FixPak 2 이상일 경우:

     ```
     COM.ibm.db2.jdbc.DB2Exception: [IBM][JDBC Driver]
     CLI0621E  Unsupported JDBC server configuration.
     ```

   - db2java.zip이 FixPak 2 이전일 경우:

     ```
     COM.ibm.db2.jdbc.DB2Exception: [IBM][JDBC Driver]
     CLI0601E  Invalid statement handle or statement is closed.
     SQLSTATE=S1000
     ```

불일치가 발생하면 JDBC 애플릿 서버는 jdbcerr.log 파일에 다음 메시지 중 하나를 기록합니다.

- JDBC 애플릿 서버가 DB2 버전 7 FixPak 2 이상일 경우:

```
jdbcFSQLConnect: JDBC Applet Server and client (db2java.zip)
versions do not match. Unable to proceed with connection., einfo= -111
```

- JDBC 애플릿 서버가 FixPak 2 이전일 경우:

```
jdbcServiceConnection(): Invalid Request Received., einfo= 0
```

JDBC 환경을 테스트하기 위해 sqllib\samples\java(Windows)나 sqllib/samples/java(UNIX)에서 샘플 파일 db2JDBCVersion.java를 사용할 수 있습니다. db2JDBCVersion 프로그램은 현재 사용 중인 DB2 JDBC 드라이버 버전과 JDBC 환경이 이에 알맞게 설정되어 있는지 여부를 점검합니다. 이 프로그램은 OS/2에서 사용 불가능합니다.

2. 몇 가지 Java 클래스로 이루어진 보다 큰 JDBC 또는 SQLJ 애플릿의 경우, 모든 클래스를 단일 JAR 파일에 패키지화하도록 선택할 수도 있습니다. SQLJ 애플릿의 경우, 직렬화된 프로파일을 또한 클래스와 함께 패키지화해야 합니다. 이를 수행하도록 선택하는 경우, JAR 파일을 "applet" 태그에 있는 archive 매개변수에 추가하십시오. 자세한 내용은 JDK 버전 1.1 문서를 참조하십시오.

   SQLJ 애플릿의 경우: 일부 브라우저에는 애플릿과 연관된 자원 파일에서 직렬화된 오브젝트를 로드하기 위한 지원이 아직 없습니다. 예를 들어, 해당 브라우저에서 Applt 애플릿을 로드하려고 시도할 때 다음 오류 메시지가 표시됩니다.

   ```
   java.lang.ClassNotFoundException: Applt_SJProfile0
   ```

   일시적인 해결책으로, 직렬화된 프로파일을 Java 클래스 형식으로 저장된 프로파일로 변환하는 유틸리티가 있습니다. 유틸리티는 sqlj.runtime.profile.util.SerProfileToClass라는 Java 클래스입니다. 직렬화된 프로파일 자원 파일을 입력으로 취하여 프로파일을 출력으로 포함하는 Java 클래스를 생성합니다. 다음 명령 중 하나를 사용하여 프로파일을 변환할 수 있습니다.

   ```
   profconv Applt_SJProfile0.ser
   ```

   또는

```
java sqlj.runtime.profile.util.SerProfileToClass Applt_SJProfile0.ser
```

결과로서 Applt_SJProfile0.class 클래스가 작성됩니다. 애플릿이 사용하는 .ser 형식의 모든 프로파일을 .class 형식의 프로파일로 바꾸면, 문제가 해결됩니다.

3. db2java.zip 파일(SQLJ 애플릿의 경우, runtime.zip 파일 또한)을 웹 사이트에서 로드할 수 있는 몇 개의 애플릿을 공유하는 디렉토리에 배치할 수 있습니다. 이들 파일은 OS/2 및 Windows 32비트 운영 체제의 sqllib\java 디렉토리와 UNIX의 sqllib/java 디렉토리에 있습니다. 디렉토리를 식별하기 위해 HTML 파일의 "applet" 태그에 codebase 매개변수를 추가할 수 있습니다. 자세한 내용은 JDK 버전 1.1 문서를 참조하십시오.
4. DB2 버전 5.2 이후 JDBC 애플릿 서버(리스너) db2jd에 신호 핸들링이 추가되어 보다 강력해집니다. 그 결과 CTRL-C 명령을 사용하여 db2jd를 종료할 수 없습니다. 따라서 리스너를 종료하는 유일한 방법은 프로세스를 종료하는 것입니다.
5. 웹 서버에서(특히, Domino Go Webserver) DB2 Java 애플릿 수행에 대한 자세한 내용은 다음을 참조하십시오.

   ```
   http://www.ibm.com/software/data/db2/db2lotus/gojava.htm
   ```

제5장 SQL 프로시듀어 빌드

이 장에서는 DB2 SQL 프로시듀어 빌드에 대한 자세한 정보를 제공합니다.

DB2 SQL 프로시듀어 샘플 프로그램은 UNIX 플랫폼의 `sqllib/samples/sqlproc` 디렉토리와 OS/2 및 Windows 32비트 운영 체제의 `%DB2PATH%\samples\sqlproc` 디렉토리에 있습니다. 샘플 프로그램에 대한 자세한 내용은 32 페이지의 표10을 참조하십시오.

사용자 환경 설정 방법과 SQL 프로시듀어 작성 및 호출 방법의 예

이 장에서는 지원되는 모든 플랫폼에서 SQL 프로시듀어 환경을 설정하는 방법, SQL 프로시듀어를 작성하는 방법, 그리고 이러한 환경에서 SQL 프로시듀어를 호출하는 방법에 대해 자세히 설명합니다. 이 소개 절은 Windows NT 또는 Windows 2000 환경에서 Microsoft Visual C++ 버전 6.0 컴파일러를 사용하여 세 가지를 모두 수행하는 방법에 대한 예를 제공합니다. 이 장의 나머지 부분에 자세히 설명한 대로 적절하게 변경하여 지원되는 모든 환경에 대해 이 단계를 적용할 수 있습니다.

다음을 잘라내어 배치 파일에 붙여넣은 다음 DB2CLP 창에서 수행하여 Microsoft Visual C++ 버전 6.0 컴파일러에 대한 환경 설정 명령을 실행할 수 있습니다. 사용자의 특정 환경에 맞게 경로 설정을 포함하여 필요한 모든 변경을 수행하였는지 확인하십시오.

```
@echo on
rem Setting the SQL PROCEDURE environment:
db2set DB2_SQLROUTINE_COMPILER_PATH="c:\Micros~1\vc98\bin\vcvars32.bat"
db2set DB2_SQLROUTINE_COMPILE_COMMAND="cl -Od -W2 /TC -D_X86_=1
         -I%DB2PATH%\include SQLROUTINE_FILENAME.c /link -dll
         -def:SQLROUTINE_FILENAME.def /out:SQLROUTINE_FILENAME.dll
         %DB2PATH%\lib\db2api.lib"
@echo off
```

환경을 설정하면 DB2CLP 창에서 SQL 프로시듀어를 작성하고 호출할 수 있습니다. 다음은 whiles.db2 샘플 파일을 사용한 예입니다.

1. SQL 프로시듀어 샘플 파일을 작업 디렉토리에 복사하십시오.

   ```
   C:\> xcopy /I c:\sqllib\samples\sqlproc c:\sqlproc
   ```

2. 작업 디렉토리로 이동하십시오.

   ```
   C:\> cd c:\sqlproc
   ```

3. 데이터베이스에 연결하십시오.

   ```
   C:\sqlproc> db2 connect to sample
   ```

4. 다음과 같이 whiles.db2 스크립트를 수행하십시오.

   ```
   C:\sqlproc> db2 -td@ -vf whiles.db2
   ```

5. 프로시듀어 이름, IN 매개변수에 대한 값 및 다음 OUT 매개변수에 대한 "?"를 사용하여 CALL 명령을 입력하십시오.

   ```
   C:\sqlproc> db2 "call dept_median (51,?)"
   ```

   다음 결과를 수신합니다.

   ```
   MEDIANSALARY: 1.76545000000000e+004
   "DEPT_MEDIAN" RETURN_STATUS: "0"
   ```

   위의 명령을 실행하는 동안 다음 오류 메시지가 발생할 경우,

   ```
   SQL0805N  "NULLID.SQLLD202" 패키지를 찾을 수 없습니다.  SQLSTATE=51002
   ```

   DB2 유틸리티를 데이터베이스에 리바인드할 수도 있습니다. SQLLIB\bnd 디렉토리에서 다음 명령을 발행하십시오.

   ```
   C:\SQLLIB\bnd> DB2 bind @db2ubind.lst blocking all grant public
   C:\SQLLIB\bnd> DB2 bind @db2cli.lst blocking all grant public
   ```

   그런 다음 CALL 명령을 다시 수행하십시오.

SQL 프로시듀어 환경 설정

다음은 39 페이지의 『제2장 설정』에서 DB2 환경을 설정하기 위한 지침과 함께 추가적으로 제공되는 지침입니다.

주:

1. S/2 FAT 파일 시스템에서는 SQL 프로시듀어의 *스키마* 이름이 8자 이하로 제한됩니다. 8문자보다 긴 스키마 이름을 사용할 때는 HPFS 파일 시스템이 있어야 합니다.
2. UNIX 시스템에서는 `.fenced` 파일의 소유자 사용권한으로 분리(fenced) SQL 루틴이 실행됩니다. `.fenced` 파일의 소유자가 파일이 상주하는 `$HOME/sqllib/adm` 디렉토리도 소유하는지 확인하십시오. 또한 `.fenced` 파일의 소유자가 변경될 때마다 다음 디렉토리에 대한 파일 사용권한을 적절하게 변경하십시오(이 디렉토리가 있는 경우).

   UNIX

   ```
   $HOME/sqllib/function/routine/sqlproc/<db_name>/<schema_name>/tmp
   ```

   OS/2 및 Windows

   ```
   %DB2PATH%\function\routine\sqlproc\<db_name>\<schema_name>\tmp
   ```

   여기서 <*db_name*> 및 <*schema_name*>은 SQL 프로시듀어를 작성하는 데 사용되는 데이터베이스 및 스키마입니다.

SQL 프로시듀어 자원의 경우, 서버에 Application Development Client를 설치해야 합니다. Application Development Client 설치에 대한 자세한 내용은 플랫폼에 대한 *빠른 시작* 책을 참조하십시오. 사용하는 플랫폼에서 DB2가 지원하는 C 및 C++ 컴파일러에 대한 자세한 내용은 8 페이지의 『플랫폼별로 지원되는 소프트웨어』를 참조하십시오.

컴파일러 구성은 두 변수를 사용하여 수행됩니다.

DB2_SQLROUTINE_COMPILER_PATH

컴파일러의 2진, 라이브러리 및 포함 파일에 대한 경로를 제공하여 환경 변수를 설정합니다.

DB2_SQLROUTINE_COMPILE_COMMAND

SQL 프로시듀어에 대해 생성된 C 파일을 컴파일하기 위해 DB2가 사용하는 전체 명령이 지정됩니다.

Stored Procedure Builder에서 db2set 명령을 사용하거나 SQL 저장 프로시듀어 빌드 옵션 대화 상자를 사용하여 DB2 레지스트리 변수의 값을 설정할 수 있습니다. SQL 저장 프로시듀어 빌드 옵션 대화 상자를 사용하면 실제로 데이터베이스 서버에 액세스하거나 변경사항을 적용하도록 재시작하지 않아도 됩니다.

컴파일러 환경 변수 설정

OS/2, Windows 및 UNIX 운영 체제에는 환경을 구성하기 위한 서로 다른 규칙이 있습니다. 어떤 경우에는 구성이 필요하지 않고, 또 다른 경우에는 적절하게 환경 변수를 설정하는 실행 스크립트를 가리키도록 DB2_SQLROUTINE_COMPILER_PATH DB2 레지스트리 변수를 설정해야 합니다. db2start 명령은 설정을 적용하도록 실행해야 합니다. 다음은 지원되는 플랫폼에 대해 컴파일러 환경 변수를 설정하는 방법을 보여줍니다.

OS/2 OS/2용 IBM VisualAge C++ 버전 3.6:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="c:\ibmcxxo\bin\setenv.cmd"
```

OS/2용 IBM VisualAge C++ 버전 4:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="c:\ibmcpp40\bin\setenv.cmd"
```

주: 이들 명령의 경우, C++ 컴파일러가 c: 드라이브에 설치된 것으로 가정합니다. 필요하면 드라이브나 경로를 변경하여 사용자의 시스템에서 C++ 컴파일러의 위치를 반영하십시오.

UNIX 처음으로 저장 프로시듀어를 컴파일할 경우, DB2는 실행 스크립트 파일 $HOME/sqllib/function/routine/sr_cpath(컴파일러 환경 변수에 대한 기본값이 있는 파일)를 생성합니다. 기본값이 사용자의 컴파일러에 적절하지 않으면 이 파일을 편집할 수 있습니다. 또한 원하는 설정을 지정하는 또 다른 실행 스크립트의 전체 경로 이름을 포함하도록 DB2_SQLROUTINE_COMPILER_PATH DB2 레지스트리 변수를 설정할 수 있습니다.

Windows

Windows 32비트 운영 체제에서 사용자의 컴파일러에 대한 환경 변수가 SYSTEM 변수로 설정되면 구성이 필요하지 않습니다. 그렇지 않으면 DB2_SQLROUTINE_COMPILER_PATH DB2 레지스트리 변수를 다음과 같이 설정하십시오.

Microsoft Visual C++ 버전 5.0:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="c:\devstudio\vc\bin\vcvars32.bat"
```

Microsoft Visual C++ 버전 6.0:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="c:\Micros~1\vc98\bin\vcvars32.bat"
```

Windows용 IBM VisualAge C++ 버전 3.6:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="c:\ibmcxxw\bin\setenv.bat"
```

Windows용 IBM VisualAge C++ 버전 4:

```
db2set DB2_SQLROUTINE_COMPILER_PATH="c:\ibmcppw40\bin\setenv.bat"
```

주: 이들 명령의 경우, C++ 컴파일러가 c: 드라이브에 설치된 것으로 가정합니다. 필요하면 드라이브나 경로를 변경하여 사용자의 시스템에서 C++ 컴파일러의 위치를 반영하십시오.

컴파일 명령 사용자 정의

Application Development Client를 설치하면 각 서버 플랫폼에서 지원되는 컴파일러 중 최소한 하나에서 작동하는 기본 컴파일 명령을 제공합니다.

AIX AIX용 IBM C Set++ 버전 3.6.6

HP-UX

HP aC++ 버전 A.03.25

Linux GNU/Linux g++

OS/2 OS/2용 IBM VisualAge C++ 버전 3.6.5

PTX ptx/C++ 버전 5.2

Solaris

Forte/WorkShop C++ 버전 4.2, 5.0, 6 및 6.1

Windows NT 및 Windows 2000

Microsoft Visual C++ 버전 5.0 및 6.0

다른 컴파일러를 사용하거나 기본 명령을 사용자 정의하려면 다음과 같이 DB2_SQLROUTINE_COMPILE_COMMAND DB2 레지스트리 변수를 설정해야 합니다.

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=<compile_command>
```

여기서 <compile_command>는 저장 프로시듀어를 작성하는 데 필요한 옵션 및 매개변수를 포함하는 C 또는 C++ 컴파일 명령입니다. db2start 명령은 설정을 적용하도록 실행해야 합니다.

컴파일 명령에서 키워드 SQLROUTINE_FILENAME을 사용하여 생성된 SQC, C, PDB, DEF, EXP, 메시지 로그 및 공유 라이브러리 파일의 파일 이름을 바꾸십시오. AIX에서만 키워드 SQLROUTINE_ENTRY를 사용하여 시작점 이름을 바꾸십시오. 다음은 지원되는 서버 플랫폼에서 C 또는 C++ 컴파일러에 대한 DB2_SQLROUTINE_COMPILE_COMMAND 기본값입니다.

AIX AIX용 IBM C 버전 3.6.6을 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=xlc -H512 -T512 \
-I$HOME/sqllib/include SQLROUTINE_FILENAME.c -bE:SQLROUTINE_FILENAME.exp \
-e SQLROUTINE_ENTRY -o SQLROUTINE_FILENAME -L$HOME/sqllib/lib -lc -ldb2
```

AIX용 IBM C Set++ 버전 3.6.6을 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=xlC -H512 -T512 \
-I$HOME/sqllib/include SQLROUTINE_FILENAME.C -bE:SQLROUTINE_FILENAME.exp \
-e SQLROUTINE_ENTRY -o SQLROUTINE_FILENAME -L$HOME/sqllib/lib -lc -ldb2
```

이것은 DB2_SQLROUTINE_COMPILE_COMMAND 레지스트리 변수가 설정되지 않은 경우의 기본 컴파일 명령입니다.

주: AIX에서 64비트 SQL 프로시듀어를 컴파일하려면 위의 명령에 -q64 옵션을 추가하십시오.

AIX용 IBM VisualAge C++ 버전 4를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="vacbld"
```

vacbld 명령 다음에 구성 파일을 지정하지 않으면 처음 SQL 프로시듀어를 작성하려고 할 때 DB2가 다음 기본 구성 파일을 작성합니다.

```
$HOME/sqllib/function/routine/sqlproc.icc
```

DB2_SQLROUTINE_COMPILE_COMMAND에 대한 DB2 레지스트리 값을 설정할 때 사용자 고유의 구성 파일을 지정할 수 있습니다.

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="vacbld $HOME/sqllib/function/sqlproc.icc"
```

HP-UX

HP C 컴파일러 버전 A.11.00.03을 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=cc +DAportable +ul -Aa +z \
-I$HOME/sqllib/include -c SQLROUTINE_FILENAME.c; \
ld -b -o SQLROUTINE_FILENAME SQLROUTINE_FILENAME.o \
-L$HOME/sqllib/lib -ldb2
```

HP aC++ 버전 A.03.25를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=aCC +DAportable +u1 +z -ext \
-I$HOME/sqllib/include SQLROUTINE_FILENAME.C -b \
-o SQLROUTINE_FILENAME -L$HOME/sqllib/lib -ldb2
```

이것은 DB2_SQLROUTINE_COMPILE_COMMAND 레지스트리 변수가 설정되지 않은 경우의 기본 컴파일 명령입니다.

Linux GNU/Linux gcc를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=cc \
-I$HOME/sqllib/include SQLROUTINE_FILENAME.c \
-shared -o SQLROUTINE_FILENAME -L$HOME/sqllib/lib -ldb2
```

GNU/Linux g++를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=g++ \
-I$HOME/sqllib/include SQLROUTINE_FILENAME.C \
-shared -o SQLROUTINE_FILENAME -L$HOME/sqllib/lib -ldb2
```

이것은 DB2_SQLROUTINE_COMPILE_COMMAND 레지스트리 변수가 설정되지 않은 경우의 기본 컴파일 명령입니다.

PTX ptx/C 버전 4.5를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=cc -KPIC \
  -I$HOME/sqllib/include SQLROUTINE_FILENAME.c \
-G -o SQLROUTINE_FILENAME.so -L$HOME/sqllib/lib -ldb2 ; \
cp SQLROUTINE_FILENAME.so SQLROUTINE_FILENAME
```

ptx/C++ 버전 5.2를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=c++ -KPIC \
-D_RWSTD_COMPILE_INSTANTIATE=0
-I$HOME/sqllib/include SQLROUTINE_FILENAME.C \
-G -o SQLROUTINE_FILENAME.so -L$HOME/sqllib/lib -ldb2 ; \
cp SQLROUTINE_FILENAME.so SQLROUTINE_FILENAME
```

이것은 DB2_SQLROUTINE_COMPILE_COMMAND 레지스트리 변수가 설정되지 않은 경우의 기본 컴파일 명령입니다.

OS/2 OS/2용 IBM VisualAge C++ 버전 3.6.5를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="icc -Ge- -Gm+ -W2
-I%DB2PATH%\include SQLROUTINE_FILENAME.c
/B\"/NOFREE /NOI /ST:64000\" SQLROUTINE_FILENAME.def
%DB2PATH%\lib\db2api.lib"
```

이것은 DB2_SQLROUTINE_COMPILE_COMMAND 레지스트리 변수가 설정되지 않은 경우의 기본 컴파일 명령입니다.

OS/2용 IBM VisualAge C++ 버전 4를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="vacbld"
```

vacbld 명령 다음에 구성 파일을 지정하지 않으면 처음 SQL 프로시듀어를 작성하려고 할 때 DB2가 다음 기본 구성 파일을 작성합니다.

```
%DB2PATH%\function\routine\sqlproc.icc
```

고유한 구성 파일을 사용하는 경우, DB2_SQLROUTINE_COMPILE_COMMAND에 대한 DB2 레지스트리 값을 설정할 때 사용자 고유의 구성 파일을 지정할 수 있습니다.

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="vacbld
%DB2PATH%\function\sqlproc.icc"
```

Solaris

Forte/WorkShop C 버전 4.2 및 5.0을 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=cc -xarch=v8plusa -Kpic \
-I$HOME/sqllib/include SQLROUTINE_FILENAME.c \
-G -o SQLROUTINE_FILENAME -L$HOME/sqllib/lib \
-R$HOME/sqllib/lib -ldb2
```

Forte/WorkShop C++ 버전 4.2 및 5.0을 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=CC -xarch=v8plusa -Kpic \
-I$HOME/sqllib/include SQLROUTINE_FILENAME.C \
-G -o SQLROUTINE_FILENAME -L$HOME/sqllib/lib \
-R$HOME/sqllib/lib -ldb2
```

이것은 DB2_SQLROUTINE_COMPILE_COMMAND 레지스트리 변수가 설정되지 않은 경우의 기본 컴파일 명령입니다.

주:

1. libdb2.so와 링크할 때 컴파일러가 유효한 실행 파일을 생성하지 못하는 문제점을 피하기 위해 기본 컴파일러 명령에 컴파일러 옵션 -xarch=v8plusa가 추가되었습니다.
2. Solaris에서 64비트 SQL 프로시듀어를 컴파일하려면 -xarch=v8plusa 옵션을 가져와서 위의 명령에 -xarch=v9 옵션을 추가하십시오.

Windows NT 및 Windows 2000

주: SQL 프로시듀어는 Windows 95, Windows 98 또는 Windows Millennium Edition에서 지원되지 않습니다.

Microsoft Visual C++ 버전 5.0 및 6.0을 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=cl -Od -W2 /TC -D_X86_=1
-I%DB2PATH%\include SQLROUTINE_FILENAME.c /link -dll
-def:SQLROUTINE_FILENAME.def /out:SQLROUTINE_FILENAME.dll
%DB2PATH%\lib\db2api.lib
```

이것은 DB2_SQLROUTINE_COMPILE_COMMAND 레지스트리 변수가 설정되지 않은 경우의 기본 컴파일 명령입니다.

Windows용 IBM VisualAge C++ 버전 3.6.5를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="ilib /GI
SQLROUTINE_FILENAME.def & icc -Ti -Ge- -Gm+ -W2
-I%DB2PATH%\include SQLROUTINE_FILENAME.c
/B\"/ST:64000 /PM:VIO /DLL\" SQLROUTINE_FILENAME.exp
%DB2PATH%\lib\db2api.lib"
```

Windows용 IBM VisualAge C++ 버전 4를 사용하려면:

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="vacbld"
```

vacbld 명령 다음에 구성 파일을 지정하지 않으면 처음 SQL 프로시듀어를 작성하려고 할 때 DB2가 다음 기본 구성 파일을 작성합니다.

```
%DB2PATH%\function\routine\sqlproc.icc
```

고유한 구성 파일을 사용하는 경우, DB2_SQLROUTINE_COMPILE_COMMAND에 대한 DB2 레지스트리 값을 설정할 때 사용자 고유의 구성 파일을 지정할 수 있습니다.

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND="vacbld
%DB2PATH%\function\sqlproc.icc"
```

기본 컴파일러 옵션으로 리턴하려면 다음 명령을 사용하여 DB2_SQLROUTINE_COMPILE_COMMAND에 대한 DB2 레지스트리 값을 널(NULL)로 설정하십시오.

```
db2set DB2_SQLROUTINE_COMPILE_COMMAND=
```

중간 파일 보유

CREATE PROCEDURE문을 발행할 때 DB2는 해당 명령문을 정상적으로 완료하는 경우 정상적으로 삭제되는 많은 수의 중간 파일을 작성합니다. SQL 프로시듀어가 예상대로 수행되지 않으면 DB2가 작성한 SQC, C 및 메시지 로그 파일을 검토하는 것이 유용할 수 있습니다. CREATE PROCEDURE문의 성공적인 실행 중에 DB2가 작성하는 파일을 보관하려면 다음 명령과 같은 DB2_SQLROUTINE_KEEP_FILES DB2 레지스트리 변수 값을 ″1″, ″y″ 또는 ″yes″로 설정해야 합니다.

```
db2set DB2_SQLROUTINE_KEEP_FILES=1
```

SQL 프로시듀어가 정상적으로 작성되지 않은 경우에 남아 있을 수 있는 중간 파일은 수동으로 삭제해야 합니다. 이들 파일은 다음 디렉토리에 있습니다.

UNIX

```
$HOME/sqllib/function/routine/sqlproc/<db_name>/<schema_name>/tmp
```

OS/2 및 Windows

```
%DB2PATH%\function\routine\sqlproc\<db_name>\<schema_name>\tmp
```

여기서 <*db_name*> 및 <*schema_name*>은 SQL 프로시듀어를 작성하는 데 사용되는 데이터베이스 및 스키마입니다.

사전 처리 컴파일 및 바인드 옵션 사용자 정의

사전 처리 컴파일 및 바인드 옵션은 DB2_SQLROUTINE_PREPOPTS DB2 레지스트리 변수를 설정하여 사용자 정의할 수 있습니다. 이들 옵션은 프로시듀어 레벨에서는 사용자 정의할 수 없습니다. SQL 프로시듀어에 대해 사용자 정의된 사전 처리 컴파일 옵션을 지정하려면 다음 명령을 사용하여 DB2 사전 처리 컴파일러에서 사용할 사전 처리 컴파일 옵션 목록을 DB2 레지스트리에 배치하십시오.

```
db2set DB2_SQLROUTINE_PREPOPTS=options
```

여기서 *options*은 DB2 사전 처리 컴파일러가 사용할 사전 처리 컴파일 옵션 목록을 지정합니다. 다음 옵션만 사용할 수 있습니다.

```
BLOCKING {UNAMBIG | ALL | NO}
DATETIME {DEF | USA | EUR | ISO | JIS | LOC}
DEGREE {1 | degree-of-parallelism | ANY}
DYNAMICRULES {BIND | RUN}
EXPLAIN {NO | YES | ALL}
EXPLAINSNAP {NO | YES | ALL}
FEDERATED {NO | YES}
INSERT {DEF | BUF}
ISOLATION {CS |RR |UR |RS |NC}
QUERYOPT optimization-level
SYNCPOINT {ONEPHASE | TWOPHASE | NONE}
```

백업 및 복원

SQL 프로시듀어가 작성될 때 DLL이 2MB보다 작으면 카탈로그 테이블에도 생성된 동적 링크 라이브러리(DLL)가 보존됩니다. 데이터베이스가 백업되거나 복원될 때 생성된 공유 DLL이 2MB보다 작은 SQL 프로시듀어는 버전이 카탈로그 테이블에 보존된 상태로 백업 및 복원됩니다. 생성된 공유 DLL이 2MB보다 큰 SQL 프로시듀어가 있으면 데이터베이스 백업 및 복원과 함께 파일 시스템 백업 및 복원도 수행해야 합니다. 그렇지 않으면 `syscat.procedures` 카탈로그 테이블에서 소스를 사용하여 수동으로 DLL을 다시 작성해야 합니다.

주: 데이터베이스 복구시, 복구되는 데이터베이스에 속하는 파일 시스템의 모든 SQL 프로시듀어 실행 파일이 제거됩니다. 색인 작성 구성 매개변수 `indexrec`가

RESTART로 설정되면 모든 SQL 프로시듀어 실행 파일은 카탈로그 테이블에서 추출되어 다음 연결시 다시 파일 시스템에 배치됩니다. 그렇지 않으면 SQL 실행 파일은 처음 SQL 프로시듀어를 실행할 때 추출됩니다.

실행 파일은 다음 디렉토리에 다시 배치됩니다.

UNIX $HOME/sqllib/function/routine/sqlproc/<*database_name*>

OS/2 및 Windows

%DB2PATH%\function\routine\sqlproc\<*database_name*>

여기서 <*database_name*>은 SQL 프로시듀어가 작성된 데이터베이스를 나타냅니다.

주: 복원 조작 후에 처음으로 데이터베이스에 연결하려고 하면 다음이 리턴됩니다.

```
SQL2048N "SQL PROCEDURE FILES" 오브젝트에 액세스하는 동안 오류가 발생했습니다.
이유 코드: "7".
```

그러면 DB2는 db2stop을 사용하여 중지한 다음 db2start를 사용하여 재시작해야 합니다.

SQL 프로시듀어 작성

UNIX의 sqllib/samples/sqlproc 디렉토리와 OS/2 및 Windows의 %DB2PATH%\samples\sqlproc 디렉토리에 있는 DB2 명령행 처리기 스크립트(.db2 확장자로 끝남)는 서버에서 저장 프로시듀어를 작성하는 CREATE PROCEDURE 문을 실행합니다. 각 CLP 스크립트에는 확장자가 .sqc 또는 .c인 동일한 이름의 해당 클라이언트 응용프로그램 파일이 있습니다.

CREATE PROCEDURE CLP 스크립트를 수행하기 전에 다음 명령을 사용하여 샘플 데이터베이스에 연결하십시오.

```
db2 connect to sample user userid using password
```

여기서 *userid* 및 *password*는 sample 데이터베이스가 있는 인스턴스의 사용자 ID와 암호입니다.

rsultset.db2 스크립트 파일에 있는 CREATE PROCEDURE문을 실행하려면 다음 명령을 입력하십시오.

```
db2 -td@ -vf rsultset.db2
```

이제 『SQL 프로시듀어 호출』에서 설명한 대로 SQL 프로시듀어를 호출할 수 있습니다.

SQL 프로시듀어 호출

명령행 처리기(CLP) call 명령을 사용하거나 클라이언트 응용프로그램을 빌드하여 SQL 프로시듀어를 호출할 수 있습니다.

CALL 명령 사용

call 명령을 사용하려면 저장 프로시듀어 이름과 IN 및 INOUT 매개변수 그리고 각 OUT 매개변수에 대한 플레이스 홀더로 '?'를 입력해야 합니다.

먼저 128 페이지의 『SQL 프로시듀어 작성』의 단계에 따라 SQL 프로시듀어를 작성하십시오.

SQL 프로시듀어를 호출하려면 먼저 데이터베이스에 연결해야 합니다.

```
db2 connect to sample user userid using password
```

여기서 *userid* 및 *password*는 sample 데이터베이스가 있는 인스턴스의 사용자 ID와 암호입니다.

저장 프로시듀어에 대한 매개변수는 프로그램 소스 파일에서 저장 프로시듀어에 대한 CREATE PROCEDURE문에 제공됩니다. 예를 들어, 소스 파일 whiles.db2에서 DEPT_MEDIAN 프로시듀어에 대한 CREATE PROCEDURE문은 다음으로 시작합니다.

```
CREATE PROCEDURE DEPT_MEDIAN
(IN deptNumber SMALLINT, OUT medianSalary DOUBLE)
```

이 프로시듀어를 호출하려면 IN 매개변수 deptNumber에 대한 유효한 SMALLINT 값과 OUT 매개변수에 대한 '?'를 배치해야 합니다. 샘플 데이터베이스의 해당 테이블에서 유효한 값을 얻거나 프로그램 소스 파일을 호출하는 클라이언트에서 이 클라이언트가 사용하는 값을 점검하여 유효한 값을 얻을 수 있습니다. whiles.sqc에서는 값 "51"이 사용되었음을 알 수 있습니다.

```
printf("Use CALL with Host Variables to invoke the Server Procedure "
   "named %s\n", procname);
dept = 51;                      /* get median for dept. 51 */
```

프로시듀어 이름, IN 매개변수에 대한 값 및 OUT 매개변수의 값에 대한 '?'를 사용하여 call 명령을 입력하십시오. 프로시듀어의 매개변수는 다음과 같이 괄호를 사용하고 큰따옴표로 묶어야 합니다.

```
db2 "call dept_median (51, ?)"
```

다음 결과를 수신합니다.

```
MEDIANSALARY: 1.76545000000000e+004
"DEPT_MEDIAN" RETURN_STATUS: "0"
```

위의 명령을 호출하는 동안 다음 오류 메시지가 발생할 경우:

```
SQL0805N  "NULLID.SQLLD202" 패키지를 찾을 수 없습니다.  SQLSTATE=51002
```

DB2 유틸리티를 데이터베이스에 리바인드할 수도 있습니다. SQLLIB\bnd 디렉토리에서 다음 명령을 발행하십시오.

```
C:\SQLLIB\bnd> DB2 bind @db2ubind.lst blocking all grant public
C:\SQLLIB\bnd> DB2 bind @db2cli.lst blocking all grant public
```

그런 다음 CALL 명령을 다시 수행하십시오.

call 명령을 사용할 때는 다음 사항에 유의하십시오.

- 하나의 결과 컬럼에 대해 최대 1023문자가 있을 수 있습니다.
- 호출되는 저장 프로시듀어는 카탈로그에 정의되어 있어야 합니다.

OS/2 DB2 CLI 클라이언트 응용프로그램

%DB2PATH%\samples\splproc에 있는 명령 파일 bldcli.cmd에는 SQL 프로시듀어에 대해 DB2 CLI 클라이언트 응용프로그램을 빌드하기 위한 명령이 들어 있습니다. bldcli.cmd에 대한 자세한 내용은 276 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 rsultset.c에서 DB2 CLI 클라이언트 응용프로그램 rsultset을 빌드하려면 다음을 입력하십시오.

```
bldcli rsultset
```

이 명령은 실행 파일 rsultset을 작성합니다.

저장 프로시듀어를 호출하려면 실행 파일 이름, 연결 중인 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

rsultset *database userid password*

OS/2 Embedded SQL 클라이언트 응용프로그램

%DB2PATH%\samples\splproc에 있는 명령 파일 bldapp.cmd에는 SQL 프로시듀어에 대해 Embedded SQL 클라이언트 응용프로그램을 빌드하기 위한 명령이 들어 있습니다. bldapp.cmd에 대한 자세한 내용은 282 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 basecase.sqc에서 Embedded SQL 클라이언트 응용프로그램 basecase를 빌드하려면 명령 파일 이름, 실행 파일 이름, 연결 중인 데이터베이스 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
bldapp basecase database userid password
```

그 결과 실행 파일 basecase가 작성됩니다.

저장 프로시듀어를 호출하려면 다음을 입력하여 클라이언트 응용프로그램을 수행하십시오.

basecase *database userid password*

UNIX DB2 CLI 클라이언트 응용프로그램

sqllib/samples/sqlproc에 있는 스크립트 파일 bldcli에는 SQL 프로시듀어에 대해 DB2 CLI 클라이언트 응용프로그램을 빌드하기 위한 명령이 들어 있습니다. bldcli 스크립트 파일에 대한 자세한 내용은 UNIX 플랫폼에 대한 "응용프로그램 빌드" 장에서 "DB2 CLI 응용프로그램" 절을 참조하십시오.

소스 파일 rsultset.c에서 DB2 CLI 클라이언트 응용프로그램 rsultset을 빌드하려면 다음을 입력하십시오.

```
bldcli rsultset
```

이 명령은 실행 파일 rsultset을 작성합니다.

저장 프로시듀어를 호출하려면 실행 파일 이름, 연결 중인 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

rsultset *database userid password*

UNIX Embedded SQL 클라이언트 응용프로그램

sqllib/samples/sqlproc에 있는 스크립트 파일 bldapp에는 SQL 프로시듀어에 대해 Embedded SQL 클라이언트 응용프로그램을 빌드하기 위한 명령이 들어 있습니다. bldapp 스크립트 파일에 대한 자세한 내용은 UNIX 플랫폼에 대한 ″응용프로그램 빌드″ 장에서 ″DB2 API 및 Embedded SQL 응용프로그램″ 절을 참조하십시오.

소스 파일 basecase.sqc에서 Embedded SQL 클라이언트 응용프로그램 basecase를 빌드하려면 스크립트 파일 이름, 실행 파일 이름, 연결 중인 데이터베이스 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
bldapp basecase database userid password
```

그 결과 실행 파일 basecase가 작성됩니다.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

basecase *database userid password*

Windows DB2 CLI 클라이언트 응용프로그램

%DB2PATH%\samples\sqlproc 디렉토리에는 DB2 CLI 클라이언트 응용프로그램을 빌드하기 위한 두 개의 빌드 파일이 들어 있습니다. 하나는 Microsoft Visual C++ 컴파일러를 위한 bldmcli이고 다른 하나는 IBM VisualAge C++ 컴파일러를 위한 bldvcli입니다. bldmcli에 대한 자세한 내용은 409 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오. bldvcli에 대한 자세한 내용은 427 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 rsultset.c에서 DB2 CLI 클라이언트 응용프로그램 rsultset을 빌드하려면 사용 중인 컴파일러에 따라 다음 중 하나를 입력하십시오.

```
bldmcli rsultset
```

또는

```
bldvcli rsultset
```

이들 명령은 실행 파일 rsultset을 작성합니다.

저장 프로시듀어를 호출하려면 실행 파일 이름, 연결 중인 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

rsultset *database userid password*

Windows Embedded SQL 클라이언트 응용프로그램

%DB2PATH%\samples\sqlproc 디렉토리에는 Embedded SQL 클라이언트 응용프로그램을 빌드하기 위한 두 개의 빌드 파일이 들어 있습니다. 하나는 Microsoft Visual C++ 컴파일러를 위한 bldmapp이고 다른 하나는 IBM VisualAge C++ 컴파일러를 위한 bldvapp입니다. bldmapp에 대한 자세한 내용은 416 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오. bldvapp에 대한 자세한 내용은 433 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 basecase.sqc에서 Embedded SQL 클라이언트 응용프로그램 basecase를 빌드하려면 스크립트 파일 이름, 실행 파일 이름, 연결 중인 데이터베이스 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오. 사용 중인 컴파일러에 따라 다음 두 가지 명령 중 하나를 사용할 수 있습니다.

```
bldmapp basecase database userid password
```

또는

```
bldvapp basecase database userid password
```

그 결과 실행 파일 basecase가 작성됩니다.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

```
basecase database userid password
```

컴파일된 SQL 프로시듀어 분산

주: DB2 버전 7 FixPak 3 이전에 작성된 데이터베이스에 대한 DB2 서버들 사이에 컴파일된 SQL 프로시듀어를 분산시키려면 DB2가 해당 SQL 프로시듀어를 발췌하여 설치할 수 있도록 해야 합니다. 이를 수행하려면 컴파일된 SQL 프로시듀어에 대한 소스 또는 목적지인 모든 DB2 서버에서 다음 명령을 입력하십시오.

```
db2updv7 -d <database_name>
```

DB2 버전 7 FixPak 3 이상으로 작성된 데이터베이스의 경우, 이는 데이터베이스 작성시 내재적으로 수행됩니다.

SQL 프로시듀어를 정의하면 이는 C 프로그램으로 변환되고 사전 처리에 컴파일된 다음 목표 데이터베이스에 대해 바인드된 후 컴파일되어 공유 라이브러리에 작성되도록 링크됩니다. 컴파일 및 링크 단계에서는 데이터베이스 서버 머신에서 C 또는 C++ 컴파일러를 사용할 수 있어야 합니다. 그러나 SQL 프로시듀어를 정의하면 컴파일된 양식으로 같은 운영 체제와 같은 DB2 버전을 사용하는 DB2 데이터베이스 서버 기계에 분산시킬 수 있지만, C 또는 C++ 컴파일러에 대한 액세스가 필요하지는 않습니다. 이러한 상황에서 DB2를 사용하여 사용자가 하나의 데이터베이스에서 컴파일된 양식으로 SQL 프로시듀어를 추출하고 다른 서버 머신의 또 다른 데이터베이스에서 컴파일된 양식으로 SQL 프로시듀어를 설치할 수 있습니다.

DB2는 추출 및 설치를 위해 명령행 인터페이스와 프로그래밍 인터페이스를 둘 다 제공합니다. 명령행 인터페이스는 두 개의 CLP 명령 `GET ROUTINE` 및 `PUT ROUTINE`으로 구성됩니다. 프로그래밍 인터페이스는 두 개의 내장된 저장 프로시듀어 `GET_ROUTINE_SAR` 및 `PUT_ROUTINE_SAR`로 구성됩니다. 명령행 인터페이스에 대한 자세한 내용은 *Command Reference*를 참조하십시오. 프로그래밍 인터페이스에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.

데이터베이스 서버 사이에 컴파일된 SQL 프로시듀어를 분산시키려면 다음 단계를 수행하십시오.

1. 응용프로그램의 일부인 SQL 프로시듀어 정의를 포함하여 응용프로그램을 빌드하십시오.
2. 프로시듀어 테스트 후 GET ROUTINE 명령을 발행하거나 GET_ROUTINE_SAR 저장 프로시듀어를 호출하여 각 프로시듀어의 컴파일 버전을 다른 파일로 추출하십시오. 파일을 분산 미디어에 복사하십시오(필요한 경우).
3. 이전 단계에서 작성한 파일을 사용하여 PUT ROUTINE 명령을 발행하거나 PUT_ROUTINE_SAR 저장 프로시듀어를 호출하여 각 서버에서 각 프로시듀어의 컴파일 버전을 설치하십시오. 각 데이터베이스 서버의 운영 체제와 DB2 레벨은 같아야 합니다.

   주: PUT ROUTINE 명령이나 PUT_ROUTINE_SAR 저장 프로시듀어가 작동되도록 하려면 Application Development Client를 목표 머신에 설치해야 합니다.

제6장 AIX 응용프로그램 빌드

이 장에서는 AIX에서의 응용프로그램 빌드에 대한 자세한 정보를 제공합니다. 스크립트 파일에서 db2로 시작되는 명령은 명령행 처리기(CLP) 명령입니다. CLP 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

AIX에 대한 최신 DB2 응용프로그램 개발 갱신사항에 대한 자세한 내용은 다음 웹 사이트를 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

주: 이 장에서 빌드 파일을 사용하여 64비트 응용프로그램을 빌드하려면 각 빌드 파일에서 지정된 명령의 주석 처리를 제거하거나 다음 명령을 사용하여 64비트 오브젝트 모드 환경을 설정할 수 있습니다.

```
export OBJECT_MODE=64
```

중요한 고려사항

이 절에서는 지원되는 다양한 컴파일러에서 DB2 응용프로그램 빌드에 대한 AIX 특정 정보를 제공합니다. 다음 정보가 포함됩니다.

- IBM 및 Micro Focus COBOL 설치 및 수행
- 저장 프로시듀어 및 UDF에 대한 시작점
- 저장 프로시듀어 및 CALL문
- UDF 및 CREATE FUNCTION문

IBM 및 Micro Focus COBOL 설치 및 수행

AIX가 저장 프로시듀어를 로드하고 라이브러리 참조를 해결하는 방법으로 인하여, COBOL을 설치하는 방법에 관한 몇 가지 요구사항이 있습니다. 이들 요구사항은 COBOL 프로그램이 런타임시 공유 라이브러리(저장 프로시듀어)를 로드할 때 인수가 됩니다.

저장 프로시듀어를 로드하면 저장 프로시듀어가 참조하는 라이브러리 체인을 또한 로드해야 합니다. AIX가 사용자 프로그램이 간접적으로만 참조한 라이브러리를 검색하는 경우, 언어 제공업체(IBM COBOL 또는 Micro Focus COBOL)가 빌드할 때 참조한 라이브러리에 컴파일된 경로를 사용해야 합니다. 이 경로는 컴파일러를 설치한 경로와 다를 수도 있습니다. 체인에서 라이브러리를 찾을 수 없는 경우, 저장 프로시듀어 로드에 실패하며 SQLCODE -10013을 수신합니다.

이러한 상황이 발생하지 않게 하려면 임의의 위치에 컴파일러를 설치한 다음 설치 디렉토리에서 모든 언어 라이브러리의 기호 링크를 /usr/lib(라이브러리를 로드

해야 할 때 거의 항상 검색하는 디렉토리)에 작성하십시오. 라이브러리를 sqllib/function(저장 프로시듀어 디렉토리)으로 링크할 수 있지만, 하나의 데이터베이스 인스턴스에 대해서만 작동하며, /usr/lib는 머신에 있는 모든 사용자에 대해 작동합니다. 라이브러리를 복사하지 않는 것이 좋습니다. 이것은 특히 복수의 라이브러리 사본이 존재할 때 Micro Focus COBOL에 적용됩니다.

Micro Focus COBOL의 샘플 기호 링크가 아래에 제공됩니다(/usr/lpp/cobdir에 설치된 것으로 가정함).

```
[1]> su root
[2]> cd /usr/lib
[1]> ln -sf /usr/lpp/cobdir/coblib/*.a .
```

설정해야 할 특정 경로에 대해 사용자 컴파일러 문서를 점검하십시오.

저장 프로시듀어 및 UDF에 대한 시작점

저장 프로시듀어는 데이터베이스에 액세스하고 정보를 클라이언트 응용프로그램으로 리턴하는 프로그램입니다. 사용자 정의 함수(UDF)는 사용자 고유의 스칼라 또는 테이블 함수입니다. 저장 프로시듀어와 UDF는 서버에서 컴파일되어, 서버의 공유 라이브러리에서 저장 및 실행됩니다. 이러한 공유 라이브러리는 저장 프로시듀어와 UDF를 컴파일할 때 작성됩니다.

각각의 공유 라이브러리에는 시작점이 있으며, 공유 라이브러리의 프로시듀어에 액세스하기 위해 서버에서 호출합니다. AIX의 IBM C 컴파일러를 사용하여 라이브러리에 있는 임의로 내보낸 함수 이름을 기본 시작점으로 지정할 수 있습니다. 라이브러리 이름이 저장 프로시듀어 호출 또는 CREATE FUNCTION문에 지정되었을 경우에만 호출되는 함수입니다. 링크 단계에서 -e 옵션을 사용하여 수행할 있습니다. 예를 들면, 다음과 같습니다.

```
-e funcname
```

funcname을 기본 시작점으로 지정합니다. CREATE FUNCTION문과 관련시키는 방법에 대한 자세한 내용은 142 페이지의 『UDF 및 CREATE FUNCTION 문』을 참조하십시오.

그 밖의 UNIX 플랫폼에는 이러한 메커니즘이 존재하지 않으므로, DB2는 기본 시작점을 라이브러리 자체와 동일한 이름으로 가정합니다.

AIX는 사용자에게 외부에서 호출할 수 있는 라이브러리에 있는 전역 함수를 지정하는 내보낼 파일을 제공하도록 요구합니다. 이 파일은 모든 저장 프로시듀어 및/또는 라이브러리에 있는 사용자 정의 함수(UDF)의 이름을 포함해야 합니다. 그 밖의 UNIX 플랫폼은 단순히 라이브러리에 있는 모든 전역 함수를 내보냅니다. AIX 내보내기 파일의 예는 다음과 같습니다.

```
#! outsrv export file
outsrv
```

내보낼 파일 `outsrv.exp`는 저장 프로시듀어 `outsrv`를 나열합니다. 링커는 `outsrv.exp`를 사용하여 같은 이름의 저장 프로시듀어를 포함하는 공유 라이브러리 `outsrv`를 작성합니다.

주: 공유 라이브러리는 빌드된 후 일반적으로 DB2가 액세스할 디렉토리로 복사됩니다. 저장 프로시듀어나 사용자 정의 함수(UDF) 공유 라이브러리를 바꾸려고 할 때 `/usr/sbin/slibclean`을 수행하여 AIX 공유 라이브러리 캐쉬를 비우거나 대상 디렉토에서 라이브러리를 제거한 다음 소스 디렉토리에서 대상 디렉토리로 라이브러리를 복사해야 합니다. 그렇지 않은 경우, AIX가 참조된 라이브러리의 캐쉬를 유지하고 라이브러리 대체를 허용하지 않기 때문에 복사 조작에 실패할 수도 있습니다.

AIX 컴파일러 문서에는 내보낼 파일에 관한 추가 정보가 있습니다.

저장 프로시듀어 및 CALL문

*응용프로그램 개발 안내서*에서는 저장 프로시듀어를 코딩하는 방법에 대해 설명합니다. *SQL 참조서*에서는 CALL문을 사용하여 데이터베이스 위치에서 저장 프로시듀어를 호출하는 방법에 대해 설명합니다. 이 절에서는 저장 프로시듀어를 컴파일하여 CALL문에서 제공한 정보가 있는 행에 링크하는 방법에 대해 설명합니다.

프로그램을 컴파일하고 링크하면 다음 두 가지 방법으로 함수를 식별할 수 있습니다.

- `-e` 옵션 사용.

  예를 들어, 링크 단계에 다음과 같이 지정할 수 있습니다.

  ```
  -e modify
  ```

  링크된 라이브러리의 기본 시작점이 `modify` 함수임을 나타냅니다.

mystored 라이브러리를 /u/mydir/procs 디렉토리에 링크하는 중이고 위에서 지정된 대로 기본 시작점 modify를 사용하려는 경우, 다음과 같이 CALL문을 코딩하십시오.

```
CALL '/u/mydir/procs/mystored'
```

mystored 라이브러리가 메모리에 로드되고, DB2가 modify 함수를 기본 시작점으로 선택하여 실행합니다.

- -bE: 옵션으로 지정된 내보낼 파일 사용.

  일반적으로, 라이브러리에 둘 이상의 저장 프로시듀어가 있고 추가 함수에 저장 프로시듀어로 액세스하려고 할 때 이 링크 옵션을 사용합니다.

  위의 예를 계속하려면 mystored 라이브러리가 세 가지 저장 프로시듀어(위의 modify, remove 및 add)를 포함한다고 가정하십시오. 위와 같이 modify를 기본 시작점으로 식별하고 내보낼 파일에 포함시켜서 remove 및 add가 추가 시작점임을 링크 단계에서 표시합니다.

  링크 단계에서 다음과 같이 지정합니다.

  ```
  -bE:mystored.exp
  ```

  내보낼 파일 mystored.exp를 식별합니다.

  내보낼 파일은 저장 프로시듀어 함수의 목록을 나타내며 맨 처음에는 기본 시작점이 나열됩니다.

  ```
  modify
  remove
  add
  ```

  마지막으로, remove 및 add 함수를 호출하는 저장 프로시듀어에 대한 두 개의 CALL 문을 다음과 같이 코딩합니다.

  ```
  CALL '/u/mydir/procs/mystored!remove'
  ```

  및

  ```
  CALL '/u/mydir/procs/mystored!add'
  ```

UDF 및 CREATE FUNCTION문

*응용프로그램 개발 안내서*에서는 UDF를 코딩하는 방법에 대해 설명합니다. *SQL 참조서*에서는 CREATE FUNCTION문을 사용하여 DB2에 UDF를 등록하는 방법에 대해 설명합니다. 이 절에서는 UDF 컴파일 및 링크 간의 관계와 CREATE FUNCTION문의 EXTERNAL NAME 절에 사용자가 제공하는 정보에 대해 설명합니다.

프로그램을 컴파일하고 링크하면 다음 두 가지 방법으로 함수를 식별할 수 있습니다.

- `-e` 옵션 사용.

  예를 들어, 링크 단계에 다음과 같이 지정할 수 있습니다.

  ```
  -e modify
  ```

  링크된 라이브러리의 기본 시작점이 `modify` 함수임을 나타냅니다.

  `myudfs` 라이브러리를 `/u/mydir/procs` 디렉토리에 링크하는 중이고 위에서 지정된 대로 기본 시작점 `modify`를 사용하려는 경우, 다음을 CREATE FUNCTION문에 포함시키십시오.

  ```
  EXTERNAL NAME '/u/mydir/procs/myudfs'
  ```

  DB2는 `myudfs` 라이브러리의 기본 시작점(`modify` 함수)을 선택합니다.

- `-bE:` 옵션으로 지정된 내보낼 파일 사용.

  일반적으로, 라이브러리에 둘 이상의 UDF가 있고 추가 함수에 UDF로 액세스하려고 할 때 이 링크 옵션을 사용합니다.

  위의 예를 계속하려면 `myudfs` 라이브러리가 세 가지 저장 프로시듀어(위의 `modify`, `remove` 및 `add`)를 포함한다고 가정하십시오. 위와 같이 `modify`를 기본 시작점으로 식별하고 내보낼 파일에 포함시켜서 `remove` 및 `add`가 추가 시작점임을 링크 단계에서 표시합니다.

  링크 단계에서 다음과 같이 지정합니다.

  ```
  -bE:myudfs.exp
  ```

  내보낼 파일 `myudfs.exp`를 식별합니다.

내보낼 파일은 다음과 같이 보입니다.

```
* additional entry points for myudfs
#!
   remove
   add
```

마지막으로, remove 및 add 함수가 구현하는 UDF에 대한 두 개의 CREATE FUNCTION문이 다음 EXTERNAL NAME절을 포함합니다.

```
EXTERNAL NAME '/u/mydir/procs/myudfs!remove'
```

및

```
EXTERNAL NAME '/u/mydir/procs/myudfs!add'
```

IBM C

이 절에서는 다음과 같은 유형의 DB2 인터페이스에서 IBM C를 사용하는 방법에 대해 설명합니다.

- DB2 CLI
- DB2 API
- Embedded SQL

DB2 CLI 응용프로그램

sqllib/samples/cli의 스크립트 파일 bldcli에는 DB2 CLI 프로그램을 빌드하기 위한 명령이 있습니다. 매개변수 $1은 소스 파일의 이름을 지정합니다.

이것은 유일한 필수 매개변수로 Embedded SQL이 없는 CLI 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

프로그램에 .sqc 확장자로 표시되는 Embedded SQL이 들어 있는 경우, 프로그램을 사전 처리 컴파일하기 위해 embprep 스크립트가 호출되고 .c 확장자를 가진 프로그램 파일을 생성합니다.

```
#! /bin/ksh
# bldcli script file -- AIX
# Builds a CLI program with IBM C.
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
fi

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-q64
else
  CFLAGS_64=
fi
# Compile the error-checking utility.
xlc $CFLAGS_64 -I$DB2PATH/include -c utilcli.c

# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c
# Link the program.
xlc $CFLAGS_64 -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2
```

bldcli에 대한 컴파일 및 링크 옵션

컴파일 옵션:

xlc IBM C 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 갖고 있습니다.

bldcli에 대한 컴파일 및 링크 옵션
링크 옵션:
xlc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$CFLAGS_64 ‘BUILD_64BIT=true’가 주석 처리되지 않으면 ″-q64″ 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 실행 프로그램을 지정합니다.
$1.o 오브젝트 파일을 지정합니다.
utilcli.o 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다.
-L$DB2PATH/lib DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 컴파일러는 /usr/lib:/lib 경로로 지정합니다.
-ldb2 DB2 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 tbinfo.c에서 샘플 프로그램 tbinfo를 빌드하려면 다음을 입력하십시오.

```
bldcli tbinfo
```

그 결과 실행 파일 tbinfo가 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
tbinfo
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 dbusemx.sqc에서 Embedded SQL 응용프로그램 dbusemx을 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldcli dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldcli dbusemx database
```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

```
bldcli dbusemx database userid password
```

그 결과 실행 파일 dbusemx가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름을 입력하십시오.

```
dbusemx
```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

```
dbusemx database
```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
dbusemx database userid password
```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다.

sqllib/samples/cli에 있는 스크립트 파일 bldapi에 DB2 API가 있는 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 이 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 bldcli 스크립트 간의 유일한 차이점입니다. bldapi 및 bldcli에 공통되는 컴파일 및 링크 옵션에 대한 자세한 내용은 143 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 dbmconn.c에서 샘플 프로그램 dbmconn을 빌드하려면 다음을 입력하십시오.

```
bldapi dbmconn
```

그 결과 실행 파일 dbmconn이 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
dbmconn
```

DB2 CLI 저장 프로시듀어

sqllib/samples/cli에 있는 스크립트 파일 bldclisp에 DB2 CLI 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 매개변수 $1은 소스 파일의 이름을 지정하고 $2는 공유 라이브러리에 대한 시작점이 되는 저장 프로시듀어 함수를 지정합니다.

```
#! /bin/ksh
# bldclisp script file -- AIX
# Builds a CLI stored procedure in IBM C.
# Usage: bldclisp <prog_name> [ <entry_point> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-q64
else
  CFLAGS_64=
fi
# Compile the error-checking utility.
xlc $CFLAGS_64 -I$DB2PATH/include -c utilcli.c

# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c

# Link the program.
xlc $CFLAGS_64 -o $1 $1.o utilcli.o -L$DB2PATH/lib
\-ldb2 -lm -H512 -T512 -bE:$1.exp -e $2
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldclisp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

xlc IBM C 컴파일러.

$CFLAGS_64
'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

링크 옵션:

xlc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.

$CFLAGS_64
'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-o $1 실행 프로그램을 지정합니다.

$1.o 오브젝트 파일을 지정합니다.

utilcli.o
오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다.

-L$DB2PATH/lib
DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 컴파일러는 /usr/lib:/lib 경로로 지정합니다.

-ldb2 DB2 라이브러리로 링크합니다.

-lm math 라이브러리로 링크합니다.

-H512 출력 파일 정렬을 지정합니다.

-T512 주소를 시작하는 출력 파일 텍스트 세그먼트를 지정합니다.

-bE:$.exp
내보낼 파일을 지정합니다. 내보낼 파일에는 저장 프로시듀어의 목록이 들어 있습니다.

-e $2 기본 시작점을 공유 라이브러리로 지정합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 spserver.c에서 샘플 프로그램 spserver를 빌드하려면 빌드 파일 이름, 프로그램 이름 및 공유 라이브러리에 대한 시작점이 되는 저장 프로시듀어 함수의 이름을 입력하십시오.

```
bldclisp spserver outlanguage
```

스크립트 파일은 저장 프로시듀어를 sqllib/function 디렉토리로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 실행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화할 수 있습니다.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리 내의 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldcli를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 143 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

공유 라이브러리에 액세스하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 또 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/c에 있는 빌드 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 DB2 API 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- AIX
# Builds a C application program.
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
```

```
then
  CFLAGS_64=-q64
else
  CFLAGS_64=
fi
# If embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  xlc $CFLAGS_64 -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  xlc $CFLAGS_64 -I$DB2PATH/include -c utilapi.c
fi
# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c
if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o
  xlc $CFLAGS_64 -o $1 $1.o utilemb.o -ldb2 -L$DB2PATH/lib
else
  # Link the program with utilapi.o
  xlc $CFLAGS_64 -o $1 $1.o utilapi.o -ldb2 -L$DB2PATH/lib
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

xlc IBM C 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다. 예를 들면 $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션:
xlc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$CFLAGS_64 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $ 실행 프로그램을 지정합니다.
$1.o 프로그램 오브젝트 파일을 지정합니다.
utilemb.o Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다.
utilapi.o Embedded SQL 프로그램이 아닌 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다.
-ldb2 데이터베이스 관리자 라이브러리로 링크합니다.
-L$DB2PATH/lib DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 컴파일러는 /usr/lib:/lib 경로로 지정합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.c에서 DB2 API 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다.

실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqc에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

sqllib/samples/c에 있는 스크립트 파일 bldsrv에 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 공유 라이브러리에 대한 시작점가 되는 저장 프로시듀어 함수를 지정합니다. 세 번째 매개변수 $3은 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

처음 두 개의 매개변수인 소스 파일 이름과 시작점만 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- AIX
# Builds a C stored procedure
# Usage: bldsrv <prog_name> <entry_point> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $3
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-q64
else
  CFLAGS_64=
fi
# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c
# Link the program using the export file $1.exp,
# creating shared library $1 with entry point $2.
xlc $CFLAGS_64 -o $1 $1.o -ldb2 -L$DB2PATH/lib -H512 -T512 -bE:$1.exp -e $2
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

xlc IBM C 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션:
xlc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$CFLAGS_64 ‘BUILD_64BIT=true’가 주석 처리되지 않으면 ″-q64″ 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 공유 라이브러리 파일로 출력을 지정합니다.
$1.o 저장 프로시듀어 오브젝트 파일을 지정합니다.
-ldb2 DB2 라이브러리로 링크합니다.
-L$DB2PATH/lib DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 컴파일러는 /usr/lib:/lib 경로로 지정합니다.
-H512 출력 파일 정렬을 지정합니다.
-T512 주소를 시작하는 출력 파일 텍스트 세그먼트를 지정합니다.
-bE:$1.exp 내보낼 파일을 지정합니다. 내보낼 파일에는 저장 프로시듀어의 목록이 들어 있습니다.
-e $1 기본 시작점을 공유 라이브러리로 지정합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqc에서 샘플 프로그램 spserver를 빌드하려면 빌드 파일 이름, 프로그램 이름 및 공유 라이브러리에 대한 시작점이 되는 저장 프로시듀어 함수의 이름을 입력하십시오.

```
bldsrv spserver outlanguage
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver outlanguage database
```

스크립트 파일은 저장 프로시듀어를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리에 액세스하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 150 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

사용자 정의 함수(UDF)

sqllib/samples/c에 있는 스크립트 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다. UDF는 저장 프로시듀어처럼 컴파일됩니다. SQL문을 포함할 수 없습니다. 이것은 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하고 사전 처리 컴파일하며 프로그램을 바인드할 필요가 없음을 의미합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 공유 라이브러리에 대한 시작점이 되는 저장 프로시듀어 함수를 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름 $1을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- AIX
# Builds a C UDF library
# Usage: bldudf <prog_name> <entry_point>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-q64
else
  CFLAGS_64=
fi
# Compile the program.
xlc $CFLAGS_64 -I$DB2PATH/include -c $1.c
# Link the program.
xlc $CFLAGS_64 -o $1 $1.o -ldb2 -ldb2apie -L$DB2PATH/lib -H512 -T512 -bE:$1.exp -e $2
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

xlc IBM C 컴파일러.

$CFLAGS_64
'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

링크 옵션:

xlc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.

$CFLAGS_64
'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-o $1 공유 라이브러리 파일로 출력을 지정합니다.

$1.o 공유 라이브러리 오브젝트 파일을 지정합니다.

-ldb2 데이터베이스 관리자 라이브러리로 링크합니다.

-ldb2apie
LOB 위치 지정자를 사용할 수 있도록 DB2 API 엔진 라이브러리와 링크합니다.

-L$DB2PATH/lib
DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면, 컴파일러는 /usr/lib:/lib 경로로 지정합니다.

-H512 출력 파일 정렬을 지정합니다.

-T512 주소를 시작하는 출력 파일 텍스트 세그먼트를 지정합니다.

-bE:$1.exp
내보낼 파일을 지정합니다. 내보낼 파일에는 UDF의 목록이 들어 있습니다.

-e $2 기본 시작점을 공유 라이브러리로 지정합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오. UDF 작성에 대한 자세한 내용은 142 페이지의 『UDF 및 CREATE FUNCTION문』을 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 빌드 파일 이름, 프로그램 이름 및 공유 라이브러리에 대한 시작점이 되는 UDF 함수의 이름을 입력하십시오.

```
bldudf udfsrv ScalarUDF
```

스크립트 파일은 UDF를 sqllib/function 디렉토리로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 이 프로그램의 Embedded SQL 버전 및 DB2 CLI가 제공됩니다.

스크립트 파일 bldcli를 사용하여 sqllib/samples/cli의 소스 파일 udfcli.c에서 DB2 CLI udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 143 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

스크립트 파일 bldapp를 사용하여 sqllib/samples/c의 소스 파일 udfcli.sqc에서 Embedded SQL udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 150 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

AIX 버전 4 및 5의 C 멀티스레드 응용프로그램은 xlc 컴파일러 대신 xlc_r 컴파일러를 사용하며, C++의 경우 xlC 컴파일러 대신 xlC_r 컴파일러를 사용하여 컴파일하고 링크해야 합니다. _r 버전은 멀티스레드 컴파일에 대한 적절한 선행 처리기 정의를 설정하고 링커에 적절한 스레드 라이브러리 이름을 제공합니다.

멀티스레드 컴파일러 프론트 엔드를 사용하는 컴파일러 및 링크 플래그 설정에 대한 추가 정보는 사용자 컴파일러 문서에서 찾을 수 있습니다.

sqllib/samples/c에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다. 첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- AIX
# Builds a C multi-threaded embedded SQL program.
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
  embprep $1 $2 $3 $4
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-q64
else
  CFLAGS_64=
fi
# Compile the program.
xlc_r $CFLAGS_64 -I$DB2PATH/include -c $1.c
# Link the program.
xlc_r $CFLAGS_64 -o $1 $1.o -L$DB2PATH/lib -ldb2
```

위에서 언급된 xlc_r 컴파일러 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이 옵션에 대한 자세한 내용은 150 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqc에서 멀티스레드 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서 실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
thdsrver
```

IBM C Set++

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)
- 멀티스레드 응용프로그램

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/cpp에 있는 스크립트 파일 bldapp에 DB2 API와 Embedded SQL 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 DB2 API 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공됩니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램에 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- AIX
# Builds a C++ application program.
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile 64 bit programs, uncomment the following line.
```

```
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-q64
else
  CFLAGS_64=
fi
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqC" ]]
then
  embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  xlC $CFLAGS_64 -I$DB2PATH/include -c utilemb.C
else
  # Compile the utilapi.C error-checking utility.
  xlC $CFLAGS_64 -I$DB2PATH/include -c utilapi.C
fi
# Compile the program.
xlC $CFLAGS_64 -I$DB2PATH/include -c $1.C

if [[ -f $1".sqC" ]]
then
  # Link the program with utilemb.o
  xlC $CFLAGS_64 -o $1 $1.o utilemb.o -ldb2 -L$DB2PATH/lib
else
  # Link the program with utilapi.o
  xlC $CFLAGS_64 -o $1 $1.o utilapi.o -ldb2 -L$DB2PATH/lib
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

xlC IBM C Set++ 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션:
xlC 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$CFLAGS_64 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 실행 프로그램을 지정합니다.
-o $1 프로그램 오브젝트 파일을 지정합니다.
utilapi.o 비 Embedded SQL 프로그램에 대한 API 유틸리티 오브젝트 파일을 포함합니다.
utilemb.o Embedded SQL 프로그램에 대한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다.
-ldb2 데이터베이스 관리자 라이브러리로 링크합니다.
-L$DB2PATH/lib DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 컴파일러는 /usr/lib:/lib 경로로 지정합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.C에서 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다. 다음을 입력하여 sample 데이터베이스에 대해서 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqC에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ 저장 프로시듀어 빌드에 대한 정보를 참조하십시오.

sqllib/samples/cpp에 있는 스크립트 파일 bldsrv에 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 공유 라이브러리에 대한 시작점이 되는 해당하는 저장 프로시듀어 함수를 지정합니다. 세 번째 매개변수 $3은 연결하려는 데이터베이스의 이름을 지정합니다. 이

저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 필요 없습니다.

처음 두 개의 매개변수 소스 파일 이름과 시작점만 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- AIX
# Builds a C++ stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-q64
  LFLAGS_64=-X64
else
  CFLAGS_64=
  LFLAGS_64=
fi
# Compile the program.
xlC $CFLAGS_64 -I$DB2PATH/include -c $1.C
# Link using export file $1.exp, creating shared library $1
makeC++SharedLib $LFLAGS_64 -p 1024 -o $1 $1.o -L$DB2PATH/lib -ldb2 -E $1.exp
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

xlC IBM C Set++ 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션: **makeC++SharedLib** 정적 생성자를 사용하는 저장 프로시듀어에 대한 링커 스크립트. **$LFLAGS_64** 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-X64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다. **-p 1024** 우선순위를 1024라는 임의의 값으로 설정합니다. **-o $1** 공유 라이브러리 파일로 출력을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 지정합니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 컴파일러는 /usr/lib:/lib 경로로 지정합니다. **-ldb2** 데이터베이스 관리자 라이브러리로 링크합니다. **-E $1.exp** 내보낼 파일을 지정합니다. 내보낼 파일에는 저장 프로시듀어의 목록이 들어 있습니다. **-e $2** 공유 라이브러리에 대한 시작점을 지정합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqC에서 샘플 프로그램 spserver를 빌드하려면 빌드 파일 이름, 프로그램 이름 및 공유 라이브러리에 대한 시작점이 되는 저장 프로시듀어 함수의 이름을 입력하십시오.

```
bldsrv spserver outlanguage
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver outlanguage database
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리 내의 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 161 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행한 다음 출력을 클라이언트 응용프로그램으로 리턴합니다.

사용자 정의 함수(UDF)

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ UDF 빌드에 대한 정보를 참조하십시오.

sqllib/samples/cpp에 있는 스크립트 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다. UDF는 Embedded SQL문을 포함할 수 없습니다. 따라서 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하고 사전 처리 컴파일하며 프로그램을 바인드할 필요가 없습니다.

매개변수 $1은 소스 파일의 이름을 지정합니다. 매개변수 $2는 공유 라이브러리에 대한 시작점이 되는 사용자 정의 함수(UDF)를 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름 $1을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- AIX
# Builds a C++ UDF library
# Usage: bldudf <prog_name>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-q64
  LFLAGS_64=-X64
else
  CFLAGS_64=
  LFLAGS_64=
fi
# Compile the program.
if [[ -f $1".c" ]]
then
  xlC $CFLAGS_64 -I$DB2PATH/include -c $1.c
elif [[ -f $1".C" ]]
then
  xlC $CFLAGS_64 -I$DB2PATH/include -c $1.C
fi
# Link using export file $1.exp, creating shared library $1
makeC++SharedLib $LFLAGS_64 -p 1024  -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2apie -E $1.exp
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

xlC IBM C Set++ 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-q64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다. 예를 들면, `$HOME/sqllib/include`와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

링크 옵션:

makeC++SharedLib

정적 생성자를 사용하는 저장 프로시듀어에 대한 링커 스크립트.

$LFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-X64" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-p 1024

우선순위를 1024라는 임의의 값으로 설정합니다.

-o $1 공유 라이브러리 파일로 출력을 지정합니다.

$1.o 프로그램 오브젝트 파일을 지정합니다.

-L$DB2PATH/lib

DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, `$HOME/sqllib/lib`와 같습니다. `-L` 옵션을 지정하지 않으면 컴파일러는 `/usr/lib:/lib` 경로로 지정합니다.

-ldb2 데이터베이스 관리자 라이브러리로 링크합니다.

-ldb2apie

LOB 위치 지정자를 사용할 수 있도록 DB2 API 엔진 라이브러리와 링크합니다.

-E $1.exp

내보낼 파일을 지정합니다. 내보낼 파일에는 저장 프로시듀어의 목록이 들어 있습니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오. UDF 작성에 대한 자세한 내용은 142 페이지의 『UDF 및 CREATE FUNCTION문』을 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 빌드 파일 이름, 프로그램 이름 및 공유 라이브러리에 대한 시작점이 되는 UDF 함수를 입력하십시오.

```
bldudf udfsrv ScalarUDF
```

스크립트 파일은 UDF를 경로 sqllib/function에 있는 서버로 복사합니다.

필요하면 DB2 인스턴스가 수행할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 스크립트 파일 bldapp를 사용하여 sqllib/samples/cpp에 있는 udfcli.sqC 소스 파일에서 udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 161 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

AIX 버전 4 및 5의 C++ 멀티스레드 응용프로그램은 xlC 컴파일러 대신 xlC_r 컴파일러를 사용하며, C의 경우 xlc 컴파일러 대신 xlc_r를 사용하여 컴파일하고 링크해야 합니다. _r 버전은 멀티스레드 컴파일에 대한 적절한 선행 처리기 정의를 설정하고 링커에 적절한 스레드 라이브러리 이름을 제공합니다.

멀티스레드 컴파일러 프론트 엔드를 사용하는 컴파일러 및 링크 플래그 설정에 대한 추가 정보는 사용자 컴파일러 문서에서 찾을 수 있습니다.

sqllib/samples/cpp에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 매개변수 $3은 데이터베이스에 대

한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- AIX
# Builds a C++ multi-threaded embedded SQL program
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
  embprep $1 $2 $3 $4
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-q64
else
  CFLAGS_64=
fi
# Compile the program.
xlC_r $CFLAGS_64 -I$DB2PATH/include -c $1.C
# Link the program.
xlC_r $CFLAGS_64 -o $1 $1.o -L$DB2PATH/lib -ldb2
```

위에서 언급한 xlC_r 컴파일러 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이 옵션에 대한 자세한 내용은 161 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqC에서 멀티스레드 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서 실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
thdsrver
```

VisualAge C++ 버전 4.0

이 VisualAge C++ 컴파일러는 AIX, OS/2 및 Windows 32비트 운영체제에 사용됩니다. 이 절에 나오는 정보는 이들 플랫폼에 모두 적용됩니다.

VisualAge C++ 컴파일러는 이 책에서 설명하는 다른 컴파일러와는 다릅니다. VisualAge C++ 버전 4.0을 사용하는 프로그램을 컴파일하려면 먼저 구성 파일을 작성해야 합니다. 자세한 내용은 컴파일러에 제공된 문서를 참조하십시오.

DB2는 VisualAge C++ 컴파일러를 사용하여 빌드할 수 있는 다양한 유형의 DB2 프로그램에 대한 구성 파일을 제공합니다. DB2 구성 파일을 사용하려면 먼저 환경 변수를 컴파일하려는 프로그램 이름으로 설정합니다. 그런 다음 VisualAge C++가 제공하는 명령을 사용하여 프로그램을 컴파일합니다. 다음은 DB2가 제공하는 구성 파일과 구성 파일을 사용하여 프로그램을 컴파일하는 방법에 대해 설명하는 절입니다.

cli.icc

DB2 CLI 구성 파일. 자세한 내용은 173 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

cliapi.icc

DB2 API 구성 파일이 있는 DB2 CLI. 자세한 내용은 176 페이지의 『DB2 API가 있는 DB2 CLI 응용프로그램』를 참조하십시오.

clis.icc

DB2 CLI 저장 프로시듀어 구성 파일. 자세한 내용은 178 페이지의 『DB2 CLI 저장 프로시듀어』을 참조하십시오.

api.icc

DB2 API 구성 파일. 자세한 내용은 181 페이지의 『DB2 API 응용프로그램』을 참조하십시오.

emb.icc

Embedded SQL 구성 파일. 자세한 내용은 183 페이지의 『Embedded SQL 응용프로그램』을 참조하십시오.

stp.icc

Embedded SQL 저장 프로시듀어 구성 파일. 자세한 내용은 185 페이지의 『Embedded SQL 저장 프로시듀어』를 참조하십시오.

udf.icc

사용자 정의 함수(UDF) 구성 파일. 자세한 내용은 188 페이지의 『사용자 정의 함수(UDF)』을 참조하십시오.

DB2 CLI 응용프로그램

AIX에서는 sqllib/samples/cli에 있으며 OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples\cli에 있는 구성 파일 cli.icc를 사용하여 DB2 CLI 프로그램을 빌드할 수 있습니다.

```
// cli.icc configuration file for DB2 CLI applications
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export CLI=prog_name'
// To use on OS/2 and Windows, enter: 'set CLI=prog_name'
// Then compile the program by entering: 'vacbld cli.icc'
if defined( $CLI )
{
  prog_name = $CLI
}
else
{
  error "Environment Variable CLI is not defined."
}
infile = prog_name".c"
utilcli = "utilcli.c"
if defined( $__TOS_AIX__ )
{
  // Set db2path to where DB2 will be accessed.
  // The default is the standard instance path.
  db2path     = $HOME"/sqllib"
  outfile     = prog_name
  group lib   = "libdb2.a"
  option opts = link( libsearchpath, db2path"/lib" ),
                incl( searchPath, db2path"/include" )
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
{
  db2path     = $DB2PATH
  outfile     = prog_name".exe"
  group lib   = "db2cli.lib"
  option opts = link( libsearchpath, db2path"\\lib" ),
```

```
                incl( searchPath, db2path"\\include" )
}
option opts
{
  target type(exe) outfile
  {
    source infile
    source utilcli
    source lib
  }
}
```

VisualAge C++ 버전 4.0은 설치된 운영 체제에 따라 환경 변수 __TOS_AIX__, __TOS_OS2__, __TOS_WIN__ 중 하나를 정의합니다.

구성 파일을 사용하여 소스 파일 tbinfo.c에서 DB2 CLI 샘플 프로그램 tbinfo를 빌드하려면 다음을 수행하십시오.

1. 다음을 입력하여 CLI 환경 변수를 프로그램 이름으로 설정하십시오.

   AIX에서:

   ```
   export CLI=tbinfo
   ```

   OS/2 및 Windows에서:

   ```
   set CLI=tbinfo
   ```

2. cli.icc 파일로 다른 프로그램을 빌드하여 생성된 cli.ics 파일이 작업 디렉토리에 있는 경우, 다음 명령을 사용하여 cli.ics 파일을 삭제하십시오.

   ```
   rm cli.ics
   ```

   다시 빌드하려는 동일한 프로그램에 대해 생성된 기존의 cli.ics 파일은 삭제할 필요가 없습니다.

3. 다음을 입력하여 샘플 프로그램을 컴파일하십시오.

   ```
   vacbld cli.icc
   ```

   주: vacbld 명령은 VisualAge C++ 버전 4.0에서 지원됩니다.

그 결과 실행 파일 tbinfo가 작성됩니다. 다음 실행 파일 이름을 입력하여 프로그램을 수행할 수 있습니다.

```
tbinfo
```

Embedded SQL 응용프로그램 빌드 및 수행

프로그램이 AIX에서 embprep 파일, OS/2에서 embprep.cmd 파일 또는 Windows 32비트 운영 체제에서 embprep.bat 파일로 사전 처리 컴파일된 후에 구성 파일을 사용합니다. embprep 파일은 소스 파일을 사전 처리 컴파일하고 프로그램을 데이터베이스에 바인드합니다. AIX에서 사전 처리 컴파일된 파일을 컴파일하려면 cli.icc 구성 파일을 사용합니다. OS/2 및 Windows에서 Embedded SQL 응용프로그램에는 cliapi.icc에 의해 링크된 db2api.lib 라이브러리가 필요하므로 cli.icc 파일 대신 cliapi.icc 파일을 사용해야 합니다.

소스 파일 dbusemx.sqc에서 Embedded SQL 응용프로그램 dbusemx을 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   embprep dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   embprep dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   embprep dbusemx database userid password
   ```

그 결과 사전 처리 컴파일된 C 파일 dbusemx.c가 작성됩니다.

사전 처리 컴파일된 후 C 파일은 다음과 같이 AIX의 cli.icc 파일 및 OS/2의 cliapi.icc 파일과 함께 컴파일될 수 있습니다.

1. 다음을 입력하여 CLI 환경 변수를 프로그램 이름으로 설정하십시오.

   AIX에서:

   ```
   export CLI=dbusemx
   ```

   OS/2 및 Windows에서:

   ```
   set CLIAPI=dbusemx
   ```

2. cli.icc 또는 cliapi.icc 파일로 다른 프로그램을 빌드하여 생성된 cli.ics 또는 cliapi.ics 파일이 작업 디렉토리에 있는 경우, 다음 명령을 사용하여 cli.ics 또는 cliapi.ics 파일을 삭제하십시오.

```
rm cli.ics
```

또는

```
rm cliapi.ics
```

다시 빌드하려는 동일한 프로그램에 대해 생성된 기존의 cli.ics 또는 cliapi.ics 파일은 삭제할 필요가 없습니다.

3. 다음을 입력하여 샘플 프로그램을 컴파일하십시오.

```
vacbld cli.icc
```

또는

```
vacbld cliapi.icc
```

주: vacbld 명령은 VisualAge C++ 버전 4.0에서 지원됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

```
dbusemx
```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

```
dbusemx database
```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
dbusemx database userid password
```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다. AIX에서는 sqllib/samples/cli에 있으며,

OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples\cli에 있는 구성 파일 cliapi.icc를 사용하면 DB2 API가 있는 DB2 CLI 프로그램을 빌드할 수 있습니다.

이 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 cli.icc 구성 파일 간의 유일한 차이점입니다.

소스 파일 dbmconn.c에서 DB2 CLI 샘플 프로그램 dbmconn을 빌드하려면 다음을 수행하십시오.

1. 다음을 입력하여 CLIAPI 환경 변수를 프로그램 이름으로 설정하십시오.

   AIX에서:

   ```
   export CLIAPI=dbmconn
   ```

   OS/2 및 Windows에서:

   ```
   set CLIAPI=dbmconn
   ```

2. cliapi.icc 파일로 다른 프로그램을 빌드하여 생성된 cliapi.ics 파일이 작업 디렉토리에 있는 경우, 다음 명령을 사용하여 cliapi.ics 파일을 삭제하십시오.

   ```
   rm cliapi.ics
   ```

   다시 빌드하려는 동일한 프로그램에 대해 생성된 기존의 cliapi.ics 파일은 삭제할 필요가 없습니다.

3. 다음을 입력하여 샘플 프로그램을 컴파일하십시오.

   ```
   vacbld cliapi.icc
   ```

   주: vacbld 명령은 VisualAge C++ 버전 4.0에서 지원됩니다.

그 결과 실행 파일 dbmconn이 작성됩니다. 다음 실행 파일 이름을 입력하여 프로그램을 수행할 수 있습니다.

```
dbmconn
```

DB2 CLI 저장 프로시듀어

AIX에서는 sqllib/samples/cli에 있으며 OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples\cli에 있는 구성 파일 clis.icc를 사용하여 DB2 CLI 저장 프로시듀어를 빌드할 수 있습니다.

```
// clis.icc configuration file for DB2 CLI stored procedures
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export CLIS=prog_name'
// To use on OS/2 and Windows, enter: 'set CLIS=prog_name'
// Then compile the program by entering: 'vacbld clis.icc'
if defined( $CLIS )
{
  prog_name = $CLIS
}
else
{
  error "Environment Variable CLIS is not defined."
}
infile = prog_name".c"
utilcli = "utilcli.c"
expfile = prog_name".exp"
if defined( $__TOS_AIX__ )
{
  // Set db2path to where DB2 will be accessed.
  // The default is the standard instance path.
  db2path     = $HOME"/sqllib"
  outfile     = prog_name
  group lib   = "libdb2.a"
  option opts = link( exportList, expfile ),
                link( libsearchpath, db2path"/lib" ),
                incl( searchPath, db2path"/include" )
  cpcmd       = "cp"
  funcdir     = db2path"/function"
}
else /* if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ ) */
{
  db2path     = $DB2PATH
  outfile     = prog_name".dll"
  if defined( $__TOS_WIN__ )
  {
    expfile = prog_name"v4.exp"
  }
  group lib   = "db2cli.lib"
  option opts = link( exportList, expfile ),
                link( libsearchpath, db2path"\\lib" ),
                incl( searchPath, db2path"\\include" )
```

```
  cpcmd       = "copy"
  funcdir     = db2path"\\function"
}
option opts
{
  target type(dll) outfile
  {
    source infile
    source utilcli
    source lib
  }
}
if defined( $__TOS_AIX__ )
{
  rmcmd       = "rm -f"
  run after rmcmd " " funcdir "/" outfile
}
run after cpcmd " " outfile " " funcdir
```

VisualAge C++ 버전 4.0은 설치된 운영 체제에 따라 환경 변수 __TOS_AIX__, __TOS_OS2__, __TOS_WIN__ 중 하나를 정의합니다.

이 구성 파일을 사용하여 소스 파일 spserver.c에서 DB2 CLI 저장 프로시듀어 spserver를 빌드하려면 다음을 수행하십시오.

1. 다음을 입력하여 CLIS 환경 변수를 프로그램 이름으로 설정하십시오.

   AIX에서:

   ```
   export CLIS=spserver
   ```

   OS/2 및 Windows에서:

   ```
   set CLIS=spserver
   ```

2. clis.icc 파일로 다른 프로그램을 빌드하여 생성된 clis.ics 파일이 작업 디렉토리에 있는 경우, 다음 명령을 사용하여 clis.ics 파일을 삭제하십시오.

   ```
   rm clis.ics
   ```

   다시 빌드하려는 동일한 프로그램에 대해 생성된 기존의 clis.ics 파일은 삭제할 필요가 없습니다.

3. 다음을 입력하여 샘플 프로그램을 컴파일하십시오.

   ```
   vacbld clis.icc
   ```

주: vacbld 명령은 VisualAge C++ 버전 4.0에서 지원됩니다.

저장 프로시듀어는 AIX에서는 경로 sqllib/function, OS/2 및 Windows 32 비트 운영 체제에서는 경로 %DB2PATH%\function에 있는 서버로 복사됩니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 사용하여 데이터베이스에 연결하십시오.

```
db2 connect to sample userid password
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 저장 프로시듀어 spserver를 빌드하면 이 저장 프로시듀어를 호출하는 CLI 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다. 구성 파일 cli.icc를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 173 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 원격 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

DB2 API 응용프로그램

AIX에서는 sqllib/samples/c 및 sqllib/samples/cpp에 있으며 OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 구성 파일 api.icc를 사용하여 C 및 C++에서 DB2 API 프로그램을 빌드할 수 있습니다.

```
// api.icc configuration file for DB2 API programs
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export API=prog_name'
// To use on OS/2 and Windows, enter: 'set API=prog_name'
// Then compile the program by entering: 'vacbld api.icc'
if defined( $API )
{
  prog_name = $API
}
else
{
  error "Environment Variable API is not defined."
}
infile = prog_name".c"
util   = "utilapi.c"
if defined( $__TOS_AIX__ )
{
  // Set db2path to where DB2 will be accessed.
  // The default is the standard instance path.
  db2path     = $HOME"/sqllib"
  outfile     = prog_name
  group lib   = "libdb2.a"
  option opts = link( libsearchpath, db2path"/lib" ),
                incl( searchPath, db2path"/include" )
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
{
  db2path     = $DB2PATH
  outfile     = prog_name".exe"
```

```
  group lib   = "db2api.lib"
  option opts = link( libsearchpath, db2path"\\lib" ),
                incl( searchPath, db2path"\\include" )
}
option opts
{
  target type(exe) outfile
  {
    source infile
    source util
    source lib
  }
}
```

VisualAge C++ 버전 4.0은 설치된 운영 체제에 따라 환경 변수 __TOS_AIX__, __TOS_OS2__, __TOS_WIN__ 중 하나를 정의합니다.

구성 파일을 사용하여 client.c 소스 파일에서 DB2 API 샘플 프로그램 client를 빌드하려면 다음을 수행하십시오.

1. 다음을 입력하여 API 환경 변수를 프로그램 이름으로 설정하십시오.

   AIX에서:

   ```
   export API=client
   ```

   OS/2 및 Windows에서:

   ```
   set API=client
   ```

2. api.icc 파일로 다른 프로그램을 빌드하여 생성된 api.ics 파일이 작업 디렉토리에 있는 경우, 다음 명령을 사용하여 api.ics 파일을 삭제하십시오.

   ```
   rm api.ics
   ```

   다시 빌드하려는 동일한 프로그램에 대해 생성된 기존의 api.ics 파일은 삭제할 필요가 없습니다.

3. 다음을 입력하여 샘플 프로그램을 컴파일하십시오.

   ```
   vacbld api.icc
   ```

   주: vacbld 명령은 VisualAge C++ 버전 4.0에서 지원됩니다.

그 결과 실행 파일 client가 작성됩니다. 다음 실행 파일 이름을 입력하여 프로그램을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램

AIX에서는 sqllib/samples/c 및 sqllib/samples/cpp에 있으며 OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 구성 파일 구성 파일 emb.icc를 사용하여 C 및 C++에서 DB2 Embedded SQL 응용프로그램을 빌드할 수 있습니다.

```
// emb.icc configuration file for embedded SQL applications
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export EMB=prog_name'
// To use on OS/2 and Windows, enter: 'set EMB=prog_name'
// Then compile the program by entering: 'vacbld emb.icc'
if defined( $EMB )
{
  prog_name = $EMB
}
else
{
  error "Environment Variable EMB is not defined."
}
// To connect to another database, replace "sample"
// For user ID and password, update 'user' and 'passwd'
// and take out the comment in the line: 'run before "embprep "'
dbname = "sample"
user   = ""
passwd = ""
// Precompiling the source program file
run before "embprep " prog_name " " dbname // " " user " " passwd
infile = prog_name".c"
util = "utilemb.sqc"
if defined( $__TOS_AIX__ )
{
  // Set db2path to where DB2 will be accessed.
  // The default is the standard instance path.
  db2path     = $HOME"/sqllib"
  outfile     = prog_name
  group lib   = "libdb2.a"
  option opts = link( libsearchpath, db2path"/lib" ),
                incl( searchPath, db2path"/include" )
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
{
  db2path     = $DB2PATH
  outfile     = prog_name".exe"
```

```
  group lib   = "db2api.lib"
  option opts = link( libsearchpath, db2path"\\lib" ),
                incl( searchPath, db2path"\\include" )
}
option opts
{
  target type(exe) outfile
  {
    source infile
    source util
    source lib
  }
}
```

VisualAge C++ 버전 4.0은 설치된 운영 체제에 따라 환경 변수 __TOS_AIX__, __TOS_OS2__, __TOS_WIN__ 중 하나를 정의합니다.

구성 파일을 사용하여 updat.sqc 소스 파일에서 Embedded SQL 응용프로그램 updat를 빌드하려면 다음을 수행하십시오.

1. 다음을 입력하여 EMB 환경 변수를 프로그램 이름으로 설정하십시오.

   AIX에서:

   ```
   export EMB=updat
   ```

   OS/2 및 Windows에서:

   ```
   set EMB=updat
   ```

2. emb.icc 파일로 다른 프로그램을 빌드하여 생성된 emb.ics 파일이 작업 디렉토리에 있는 경우, 다음 명령을 사용하여 emb.ics 파일을 삭제하십시오.

   ```
   rm emb.ics
   ```

   다시 빌드하려는 동일한 프로그램에 대해 생성된 기존의 emb.ics 파일은 삭제할 필요가 없습니다.

3. 다음을 입력하여 샘플 프로그램을 컴파일하십시오.

   ```
   vacbld emb.icc
   ```

   주: vacbld 명령은 VisualAge C++ 버전 4.0에서 지원됩니다.

그 결과 실행 파일 updat가 작성됩니다. 다음 실행 파일 이름을 입력하여 프로그램을 수행할 수 있습니다.

```
updat
```

Embedded SQL 저장 프로시듀어

AIX에서는 sqllib/samples/c 및 sqllib/samples/cpp에 있으며 OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 구성 파일 stp.icc를 사용하여 C 및 C++에서 DB2 Embedded SQL 저장 프로시듀어를 빌드할 수 있습니다.

```
// stp.icc configuration file for embedded SQL stored procedures
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export STP=prog_name'
// To use on OS/2 and Windows, enter: 'set STP=prog_name'
// Then compile the program by entering: 'vacbld emb.icc'
if defined( $STP )
{
  prog_name = $STP
}
else
{
  error "Environment Variable STP is not defined."
}
// To connect to another database, replace "sample"
// For user ID and password, update 'user' and 'passwd'
// and take out the comment in the line: 'run before "embprep "'
dbname = "sample"
user   = ""
passwd = ""
// Precompiling the source program file
run before "embprep " prog_name " " dbname // " " user " " passwd
infile = prog_name".c"
expfile = prog_name".exp"
if defined( $__TOS_AIX__ )
{
  // Set db2path to where DB2 will be accessed.
  // The default is the standard instance path.
  db2path     = $HOME"/sqllib"
  outfile     = prog_name
  group lib   = "libdb2.a"
  option opts = link( exportList, expfile ),
                link( libsearchpath, db2path"/lib" ),
                incl( searchPath, db2path"/include" )
  cpcmd       = "cp"
  funcdir     = db2path"/function"
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
```

```
{
  db2path     = $DB2PATH
  outfile     = prog_name".dll"
  if defined( $__TOS_WIN__ )
  {
    expfile = prog_name"v4.exp"
  }
  group lib   = "db2api.lib"
  option opts = link( exportList, expfile ),
                link( libsearchpath, db2path"\\lib" ),
                incl( searchPath, db2path"\\include" )
  cpcmd       = "copy"
  funcdir     = db2path"\\function"
}
option opts
{
  target type(dll) outfile
  {
    source infile
    source lib
  }
}
if defined( $__TOS_AIX__ )
{
  rmcmd       = "rm -f"
  run after rmcmd " " funcdir "/" outfile
}
run after cpcmd " " outfile " " funcdir
```

VisualAge C++ 버전 4.0은 설치된 운영 체제에 따라 환경 변수 __TOS_AIX__, __TOS_OS2__, __TOS_WIN__ 중 하나를 정의합니다.

구성 파일을 사용하여 소스 파일 spserver.sqc에서 Embedded SQL 저장 프로시듀어 spserver를 빌드하려면 다음을 수행하십시오.

1. 다음을 입력하여 STP 환경 변수를 프로그램 이름으로 설정하십시오.

   AIX에서:

   ```
   export STP=spserver
   ```

   OS/2 및 Windows에서:

   ```
   set STP=spserver
   ```

2. stp.icc 파일로 다른 프로그램을 빌드하여 생성된 stp.ics 파일이 작업 디렉토리에 있는 경우, 다음 명령을 사용하여 stp.ics 파일을 삭제하십시오.

```
rm stp.ics
```

다시 빌드하려는 동일한 프로그램에 대해 생성된 기존의 stp.ics 파일은 삭제할 필요가 없습니다.

3. 다음을 입력하여 샘플 프로그램을 컴파일하십시오.

```
vacbld stp.icc
```

주: vacbld 명령은 VisualAge C++ 버전 4.0에서 지원됩니다.

저장 프로시듀어는 AIX에서는 경로 sqllib/function, OS/2 및 Windows 32 비트 운영 체제에서는 경로 %DB2PATH%\function에 있는 서버로 복사됩니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 저장 프로시듀어 spserver를 빌드하면 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다. 구성 파일 emb.icc를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 183 페이지의 『Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 원격 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

사용자 정의 함수(UDF)

AIX에서는 sqllib/samples/c 및 sqllib/samples/cpp에 있으며 OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 구성 파일 udf.icc를 사용하여 C 및 C++에서 사용자 정의 함수(UDF)를 빌드할 수 있습니다.

```
// udf.icc configuration file for user-defined functions
// for VisualAge C++ Version 4.0
// To use on AIX, enter: 'export UDF=prog_name'
// To use on OS/2 and Windows, enter: 'set UDF=prog_name'
// Then compile the program by entering: 'vacbld udf.icc'
if defined( $UDF )
{
  prog_name = $UDF
}
else
{
  error "Environment Variable UDF is not defined."
}
infile = prog_name".c"
expfile = prog_name".exp"
if defined( $__TOS_AIX__ )
{
  // Set db2path to where DB2 will be accessed.
  // The default is the standard instance path.
  db2path     = $HOME"/sqllib"
  outfile     = prog_name
  group lib   = "libdb2.a", "libdb2apie.a"
```

```
  option opts = link( exportList, expfile ),
                link( libsearchpath, db2path"/lib" ),
                incl( searchPath, db2path"/include" )
  cpcmd       = "cp"
  funcdir     = db2path"/function"
}
else // if defined( $__TOS_OS2__ ) | defined( $__TOS_WIN__ )
{
  db2path     = $DB2PATH
  outfile     = prog_name".dll"
  if defined( $__TOS_WIN__ )
  {
    expfile = prog_name"v4.exp"
  }
  group lib   = "db2api.lib", "db2apie.lib"
  option opts = link( exportList, expfile ),
                link( libsearchpath, db2path"\\lib" ),
                incl( searchPath, db2path"\\include" )
  cpcmd       = "copy"
  funcdir     = db2path"\\function"
}
option opts
{
  target type(dll) outfile
  {
    source infile
    source lib
  }
}
if defined( $__TOS_AIX__ )
{
  rmcmd       = "rm -f"
  run after rmcmd " " funcdir "/" outfile
}
run after cpcmd " " outfile " " funcdir
```

VisualAge C++ 버전 4.0은 설치된 운영 체제에 따라 환경 변수 __TOS_AIX__, __TOS_OS2__, __TOS_WIN__ 중 하나를 정의합니다.

구성 파일을 사용하여 소스 파일 udf.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 수행하십시오.

1. 다음을 입력하여 UDF 환경 변수를 프로그램 이름으로 설정하십시오.

   AIX에서:

   ```
   export UDF=udfsrv
   ```

OS/2 및 Windows에서:

```
set UDF=udfsrv
```

2. udf.icc 파일로 다른 프로그램을 빌드하여 생성된 udf.ics 파일이 작업 디렉토리에 있는 경우, 다음 명령을 사용하여 udf.ics 파일을 삭제하십시오.

   ```
   rm udf.ics
   ```

   다시 빌드하려는 동일한 프로그램에 대해 생성된 기존의 udf.ics 파일은 삭제할 필요가 없습니다.

3. 다음을 입력하여 샘플 프로그램을 컴파일하십시오.

   ```
   vacbld udf.icc
   ```

   주: vacbld 명령은 VisualAge C++ 버전 4.0에서 지원됩니다.

UDF 라이브러리는 경로 sqllib/function에 있는 서버로 복사됩니다.

필요하면 DB2 인스턴스가 실행할 수 있도록 사용자 정의 함수(UDF)에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 이 프로그램의 Embedded SQL 버전 및 DB2 CLI가 제공됩니다.

구성 파일 cli.icc를 사용하여 AIX에서는 sqllib/samples/cli에 있으며 OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples\cli에 있는 소스 파일 udfcli.c에서 DB2 CLI udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 173 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

구성 파일 emb.icc를 사용하여 AIX에서는 sqllib/samples/c에 있으며 OS/2 및 Windows 32비트 운영 체제에서는 %DB2PATH%\samples\cli에 있는 소스 파일 udfcli.sqc에서 Embedded SQL udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 183 페이지의 『Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

AIX용 IBM COBOL Set

이 절에서는 다음 주제에 대해 설명합니다.

- 컴파일러 사용
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어

컴파일러 사용

Embedded SQL 및 DB2 API 호출을 포함하는 응용프로그램을 개발하며 AIX용 IBM COBOL Set 컴파일러를 사용 중인 경우, 다음 사항에 유의하십시오.

- 명령행 처리기 명령 db2 prep를 사용하여 응용프로그램을 사전 처리 컴파일할 때 target ibmcob 옵션을 사용하십시오.
- 소스 파일에서 탭 문자를 사용하지 마십시오.
- 소스 파일의 첫 번째 행에 PROCESS 및 CBL 키워드를 사용하여 컴파일 옵션을 설정할 수 있습니다.
- 응용프로그램에 Embedded SQL만 들어 있고 DB2 API 호출이 들어 있지 않을 경우, pgmname(mixed) 컴파일 옵션을 사용할 필요가 없습니다. DB2 API 호출을 사용할 경우에는 pgmname(mixed) 컴파일 옵션을 사용해야 합니다.
- AIX용 IBM COBOL Set의 "System/390 호스트 데이터 유형 지원" 기능을 사용 중인 경우, 응용프로그램에 대한 DB2 포함 파일은 다음 디렉토리에 있습니다.

  ```
  $HOME/sqllib/include/cobol_i
  ```

  제공된 스크립트 파일을 사용하여 DB2 샘플 프로그램을 빌드 중인 경우, 스크립트 파일에 지정된 포함 파일 경로를 cobol_a 디렉토리가 아닌 cobol_i 디렉토리를 가리키도록 변경해야 합니다.

  AIX용 IBM COBOL Set의 "System/390 호스트 데이터 유형 지원" 기능을 사용하지 않거나 해당 컴파일러의 이전 버전을 사용 중인 경우, 응용프로그램에 대한 DB2 포함 파일은 다음 디렉토리에 있습니다.

```
$HOME/sqllib/include/cobol_a
```

다음과 같이 COPY 파일 이름이 .cbl 확장자를 포함하도록 지정하십시오.

```
COPY "sql.cbl".
```

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/cobol에 있는 빌드 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수, $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- AIX
# Builds an IBM COBOL application program
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqb" ]]
then
embprep $1 $2 $3 $4
fi

# Compile the checkerr.cbl error checking utility.
cob2 -qpgmname\(mixed\) -qlib -I$DB2PATH/include/cobol_a \
     -c checkerr.cbl

# Compile the program.
cob2 -qpgmname\(mixed\) -qlib -I$DB2PATH/include/cobol_a \
     -c $1.cbl
```

```
# Link the program.
cob2 -o $1 $1.o checkerr.o -ldb2 -L$DB2PATH/lib
```

bldapp에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cob2** IBM COBOL Set 컴파일러. **-qpgmname\(mixed\)** 컴파일러에 대소문자가 혼합된 이름으로 라이브러리 시작점에 대한 CALL을 허용하도록 지시합니다. **-qlib** COPY문을 처리할 컴파일러를 지시합니다. **-I$DB2PATH/include/cobol_a** DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include/cobol_a와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.
링크 옵션: **cob2** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-o $1** 실행 프로그램을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 지정합니다. **checkerr.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-ldb2** 데이터베이스 관리자 라이브러리로 링크합니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 컴파일러는 /usr/lib:/lib 경로로 지정합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.cbl에서 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다. 다음을 입력하여 sample 데이터베이스에 대해서 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqb에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

sqllib/samples/cobol에 있는 스크립트 파일 bldsrv에 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 필요 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

스크립트 파일은 공유 라이브러리 이름과 공유 라이브러리의 시작점에 대해 소스 파일 이름 $1을 사용합니다. 시작점 함수 이름이 소스 파일 이름과 다른 저장 프로시듀어를 빌드 중인 경우, 시작점에 대한 또 다른 매개변수를 승인하도록 스크립트 파일을 수정할 수 있습니다. 데이터베이스 매개변수 이름을 $3으로 바꾸는 것이 좋습니다. 그런 다음 시작점 링크 옵션을 -e $2로 변경하여 스크립트 파일을 수행할 때 명령행에서 추가 매개변수를 지정할 수 있습니다.

```
#! /bin/ksh
# bldsrv script file -- AIX
# Builds an IBM COBOL stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2
# Compile the checkerr.cbl error checking utility.
cob2 -qpgmname\(mixed\) -qlib -I$DB2PATH/include/cobol_a \
     -c checkerr.cbl

# Compile the program.
cob2 -qpgmname\(mixed\) -qlib -c -I$DB2PATH/include/cobol_a $1.cbl

# Link the program using the export file $1.exp
# creating shared library $1 with entry point $1.
cob2 -o $1 $1.o checkerr.o -H512 -T512 -e $1 -bE:$1.exp \
     -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory of the DB2 instance.
# This assumes the user has write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cob2 IBM COBOL Set 컴파일러.

-qpgmname\(mixed\)
컴파일러에 대소문자가 혼합된 이름으로 라이브러리 시작점에 대한 CALL을 허용하도록 지시합니다.

-qlib COPY문을 처리할 컴파일러를 지시합니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

-I$DB2PATH/include/cobol_a
DB2 포함 파일의 위치를 지정합니다. 예를 들면, `$HOME/sqllib/include/cobol_a`와 같습니다.

링크 옵션:

cob2 편집기를 링크하려면 컴파일러를 사용합니다.

-o $1 공유 라이브러리 파일로 출력을 지정합니다.

$1.o 저장 프로시듀어 오브젝트 파일을 지정합니다.

checkerr.o
오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다.

-H512 출력 파일 정렬을 지정합니다.

-T512 주소를 시작하는 출력 파일 텍스트 세그먼트를 지정합니다.

-e $1 기본 시작점을 공유 라이브러리로 지정합니다.

-bE:$1.exp
내보낼 파일을 지정합니다. 내보낼 파일에는 저장 프로시듀어의 목록이 들어 있습니다.

-L$DB2PATH/lib
DB2 런타임 공유 라이브러리의 위치를 지정합니다.
예를 들면, `$HOME/sqllib/lib`와 같습니다. `-L` 옵션을 지정하지 않으면, 컴파일러는 `/usr/lib:/lib` 경로로 지정합니다.

-ldb2 데이터베이스 관리자 라이브러리로 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

샘플 데이터베이스에 연결하여 소스 파일 outsrv.sqb에서 샘플 프로그램 outsrv를 빌드하려면 다음을 입력하십시오.

```
bldsrv outsrv
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 포함시키십시오.

```
bldsrv outsrv database
```

스크립트 파일은 저장 프로시듀어를 경로 sqllib/function에 있는 서버로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 저장 프로시듀어에 대한 파일 모드를 설정하십시오.

일단 저장 프로시듀어 outsrv를 빌드하면 사용자는 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 outcli를 빌드할 수 있습니다 스크립트 파일 bldapp를 사용하여 outcli를 빌드할 수 있습니다. 자세한 내용은 192 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

```
outcli database userid password
```

여기서

database
: 연결하려는 데이터베이스의 이름입니다. 이름은 sample, 원격 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
: 유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 outsrv에 액세스하고 서버 데이터베이스에서 동일한 이름의 저장 프로시듀어 함수를 실행한 다음 출력을 클라이언트 응용프로그램으로 리턴합니다.

Micro Focus COBOL

이 절에서는 다음 주제에 대해 설명합니다.

- 컴파일러 사용
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어

컴파일러 사용

Embedded SQL 및 DB2 API 호출을 포함하는 응용프로그램을 개발하며 Micro Focus COBOL 컴파일러를 사용 중인 경우, 다음 사항에 유의하십시오.

- 명령행 처리기 명령 `db2 prep`를 사용하여 응용프로그램을 사전 처리 컴파일할 때 `target mfcob` 옵션(기본값)을 사용하십시오.
- Micro Focus COBOL 버전 4.0 및 4.1과 함께 내장된 사전 처리 컴파일러 프론트 엔드, 런타임 인터프리터 또는 애니메이터 디버거를 사용하려면 다음과 같이 Micro Focus가 제공하는 `mkrts` 명령을 실행하여 DB2 Generic API 시작점을 Micro Focus 런타임 모듈 `rts32`에 추가하십시오.

  1. 루트로 로그인하십시오.
  2. 다음 디렉토리에 제공되는 인수를 사용하여 `mkrts`를 실행하십시오.

     ```
     /usr/lpp/db2_07_01/lib/db2mkrts.args
     ```

  주: 위의 사항은 Micro Focus Server Express에 적용되지 않습니다.
- DB2 COBOL COPY 파일 디렉토리를 Micro Focus COBOL 환경 변수 COBCPY에 포함시켜야 합니다. COBCPY 환경 변수는 COPY 파일의 위치를 지정합니다. Micro Focus COBOL용 DB2 COPY 파일은 데이터베이스 인스턴스 디렉토리의 `sqllib/include/cobol_mf`에 있습니다.

  이 디렉토리를 포함시키려면 다음을 입력하십시오.

  ```
  export COBCPY=$COBCPY:$HOME/sqllib/include/cobol_mf
  ```

  주: `.profile` 파일에서 COBCPY를 설정하려 할 수 있습니다.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/cobol_mf에 있는 빌드 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 매개변수 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- AIX
# Builds a Micro Focus COBOL application program
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqb" ]]
then
embprep $1 $2 $3 $4
fi

# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$DB2PATH/include/cobol_mf:$COBCPY

# Compile the checkerr.cbl error checking utility.
cob -c -x checkerr.cbl

# Compile the program.
cob -c -x $1.cbl

# Link the program.
cob -x -o $1 $1.o checkerr.o -ldb2 -ldb2gmf -L$DB2PATH/lib
```

bldmfapi에 대한 컴파일 및 링크 옵션	
컴파일 옵션:	
cob	COBOL 컴파일러.
-c	컴파일만 수행하고 링크는 수행하지 않습니다.
-x	실행 프로그램을 만듭니다.

bldmfapi에 대한 컴파일 및 링크 옵션	
링크 옵션:	
cob	링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
-x	실행 프로그램을 만듭니다.
-o $1	실행 프로그램을 지정합니다.
$1.o	프로그램 오브젝트 파일을 지정합니다.
-ldb2	DB2 라이브러리로 링크합니다.
-ldb2gmf	Micro Focus COBOL용 DB2 예외 핸들러 라이브러리로 링크합니다.
-L$DB2PATH/lib	DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 컴파일러는 /usr/lib:/lib 경로로 지정합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.	

소스 파일 client.cbl에서 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다. 다음을 입력하여 sample 데이터베이스에 대해서 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqb에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

주:

1. Micro Focus 4.1 컴파일러를 사용하여 AIX 4.2.1에 저장 프로시듀어를 빌드하기 전에 다음 명령을 실행하십시오.

   ```
   db2stop
   db2set DB2LIBPATH=$LIBPATH
   db2set DB2ENVLIST="COBDIR LIBPATH"
   db2set
   db2start
   ```

   db2stop이 데이터베이스를 중지했는지와 LIBPATH가 쉘 환경에서 적절히 설정되어 있는지 확인하십시오. 마지막 db2set 명령이 발행되어 설정을 표시합니다. DB2LIBPATH 및 DB2ENVLIST가 제대로 설정되어 있는지 확인하십시오.

2. AIX 버전 4 플랫폼에서 사용되는, 일부 최신 버전의 Micro Focus COBOL 컴파일러는 정적으로 링크된 저장 프로시듀어를 작성하는 데 사용할 수 없습

니다. 따라서 makefile과 스크립트 파일 bldsrv가 동적으로 링크된 저장 프로시듀어의 작성을 허용하도록 수정되었습니다.

원격 클라이언트 응용프로그램이 이러한 동적으로 링크된 정적 프로시듀어를 정상적으로 호출하려면 저장 프로시듀어를 실행하기 직전에 저장 프로시듀어가 상주하는 서버에서 Micro Focus COBOL 루틴 cobinit()를 호출해야 합니다. 이를 수행하는 랩퍼 프로그램이 makefile 또는 스크립트 파일 bldsrv의 실행 중에 작성됩니다. 그런 다음 저장 프로시듀어 코드로 링크되어 저장 프로시듀어 공유 라이브러리를 형성합니다. 이러한 랩퍼 프로그램의 사용으로 인해 클라이언트 응용프로그램이 x라는 저장 프로시듀어를 호출하려면 x 대신 x_wrap를 호출해야 합니다.

랩퍼 프로그램의 세부사항을 이 절의 뒷 부분에서 설명합니다.

sqllib/samples/cobol_mf에 있는 스크립트 파일 bldsrv에 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 필요 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

스크립트 파일은 공유 라이브러리 이름과 공유 라이브러리의 시작점에 대하여 소스 파일 이름 $1을 사용합니다. 시작점 함수 이름이 소스 파일 이름과 다른 저장 프로시듀어를 빌드 중인 경우, 시작점에 대한 또 다른 매개변수를 승인하도록 스크립트 파일을 수정할 수 있습니다. 데이터베이스 매개변수 이름을 $3으로 바꾸는 것이 좋습니다. 그런 다음 시작점 링크 옵션을 -e $2로 변경하여 스크립트 파일을 수행할 때 명령행에서 추가 매개변수를 지정할 수 있습니다.

```
#! /bin/ksh
# bldsrv script file -- AIX
# Builds a Micro Focus COBOL stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
```

```
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2

# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$DB2PATH/include/cobol_mf:$COBCPY
# Compile the program.
cob -c -x $1.cbl

# Create the wrapper program for the stored procedure.
wrapsrv $1
# Link the program using export file ${1}_wrap.exp
# creating shared library $1 with entry point ${1}_wrap.
cob -x -o $1 ${1}_wrap.c $1.o -Q -bE:${1}_wrap.exp -Q "-e $1" \
-Q -bI:$DB2PATH/lib/db2g.imp -ldb2gmf -L$DB2PATH/lib
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션	
컴파일 옵션:	
cob	COBOL 컴파일러.
-c	컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.
-x	실행 프로그램을 만듭니다.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션: **cob** 편집기를 링크하려면 컴파일러를 사용합니다. **-x** 실행 프로그램을 만듭니다. **-o $1** 실행 프로그램을 지정합니다. **-o ${1}_wrap.c** 랩퍼 프로그램을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 지정합니다. **-Q -bE:${1}_wrap.exp** 내보낼 파일을 지정합니다. 내보낼 파일에는 저장 프로시듀어 시작점의 목록이 들어 있습니다. 저장 프로시듀어가 x이면 해당 시작점은 x_wrap입니다. **-Q "-e $1"** 기본 시작점을 공유 라이브러리로 지정합니다. **-Q -bI:$DB2PATH/lib/db2g.imp** DB2 응용프로그램 라이브러리에 대한 시작점 목록을 제공합니다. **-ldb2gmf** Micro Focus COBOL용 DB2 예외 핸들러 라이브러리로 링크합니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면, 컴파일러는 /usr/lib:/lib 경로로 지정합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

랩퍼 프로그램 wrapsrv로 인해 저장 프로시듀어가 실행되기 직전에 Micro Focus COBOL 루틴 cobinit()가 호출됩니다. 내용은 다음과 같습니다.

```
#! /bin/ksh
# wrapsrv script file
# Creates the wrapper program for Micro Focus COBOL stored procedures
# Usage: wrapsrv <stored_proc>
# Note: The client program calls "<stored_proc>_wrap" not "<stored_proc>"

# Create the wrapper program for the stored procedure.
cat << WRAPPER_CODE > ${1}_wrap.c
#include <stdio.h>
void cobinit(void);
int $1(void *p0, void *p1, void *p2, void *p3);
int main(void)
{
  return 0;
}
int ${1}_wrap(void *p0, void *p1, void *p2, void *p3)
{
  cobinit();
  return $1(p0, p1, p2, p3);
}
WRAPPER_CODE
# Create the export file for the wrapper program
echo $1_wrap > ${1}_wrap.exp
```

소스 파일 outsrv.sqb에서 샘플 프로그램 outsrv를 빌드하려면 샘플 데이터베이스에 연결하고 있는 경우, 다음을 입력하십시오.

```
bldsrv outsrv
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv outsrv database
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 저장 프로시듀어 outsrv를 빌드하면 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 outcli를 빌드할 수 있습니다. 스크립트 파일 bldapp를 사용하여 outcli를 빌드할 수 있습니다. 자세한 내용은 199 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

```
outcli database userid password
```

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 outsrv에 액세스하고, 서버 데이터베이스에서 동일한 이름의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

저장 프로시듀어 나감

저장 프로시듀어를 개발할 때 다음 명령문을 사용하여 저장 프로시듀어를 나가십시오.

```
move SQLZ-HOLD-PROC to return-code.
```

이 명령문을 통해 저장 프로시듀어가 클라이언트 응용프로그램으로 올바로 리턴합니다. 이것은 저장 프로시듀어가 지역 COBOL 클라이언트 응용프로그램에 의해 호출될 때 특히 중요합니다.

REXX

REXX 프로그램을 사전 처리 컴파일하거나 바인드하지 마십시오.

AIX에서 DB2 REXX/SQL 프로그램을 수행하려면 DB2 설치 디렉토리에 sqllib/lib를 포함시키도록 LIBPATH 환경 변수를 설정해야 합니다.

다음을 입력하십시오.

```
export LIBPATH=$LIBPATH:/lib:/usr/lib:/usr/lpp/db2_07_01/sqllib/lib
```

AIX에서 응용프로그램 파일은 임의의 파일 확장자를 가질 수 있습니다. 다음 두 가지 방법 중 하나를 사용하여 응용프로그램을 수행할 수 있습니다.

1. 쉘 명령 프롬프트에서 rexx *name*을 입력하십시오. 여기서 *name*은 REXX 프로그램의 이름입니다.

2. REXX 프로그램의 첫 번째 행은 "마법 수"(#!)를 포함하고 REXX/6000 인터프리터가 상주하는 디렉토리를 식별하는 경우, 쉘 명령 프롬프트에서 명령을 입력하여 REXX 프로그램을 수행할 수 있습니다. 예를 들어, REXX/6000 인터프리터 파일이 /usr/bin 디렉토리에 있는 경우, REXX 프로그램의 첫 번째 행으로 다음을 포함시키십시오.

   ```
   #! /usr/bin/rexx
   ```

   그런 다음 쉘 명령 프롬프트에서 다음 명령을 입력하여 프로그램을 실행 가능하게 하십시오.

   ```
   chmod +x name
   ```

   쉘 명령 프롬프트에서 파일 이름을 입력하여 REXX 프로그램을 수행하십시오.

REXX 샘플 프로그램은 sqllib/samples/rexx 디렉토리에 있습니다. 샘플 REXX 프로그램 updat.cmd를 수행하려면 다음 중 하나를 수행하십시오.

- 존재하지 않는 경우 "#! /usr/bin/rexx" 행을 프로그램 소스 파일의 맨 위에 추가하십시오. 그런 다음, 다음을 입력하여 프로그램을 직접 수행하십시오.

  ```
  updat.cmd
  ```

- 다음을 입력하여 REXX 인터프리터와 프로그램을 지정하십시오.

  ```
  rexx updat.cmd
  ```

REXX 및 DB2에 대한 자세한 내용은 *응용프로그램 개발 안내서*의 "REXX 프로그래밍"을 참조하십시오.

제7장 HP-UX 응용프로그램 빌드

이 장에서는 HP-UX에서의 DB2 응용프로그램 빌드에 대한 자세한 정보를 제공합니다. 스크립트 파일에서 db2로 시작되는 명령은 명령행 처리기(CLP) 명령입니다. CLP 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

HP-UX에 대한 최신 DB2 응용프로그램 개발 갱신사항에 대대한 자세한 내용은 다음의 DB2 응용프로그램 개발 웹 페이지를 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

주:

1. +DAportable 옵션은 DB2 빌드 파일과 makefile의 컴파일 및 링크 단계에서 사용됩니다. 이 옵션은 PA_RISC 1.1 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다. 이 옵션을 사용하면 약간의 성능 저하가 발생합니다. 성능을 향상시키기 위해 sqllib/samples 디렉토리에 제공된 빌드 파일과 makefile에서 +DAportable 옵션을 제거할 수 있습니다. 이 옵션을 사용하지 않으면 HP-UX 프로그램을 빌드할 때 다음과 유사한 경고가 발생할 수 있습니다.

(경고) 최소한 하나의 PA 2.0 오브젝트 파일(<filename>.o)이 검출되었습니다.
링크된 오브젝트가 PA 1.x 시스템에서 수행되지 않을 수 있습니다.

여기서 <filename>는 컴파일 중인 프로그램 파일입니다.

PA_RISC 1.1 또는 2.0 시스템이 없으면 이 경고는 적용되지 않습니다.

2. HP-UX 버전 10 이하에서 HP-UX 버전 11로 DB2를 이주할 경우, DB2 프로그램은 HP-UX 버전 11에 있는 DB2에서 다시 사전 처리 컴파일된 다음 (Embedded SQL을 포함하는 경우) 다시 컴파일되어야 합니다. 모든 DB2 응용프로그램, 저장 프로시듀어, 사용자 정의 함수(UDF) 및 User Exit 프로그램을 포함합니다. 또한 HP-UX 버전 11에서 컴파일된 DB2 프로그램은 HP-UX 버전 10 이하에서는 수행되지 않을 수 있습니다. HP-UX 버전 10에서 컴파일되고 수행되는 DB2 프로그램은 HP-UX 버전 11 서버에 원격으로 연결될 수 있습니다.
3. 64비트 응용프로그램의 경우, 샘플 디렉토리에 64비트 빌드 스크립트를 사용하십시오. 스크립트 이름은 끝에 "64"가 추가되는 것을 제외하고 32비트 스크립트와 같습니다.

HP-UX C

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 CLI 응용프로그램
- DB2 CLI 저장 프로시듀어
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)
- 멀티스레드 응용프로그램

DB2 CLI 응용프로그램

sqllib/samples/cli에 있는 스크립트 파일 bldcli에 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 DB2 CLI 프로그램의 경우, bldcli64 스크립트를 사용하십시오.

매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 CLI 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

프로그램에 .sqc 확장자로 표시되는 Embedded SQL이 들어 있는 경우, 프로그램을 사전 처리 컴파일하기 위해 embprep 스크립트가 호출되고 .c 확장자를 가진 프로그램 파일을 생성합니다.

```
#! /bin/ksh
# bldcli script file -- HP-UX
# Builds a CLI program with HP-UX C.
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
fi

# Compile the error-checking utility.
cc +DAportable -Aa +e -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc +DAportable -Aa +e -I$DB2PATH/include -c $1.c

# Link the program.
cc +DAportable -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2
```

bldcli에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cc** C 컴파일러를 사용합니다. **+DAportable** PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다. **-Aa** ANSI 표준 모드를 사용합니다. **+e** ANSI C 모드로 컴파일하는 동안 HP 부가 가치 기능을 사용할 수 있게 합니다. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, `$HOME/sqllib/include`와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.
링크 옵션: **cc** 링커에 대한 처음과 끝으로서 컴파일러를 사용합니다. **+DAportable** PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 사용하십시오. **-o $1** 실행 프로그램을 지정합니다. **-o $1.o** 오브젝트 파일을 지정합니다. **utilcli.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, `$HOME/sqllib/lib`와 같습니다. **-ldb2** 데이터베이스 관리자 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 `tbinfo.c`에서 샘플 프로그램 `tbinfo`를 빌드하려면 다음을 입력하십시오.

```
bldcli tbinfo
```

그 결과 실행 파일 `tbinfo`가 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
tbinfo
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 dbusemx.sqc에서 Embedded SQL 응용프로그램 dbusemx를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldcli dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldcli dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldcli dbusemx database userid password
   ```

그 결과 실행 파일 dbusemx가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   dbusemx database userid password
   ```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다.

sqllib/samples/cli에 있는 스크립트 파일 bldapi에 DB2 API가 있는 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다.

주: DB2 API가 있는 64비트 DB2 CLI 프로그램의 경우, bldapi64 스크립트를 사용하십시오.

bldapi 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 bldcli 스크립트 간의 유일한 차이점입니다. bldapi 및 bldcli에 공통되는 컴파일 및 링크 옵션에 대한 자세한 내용은 210 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 dbmconn.c에서 샘플 프로그램 dbmconn을 빌드하려면 다음을 입력하십시오.

```
bldapi dbmconn
```

그 결과 실행 파일 dbmconn이 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
dbmconn
```

DB2 CLI 저장 프로시듀어

sqllib/samples/cli에 있는 스크립트 파일 bldclisp에 DB2 CLI 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 DB2 CLI 저장 프로시듀어의 경우, bldclisp64 스크립트를 사용하십시오.

매개변수 $1은 소스 파일의 이름을 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldclisp script file -- HP-UX
# Builds a CLI stored procedure in HP-UX C.
# Usage: bldclisp <prog_name>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the error-checking utility.
cc +DAportable +u1 +z -Aa +e -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc +DAportable +u1 +z -Aa +e -I$DB2PATH/include -c $1.c

# Link the program.
ld -b -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldclisp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

+DAportable
PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다.

+u1 정렬되지 않은 데이터 액세스를 허용합니다. 응용프로그램이 정렬되지 않은 데이터를 사용할 경우에만 사용하십시오.

+z 위치 독립적(position-independent) 코드를 생성합니다.

-Aa ANSI 표준 모드를 사용합니다(C 컴파일러 전용).

+e ANSI C 모드로 컴파일하는 동안 HP 부가 가치 기능을 사용할 수 있게 합니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, `$HOME/sqllib/include`와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 마십시오. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

bldclisp에 대한 컴파일 및 링크 옵션
링크 옵션: **ld** 편집기를 링크하려면 링커를 사용합니다. **-b** 일반적인 실행 파일 대신 공유 라이브러리를 작성합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 오브젝트 파일을 지정합니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, -L$HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 spserver.c에서 샘플 프로그램 spserver를 빌드하려면 sample 데이터베이스에 연결하고 있는 경우, 다음을 입력하십시오.

```
bldclisp spserver
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리에 액세스하는 CLI 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldcli를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 210 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

공유 라이브러리에 액세스하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행한 다음 출력을 클라이언트 응용프로그램으로 리턴합니다.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/c에 있는 스크립트 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 DB2 응용프로그램의 경우, bldapp64 스크립트를 사용하십시오.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 DB2 API 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본

sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- HP-UX
# Builds a C application program
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  cc +DAportable -Aa +e -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  cc +DAportable -Aa +e -I$DB2PATH/include -c utilapi.c
fi
# Compile the program.
cc +DAportable -Aa +e -I$DB2PATH/include -c $1.c

if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o
  cc +DAportable -o $1 $1.o utilemb.o -L$DB2PATH/lib -ldb2
else
  # Link the program with utilapi.o
  cc +DAportable -o $1 $1.o utilapi.o -L$DB2PATH/lib -ldb2
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

+DAportable
PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다.

-Aa ANSI 표준 모드를 사용합니다(C 컴파일러 전용).

+e ANSI C 모드로 컴파일하는 동안 HP 부가 가치 기능을 사용할 수 있게 합니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, -I$DB2PATH/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션: **cc** 편집기를 링크하려면 컴파일러를 사용합니다. **+DAportable** PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 사용하십시오. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 지정합니다. **utilemb.o** Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다. **utilapi.o** 비 Embedded SQL 프로그램의 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, -L$DB2PATH/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.c에서 DB2 API 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다.

실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqc에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

sqllib/samples/c에 있는 스크립트 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 Embedded SQL 저장 프로시듀어의 경우, bldsrv64 스크립트를 사용하십시오.

bldsrv 스크립트는 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수를 지정할 필요가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- HP-UX
# Builds a C stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2
# Compile the program.
cc +DAportable +u1 +z -Aa +e -I$DB2PATH/include -c $1.c

# Link the program to create a shared library
ld -b -o $1 $1.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cc** C 컴파일러. **+DAportable** PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다. **+u1** 정렬되지 않은 데이터 액세스를 허용합니다. 응용프로그램이 정렬되지 않은 데이터를 사용할 경우에만 사용하십시오. **-Aa** ANSI 표준 모드를 사용합니다(C 컴파일러 전용). **+z** 위치 독립적(position-independent) 코드를 생성합니다. **+e** ANSI C 모드로 컴파일하는 동안 HP 부가 가치 기능을 사용할 수 있게 합니다. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, **-I$DB2PATH/include**와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.
링크 옵션: **ld** 편집기를 링크하려면 링커를 사용합니다. **-b** 일반적인 실행 파일 대신 공유 라이브러리를 작성합니다. **-o $1** 공유 라이브러리 파일로 출력을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 지정합니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, **$HOME/sqllib/lib**와 같습니다. **-L** 옵션을 지정하지 않으면 **/usr/lib:/lib**가 가정됩니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqc에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver database
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리에 액세스하는 CLI 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 217 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

공유 라이브러리에서 저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

사용자 정의 함수(UDF)

sqllib/samples/c에 있는 스크립트 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 UDF의 경우, bldudf64 스크립트를 사용하십시오.

UDF는 저장 프로시듀어처럼 컴파일됩니다. 그러나 Embedded SQL문을 포함할 수 없습니다. 이것은 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하고 사전 처리 컴파일하며 프로그램을 바인드할 필요가 없음을 의미합니다.

매개변수 $1은 소스 파일의 이름을 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- HP-UX
# Builds a C UDF library
# Usage: bldudf <prog_name>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Compile the program.
cc +DAportable +u1 +z -Aa +e -I$DB2PATH/include -c $1.c

# Link the program and create a shared library.
ld -b -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the sqllib/function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

+DAportable
PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다.

+u1 정렬되지 않은 데이터 액세스를 허용합니다. 응용프로그램이 정렬되지 않은 데이터를 사용할 경우에만 사용하십시오.

-Aa ANSI 표준 모드를 사용합니다(C 컴파일러 전용).

+z 위치 독립적(position-independent) 코드를 생성합니다.

+e ANSI C 모드로 컴파일하는 동안 HP 부가 가치 기능을 사용할 수 있게 합니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

링크 옵션:

ld 편집기를 링크하기 위해 링커를 사용합니다.

-b 일반적인 실행 파일 대신 공유 라이브러리를 작성합니다.

-o $1 공유 라이브러리 파일로 출력을 지정합니다.

$1.o 프로그램 오브젝트 파일을 지정합니다.

-L$DB2PATH/lib
DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면, /usr/lib:/lib가 가정됩니다.

-ldb2 DB2 라이브러리로 링크합니다.

-ldb2apie
LOB 위치 지정자를 사용할 수 있도록 DB2 API 엔진 라이브러리와 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldudf udfsrv
```

스크립트 파일은 UDF를 sqllib/function 디렉토리로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 이 프로그램의 Embedded SQL 버전 및 DB2 CLI가 제공됩니다.

스크립트 파일 bldcli를 사용하여 sqllib/samples/cli에 있는 소스 파일 udfcli.c에서 DB2 CLI udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 210 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

스크립트 파일 bldapp를 사용하여 sqllib/samples/c에 있는 소스 파일 udfcli.sqc에서 Embedded SQL udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 217 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

주: HP-UX는 POSIX 스레드 라이브러리와 DCE 스레드 라이브러리를 제공합니다. DB2는 POSIX 스레드 라이브러리를 사용하는 멀티스레드 응용프로그램을 지원합니다.

HP-UX에서 멀티스레드 응용프로그램이 컴파일되려면 _REENTRANT가 정의되어야 합니다. HP-UX 문서는 -D_POSIX_C_SOURCE=199506L을 사용하여 컴파일하는 것이 좋습니다. 또한 반드시 _REENTRANT가 정의되어야 합니다. 응용프로그램은 또한 -lpthread로 링크되어야 합니다.

sqllib/samples/c에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 Embedded SQL 멀티스레드 프로그램의 경우, bldmt64 스크립트를 사용하십시오.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- HP-UX
# Builds a C multi-threaded embedded SQL program
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
  embprep $1 $2 $3 $4
# Compile the program.
cc +DAportable -Aa -I$DB2PATH/include -D_POSIX_C_SOURCE=199506L -c $1.c

# Link the program
cc +DAportable -o $1 $1.o -L$DB2PATH/lib -ldb2 -lpthread
```

위에서 언급한 -D_POSIX_C_SOURCE=199506L 컴파일 옵션 및 -lpthread 링크 옵션 외에도 +e 컴파일 옵션 및 링크된 유틸리티 파일이 없는 경우에도 다른 컴파일 및 링크 옵션이 bldapp 파일에 사용된 것과 동일합니다. 이들 옵션에 대한 자세한 내용은 217 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqc에서 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서 실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
thdsrver
```

HP-UX C++

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)
- 멀티스레드 응용프로그램

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/c에 있는 스크립트 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 응용프로그램의 경우, bldapp64 스크립트를 사용하십시오.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- HP-UX
# Builds a C++ application program
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqC" ]]
then
  ./embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
```

```
  aCC +DAportable -Aa -ext -I$DB2PATH/include -c utilemb.C
else
  # Compile the utilapi.C error-checking utility.
  aCC +DAportable -Aa -ext -I$DB2PATH/include -c utilapi.C
fi
# Compile the program.
aCC +DAportable -Aa -ext -I$DB2PATH/include -c $1.C

if [[ -f $1".sqC" ]]
then
  # Link the program with utilemb.o.
  aCC +DAportable -o $1 $1.o utilemb.o -L$DB2PATH/lib -ldb2
else
  # Link the program with utilapi.o.
  aCC +DAportable -o $1 $1.o utilapi.o -L$DB2PATH/lib -ldb2
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

aCC HP aC++ 컴파일러.

+DAportable
PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다.

-Aa ANSI C++ 표준 기능을 켭니다.

-ext "long long" 지원을 포함하는 여러 C++ 확장자를 사용할 수 있게 합니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션: **aCC** 링커에 대한 프론트 엔드로서 HP aC++ 컴파일러를 사용합니다. **+DAportable** PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 사용하십시오. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 지정합니다. **utilemb.o** Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다. **utilapi.o** 비 Embedded SQL 프로그램의 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, `$HOME/sqllib/lib`와 같습니다. `-L` 옵션을 지정하지 않으면 `/usr/lib:/lib`가 가정됩니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 `client.C`에서 비 Embedded SQL DB2 API 샘플 프로그램 `client`를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 `client`가 작성됩니다. 다음을 입력하여 `sample` 데이터베이스에 대해서 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 `updat.sqC`에서 Embedded SQL 응용프로그램 `updat`를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름만 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ 저장 프로시듀어 빌드에 대한 정보를 참조하십시오.

sqllib/samples/cpp에 있는 스크립트 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 Embedded SQL 저장 프로시듀어의 경우, bldsrv64 스크립트를 사용하십시오.

bldsrv 스크립트는 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 필요 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름 $1을 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- HP-UX
# Builds a C++ stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
./embprep $1 $2
# Compile the program. First ensure it is coded with extern "C".
aCC +DAportable +u1 -Aa +z -ext -I$DB2PATH/include -c $1.C
# Link the program to create a shared library.
aCC +DAportable -b -o $1 $1.o -L$DB2PATH/lib -ldb2
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

aCC HP aC++ 컴파일러.

+DAportable
PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다.

+u1 정렬되지 않은 데이터 액세스를 허용합니다.

-Aa ANSI C++ 표준 기능을 켭니다.

+z 위치 독립적(position-independent) 코드를 생성합니다.

-ext "long long" 지원을 포함하는 여러 C++ 확장자를 사용할 수 있게 합니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $DB2PATH/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

링크 옵션:

aCC 링커에 대한 프론트 엔드로서 HP aC++ 컴파일러를 사용합니다.

+DAportable
PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 사용하십시오.

-b 일반적인 실행 파일 대신 공유 라이브러리를 작성합니다.

-o $1 실행 파일을 지정합니다.

$1.o 프로그램 오브젝트 파일을 지정합니다.

-L$DB2PATH/lib
DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, -L$DB2PATH/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.

-ldb2 DB2 라이브러리로 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqC에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver database
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결합니다.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리 내의 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 228 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

공유 라이브러리에서 저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
: 연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 저장 프로시듀어는 클라이언트 응용프로그램으로 출력을 리턴합니다.

사용자 정의 함수(UDF)

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ UDF 빌드에 대한 정보를 참조하십시오.

sqllib/samples/c의 스크립트 파일 bldudf에는 UDF를 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 UDF의 경우, bldudf64 스크립트를 사용하십시오.

사용자 정의 프로그램은 Embedded SQL문을 포함할 수 없습니다. 이것은 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하고 사전 처리 컴파일하며 프로그램을 바인드할 필요가 없음을 의미합니다.

매개변수 $1은 소스 파일의 이름을 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- HP-UX
# Builds a C or C++ UDF library
# Usage: bldudf <prog_name>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Compile the program.
if [[ -f $1".c" ]]
then
  aCC +DAportable +u1 -ext -Aa +z -I$DB2PATH/include -c $1.c
elif [[ -f $1".C" ]]
then
  aCC +DAportable +u1 -ext -Aa +z -I$DB2PATH/include -c $1.C
fi
# Link the program.
aCC +DAportable -b -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

aCC HP aC++ 컴파일러.

+DAportable
PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다.

+u1 정렬되지 않은 데이터 액세스를 허용합니다.

-ext "long long" 지원을 포함하는 여러 C++ 확장자를 사용할 수 있게 합니다.

-Aa ANSI C++ 표준 기능을 켭니다.

+z 위치 독립적(position-independent) 코드를 생성합니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $DB2PATH/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

bldudf에 대한 컴파일 및 링크 옵션

링크 옵션:

aCC 링커에 대한 프론트 엔드로서 HP aC++ 컴파일러를 사용합니다.

-b 일반적인 실행 파일 대신 공유 라이브러리를 작성합니다.

-o $1 실행 파일을 지정합니다.

$1.o 프로그램 오브젝트 파일을 지정합니다.

-L$DB2PATH/lib
DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, -L$DB2PATH/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.

-ldb2 DB2 라이브러리로 링크합니다.

-ldb2apie
LOB 위치 지정자를 사용할 수 있도록 DB2 API 엔진 라이브러리와 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldudf udfsrv
```

스크립트 파일은 UDF를 sqllib/function 디렉토리로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 스크립트 파일 bldapp를 사용하여 sqllib/samples/cpp에 있는 udfcli.sqC 소스 파일에서 udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 228 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

주: HP-UX는 POSIX 스레드 라이브러리와 DCE 스레드 라이브러리를 제공합니다. HP-UX의 DB2는 POSIX 스레드 라이브러리를 사용하는 멀티스레드 응용프로그램만 지원합니다.

HP-UX C++ 컴파일러의 경우, rand_r을 정의하려면 멀티스레드 응용프로그램에 대해 -D_HPUX_SOURCE를 사용해야 합니다. RWSTD_MULTI_THREAD 및 _REENTRANT도 정의해야 하며 +W829를 사용해야 합니다. 응용프로그램은 또한 -lpthread로 링크되어야 합니다.

sqllib/samples/cpp에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다.

주: 64비트 Embedded SQL 멀티스레드 프로그램의 경우, bldmt64 스크립트를 사용하십시오.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- HP-UX
# Builds a C++ multi-threaded embedded SQL program
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
./embprep $1 $2 $3 $4
# Compile the program.
aCC +DAportable -ext -I$DB2PATH/include \
   -D_HPUX_SOURCE -DRWSTD_MULTI_THREAD -D_REENTRANT +W829 -c $1.C
# Link the program
aCC -o $1 $1.o -L$DB2PATH/lib -ldb2 -lpthread
```

위에서 언급한 옵션 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용됩니다. 이들 옵션에 대한 자세한 내용은 228 페이지의『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqC에서 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서 실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
thdsrver
```

Micro Focus COBOL

이 절에서는 다음 주제에 대해 설명합니다.

- 컴파일러 사용
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어

컴파일러 사용

Embedded SQL 및 DB2 API 호출을 포함하는 응용프로그램을 개발하며 Micro Focus COBOL 컴파일러를 사용 중이면 다음 사항에 유의하십시오.

- 명령행 처리기 명령 db2 prep을 사용하여 응용프로그램을 사전 처리 컴파일할 때 target mfcob 옵션(기본값)을 사용하십시오.
- 내장된 사전 처리 컴파일러 프론트 엔드, 런타임 인터프리터 또는 애니메이터 디버거를 사용하려면, Micro Focus가 제공하는 mkrts 명령을 실행하여 DB2 Generic API 시작점을 Micro Focus 런타임 모듈 rts32에 추가하십시오. 또한 체크 파일을 갱신하려면 mkcheck도 수행해야 합니다. 이 프로그램을 수행하지 않으면 SQLGSTRT에서 173 오류를 수신합니다.

  mkrts 및 mkcheck를 수행하기 전에 다음 단계에 따라 COBOPT를 설정해야 합니다.

1. 루트로 로그인하십시오.
2. 디렉토리 $COBDIR/src/rts에서 다음을 입력하십시오.

```
COBOPT=/opt/IBMdb2/V7.1/lib/db2mkrts.args; export COBOPT
ksh mkrts
mv $COBDIR/rts32 $COBDIR/rts32.orig
cp rts32 $COBDIR/rts32
```

3. 또한 Hewlett-Packard에서 이 제품과 함께 제공하는 check 실행 파일도 재빌드해야 합니다. $COBDIR 디렉토리에 있는 check 실행 파일을 재빌드하지 않는 경우, cob -C SQL을 사용하여 컴파일을 시도하면 DB2 선행 처리기가 DB2 라이브러리를 호출하므로 런타임 시스템 173 오류를 나타내며 실패합니다. check를 재빌드하려면 $COBDIR 디렉토리에서 src/sql 디렉토리로 변경하고 mkcheck 스크립트를 수행해야 합니다. 일단 스크립트가 완료되면 결과로 생성되는 check 실행 파일을 $COBDIR 디렉토리로 이동해야 합니다. 디렉토리 $COBDIR/src/sql에서 다음을 입력하십시오.

```
COBOPT=/opt/IBMdb2/V7.1/lib/db2mkrts.args; export COBOPT
ksh mkcheck
mv $COBDIR/check $COBDIR/check.orig
cp check $COBDIR/check
```

이제 다음 디렉토리에 제공된 인수를 사용하여 mkrts를 실행할 수 있습니다.

```
/opt/IBMdb2/V7.1/lib/db2mkrts.args
```

- DB2 COBOL COPY 파일 디렉토리를 Micro Focus COBOL 환경 변수 COBCPY에 포함시켜야 합니다. COBCPY 환경 변수는 COPY 파일의 위치를 지정합니다. Micro Focus COBOL용 DB2 COPY 파일은 데이터베이스 인스턴스 디렉토리의의 sqllib/include/cobol_mf에 있습니다.

  이 디렉토리를 포함시키려면 다음을 입력하십시오.

```
export COBCPY=$COBCPY:/opt/IBMdb2/V7.1/include/cobol_mf
```

  주: .profile 파일에서 COBCPY를 설정하려 할 수 있습니다.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/cobol_mf에 있는 스크립트 파일 bldapp에 DB2 API 및 Embedded SQL 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 DB2 API 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- HP-UX
# Builds a Micro Focus COBOL application program
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqb" ]]
then
embprep $1 $2 $3 $4
fi
# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# Compile the checkerr.cbl error checking utility.
cob +DAportable -cx checkerr.cbl

# Compile the program.
cob +DAportable -cx $1.cbl

# Link the program.
cob +DAportable -x $1.o checkerr.o -L$DB2PATH/lib -ldb2 -ldb2gmf
```

bldapp에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cob** Micro Focus COBOL 컴파일러. **+DAportable** PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다. **-cx** 오브젝트 모듈로 컴파일합니다.
링크 옵션: **cob** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **+DAportable** PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 사용하십시오. **-x** 실행 프로그램을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **checkerr.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. **-ldb2gmf** Micro Focus COBOL용 DB2 예외 핸들러 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.cbl에서 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다. 다음을 입력하여 sample 데이터베이스에 대해서 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqb에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

sqllib/samples/cobol_mf에 있는 스크립트 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 서버의 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름 $1을 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- HP-UX
# Builds a Micro Focus COBOL stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2
# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$COBCPY:$DB2PATH/include/cobol_mf

# Compile the program.
cob +DAportable +z -cx $1.cbl

# Link the program.
ld -b -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2gmf \
   -L$COBDIR/coblib -lcobol -lcrtn

# Copy the shared library to the sqllib/function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cob COBOL 컴파일러.

+DAportable
PA_RISC 1.x 및 2.0 워크스테이션과 서버에 대해 호환 가능한 코드를 생성합니다.

+z 위치 독립적(position-independent) 코드를 생성합니다.

-cx 오브젝트 모듈로 컴파일합니다.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션: **ld** 편집기를 링크하기 위해 링커를 사용합니다. **-b** 일반적인 실행 파일 대신 공유 라이브러리를 작성합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **-L$DB2PATH/lib** DB2 런타임 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 공유 라이브러리로 링크합니다. **-ldb2gmf** Micro Focus COBOL용 DB2 예외 핸들러 라이브러리로 링크합니다. **-L$COBDIR/coblib** COBOL 런타임 라이브러리의 위치를 지정합니다. **-lcobol** COBOL 라이브러리로 링크합니다. **-lcrtn** crtn 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

샘플 프로그램에 연결하고 있는 경우, 소스 파일 outsrv.sqb에서 샘플 프로그램 outsrv를 빌드하려면 다음을 입력하십시오.

```
bldsrv outsrv
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv outsrv database
```

스크립트 파일은 저장 프로시듀어를 sqllib/function 디렉토리로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 저장 프로시듀어에 대한 파일 모드를 설정하십시오.

일단 저장 프로시듀어 outsrv를 빌드하면 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 outcli를 빌드할 수 있습니다 스크립트 파일 bldapp를 사용하여 outcli를 빌드할 수 있습니다. 자세한 내용은 240 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

```
outcli database userid password
```

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 저장 프로시듀어 라이브러리 outsrv에 액세스하고 서버 데이터베이스에서 동일한 이름의 저장 프로시듀어 함수를 실행한 다음 출력을 클라이언트 응용프로그램으로 리턴합니다.

저장 프로시듀어 나감

저장 프로시듀어를 개발할 때 다음 명령문을 사용하여 저장 프로시듀어를 나가십시오.

```
move SQLZ-HOLD-PROC to return-code.
```

이 명령문을 통해 저장 프로시듀어가 클라이언트 응용프로그램으로 올바로 리턴합니다.

제8장 Linux 응용프로그램 빌드

이 장에서는 Lunux에서의 응용프로그램 빌드에 대한 자세한 정보를 제공합니다. 스크립트 파일에서 db2로 시작되는 명령은 명령행 처리기(CLP) 명령입니다. CLP 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

Linux에 대한 최신 DB2 응용프로그램 개발 갱신사항에 대한 자세한 내용은 다음 웹 페이지를 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

Linux C

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 CLI 응용프로그램
- DB2 CLI 저장 프로시듀어
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)
- 멀티스레드 응용프로그램

DB2 CLI 응용프로그램

sqllib/samples/cli에 있는 스크립트 파일 bldcli에 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 매개변수 $1은 소스 파일의 이름을 지정합니다.

이것은 유일한 필수 매개변수로 Embedded SQL이 없는 CLI 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

프로그램에 .sqc 확장자로 표시되는 Embedded SQL이 들어 있는 경우, 프로그램을 사전 처리 컴파일하기 위해 embprep 스크립트가 호출되고 .c 확장자를 가진 프로그램 파일을 생성합니다.

```
#! /bin/ksh
# bldcli script file -- Linux
# Builds a CLI program with Linux C
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
fi
# Compile the error-checking utility.
cc -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc -I$DB2PATH/include -c $1.c

# Link the program.
cc -o $1 $1.o utilcli.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
```

bldcli에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cc** C 컴파일러. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면 $HOME/sqllib/include와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.
링크 옵션: **cc** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **utilcli.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-Wl,-rpath,$DB2PATH/lib** 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 tbinfo.c에서 샘플 프로그램 tbinfo를 빌드하려면 다음을 입력하십시오.

```
bldcli tbinfo
```

그 결과 실행 파일 tbinfo가 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
tbinfo
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 dbusemx.sqc에서 Embedded SQL 응용프로그램 dbusemx를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldcli dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldcli dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldcli dbusemx database userid password
   ```

그 결과 실행 파일 dbusemx가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   dbusemx database userid password
   ```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다.

sqllib/samples/cli에 있는 스크립트 파일 bldapi에 DB2 API가 있는 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 이 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 bldcli 스크립트 간의 유일한 차이점입니다. bldapi 및 bldcli에 공통되는 컴파일 및 링크 옵션에 대한 자세한 내용은 248 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 dbmconn.c에서 샘플 프로그램 dbmconn을 빌드하려면 다음을 입력하십시오.

```
bldapi dbmconn
```

그 결과 실행 파일 dbmconn이 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
dbmconn
```

DB2 CLI 저장 프로시듀어

sqllib/samples/cli에 있는 스크립트 파일 bldclisp에 DB2 CLI 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 매개변수 $1은 소스 파일의 이름을 지정합니다.

```
#! /bin/ksh
# bldclisp script file -- Linux
# Builds a CLI stored procedure in Linux C.
# Usage: bldclisp <prog_name>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the error-checking utility.
cc -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc -I$DB2PATH/include -c $1.c

# Link the program.
cc -o $1 $1.o utilcli.o -shared -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2

# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldclisp에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cc** C 컴파일러. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.
링크 옵션: **cc** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **utilcli.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-shared** 공유 라이브러리를 생성합니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-Wl,-rpath,$DB2PATH/lib** 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 spserver.c에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldclisp spserver
```

스크립트 파일은 공유 라이브러리를 sqllib/function 디렉토리로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리를 호출하는 CLI 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldcli를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 248 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

공유 라이브러리에 액세스하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
: 연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
: 유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/c에 있는 스크립트 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 DB2 API 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 스크립트 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- Linux
# Builds a C application program.
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  cc -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  cc -I$DB2PATH/include -c utilapi.c
fi
# Compile the program.
cc -I$DB2PATH/include -c $1.c
if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o.
  cc -o $1 $1.o utilemb.o -L$DB2PATH/lib \
   -Wl,-rpath,$DB2PATH/lib -ldb2
else
  # Link the program with utilapi.o.
  cc -o $1 $1.o utilapi.o -L$DB2PATH/lib \
   -Wl,-rpath,$DB2PATH/lib -ldb2
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션: **cc** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 오브젝트 파일을 지정합니다. **utilemb.o** Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다. **utilapi.o** 비 Embedded SQL 프로그램의 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-Wl,-rpath,$DB2PATH/lib** 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.c에서 DB2 API 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다.

실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqc에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

sqllib/samples/c의 스크립트 파일 bldsrv에는 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수를 지정할 필요가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- Linux
# Builds a C stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2
# Compile the program.
cc -I$DB2PATH/include -c $1.c
# Link the program and create a shared library
cc -shared -o $1 $1.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션:
cc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
-shared 공유 라이브러리를 생성합니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
-L$DB2PATH/lib 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-Wl,-rpath,$DB2PATH/lib 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다.
-ldb2 DB2 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqc에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver database
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리에 액세스하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 254 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

공유 라이브러리에서 저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
: 연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
: 유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

사용자 정의 함수(UDF)

sqllib/samples/c에 있는 스크립트 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다. UDF는 Embedded SQL문을 포함하지 않습니다. 그러므로 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하거나 프로그램을 사전 처리 컴파일하고 바인드할 필요가 없습니다.

매개변수 $1은 소스 파일의 이름을 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대해 이 소스 파일 이름을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- Linux
# Builds a C UDF library
# Usage: bldudf <prog_name>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Compile the program.
cc -I$DB2PATH/include -c $1.c
# Link the program and create a shared library.
cc -o $1 $1.o -shared -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2 -ldb2apie
# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.

bldudf에 대한 컴파일 및 링크 옵션
링크 옵션: **cc** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **-shared** 공유 라이브러리를 생성합니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-Wl,-rpath,$DB2PATH/lib** 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. **-ldb2apie** LOB 위치 지정자를 사용할 수 있도록 DB2 API 엔진 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldudf udfsrv
```

스크립트 파일은 UDF를 sqllib/function 디렉토리로 복사합니다.

필요하면 DB2 인스턴스가 실행할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 이 프로그램의 Embedded SQL 버전 및 DB2 CLI가 제공됩니다.

스크립트 파일 bldcli를 사용하여 sqllib/samples/cli의 소스 파일 udfcli.c에서 DB2 CLI udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 248 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

스크립트 파일 bldapp를 사용하여 sqllib/samples/c의 소스 파일 udfcli.sqc에서 Embedded SQL udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 254 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

Linux C를 사용하는 멀티스레드 응용프로그램은 -D_REENTRANT로 컴파일되고 -lpthread로 링크되어야 합니다.

sqllib/samples/c에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- Linux
# Builds a C multi-threaded embedded SQL program.
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
  embprep $1 $2 $3 $4
# Compile the program.
```

```
cc -I$DB2PATH/include -c $1.c -D_REENTRANT
# Link the program.
cc -o $1 $1.o -lpthread -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
```

위에서 언급한 -D_REENTRANT와 -lpthread 옵션 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이들 옵션에 대한 자세한 내용은 254 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqc에서 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서 실행 파일을 수행하려면 다음을 입력하십시오.

```
thdsrver
```

Linux C++

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)
- 멀티스레드 응용프로그램

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/cpp에 있는 스크립트 파일 bldapp에 샘플 C++ 프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 DB2 API 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- Linux
# Builds a C++ application program.
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqC" ]]
then
embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  g++ -I$DB2PATH/include -c utilemb.C
else
  # Compile the utilapi.C error-checking utility.
  g++ -I$DB2PATH/include -c utilapi.C
fi
# Compile the program.
g++ -I$DB2PATH/include -c $1.C
if [[ -f $1".sqC" ]]
then
  # Link the program with utilemb.o
  g++ -o $1 $1.o utilemb.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
else
  # Link the program with utilapi.o
  g++ -o $1 $1.o utilapi.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

g++ C++ 컴파일러.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션: **g++** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함시키십시오. **utilemb.o** Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다. **utilapi.o** 비 Embedded SQL 프로그램의 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-Wl,-rpath,$DB2PATH/lib** 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.C에서 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다. 다음을 입력하여 sample 데이터베이스에 대해서 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqC에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ 저장 프로시듀어 빌드에 대한 정보를 참조하십시오.

sqllib/samples/cpp에 있는 스크립트 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 필요 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름 $1을 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- Linux
# Builds a C++ stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2
# Compile the program.
g++ -I$DB2PATH/include -c $1.C
# Link the program and create a shared library.
g++ -shared -o $1 $1.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2

# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션
컴파일 옵션: **g++** C++ 컴파일러. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.
링크 옵션: **g++** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-shared** 공유 라이브러리를 생성합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-Wl,-rpath,$DB2PATH/lib** 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqC에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver database
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대한 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리 내의 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 264 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

공유 라이브러리에서 저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
: 연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 저장 프로시듀어는 클라이언트 응용프로그램으로 출력을 리턴합니다.

사용자 정의 함수(UDF)

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ UDF 빌드에 대한 정보를 참조하십시오.

sqllib/samples/cpp에 있는 스크립트 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다. UDF는 Embedded SQL문을 포함하지 않습니다. 이것은 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하고 사전 처리 컴파일하며 프로그램을 바인드할 필요가 없음을 의미합니다.

매개변수 $1은 소스 파일의 이름을 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 이 소스 파일 이름을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- Linux
# Builds a C++ UDF library
# Usage: bldudf <prog_name>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Compile the program.
if [[ -f $1".c" ]]
then
g++ -I$DB2PATH/include -c $1.c

elif [[ -f $1".C" ]]
then
g++ -I$DB2PATH/include -c $1.C
fi
# Link the program.
g++ -o $1 $1.o -shared -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf에 대한 컴파일 및 링크 옵션
컴파일 옵션: **g++** C++ 컴파일러. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.
링크 옵션: **g++** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **-shared** 공유 라이브러리를 생성합니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-Wl,-rpath,$DB2PATH/lib** 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. **-ldb2apie** LOB 위치 지정자를 사용할수 있도록 DB2 API 엔진 라이브러리와 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldudf udfsrv
```

스크립트 파일은 UDF를 sqllib/function 디렉토리로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 스크립트 파일 bldapp를 사용하여 sqllib/samples/cpp에 있는 udfcli.sqC 소스 파일에서 udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 264 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

Linux C++를 사용하는 멀티스레드 응용프로그램은 -D_REENTRANT로 컴파일하고 -lpthread로 링크되어야 합니다.

sqllib/samples/cpp에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- Linux
# Builds a C++ multi-threaded embedded SQL program.
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
  embprep $1 $2 $3 $4
```

```
# Compile the program.
g++ -D_REENTRANT -I$DB2PATH/include -c $1.C
# Link the program.
g++ -lpthread -o $1 $1.o -L$DB2PATH/lib -Wl,-rpath,$DB2PATH/lib -ldb2
```

위에서 언급한 -D_REENTRANT와 -lpthread 옵션 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이들 옵션에 대한 자세한 내용은 264 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqC에서 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서 실행 파일을 수행하려면 다음을 입력하십시오.

```
thdsrver
```

제9장 OS/2 응용프로그램 빌드

이 장에서는 OS/2에서의 응용프로그램 빌드에 대한 자세한 정보를 제공합니다. 명령 파일에서 db2로 시작되는 명령은 명령행 처리기(CLP) 명령입니다. CLP 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

OS/2에 대한 최신 DB2 응용프로그램 개발 갱신사항에 대한 자세한 내용은 다음 웹 페이지를 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

주: 사용자 정의 SQLDA를 포함하는 복합 SQL문은 OS/2의 16비트 응용프로그램에서 사용할 수 없습니다.

OS/2용 IBM VisualAge C++ 버전 3

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 CLI 응용프로그램
- DB2 CLI 저장 프로시듀어
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어

- 사용자 정의 함수(UDF)

주: VisualAge C++ 컴파일러는 %DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp 디렉토리에 제공되는 C 및 C++ 샘플 프로그램에 모두 사용됩니다. 동일한 명령 파일이 두 디렉토리에 있습니다. 이 두 디렉토리에는 파일 확장자에 따라 C 또는 C++ 소스 파일을 승인하기 위한 명령이 들어 있습니다.

DB2 CLI 응용프로그램

%DB2PATH%\samples\cli에 있는 명령 파일 bldcli에 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 매개변수 %1은 소스 파일의 이름을 지정합니다.

이것은 유일한 필수 매개변수로 Embedded SQL이 없는 CLI 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

프로그램에 .sqc 확장자로 표시되는 Embedded SQL이 들어 있는 경우, 프로그램을 사전 처리 컴파일하기 위해 embprep 명령 파일이 호출되고 .c 확장자를 가진 프로그램 파일을 생성합니다.

```
@echo off
rem  bldcli command file - OS/2
rem  Builds a CLI program with IBM VisualAge C++.
rem  Usage: bldcli prog_name [ db_name [ userid password ]]
if exist "%1.sqc" call embprep %1 %2 %3 %4
if exist "%1.sqx" call embprep %1 %2 %3 %4
if "%1" == "" goto error

rem Compile the error-checking utility.
icc -C+ -O- -Ti+ utilcli.c

rem  Compile the program.
if exist "%1.sqx" goto cpp
icc -C+ -O- -Ti+ %1.c
goto link_step
:cpp
icc -C+ -O- -Ti+ %1.cxx

rem  Link the program.
:link_step
ilink /NOFREE /NOI /DEBUG /ST:64000 /PM:VIO %1.obj utilcli.obj,%1.exe,NUL,db2cli.lib;
goto exit
```

```
:error
echo Usage: bldcli prog_name [ db_name [ userid password ]]
:exit
@echo on
```

bldcli에 대한 컴파일 및 링크 옵션

컴파일 옵션:

icc IBM VisualAge C++ 컴파일러.

-C+ 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

-O- 최적화하지 않음. 최적화를 사용하지 않고 디버거를 사용하는 것이 더 쉽습니다.

-Ti+ 디버거 정보를 생성합니다.

링크 옵션:

ilink 편집기를 링크하기 위해 ilink 링커를 사용합니다.

/NOFREE
free format을 하지 않음.

/NOI 대소문자를 무시하지 않음. 대소문자를 구별합니다.

/DEBUG 디버깅 정보를 포함시킵니다.

/ST:64000
최소한 64 000의 스택 크기를 지정합니다.

/PM:VIO
프로그램이 OS/2 창에서 수행할 수 있도록 합니다.

%1.obj 오브젝트 파일을 포함시킵니다.

utilcli.obj
오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다.

%1.exe 실행 파일을 지정합니다.

NUL 기본값을 승인합니다.

db2cli.lib
DB2 CLI 라이브러리로 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 tbinfo.c에서 샘플 프로그램 tbinfo를 빌드하려면 다음을 입력하십시오.

```
bldcli tbinfo
```

그 결과 실행 파일 tbinfo.exe가 작성됩니다. 다음과 같이 확장자 없이 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
tbinfo
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 dbusemx.sqc에서 Embedded SQL 응용프로그램 dbusemx를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldcli dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldcli dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldcli dbusemx database userid password
   ```

그 결과 실행 파일 dbusemx.exe가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 확장자 없이 실행 파일 이름만 입력하십시오.

   ```
   dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   dbusemx database userid password
   ```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다.

%DB2PATH%\samples\cli에 있는 명령 파일 bldapi에 DB2 API가 있는 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 이 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 bldcli 명령 파일 간의 유일한 차이점입니다. bldapi 및 bldcli에 공통되는 컴파일 및 링크 옵션에 대한 자세한 내용은 276 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 dbmconn.c에서 샘플 프로그램 dbmconn을 빌드하려면 다음을 입력하십시오.

```
bldapi dbmconn
```

그 결과 실행 파일 dbmconn.exe가 작성됩니다. 다음과 같이 확장자 없이 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
dbmconn
```

DB2 CLI 저장 프로시듀어

%DB2PATH%\samples\cli의 명령 파일 bldclisp에는 CLI 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 명령 파일은 저장 프로시듀어를 서버의 DLL로 빌드합니다.

매개변수 %1은 소스 파일의 이름을 지정합니다. 명령 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem  bldclisp command file - OS/2
rem  Builds a CLI stored procedure using the IBM VisualAge C++ compiler.
rem  Usage: bldclisp <prog_name>

if "%1" == "" goto error

rem Compile the error-checking utility.
icc -C+ -Ti+ -Ge- -Gm+ -W2 utilcli.c
rem  Compile the program.
```

```
if exist "%1.cxx" goto cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.c
goto link_step
:cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.cxx
:link_step
rem Link the program and produce a DLL.
ilink /NOFREE /MAP /NOI /DEBUG /ST:64000 %1.obj utilcli.obj,%1.dll,,db2cli.lib,%1.def;
rem Copy the stored procedure DLL to the 'function' directory
copy %1.dll %DB2PATH%\function

goto exit
:error
echo Usage: bldclisp prog_name
:exit
@echo on
```

bldclisp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

icc IBM VisualAge C++ 컴파일러.

-C+ 컴파일만 수행하고 링크는 수행하지 않습니다. 이 명령 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.

-Ti+ 디버거 정보를 작성합니다.

-Ge- .DLL 파일을 빌드합니다. 정적으로 링크된 런타임 라이브러리 버전을 사용합니다.

-Gm+ 멀티타스킹 라이브러리로 링크합니다.

-W2 출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지.

링크 옵션:

ilink 편집기를 링크하기 위해 ilink 링커를 사용합니다.

/NOFREE
free format을 하지 않음.

/MAP 맵 파일을 작성합니다.

/NOI 대소문자를 무시하지 않음. 대소문자를 구별합니다.

/DEBUG 디버깅 정보를 포함시킵니다.

/ST:64000
최소한 64000의 스택 크기를 지정합니다.

%1.obj 오브젝트 파일을 포함시킵니다.

%1.dll 동적 링크 라이브러리를 작성합니다.

db2cli.lib
DB2 CLI 라이브러리로 링크합니다.

%1.def 모듈 정의 파일.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 spserver.c에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldclisp spserver
```

스크립트 파일은 공유 라이브러리를 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리 내의 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

명령 파일 bldcli를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 276 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

공유 라이브러리에 액세스하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

DB2 API 및 Embedded SQL 응용프로그램

%DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 명령 파일 bldapp.cmd에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일하게 필요한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하며 %4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
@echo off
rem bldapp command file -- OS/2
rem Builds a VisualAge C++ application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
if exist "%1.sqx" goto embedded
if exist "%1.sqc" goto embedded
goto non_embedded
:embedded
rem Precompile and bind the program.
call embprep %1 %2 %3 %4
rem  Compile the program.
if exist "%1.cxx" goto cpp_embedded
```

```
icc -c utilemb.c
icc -C+ -O- -Ti+ %1.c
goto link_embedded
:cpp_embedded
icc -c utilemb.cxx
icc -C+ -O- -Ti+ %1.cxx
goto link_embedded
:non_embedded
rem  Compile the program.
if exist "%1.cxx" goto cpp
icc -c utilapi.c
icc -C+ -O- -Ti+ %1.c
goto link_non_embedded
:cpp
icc -c utilapi.cxx
icc -C+ -O- -Ti+ %1.cxx
goto link_non_embedded
rem  Link the program.
:link_embedded
ilink /NOFREE /NOI /DEBUG /ST:64000 /PM:VIO %1.obj utilemb.obj,,,db2api;
goto exit
:link_non_embedded
ilink /NOFREE /NOI /DEBUG /ST:64000 /PM:VIO %1.obj utilapi.obj,,,db2api;
:exit
@echo on
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

icc IBM VisualAge C++ 컴파일러.

-C+ 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

-O- 최적화하지 않음. 최적화를 사용하지 않고 디버거를 사용하는 것이 더 쉽습니다.

-Ti+ 디버거 정보를 생성합니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션: **ilink** 편집기를 링크하려면 ilink 링커를 사용합니다. **/NOFREE** free format을 하지 않음. **/NOI** 대소문자를 무시하지 않음. 대소문자를 구별합니다. **/DEBUG** 디버깅 정보를 포함시키십시오. **/ST:64000** 최소한 64000의 스택 크기를 지정합니다. **/PM:VIO** 프로그램이 OS/2 창에서 수행할 수 있도록 합니다. **utilemb.obj** Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다. **utilapi.obj** Embedded SQL 프로그램이 아닌 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다. **db2api** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.c에서 DB2 API 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client.exe가 작성됩니다.

실행 파일을 수행하려면 확장자 없이 실행 파일 이름을 입력하십시오.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqc에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat.exe가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 확장자 없이 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

%DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 명령 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 명령 파일은 저장 프로시듀어를 서버의 DLL로 컴파일합니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

명령 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldsrv command file -- OS/2
rem Builds a VisualAge C++ stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]
rem Precompile and bind the program.
call embprep %1 %2

rem  Compile the program.
if exist "%1.cxx" goto cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.c
goto link_step
:cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.cxx
:link_step
rem  Link the program.
ilink /NOFREE /NOI /DEBUG /ST:64000 %1.obj,%1.dll,,db2api,%1.def;
rem Copy the stored procedure to the %DB2PATH%\function directory.
copy %1.dll %DB2PATH%\function
@echo on
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

icc IBM VisualAge C++ 컴파일러.

-C+ 컴파일만 수행하고 링크는 수행하지 않습니다. 이 명령 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.

-Ti+ 디버거 정보를 작성합니다.

-Ge- .DLL 파일을 빌드합니다. 정적으로 링크된 런타임 라이브러리 버전을 사용합니다.

-Gm+ 멀티타스킹 라이브러리로 링크합니다.

-W2 출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션: **ilink** 편집기를 링크하기 위해 ilink 링커를 사용합니다. **/NOFREE** free format을 하지 않음. **/NOI** 대소문자를 무시하지 않음. 대소문자를 구별합니다. **/DEBUG** 디버깅 정보를 포함시킵니다. **/ST:64000** 최소한 64000의 스택 크기를 지정합니다. **%1.dll** 동적 링크 라이브러리를 작성합니다. **db2api** DB2 라이브러리로 링크합니다. **%1.def** 모듈 정의 파일. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqc에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver database
```

명령 파일은 공유 라이브러리를 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리에 액세스하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

명령 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 282 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

사용자 정의 함수(UDF)

%DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 명령 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다.

UDF는 Embedded SQL문을 포함할 수 없습니다. 따라서 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하여 프로그램을 사전 처리 컴파일하고 바인드할 필요가 없습니다.

명령 파일은 소스 파일의 이름을 지정하는 하나의 매개변수 %1을 가집니다. 명령 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem  bldudf command file -- OS/2
rem Builds a VisualAge C++ user-defined function (UDF)
rem  Usage: bldudf <prog_name>
if "%1" == "" goto error

rem  Compile the program.
if exist "%1.cxx" goto cpp
icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.c
goto link_step
:cpp
rem icc -C+ -Ti+ -Ge- -Gm+ -W2 %1.cxx
:link_step
rem  Link the program.
ilink /NOFREE /MAP /NOI /DEBUG /ST:64000 %1.obj,%1.dll,,db2api db2apie,%1.def;

rem Copy the UDF to the %DB2PATH%\function directory
copy %1.dll %DB2PATH%\function
goto exit
:error
echo Usage: bldudf prog_name
:exit
@echo on
```

bldudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

icc IBM VisualAge C++ 컴파일러.

-C+ 컴파일만 수행하고 링크는 수행하지 않습니다. 이 명령 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.

-Ti+ 디버거 정보를 작성합니다.

-Ge- .DLL 파일을 빌드합니다. 정적으로 링크된 런타임 라이브러리 버전을 사용합니다.

-Gm+ 멀티타스킹 라이브러리로 링크합니다.

-W2 출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지.

bldudf에 대한 컴파일 및 링크 옵션
링크 옵션: **ilink** 편집기를 링크하기 위해 ilink 링커를 사용합니다. **/NOFREE** free format을 하지 않음. **/MAP** 맵 파일을 작성합니다. **/NOI** 대소문자를 무시하지 않음. 대소문자를 구별합니다. **/DEBUG** 디버깅 정보를 포함시킵니다. **/ST:64000** 최소한 64000의 스택 크기를 지정합니다. **%1.dll** 동적 링크 라이브러리를 작성합니다. **db2api** DB2 라이브러리로 링크합니다. **db2apie** DB2 API 엔진 라이브러리로 링크합니다. **%1.def** 모듈 정의 파일. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldudf udfsrv
```

스크립트 파일은 UDF를 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 UDF에 대한 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 이 프로그램의 Embedded SQL 버전 및 DB2 CLI가 제공됩니다.

명령 파일 bldcli.cmd를 사용하여 %DB2PATH%\samples\cli의 udfcli.c 소스 파일에서 DB2 CLI udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 276 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

명령 파일 bldapp를 사용하여 %DB2PATH%\samples\c의 소스 파일 udfcli.sqc에서 Embedded SQL udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 282 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 확장자 없이 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

OS/2용 IBM VisualAge C++ 버전 4.0

VisualAge C++ 버전 4 컴파일러에 대한 응용프로그램 빌드 정보는 AIX, OS/2 및 Windows 32비트 운영 체제에서 공통됩니다. 이 정보에 대해서는 172 페이지의 『VisualAge C++ 버전 4.0』을 참조하십시오.

OS/2용 IBM VisualAge COBOL

이 절에서는 다음 주제에 대해 설명합니다.

- 컴파일러 사용
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어

컴파일러 사용

다음 내용은 DB2가 IBM VisualAge COBOL 컴파일러를 사용하는 데 도움을 줄 것입니다.

바인드 파일 작성을 위한 일시적인 해결책

OS/2용 DB2 및 IBM Cobol을 사용하는 응용프로그램을 작성할 때 DB2 사전처리 컴파일러가 바인드 파일 작성에 실패하는 경우가 있습니다. 이것은 OS/2 내에서의 파일 핸들 한계에 의한 것입니다.

그 해결책은 OS/2가 컴파일을 수행하는 머신에 더 많은 파일 핸들을 허용하게 하는 것입니다. 다음의

```
SET SHELLHANDLESINC=20
```

행을 OS/2용 DB2가 설치되어 있는 머신의 CONFIG.SYS 파일에 삽입해야 합니다. 또는 컴파일할 때 NODATA 옵션을 사용할 수 있습니다(IBM Cobol 옵션임).

Embedded SQL 및 DB2 API 호출

Embedded SQL 및 DB2 API 호출을 포함하는 응용프로그램을 개발하며 IBM VisualAge COBOL 컴파일러를 사용하고 있으면 다음 사항에 유의하십시오.

- 명령행 처리기 명령 db2 prep를 사용하여 응용프로그램을 사전 처리 컴파일할 때 target ibmcob 옵션을 사용하십시오.
- 소스 파일에서 탭 문자를 사용하지 마십시오.
- 소스 파일에서 PROCESS 및 CBL 키워드를 사용하여 컴파일 옵션을 설정할 수 있습니다. 키워드를 8-72 컬럼에만 배치하십시오.
- 응용프로그램에 Embedded SQL만 들어 있고 DB2 API 호출을 포함하지 않을 경우, pgmname(mixed) 컴파일 옵션을 사용할 필요가 없습니다. DB2 API 호출을 사용할 경우에는 pgmname(mixed) 컴파일 옵션을 사용해야 합니다.
- IBM VisualAge COBOL 컴파일러의 "System/390 호스트 데이터 유형 지원" 기능을 사용 중인 경우, 응용프로그램에 대한 DB2 포함 파일은 다음 디렉토리에 있습니다.

  ```
  %DB2PATH%\include\cobol_i
  ```

  제공된 명령 파일을 사용하여 DB2 샘플 프로그램을 빌드하는 경우, 명령 파일에 지정된 포함 파일 경로가 cobol_a 디렉토리가 아닌 cobol_i 디렉토리를 가리키도록 변경해야 합니다.

  IBM VisualAge COBOL 컴파일러의 "System/390 호스트 데이터 유형 지원" 기능을 사용하지 않거나 해당 컴파일러의 이전 버전을 사용 중인 경우, 응용프로그램에 대한 DB2 포함 파일은 다음 디렉토리에 있습니다.

  ```
  %DB2PATH%\include\cobol_a
  ```

  다음과 같이 COPY 파일 이름이 .cbl 확장자를 포함시키도록 지정하십시오.

  ```
  COPY "sql.cbl".
  ```

Embedded SQL 응용프로그램

%DB2PATH%\samples\cobol에 있는 명령 파일 bldapp.cmd에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하며 %4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 절달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
@echo off
rem bldapp command file -- OS/2
rem Builds a VisualAge COBOL application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

rem If an embedded SQL program, precompile and bind it.
if exist "%1.sqb" goto prepbind
goto compile_step
:prepbind
call embprep %1 %2 %3 %4
:compile_step
rem Compile the checkerr error checking utility.
cob2 -c -g -qpgmname(mixed) -qlib -I%DB2PATH%\include\cobol_a checkerr.cbl

rem  Compile the program.
cob2 -c -g -qpgmname(mixed) -qlib -I%DB2PATH%\include\cobol_a %1.cbl

rem  Link the program.
ilink %1.obj checkerr.obj db2api.lib /ST:64000 /PM:VIO /NOI /DEBUG
@echo on
```

bldapp에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cob2** IBM VisualAge COBOL 컴파일러. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다. **-g** 디버그 정보를 포함시킵니다. **-qpgmname(mixed)** 컴파일러에 대소문자가 혼합된 이름으로 라이브러리 시작점에 대한 CALL을 허용하도록 지시합니다. **-qlib** COPY문을 처리할 컴파일러를 지시합니다. **-I***path* DB2 포함 파일의 위치를 지정합니다. 예를 들면, `-I%DB2PATH%\include\cobol_a`와 같습니다.
링크 옵션: **ilink** 편집기를 링크하기 위해 ilink 링커를 사용합니다. **checkerr.obj** 오류 체크 유틸리티 오브젝트 파일을 포함시킵니다. **db2api.lib** DB2 라이브러리로 링크합니다. **/ST:64000** 최소한 64000의 스택 크기를 지정합니다. **/PM:VIO** 프로그램이 OS/2 창에서 수행할 수 있도록 합니다. **/NOI** 링크될 때 대소문자를 무시하지 않습니다. **/DEBUG** 디버깅 정보를 포함시킵니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 `client.cbl`에서 비 Embedded SQL 샘플 프로그램 `client`를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 `client.exe`가 작성됩니다. 다음과 같이 확장자 없이 실행 파일 이름을 입력하여 `sample` 데이터베이스에 대해 실행 파일을 수행할 수 있습니다.

client

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqb에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat.exe가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 파일 확장자 없이 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

%DB2PATH%\samples\cobol에 있는 명령 파일 bldsrv에 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 명령 파일은 저장 프로시듀어를 서버의 DLL에 컴파일합니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

명령 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldsrv command file -- OS/2
rem Builds a VisualAge COBOL stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
call embprep %1 %2

rem  Compile the program.
cob2 -c -g -qpgmname(mixed) -qlib -I%DB2PATH%\include\cobol_a %1.cbl

rem  Link the program.
ilink %1.obj checkerr.obj %1.def db2api.lib /ST:64000 /PM:VIO /NOI /DEBUG

rem Copy stored procedure to the %DB2PATH%\function directory.
copy %1.dll %DB2PATH%\function
@echo on
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cob2 IBM VisualAge COBOL 컴파일러.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

-g 디버그 정보를 포함시킵니다.

-qpgmname(mixed)
컴파일러에 대소문자가 혼합된 이름으로 라이브러리 시작점에 대한 CALL을 허용하도록 지시합니다.

-qlib COPY문을 처리할 컴파일러를 지시합니다.

-I*path* DB2 포함 파일의 위치를 지정합니다. 예를 들면, -I%DB2PATH%\include\cobol_a와 같습니다.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션: **ilink** 편집기를 링크하기 위해 ilink 링커를 사용합니다. **checkerr.obj** 오류 체크 유틸리티 오브젝트 파일을 포함시킵니다. **%1.def** 모듈 정의 파일. **db2api.lib** DB2 라이브러리로 링크합니다. **/ST:64000** 최소한 64000의 스택 크기를 지정합니다. **/PM:VIO** 프로그램이 OS/2 창에서 수행할 수 있도록 합니다. **/NOI** 링크될 때 대소문자를 무시하지 않습니다. **/DEBUG** 디버깅 정보를 포함시킵니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

샘플 데이터베이스에 연결하여 소스 파일 outsrv.sqb에서 샘플 프로그램 outsrv를 빌드하려면 다음을 입력하십시오.

```
bldsrv outsrv
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 포함시키십시오.

```
bldsrv outsrv database
```

명령 파일은 샘플 프로그램과 동일한 디렉토리에 있는 모듈 정의 파일 outsrv.def를 사용하여 DLL을 빌드합니다. 명령 파일은 저장 프로시듀어 DLL outsrv.dll을 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 DLL에 대해 파일 모드를 설정하십시오.

일단 DLL outsrv를 빌드하면 이 DLL에 액세스하는 클라이언트 응용프로그램 outcli를 빌드할 수 있습니다. 명령 파일 bldapp를 사용하여 outcli를 빌드할 수 있습니다. 자세한 내용은 293 페이지의 『Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면, 다음과 같이 입력하여 클라이언트 응용프로그램을 수행하십시오.

```
outcli database userid password
```

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 원격 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 DLL outsrv에 액세스하고 서버 데이터베이스에서 동일한 이름의 저장 프로시듀어 함수를 실행한 다음 출력을 클라이언트 응용프로그램으로 리턴합니다.

Micro Focus COBOL

이 절에서는 다음 주제에 대해 설명합니다.

- 컴파일러 사용
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어

컴파일러 사용

DB2는 Micro Focus COBOL 컴파일러와 함께 제공되는 link385 링커를 지원하지 않습니다. DB2 Micro Focus COBOL 프로그램을 링크하려면 IBM 컴파일러 제품에서 사용 가능한 ilink 링커를 사용해야 합니다. 이 절에 있는 스크립트 파일에서 사용되는 cbllink 명령은 ilink 링커를 호출합니다.

Micro Focus COBOL 컴파일러를 사용하여 Embedded SQL 및 DB2 API 호출을 포함하는 응용프로그램을 빌드할 때 다음 사항에 유의하십시오.

- 명령행 처리기 명령 db2 prep를 사용하여 응용프로그램을 사전 처리 컴파일할 때 기본값 target mfcob 옵션을 사용하십시오.
- LIB 환경 변수가 다음과 같이 %DB2PATH%\lib를 가리키는지 확인하십시오.

  ```
  set LIB=%DB2PATH%\lib;%LIB%
  ```

- Micro Focus COBOL용 DB2 COPY 파일은 %DB2PATH%\include\cobol_mf에 있습니다. 다음 디렉토리를 포함시키도록 COBCPY 환경 변수를 설정하십시오.

  ```
  set COBCPY=%DB2PATH%\include\cobol_mf;%COBCPY%
  ```

모든 DB2 API에 대한 호출은 호출 규칙 8을 사용하여 이루어져야 합니다. DB2 COBOL 사전 처리 컴파일러는 자동으로 CALL-CONVENTION 절을 SPECIAL-NAMES 단락에 추가합니다. SPECIAL-NAMES 단락이 없을 경우, DB2 COBOL 사전 처리 컴파일러가 다음과 같이 이를 작성합니다.

```
Identification Division
Program-ID. "static".
special-names.
    call-convention 8 is DB2API.
```

또한 사전 처리 컴파일러는 DB2 API가 호출될 때마다 "call" 키워드 다음에 호출 규칙을 식별하는 데 사용되는 기호 DB2API를 배치합니다. 이것은, 예를 들어, 사전 처리 컴파일러가 Embedded SQL문으로부터 DB2 API 런타임 호출을 생성할 때마다 발생합니다.

사전 처리 컴파일되지 않은 응용프로그램에서 DB2 API에 대한 호출이 이루어지면 사용자는 수동으로 위에 제공된 것과 비슷하게 응용프로그램에 SPECIAL-NAMES 단락을 작성해야 합니다. DB2 API를 직접 호출하는 경우에는 수동으로 DB2API 기호를 "call" 키워드 다음에 추가해야 합니다.

DB2 API 및 Embedded SQL 응용프로그램

%DB2PATH%\samples\cobol_mf에 있는 명령 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함

께 제공되어야 합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하며 %4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 제공합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
@echo off
rem bldapp command file -- OS/2
rem Builds a Micro Focus COBOL application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

rem If an embedded SQL program, precompile and bind it.
if exist "%1.sqb" goto prepbind
goto compile_step
:prepbind
call embprep %1 %2 %3 %4
:compile_step
rem Compile the error-checking utility.
cobol checkerr.cbl;

rem  Compile the program.
cobol %1.cbl;

rem  Link the program.
cbllink %1.obj checkerr.obj db2api.lib db2gmf32.lib
@echo on
```

bldapp에 대한 컴파일 및 링크 옵션
컴파일 옵션:
cobol Micro Focus COBOL 컴파일러.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션: **cbllink** 편집기를 링크하기 위해 링커를 사용합니다. **checkerr.obj** 오류 체크 유틸리티 오브젝트 파일을 포함시킵니다. **db2api.lib** DB2 API 라이브러리로 링크합니다. **db2gmf32.lib** M. F. COBOL용 DB2 예외 핸들러로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.cbl에서 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client.exe가 작성됩니다. 다음과 같이 확장자 없이 실행 파일 이름을 입력하여 sample 데이터베이스에 대해 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqb에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

```
bldapp updat database userid password
```

그 결과 실행 파일 updat.exe가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 파일 확장자 없이 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

%DB2PATH%\samples\cobol_mf에 있는 명령 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 명령 파일은 저장 프로시듀어를 서버의 DLL로 컴파일합니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다. 명령 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldsrv command file -- OS/2
rem Builds a Micro Focus COBOL stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
```

```
call embprep %1 %2

rem  Compile the stored procedure.
cobol %1.cbl;

rem  Link the stored procedure and create a shared library.
cbllink /d %1.obj db2api.lib db2gmf32.lib

rem Copy the stored procedure to the %DB2PATH%\function directory.
copy %1.dll %DB2PATH%\function
@echo on
```

bldsrv에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cobol** Micro Focus COBOL 컴파일러.
링크 옵션: **cbllink** 편집기에 링크하기 위해 Micro Focus COBOL 링커를 사용합니다. **/d** .dll 파일을 작성합니다. **db2api.lib** DB2 API 라이브러리를 포함시킵니다. **db2gmf32.lib** M. F. COBOL용 DB2 예외 핸들러로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

샘플 프로그램에 연결하고 있는 경우, 소스 파일 outsrv.sqb에서 샘플 프로그램 outsrv를 빌드하려면 다음을 입력하십시오.

```
bldsrv outsrv
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv outsrv database
```

링커는 사용자에 의해 지정되지 않은 기본 시작점을 사용합니다. 저장 프로시듀어를 빌드하기 위해 .dll 파일을 작성할 때 /d 옵션이 사용됩니다. 명령 파일은 저장 프로시듀어 DLL outsrv.dll을 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 DLL에 대해 파일 모드를 설정하십시오.

일단 DLL outsrv를 빌드하면 이 DLL에 액세스하는 클라이언트 응용프로그램 outcli를 빌드할 수 있습니다. 명령 파일 bldapp를 사용하여 outcli를 빌드할 수 있습니다. 자세한 내용은 299 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

```
outcli database userid password
```

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 DLL, outsrv에 액세스하고 서버 데이터베이스에서 동일한 이름의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

REXX

REXX 프로그램을 컴파일하거나 바인드하지 마십시오.

OS/2의 경우, 응용프로그램 파일에는 .cmd 확장자가 있어야 합니다. 작성한 다음 운영 체제 명령 프롬프트에서 직접 응용프로그램을 수행할 수 있습니다.

OS/2 REXX 프로그램은 일괄처리 명령과 구별하기 위해 처음 행의 첫 번째 컬럼에서 시작하는 주석을 포함해야 합니다.

```
/* Any comment will do. */
```

REXX 샘플 프로그램은 %DB2PATH%\samples\rexx 디렉토리에 있습니다. 샘플 REXX 프로그램 updat를 수행하려면 다음을 입력하십시오.

```
updat
```

REXX 및 DB2에 대한 자세한 내용은 *응용프로그램 개발 안내서*의 "REXX 프로그래밍"을 참조하십시오.

제10장 PTX 응용프로그램 빌드

이 장에서는 NUMA-Q용 DB2를 사용하여 PTX에서의 응용프로그램 빌드에 대한 자세한 정보를 제공합니다. 스크립트 파일에서 db2로 시작되는 명령은 명령행 처리기(CLP) 명령입니다. CLP 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

PTX에 대한 최신 DB2 응용프로그램 개발 갱신사항에 대한 자세한 내용은 다음 웹 페이지를 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

ptx/C

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 CLI 응용프로그램
- DB2 API가 있는 DB2 CLI 응용프로그램
- DB2 CLI 저장 프로시듀어
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)
- 멀티스레드 응용프로그램

DB2 CLI 응용프로그램

sqllib/samples/cli에 있는 스크립트 파일 bldcli에 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 매개변수 $1은 소스 파일의 이름을 지정합니다.

이것은 유일한 필수 매개변수로 Embedded SQL이 없는 CLI 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

프로그램에 .sqc 확장자로 표시되는 Embedded SQL이 들어 있는 경우, 프로그램을 사전 처리 컴파일하기 위해 embprep 스크립트가 호출되고 .c 확장자를 가진 프로그램 파일을 생성합니다.

```
#! /bin/ksh
# bldcli script file -- PTX
# Builds a DB2 CLI program
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
fi

# Compile the error-checking utility.
cc -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc -I$DB2PATH/include -c $1.c

# Link the program.
cc -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2
```

bldcli에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cc** C 컴파일러를 사용합니다. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.
링크 옵션: **cc** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-o $1** 실행 프로그램을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **utilcli.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 tbinfo.c에서 샘플 프로그램 tbinfo를 빌드하려면 다음을 입력하십시오.

```
bldcli tbinfo
```

그 결과 실행 파일 tbinfo가 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
tbinfo
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 dbusemx.sqc에서 Embedded SQL 응용프로그램 dbusemx를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

```
bldcli dbusemx
```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldcli dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldcli dbusemx database userid password
   ```

그 결과 실행 파일 dbusemx가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   dbusemx database userid password
   ```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다.

sqllib/samples/cli에 있는 스크립트 파일 bldapi에 DB2 API가 있는 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 이 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 bldcli 스크립트 간의 유일한 차이

점입니다. bldapi 및 bldcli에 공통되는 컴파일 및 링크 옵션에 대한 자세한 내용은 308 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 dbmconn.c에서 샘플 프로그램 dbmconn을 빌드하려면 다음을 입력하십시오.

```
bldapi dbmconn
```

그 결과 실행 파일 dbmconn이 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
dbmconn
```

DB2 CLI 저장 프로시듀어

sqllib/samples/cli에 있는 스크립트 파일 bldclisp에 DB2 CLI 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 매개변수 $1은 소스 파일의 이름을 지정합니다.

```
#! /bin/ksh
# bldclisp script file -- PTX
# Builds a DB2 CLI stored procedure
# Usage: bldclisp <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the error-checking utility.
cc -KPIC -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc -KPIC -I$DB2PATH/include -c $1.c

# Link the program.
cc -G -o $1 $1.o utilcli.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldclisp에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cc** C 컴파일러. **-KPIC** 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.
링크 옵션: **cc** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-G** 공유 라이브러리를 생성합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **utilcli.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 spserver.c에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldclisp spserver
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리에 액세스하는 CLI 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldcli를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 308 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

공유 라이브러리에 액세스하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행한 다음 출력을 클라이언트 응용프로그램으로 리턴합니다.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/c에 있는 스크립트 파일 bldapp에 DB2 API와 Embedded SQL 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 DB2 API 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- PTX
# Builds a C application program.
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to the location where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# if an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  cc -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  cc -I$DB2PATH/include -c utilapi.c
fi
# Compile the program.
cc -I$DB2PATH/include -c $1.c

if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o
  cc -o $1 $1.o utilemb.o -L$DB2PATH/lib -ldb2
else
  # Link the program with utilapi.o
  cc -o $1 $1.o utilapi.o -L$DB2PATH/lib -ldb2
fi
```

bldapp에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cc** C 컴파일러. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.
링크 옵션: **cc** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **util.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

비 Embedded SQL 샘플 프로그램 client를 빌드하려면 소스 파일 client.c에서 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다.

실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqc에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

```
bldapp updat
```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

sqllib/samples/c에 있는 스크립트 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수를 지정할 필요가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- PTX
# Builds a C stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2

# Compile the program.
cc -KPIC -I$DB2PATH/include -c $1.c

# Link the program and create a shared library.
cc -G -o $1 $1.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

-KPIC 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, -I$DB2PATH/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션:
cc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
-G 공유 라이브러리를 생성합니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
-L$DB2PATH/lib 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-ldb2 DB2 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqc에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver database
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리에 액세스하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 313 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

공유 라이브러리에서 저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

사용자 정의 함수(UDF)

sqllib/samples/c에 있는 스크립트 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다. UDF는 Embedded SQL문을 포함하지 않습니다. 그러므로 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하거나 프로그램을 사전 처리 컴파일하고 바인드할 필요가 없습니다.

매개변수 $1은 소스 파일의 이름을 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- PTX
# Builds a C user-defined function library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the program.
cc -KPIC -I$DB2PATH/include -c $1.c

# Link the program and create a shared library.
cc -G -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

-KPIC 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면 $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.

bldudf에 대한 컴파일 및 링크 옵션
링크 옵션:
cc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
-G 공유 라이브러리를 생성합니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
-L$DB2PATH/lib 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-ldb2 DB2 라이브러리로 링크합니다.
-ldb2apie LOB 위치 지정자를 사용할 수 있도록 DB2 API 엔진 라이브러리와 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldudf udfsrv
```

스크립트 파일은 UDF를 sqllib/function 디렉토리로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 이 프로그램의 Embedded SQL 버전 및 DB2 CLI가 제공됩니다.

스크립트 파일 bldcli를 사용하여 sqllib/samples/cli의 소스 파일 udfcli.c에서 DB2 CLI udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 308 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

스크립트 파일 bldapp를 사용하여 sqllib/samples/c의 소스 파일 udfcli.sqc에서 Embedded SQL udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 313 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

ptx/C를 사용하는 멀티스레드 응용프로그램은 -Kthread로 컴파일되고 링크되어야 합니다.

sqllib/samples/c에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- PTX
# Builds a C multi-threaded embedded SQL program.
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to the location where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
  embprep $1 $2 $3 $4
# Compile the program.
cc -Kthread -I$DB2PATH/include -c $1.c

# Link the program.
cc -Kthread -o $1 $1.o -L$DB2PATH/lib -ldb2
```

위에서 언급한 -Kthread 옵션 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이들 옵션에 대한 자세한 내용은 313 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqc에서 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서 실행 파일을 수행하려면 다음을 입력하십시오.

```
thdsrver
```

ptx/C++

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)
- 멀티스레드 응용프로그램

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/cpp에 있는 빌드 파일 bldapp에 DB2 API와 Embedded SQL 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 DB2 API 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본

sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- PTX
# Builds a C++ application program
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to the location where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# if an embedded SQL program, precompile and bind it.
if [[ -f $1".sqC" ]]
then
embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  c++ -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c utilemb.C
else
  # Compile the utilapi.C error-checking utility.
  c++ -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c utilapi.C
fi

# Compile the program.
c++ -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c $1.C
if [[ -f $1".sqC" ]]
then
  # Link the program with utilemb.o
  c++ -o $1 $1.o utilemb.o -L$DB2PATH/lib -ldb2 -lseq
else
  # Link the program with utilapi.o
  c++ -o $1 $1.o utilapi.o -L$DB2PATH/lib -ldb2 -lseq
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

c++ C++ 컴파일러.

-I$DB2PATH/include
DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-D_RWSTD_COMPILE_INSTANTIATE=0
rogue wave 클래스를 인스턴스화하지 마십시오.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 컴파일과 링크는 서로 다른 단계입니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션:
c++ 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
utilemb.o Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다.
utilapi.o 비 Embedded SQL 프로그램의 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다.
-L$DB2PATH/lib 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-ldb2 DB2 라이브러리로 링크합니다.
-lseq Sequent 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.C에서 DB2 API 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다.

실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqC에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

```
bldapp updat
```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름을 추가하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호를 추가하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ 저장 프로시듀어 빌드에 대한 정보를 참조하십시오.

sqllib/samples/cpp에 있는 스크립트 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하

므로 선택적인 추가 매개변수 $2가 연결하려는 데이터베이스의 이름을 지정해야 합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 추가 매개변수가 필요 없습니다. 스크립트 파일 bldsrv는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다.

소스 파일 이름 $1이 공유 라이브러리 이름에 사용됩니다.

```
#! /bin/ksh
# bldsrv script file -- PTX
# Builds a C++ stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2

# Compile the program. First ensure it is coded with extern "C".
c++ -KPIC -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c $1.C
# Link the program and create a shared library.
c++ -G -o $1 $1.o -L$DB2PATH/lib -ldb2

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1.so $DB2PATH/function/$1
```

bldsrv에 대한 컴파일 및 링크 옵션
컴파일 옵션: **c++** C++ 컴파일러. **-KPIC** 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, -I$DB2PATH/include와 같습니다. **-D_RWSTD_COMPILE_INSTANTIATE=0** rogue wave 클래스를 인스턴스화하지 마십시오. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.
링크 옵션: **c++** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-G** 공유 라이브러리를 생성합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqC에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver database
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리 내의 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 323 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

공유 라이브러리에서 저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 저장 프로시듀어는 클라이언트 응용프로그램으로 출력을 리턴합니다.

사용자 정의 함수(UDF)

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ UDF 빌드에 대한 정보를 참조하십시오.

sqllib/samples/cpp에 있는 스크립트 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다. UDF는 Embedded SQL문을 포함하지 않습니다. 그러므로 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하거나 프로그램을 사전 처리 컴파일하고 바인드할 필요가 없습니다.

매개변수 $1은 소스 파일의 이름을 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- PTX
# Builds a C++ user-defined function library
# Usage: bldudf <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# Compile the program.
c++ -KPIC -I$DB2PATH/include  -D_RWSTD_COMPILE_INSTANTIATE=0 -c $1.c

# Link the program and create a shared library.
c++ -G -o $1 $1.o -L$DB2PATH/lib -ldb2 -ldb2apie

# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1.so $DB2PATH/function/$1
```

bldudf에 대한 컴파일 및 링크 옵션
컴파일 옵션: **c++** C++ 컴파일러. **-KPIC** 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다. **-I$DB2PATH/include** DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다. **-D_RWSTD_COMPILE_INSTANTIATE=0** rogue wave 클래스를 인스턴스화하지 마십시오. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.
링크 옵션: **c++** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-G** 공유 라이브러리를 생성합니다. **-o $1** 실행 파일을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다. **-ldb2** DB2 라이브러리로 링크합니다. **-ldb2apie** LOB 위치 지정자를 사용할 수 있도록 DB2 API 엔진 라이브러리와 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldudf udfsrv
```

스크립트 파일은 UDF를 경로 sqllib/function에 있는 서버로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 스크립트 파일 bldapp를 사용하여 sqllib/samples/cpp에 있는 udfcli.sqC 소스 파일에서 udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 323 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

ptx/C++를 사용하는 멀티스레드 응용프로그램은 -Kthread로 컴파일되고 링크되어야 합니다.

sqllib/samples/cpp에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- PTX
# Builds a C++ multi-threaded embedded SQL program
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to the location where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
  embprep $1 $2 $3 $4

# Compile the program.
```

```
c++ -Kthread -I$DB2PATH/include -D_RWSTD_COMPILE_INSTANTIATE=0 -c $1.C
# Link the program.
c++ -Kthread -o $1 $1.o -L$DB2PATH/lib -ldb2 -lseq
```

위에서 언급한 -Kthread 옵션 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이 옵션에 대한 자세한 내용은 323 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqC에서 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서 실행 파일을 수행하려면 다음을 입력하십시오.

```
thdsrver
```

제11장 Silicon Graphics IRIX 응용프로그램 빌드

이 장에서는 Silicon Graphics IRIX에서의 DB2 응용프로그램 빌드에 대한 자세한 정보를 제공합니다. 스크립트 파일에서 db2로 시작되는 명령은 명령행 처리기(CLP) 명령입니다. CLP 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

Silicon Graphics IRIX에 대한 최신 DB2 응용프로그램 개발 갱신사항에 대한 자세한 내용은 다음 웹 페이지를 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

Silicon Graphics IRIX용 DB2는 클라이언트 전용입니다. DB2 응용프로그램을 수행하고 DB2 Embedded SQL 응용프로그램을 빌드하려면 클라이언트 머신에서 서버 머신의 DB2 데이터베이스에 액세스해야 합니다. 이 서버 머신에서는 다른 운영 체제를 수행 중입니다. 클라이언트에서 서버로의 통신 구성에 대한 자세한 내용은 *UNIX용 DB2 빠른 시작*을 참조하십시오.

또한 다른 운영 체제에서 수행하는 원격 클라이언트에서 서버의 데이터베이스에 액세스한 이후에는 DB2 CLI를 포함하여 데이터베이스 유틸리티를 데이터베이스에 바인드해야 합니다. 자세한 내용은 51 페이지의 『바인딩』을 참조하십시오.

DB2 라이브러리 지원

Silicon Graphics IRIX는 세 가지의 호환되지 않는 별도의 오브젝트 유형인 o32(기본값), n32(새로운 32 오브젝트 유형) 및 64(64 오브젝트 유형)를 제공합니다. DB2는 아직 64를 지원하지 않으나 o32 및 n32 오브젝트 유형은 지원합니다.

이 운영 체제에는 또한 두 가지의 호환되지 않는 별도의 스레드 API 버전인 sproc 인터페이스와 POSIX 스레드 API가 있습니다. DB2는 POSIX 스레드 API를 지원합니다.

Sproc 인터페이스를 사용하는 응용프로그램은 DB2 라이브러리의 스레드 없는 버전으로, 스레드가 안전하지 않는 libdb2를 사용해야 합니다. libdb2는 sproc가 안전하지 않으므로 sproc 인터페이스를 사용할 때 신중해야 합니다.

이 범위의 기능성을 제공하기 위해 DB2는 다음 라이브러리를 지원합니다.

lib/libdb2.so
: 스레드 없는 o32

lib/libdb2_th.so
: POSIX 스레드 o32

lib32/libdb2.so
: 스레드 없는 n32

lib32/libdb2_th.so
: POSIX 스레드 n32

n32 오브젝트 유형을 사용하려면 프로그램이 lib32/libdb2.so 또는 lib32/libdb2_th.so 라이브러리로 링크될 뿐 아니라 -n32 옵션으로 컴파일되고 링크되어야 합니다. 기본 o32 오브젝트 유형을 사용하려면 프로그램이 -n32 옵션을 사용하지 말고 lib/libdb2.so 또는 lib/libdb2_th.so 라이브러리에 링크되어야 합니다.

주: 이 장에서 설명하는 빌드 파일을 사용하여 n32 오브젝트 유형 응용프로그램을 빌드하려면 표시된 명령의 주석 처리를 제거하십시오.

MIPSpro C

이 절에서는 다음 DB2 인터페이스에서 MIPSpro C를 사용하는 방법에 대해 설명합니다.

- DB2 CLI
- DB2 API
- Embedded SQL

DB2 CLI 응용프로그램

sqllib/samples/cli에 있는 스크립트 파일 bldcli에 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 매개변수 $1은 소스 파일의 이름을 지정합니다.

이것은 유일한 필수 매개변수로 Embedded SQL이 없는 CLI 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

프로그램에 .sqc 확장자로 표시되는 Embedded SQL이 들어 있는 경우, 프로그램을 사전 처리 컴파일하기 위해 embprep 스크립트가 호출되고 .c 확장자를 가진 프로그램 파일을 생성합니다.

```
#! /bin/ksh
# bldcli script file -- Silicon Graphics IRIX
# Builds a CLI program with MIPSpro C.
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
fi

# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32
if [ "$IRIX_OBJECT_MODE" = "-n32" ] ; then
  # Link with db2 n32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib32
else
  # Link with db2 o32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib
fi
# Compile the error-checking utility.
cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilcli.c
```

```
# Compile the program.
cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.c

# Link the program.
cc $IRIX_OBJECT_MODE -o $1 $1.o utilcli.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
```

bldcli에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러를 사용합니다.

$IRIX_OBJECT_MODE

'IRIX_OBJECT_MODE=-n32'가 주석 처리되지 않으면 "-n32" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

bldcli에 대한 컴파일 및 링크 옵션
링크 옵션:
cc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$IRIX_OBJECT_MODE ‘IRIX_OBJECT_MODE=-n32’가 주석 처리되지 않으면 ″-n32″ 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
utilcli.o 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다.
-L$DB2_LIBPATH 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. o32 오브젝트 유형에 대해서는 $DB2PATH/lib를 가리킵니다. n32 오브젝트 유형에 대해서는 $DB2PATH/lib32를 가리킵니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-rpath $DB2_LIBPATH 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. o32 오브젝트 유형에 대해서는 $DB2PATH/lib를 가리킵니다. n32 오브젝트 유형에 대해서는 $DB2PATH/lib32를 가리킵니다.
-lm math 라이브러리로 링크합니다.
-ldb2 DB2 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 tbinfo.c에서 샘플 프로그램 tbinfo를 빌드하려면 다음을 입력하십시오.

```
bldcli tbinfo
```

그 결과 실행 파일 tbinfo가 작성됩니다. 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하여 실행 파일을 수행할 수 있습니다.

```
tbinfo database userid password
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 dbusemx.sqc에서 dbusemx를 빌드하려면 데이터베이스에 대한 매개변수, 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 포함시키십시오.

```
bldcli dbusemx database userid password
```

그 결과 실행 파일 dbusemx가 작성됩니다. Embedded SQL 응용프로그램을 수행하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
dbusemx database userid password
```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다.

sqllib/samples/cli에 있는 스크립트 파일 bldapi에 DB2 API가 있는 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 이 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 bldcli 스크립트 간의 유일한 차이점입니다. bldapi 및 bldcli에 공통되는 컴파일 및 링크 옵션에 대한 자세한 내용은 337 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 dbmconn.c에서 샘플 프로그램 dbmconn을 빌드하려면 다음을 입력하십시오.

```
bldapi dbmconn
```

그 결과 실행 파일 dbmconn이 작성됩니다. 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하여 실행 파일을 수행할 수 있습니다.

```
dbmconn database userid password
```

저장 프로시듀어용 DB2 CLI 클라이언트 응용프로그램

저장 프로시듀어는 데이터베이스에 액세스하고 클라이언트 응용프로그램으로 정보를 리턴하는 프로그램입니다. 서버에서 저장 프로시듀어를 컴파일하고 저장합니다. 이 서버는 다른 플랫폼에서 수행합니다.

DB2 CLI 저장 프로시듀어 spserver를 DB2 지원 플랫폼 서버에서 빌드하려면 이 책의 내용 중 해당 플랫폼에 대한 "응용프로그램 빌드" 장을 참조하십시오. DB2 클라이언트가 액세스할 수 있는 다른 서버에 대한 자세한 내용은 7 페이지의 『지원되는 서버』를 참조하십시오.

일단 저장 프로시듀어 spserver를 빌드하면 사용자는 스크립트 파일 bldcli를 사용하여 소스 파일 spclient.c에서 저장 프로시듀어 spclient를 호출하는 클라이언트 응용프로그램을 빌드할 수 있습니다. 자세한 내용은 337 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하여 저장 프로시듀어를 호출할 수 있습니다.

```
spclient database userid password
```

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

UDF용 DB2 CLI 클라이언트 응용프로그램

사용자 정의 함수(UDF)는 사용자가 서버에서 컴파일하고 저장하는 사용자 고유 스칼라 및 테이블 함수입니다. 이 서버는 다른 플랫폼에서 수행합니다. 사용자 정의 함수(UDF) 프로그램 udfsrv를 DB2 지원 플랫폼 서버에서 빌드하려면 이 책의 내용 중 해당 플랫폼에 대한 "응용프로그램 빌드" 장을 참조하십시오. DB2 클라이언트가 액세스할 수 있는 다른 서버에 대한 자세한 내용은 7 페이지의 『지원되는 서버』를 참조하십시오.

일단 udfsrv를 빌드하면 DB2 CLI 스크립트 파일 bldcli를 사용하여 sqllib/samples/cli의 udfcli.c 소스 파일에서 이 udfsrv를 호출하는 DB2 CLI

클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 자세한 내용은 337 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

UDF 프로그램을 호출하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하여 호출 응용프로그램을 수행하십시오.

```
udfcli database userid password
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/c에 있는 스크립트 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다. 첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 유일한 필수 매개변수로 Embedded SQL이 없는 DB2 API 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- Silicon Graphics IRIX
# Builds a C application program.
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32
if [ "$IRIX_OBJECT_MODE" = "-n32" ] ; then
  # Link with db2 n32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib32
else
  # Link with db2 o32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib
fi
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
```

```
  # Compile the utilemb.c error-checking utility.
  cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilapi.c
fi
# Compile the program.
cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.c

if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o
  cc $IRIX_OBJECT_MODE -o $1 $1.o utilemb.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
else
  # Link the program with utilapi.o
  cc $IRIX_OBJECT_MODE -o $1 $1.o utilapi.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러를 사용합니다.

$IRIX_OBJECT_MODE

'IRIX_OBJECT_MODE=-n32'가 주석 처리되지 않으면 "-n32" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션:
cc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$IRIX_OBJECT_MODE 'IRIX_OBJECT_MODE=-n32'가 주석 처리되지 않으면 "-n32" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
utilemb.o Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다.
utilapi.o 비 Embedded SQL 프로그램의 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다.
-L$DB2_LIBPATH 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. o32 오브젝트 유형에 대해서는 $DB2PATH/lib를 가리킵니다. n32 오브젝트 유형에 대해서는 $DB2PATH/lib32를 가리킵니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-rpath $DB2_LIBPATH 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. o32 오브젝트 유형에 대해서는 $DB2PATH/lib를 가리킵니다. n32 오브젝트 유형에 대해서는 $DB2PATH/lib32를 가리킵니다.
-lm math 라이브러리로 링크합니다.
-ldb2 DB2 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.c에서 DB2 API 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다.

실행 파일을 수행하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
client database userid password
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqc에서 샘플 프로그램 updat를 빌드하려면 데이터베이스에 대한 매개변수, 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 포함시키십시오.

```
bldapp updat database userid password
```

그 결과 실행 파일 updat가 작성됩니다. sample 데이터베이스에 대해 실행 파일을 수행하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
updat database userid password
```

저장 프로시듀어용 Embedded SQL 클라이언트 응용프로그램

저장 프로시듀어는 데이터베이스에 액세스하고 클라이언트 응용프로그램으로 정보를 리턴하는 프로그램입니다. 서버에서 저장 프로시듀어를 컴파일하고 저장합니다. 이 서버는 다른 플랫폼에서 수행합니다.

Embedded SQL 저장 프로시듀어 spserver를 DB2 지원 플랫폼 서버에서 빌드하려면 이 책의 내용 중 해당 플랫폼에 대한 "응용프로그램 빌드" 장을 참조하십시오. DB2 클라이언트가 액세스할 수 있는 다른 서버에 대한 자세한 내용은 7 페이지의 『지원되는 서버』를 참조하십시오.

일단 저장 프로시듀어 spserver를 빌드하면 저장 프로시듀어를 호출하는 클라이언트 응용프로그램을 빌드할 수 있습니다. 스크립트 파일 bldapp를 사용하여 소스 파일 spclient.sqc에서 spclient를 빌드할 수 있습니다. 자세한 내용은 342 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하여 클라이언트 응용프로그램을 수행하십시오.

```
spclient database userid password
```

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

UDF용 클라이언트 응용프로그램

사용자 정의 함수(UDF)는 사용자가 서버에서 컴파일하고 저장하는 사용자 고유 스칼라 및 테이블 함수입니다. 이 서버는 다른 플랫폼에서 수행합니다. 사용자 정의 함수(UDF) 프로그램 udfsrv를 DB2 지원 플랫폼 서버에서 빌드하려면 이 책의 내용 중 해당 플랫폼에 대한 "응용프로그램 빌드" 장을 참조하십시오. DB2 클라이언트가 액세스할 수 있는 다른 서버에 대한 자세한 내용은 7 페이지의 『지원되는 서버』를 참조하십시오.

일단 udfsrv를 빌드하면 스크립트 파일 bldapp를 사용하여 sqllib/samples/c의 udfcli.sqc 소스 파일에서 이 udfsrv를 호출하는 Embedded SQL 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 자세한 내용은 342 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF 프로그램을 호출하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하여 호출 응용프로그램을 수행하십시오.

```
udfcli database userid password
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

Silicon Graphics IRIX의 멀티스레드 응용프로그램은 -ldb2_th 및 -lpthread 링크 옵션을 사용하여 o32 또는 n32 오브젝트 유형에 대한 DB2 라이브러리의 POSIX 스레드 버전으로 링크되어야 합니다.

sqllib/samples/c에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다. 첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다.

```
#! /bin/ksh
# bldmt script file -- Silicon Graphics IRIX
# Builds a C multi-threaded embedded SQL program
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32
if [ "$IRIX_OBJECT_MODE" = "-n32" ] ; then
  # Link with db2 n32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib32
else
  # Link with db2 o32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib
fi
# Precompile and bind the program.
  embprep $1 $2 $3 $4
# Compile the program.
cc $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.c

# Link the program.
cc $IRIX_OBJECT_MODE -o $1 $1.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2_th -lpthread
```

위에서 언급한 -ldb2_th와 -lpthread 링크 옵션 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이들 옵션에 대한 자세한 내용은 342 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqc에서 샘플 프로그램 thdsrver를 빌드하려면 데이터베이스에 대한 매개변수, 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 포함시키십시오.

```
bldmt thdsrver database userid password
```

그 결과 실행 파일 thdsrver이 작성됩니다.

실행 파일을 수행하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
thdsrver database userid password
```

MIPSpro C++

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 API 및 Embedded SQL 응용프로그램
- 멀티스레드 응용프로그램

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/c에 있는 스크립트 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 비 Embedded SQL 응용프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다.

```
#! /bin/ksh
# bldapp script file -- Silicon Graphics IRIX
# Builds a C++ application program
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32
if [ "$IRIX_OBJECT_MODE" = "-n32" ] ; then
  # Link with db2 n32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib32
else
  # Link with db2 o32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib
fi
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqC" ]]
then
embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  CC $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilemb.C
else
  # Compile the utilapi.c error-checking utility.
  CC $IRIX_OBJECT_MODE -I$DB2PATH/include -c utilapi.C
fi
# Compile the program.
CC $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.C
if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o
  CC $IRIX_OBJECT_MODE -o $1 $1.o utilemb.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
else
  # Link the program with utilapi.o
  CC $IRIX_OBJECT_MODE -o $1 $1.o utilapi.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2
fi
```

bldapp에 대한 컴파일 및 링크 옵션
컴파일 옵션:
CC C++ 컴파일러를 사용합니다.
$IRIX_OBJECT_MODE ‘IRIX_OBJECT_MODE=-n32’가 주석 처리되지 않으면 ″-n32″ 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-I$DB2PATH/include DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.
-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 갖고 있습니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션:
CC 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$IRIX_OBJECT_MODE ‘IRIX_OBJECT_MODE=-n32’가 주석 처리되지 않으면 ″-n32″ 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
utilemb.o Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다.
utilapi.o 비 Embedded SQL 프로그램의 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다.
-L$DB2_LIBPATH 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. o32 오브젝트 유형에 대해서는 $DB2PATH/lib를 가리킵니다. n32 오브젝트 유형에 대해서는 $DB2PATH/lib32를 가리킵니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-rpath $DB2_LIBPATH 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. o32 오브젝트 유형에 대해서는 $DB2PATH/lib를 가리킵니다. n32 오브젝트 유형에 대해서는 $DB2PATH/lib32를 가리킵니다.
-lm math 라이브러리로 링크합니다.
-ldb2 DB2 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 updat.sqC에서 샘플 프로그램 updat를 빌드하려면 데이터베이스에 대한 매개변수, 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 포함시키십시오.

```
bldapp updat database userid password
```

그 결과 실행 파일 updat가 작성됩니다. 이 실행 파일을 수행하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
updat database userid password
```

저장 프로시듀어용 Embedded SQL 클라이언트 응용프로그램

저장 프로시듀어는 데이터베이스에 액세스하고 클라이언트 응용프로그램으로 정보를 리턴하는 프로그램입니다. 서버에서 저장 프로시듀어를 컴파일하고 저장합니다. 이 서버는 다른 플랫폼에서 수행합니다.

Embedded SQL 저장 프로시듀어 spserver를 DB2 지원 플랫폼 서버에서 빌드하려면 이 책의 내용 중 해당 플랫폼에 대한 "응용프로그램 빌드" 장을 참조하십시오. DB2 클라이언트가 액세스할 수 있는 다른 서버에 대한 자세한 내용은 7 페이지의 『지원되는 서버』를 참조하십시오.

일단 저장 프로시듀어 spserver를 빌드하면 저장 프로시듀어를 호출하는 클라이언트 응용프로그램을 빌드할 수 있습니다. 스크립트 파일 bldapp를 사용하여 소스 파일 spclient.sqC에서 spclient를 빌드할 수 있습니다. 자세한 내용은 348 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하여 클라이언트 응용프로그램을 수행하십시오.

```
spclient database userid password
```

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 저장 프로시듀어는 클라이언트 응용프로그램으로 출력을 리턴합니다.

UDF용 Embedded SQL 클라이언트 응용프로그램

사용자 정의 함수(UDF)는 사용자가 서버에서 컴파일하고 저장하는 사용자 고유 스칼라 함수입니다. 이 서버는 다른 플랫폼에서 수행합니다. 사용자 정의 함수(UDF) 프로그램 udfsrv를 DB2 지원 플랫폼 서버에서 빌드하려면 이 책의 내용 중 해

당 플랫폼에 대한 "응용프로그램 빌드" 장을 참조하십시오. DB2 클라이언트가 액세스할 수 있는 다른 서버에 대한 자세한 내용은 7 페이지의 『지원되는 서버』를 참조하십시오.

일단 udfsrv를 빌드하면 스크립트 파일 bldapp를 사용하여 sqllib/samples/cpp에 있는 udfcli.sqC 소스 파일에서 이 udfsrv를 호출하는 Embedded SQL 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 자세한 내용은 348 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF 프로그램을 호출하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하여 호출 응용프로그램을 수행하십시오.

```
udfcli database userid password
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

Silicon Graphics IRIX의 멀티스레드 응용프로그램은 -ldb2_th 및 -lpthread 링크 옵션을 사용하여 o32 또는 n32 오브젝트 유형에 대한 DB2 라이브러리의 POSIX 스레드 버전으로 링크되어야 합니다.

sqllib/samples/cpp에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다.

```
#! /bin/ksh
# bldmt script file -- Silicon Graphics IRIX
# Builds a C++ multi-threaded embedded SQL program
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile with n32 object support, uncomment the following line.
# IRIX_OBJECT_MODE=-n32
if [ "$IRIX_OBJECT_MODE" = "-n32" ] ; then
  # Link with db2 n32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib32
else
  # Link with db2 o32 object type libraries.
  DB2_LIBPATH=$DB2PATH/lib
```

```
fi
# Precompile and bind the program.
  embprep $1 $2 $3 $4
# Compile the program.
CC $IRIX_OBJECT_MODE -I$DB2PATH/include -c $1.C

# Link the program.
CC $IRIX_OBJECT_MODE -o $1 $1.o -L$DB2_LIBPATH -rpath $DB2_LIBPATH -lm -ldb2_th -lpthread
```

위에서 언급한 -ldb2_th와 -lpthread 링크 옵션 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이들 옵션에 대한 자세한 내용은 348 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqc에서 샘플 프로그램 thdsrver를 빌드하려면 데이터베이스에 대한 매개변수, 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 포함시키십시오.

```
bldmt thdsrver database userid password
```

그 결과 실행 파일 thdsrver이 작성됩니다.

sample 데이터베이스에 대해 실행 파일을 수행하려면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스가 있는 인스턴스의 사용자 ID와 암호를 입력하십시오.

```
thdsrver database userid password
```

제12장 Solaris 응용프로그램 빌드

이 장에서는 Solaris 운영 환경에서의 응용프로그램 빌드에 대한 자세한 정보를 제공합니다. 스크립트 파일에서 db2로 시작되는 명령은 명령행 처리기(CLP) 명령입니다. CLP 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

Solaris 운영 환경에 대한 최신 DB2 응용프로그램 개발 갱신사항에 대한 자세한 내용은 다음 웹 페이지를 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

주:

1. -mt 멀티스레드 옵션은 Solaris 운영 환경에서 스레드가 구현되는 방식으로 인해 D2 빌드 파일과 makefile의 링크 단계에서 사용됩니다. 이 옵션을 사용하면 약간의 성능 저하가 발생합니다. 최적의 성능을 위해서는 이 옵션을 사용하지 않고 비 스레드 libdb2.so 라이브러리를 사용하여 응용프로그램을 링크할 수 있습니다. 그러나 -mt 스위치가 사용되지 않으면 응용프로그램이 수행될 때 다음과 같은 오류가 발생할 수 있습니다.

   ```
   libc internal error: _rmutex_unlock: rmutex not held
   ```

또는 응용프로그램이 정지되거나 오류 메시지가 나타나지 않습니다.

2. 이 장에서 설명하는 빌드 파일을 사용하여 64비트 응용프로그램을 빌드하려면 표시된 명령의 주석 처리를 제거하십시오.

Forte/WorkShop C

주: Forte/WorkShop C는 이전에 "SPARCompiler C"라고 하였습니다.

이 절에서는 다음 주제에 대해 설명합니다.

- -xarch=v8plusa 옵션 사용
- DB2 CLI 응용프로그램
- DB2 API가 있는 DB2 CLI 응용프로그램
- DB2 CLI 저장 프로시듀어
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)
- 멀티스레드 응용프로그램

-xarch=v8plusa 옵션 사용

Sun WorkShop C 및 C++ 컴파일러를 사용할 때 다음과 같은 오류가 수신되어 실행 파일에 문제점이 생긴 경우,

1. 1행의 구문 오류: 예기치 않은 '('
2. ksh: <application name>: 실행할 수 없음(여기서 application name은 컴파일된 실행 파일의 이름임)

`libdb2.so`로 링크할 때 컴파일러가 유효한 실행 파일을 생성하지 못하는 문제점이 발생한 것일 수 있습니다. 이를 수정하기 위한 한 가지의 제안은 `-xarch=v8plusa` 옵션을 컴파일에 추가하고 명령을 링크하는 것입니다. 예를 들어, 다음은 샘플 응용프로그램 `dynamic.sqc`를 컴파일할 경우입니다.

```
embprep dynamic sample
embprep utilemb sample
cc -c utilemb.c -xarch=v8plusa -I/export/home/db2inst1/sqllib/include
cc -o dynamic dynamic.c utilemb.o -xarch=v8plusa -I/export/home/db2inst1/sqllib/include \
   -L/export/home/db2inst1/sqllib/lib -R/export/home/db2inst1/sqllib/lib -l db2
```

주:

1. Solaris에서 SQL 프로시듀어를 사용 중이고 DB2_SQLROUTINE_COMPILE_COMMAND 프로파일 변수를 통해 자신의 고유 컴파일 문자열을 사용 중일 경우, -xarch=v8plusa 컴파일러 옵션이 포함되어 있는지 확인하십시오. Forte/WorkShop C에 대한 기본 컴파일 명령에는 이 옵션이 있습니다.

   ```
   db2set DB2_SQLROUTINE_COMPILE_COMMAND=cc -xarch=v8plusa -Kpic \
     -I$HOME/sqllib/include SQLROUTINE_FILENAME.c \
   -G -o SQLROUTINE_FILENAME -L$HOME/sqllib/lib \
   -R$HOME/sqllib/lib -ldb2
   ```

2. Solaris에서 64비트 SQL 프로시듀어를 컴파일하려면 -xarch=v8plusa 옵션을 가져와서 위의 명령에 -xarch=v9 옵션을 추가하십시오.

DB2 CLI 응용프로그램

sqllib/samples/cli에 있는 스크립트 파일 bldcli에 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 매개변수 $1은 소스 파일의 이름을 지정합니다.

이것은 Embedded SQL이 없는 CLI 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

프로그램에 .sqc 확장자로 표시되는 Embedded SQL이 들어 있는 경우, 프로그램을 사전 처리 컴파일하기 위해 embprep 스크립트가 호출되고 .c 확장자를 가진 프로그램 파일을 생성합니다.

```
#! /bin/ksh
# bldcli script file -- Solaris
# Builds a DB2 CLI program.
# Usage: bldcli <prog_name> [ <db_name> [ <userid> <password> ]]

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
```

```
then
embprep $1 $2 $3 $4
fi

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi
# Compile the error-checking utility.
cc $CFLAGS_64 -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc $CFLAGS_64 -I$DB2PATH/include -c $1.c
# Link the program.
cc $CFLAGS_64 -o $1 $1.o utilcli.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2
```

bldcli에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러를 사용합니다.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다.

예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 갖고 있습니다.

bldcli에 대한 컴파일 및 링크 옵션
링크 옵션: **cc** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **$CFLAGS_64** 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다. **-o $1** 실행 프로그램을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **utilcli.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면 $HOME/sqllib/lib와 같습니다. **-R$DB2PATH/lib** 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 tbinfo.c에서 샘플 프로그램 tbinfo를 빌드하려면 다음을 입력하십시오.

```
bldcli tbinfo
```

그 결과 실행 파일 tbinfo가 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
tbinfo
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 dbusemx.sqc에서 Embedded SQL 응용프로그램 dbusemx를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

```
bldcli dbusemx
```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldcli dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldcli dbusemx database userid password
   ```

그 결과 실행 파일 dbusemx가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   dbusemx database userid password
   ```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다.

sqllib/samples/cli에 있는 스크립트 파일 bldapi에 DB2 API가 있는 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 이 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 bldcli 스크립트 간의 유일한 차이

점입니다. bldapi 및 bldcli에 공통되는 컴파일 및 링크 옵션에 대한 자세한 내용은 357 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 dbmconn.c에서 샘플 프로그램 dbmconn을 빌드하려면 다음을 입력하십시오.

```
bldapi dbmconn
```

그 결과 실행 파일 dbmconn이 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
dbmconn
```

DB2 CLI 저장 프로시듀어

sqllib/samples/cli에 있는 스크립트 파일 bldclisp에 DB2 CLI 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 매개변수 $1은 소스 파일의 이름을 지정합니다.

```
#! /bin/ksh
# bldclisp script file -- Solaris
# Builds a DB2 CLI stored procedure.
# Usage: bldclisp <prog_name>

# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib

# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi
# Compile the error-checking utility.
cc $CFLAGS_64 -Kpic -I$DB2PATH/include -c utilcli.c

# Compile the program.
cc $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.c

# Link the program.
cc $CFLAGS_64 -G -o $1 $1.o utilcli.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldclisp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-Kpic 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다.
예를 들면, `$HOME/sqllib/include`와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

bldclisp에 대한 컴파일 및 링크 옵션
링크 옵션:
cc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$CFLAGS_64 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
utilcli.o 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다.
-L$DB2PATH/lib 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-R$DB2PATH/lib 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다.
-ldb2 DB2 라이브러리로 링크합니다.
-G 공유 라이브러리를 생성합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 spserver.c에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldclisp spserver
```

스크립트 파일은 저장 프로시듀어를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 저장 프로시듀어 spserver를 빌드하면 이 저장 프로시듀어를 호출하는 CLI 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldcli를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 357 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
: 연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
: 유효한 암호입니다.

클라이언트 응용프로그램은 저장 프로시듀어 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/c에 있는 스크립트 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- Solaris
# Builds a C application program.
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqc" ]]
then
embprep $1 $2 $3 $4
  # Compile the utilemb.c error-checking utility.
  cc $CFLAGS_64 -I$DB2PATH/include -c utilemb.c
else
  # Compile the utilapi.c error-checking utility.
  cc $CFLAGS_64 -I$DB2PATH/include -c utilapi.c
fi
# Compile the program.
cc $CFLAGS_64 -I$DB2PATH/include -c $1.c
if [[ -f $1".sqc" ]]
then
  # Link the program with utilemb.o
  cc $CFLAGS_64 -o $1 $1.o utilemb.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2
```

```
else
  # Link the program with utilapi.o
  cc $CFLAGS_64 -o $1 $1.o utilapi.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다.

예를 들면, `$HOME/sqllib/include`와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 갖고 있습니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션:
cc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$CFLAGS_64 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
utilemb.o Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다.
utilapi.o Embedded SQL 프로그램이 아닌 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다.
-L$DB2PATH/lib 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-R$DB2PATH/lib 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다.
-ldb2 DB2 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.c에서 DB2 API 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다.

실행 파일을 수행하려면 다음 실행 파일 이름을 입력하십시오.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqc에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

sqllib/samples/c에 있는 스크립트 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- Solaris
# Builds a C stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi
# Compile the program.
cc $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.c

# Link the program and create a shared library
cc $CFLAGS_64 -G -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cc C 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-Kpic 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다.
예를 들면, `-I$DB2PATH/include`와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 갖고 있습니다.

링크 옵션:

cc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-G 공유 라이브러리를 생성합니다.

-o $1 실행 파일을 지정합니다.

$1.o 프로그램 오브젝트 파일을 포함합니다.

-L$DB2PATH/lib

링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다.
예를 들면, `$HOME/sqllib/lib`와 같습니다. `-L` 옵션을 지정하지 않으면 `/usr/lib:/lib`가 가정됩니다.

-R$DB2PATH/lib

런타임시에 DB2 공유 라이브러리의 위치를 지정합니다.
예를 들면, `$HOME/sqllib/lib`와 같습니다.

-ldb2 DB2 라이브러리로 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqc에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver database
```

스크립트 파일은 저장 프로시듀어를 sqllib/function 디렉토리로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결합니다.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 저장 프로시듀어 spserver를 빌드하면 이 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 365 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

사용자 정의 함수(UDF)

sqllib/samples/c에 있는 스크립트 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다. UDF는 Embedded SQL문을 포함하지 않습니다. 그러므로 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하거나 프로그램을 사전 처리 컴파일하고 바인드할 필요가 없습니다.

매개변수 $1은 소스 파일의 이름을 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- Solaris
# Builds a C UDF library
# Usage: bldudf <prog_name>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi
# Compile the program.
cc $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.c

# Link the program and create a shared library.
cc $CFLAGS_64 -G -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -ldb2apie
```

```
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf에 대한 컴파일 및 링크 옵션
컴파일 옵션:
cc C 컴파일러.
$CFLAGS_64 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-Kpic 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다.
-I$DB2PATH/include DB2 포함 파일의 위치를 지정합니다. 예를 들면, $HOME/sqllib/include와 같습니다.
-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 갖고 있습니다.

bldudf에 대한 컴파일 및 링크 옵션
링크 옵션:
cc 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$CFLAGS_64 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
-L$DB2PATH/lib 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-R$DB2PATH/lib 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다.
-ldb2 DB2 라이브러리로 링크합니다.
-ldb2apie LOB 위치 지정자를 사용할 수 있도록 DB2 API 엔진 라이브러리로 링크합니다.
-G 공유 라이브러리를 생성합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldudf udfsrv
```

스크립트 파일은 UDF를 sqllib/function 디렉토리로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 이 프로그램의 Embedded SQL 버전 및 DB2 CLI가 제공됩니다.

스크립트 파일 bldcli를 사용하여 sqllib/samples/cli의 소스 파일 udfcli.c에서 DB2 CLI udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 357 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

스크립트 파일 bldapp를 사용하여 sqllib/samples/c에 있는 소스 파일 udfcli.sqc에서 Embedded SQL udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 365 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

Solaris에서 Forte/WorkShop C를 사용하는 멀티스레드 응용프로그램은 -mt로 컴파일 및 링크되어야 합니다. 이 옵션은 -D_REENTRANT를 선행 처리기에 전달하고 -lthread를 링커에 전달합니다. POSIX 스레드는 링커에 전달될 때 -lpthread도 필요합니다. 그 밖에 컴파일러 옵션 -D_POSIX_PTHREAD_SEMANTICS를 사용하면 getpwnam_r()와 같은 함수의 POSIX 가변을 허용합니다.

sqllib/samples/c에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- Solaris
# Builds a C multi-threaded embedded SQL program.
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
  embprep $1 $2 $3 $4
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi
# Compile the program.
cc $CFLAGS_64 -mt -D_POSIX_PTHREAD_SEMANTICS -I$DB2PATH/include -c $1.c
# Link the program.
cc $CFLAGS_64  -mt -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -lpthread
```

위에서 언급한 -mt, -D_POSIX_PTHREAD_SEMANTICS와 -lpthread 옵션 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이들 옵션에 대한 자세한 내용은 365 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqc에서 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서실행 파일을 수행하려면 다음을 입력하십시오.

```
thdsrver
```

연결 수가 공정한 멀티스레드 프로그램의 경우, 다음 커널 매개변수는 기본값을 초과하여 설정해야 할 수도 있습니다. 이들 매개변수는 이를 요구하는 멀티스레드 프로그램이 지역 응용프로그램일 경우에만 재설정해야 합니다.

semsys:seminfo_semume
: 하나의 프로세스에서 사용할 수 있는 세마포어 실행 취소 구조의 한계

shmsys:shminfo_shmseg
: 하나의 프로세스가 작성할 수 있는 공유 메모리 세그먼트 수의 한계

이들 매개변수는 /etc/system 파일에 설정되어 있습니다. 값을 설정하기 위한 지침으로 다음을 사용하십시오. 각 지역 연결마다 DB2는 하나의 세마포어와 하나의 공유 메모리를 사용하여 통신합니다. 멀티스레드 응용프로그램이 지역 응용프로그램이고 DB2에 대해 X개의 연결을 가지고 있는 것으로 가정한 경우, 그 응용프로그램(프로세스)에는 DB2와 통신하기 위해 X개의 공유 메모리 세그먼트와 X개의 세마포어 실행 취소 구조가 필요합니다. 그러므로 두 커널 매개변수의 값은 X + 10(+ 10은 안전을 위한 여분임)으로 설정해야 합니다.

Forte/WorkShop C++

주: Forte/WorkShop C++는 이전에 SPARCompiler C++라고 하였습니다.

이 절에서는 다음 주제에 대해 설명합니다.

- -xarch=v8plusa 옵션 사용
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)
- 멀티스레드 응용프로그램

-xarch=v8plusa 옵션 사용

Sun WorkShop C 및 C++ 컴파일러를 사용할 때 다음과 같은 오류를 수신하여 실행 파일에 문제점이 생긴 경우,

1. 1행의 구문 오류: 예기치 않은 '('
2. ksh: <application name>: 실행할 수 없음(여기서 application name은 컴파일된 실행 파일의 이름임)

libdb2.so로 링크할 때 컴파일러에서 유효한 실행 파일을 생성하지 못하는 문제점이 발생한 것일 수 있습니다. 이를 수정하기 위한 한 가지의 제안은 -xarch=v8plusa 옵션을 컴파일에 추가하고 명령을 링크하는 것입니다. 예를 들어, 다음은 샘플 응용프로그램 dynamic.sqC를 컴파일할 경우입니다.

```
embprep dynamic sample
embprep utilemb sample
CC -c utilemb.C -xarch=v8plusa -I/export/home/db2inst1/sqllib/include
CC -o dynamic dynamic.C utilemb.o -xarch=v8plusa -I/export/home/db2inst1/sqllib/include \
   -L/export/home/db2inst1/sqllib/lib -R/export/home/db2inst1/sqllib/lib -l db2
```

주:

1. Solaris에서 SQL 프로시듀어를 사용 중이고 DB2_SQLROUTINE_COMPILE_COMMAND 프로파일 변수를 통해 자신의 고유 컴파일 문자열을 사용 중일 경우, -xarch=v8plusa 컴파일러 옵션이 포함되어 있는지 확인하십시오. Forte/WorkShop C++에 대한 기본 컴파일 명령에는 이 옵션이 있습니다.

   ```
   db2set DB2_SQLROUTINE_COMPILE_COMMAND=CC -xarch=v8plusa -Kpic \
     -I$HOME/sqllib/include SQLROUTINE_FILENAME.c \
   -G -o SQLROUTINE_FILENAME -L$HOME/sqllib/lib \
   -R$HOME/sqllib/lib -ldb2
   ```

2. Solaris에서 64비트 SQL 프로시듀어를 컴파일하려면 -xarch=v8plusa 옵션을 가져와서 위의 명령에 -xarch=v9 옵션을 추가하십시오.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/cpp에 있는 스크립트 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- Solaris
# Builds a C++ application program.
# Usage:  bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
```

```
then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi
# If an Embedded SQL program, precompile and bind it.
if [[ -f $1".sqC" ]]
then
embprep $1 $2 $3 $4
  # Compile the utilemb.C error-checking utility.
  CC $CFLAGS_64 -I$DB2PATH/include -c utilemb.C
else
  # Compile the utilapi.C error-checking utility.
  CC $CFLAGS_64 -I$DB2PATH/include -c utilapi.C
fi
# Compile the program.
CC $CFLAGS_64 -I$DB2PATH/include -c $1.C
if [[ -f $1".sqC" ]]
then
  # Link the program with utilemb.o
  CC $CFLAGS_64 -o $1 $1.o utilemb.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -mt
else
  # Link the program with utilapi.o
  CC $CFLAGS_64 -o $1 $1.o utilapi.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -mt
fi
```

bldapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

CC C++ 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 ″-xarch=v9″ 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다.

예를 들면, $HOME/sqllib/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 갖고 있습니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션:
CC 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$CFLAGS_64 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
utilemb.o Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다.
utilapi.o 비 Embedded SQL 프로그램의 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다.
-L$DB2PATH/lib 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, `$HOME/sqllib/lib`와 같습니다. `-L` 옵션을 지정하지 않으면 `/usr/lib:/lib`가 가정됩니다.
-R$DB2PATH/lib 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, `$HOME/sqllib/lib`와 같습니다.
-ldb2 DB2 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 `client.C`에서 비 Embedded SQL DB2 API 샘플 프로그램 `client`를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 `client`가 작성됩니다. 다음을 입력하여 `sample` 데이터베이스에 대해서 실행 파일을 수행할 수 있습니다.

client

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqC에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ 저장 프로시듀어 빌드에 대한 정보를 참조하십시오.

sqllib/samples/cpp에 있는 스크립트 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 필요 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름 $1을 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- Solaris
# Builds a C++ stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi
# Compile the program.
CC $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.C
# Link the program and create a shared library
CC $CFLAGS_64 -G -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -mt
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

CC C++ 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-Kpic 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다.
예를 들면, -I$DB2PATH/include와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 갖고 있습니다.

링크 옵션:

CC 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-G 공유 라이브러리를 생성합니다.

-o $1 실행 파일을 지정합니다.

$1.o 프로그램 오브젝트 파일을 포함합니다.

-L$DB2PATH/lib

링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다.
예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.

-R$DB2PATH/lib

런타임시에 DB2 공유 라이브러리의 위치를 지정합니다.
예를 들면, $HOME/sqllib/lib와 같습니다.

-ldb2 DB2 라이브러리로 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

sample 데이터베이스에 연결하고 있는 경우, 소스 파일 spserver.sqC에서 샘플 프로그램 spserver를 빌드하려면 다음을 입력하십시오.

```
bldsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv spserver database
```

스크립트 파일은 공유 라이브러리를 경로 sqllib/function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 공유 라이브러리 spserver를 빌드하면 공유 라이브러리 내의 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 378 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

공유 라이브러리에서 저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 저장 프로시듀어는 클라이언트 응용프로그램으로 출력을 리턴합니다.

사용자 정의 함수(UDF)

주: 72 페이지의 『UDF 및 저장 프로시듀어에 대한 C++ 고려사항』에 있는 C++ UDF 빌드에 대한 정보를 참조하십시오.

sqllib/samples/cpp에 있는 스크립트 파일 bldudf에 UDF를 빌드하기 위한 명령이 들어 있습니다. UDF는 Embedded SQL문을 포함하지 않습니다. 그러므로 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하거나 프로그램을 사전 처리 컴파일하고 바인드할 필요가 없습니다.

매개변수 $1은 소스 파일의 이름을 지정합니다. 스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름을 사용합니다.

```
#! /bin/ksh
# bldudf script file -- Solaris
# Builds a C++ UDF library
# Usage: bldudf <prog_name>
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-xarch=v9
else
  CFLAGS_64=
fi
# Compile the program.
```

```
if [[ -f $1".c" ]]
then
  CC $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.c
elif [[ -f $1".C" ]]
then
CC $CFLAGS_64 -Kpic -I$DB2PATH/include -c $1.C
fi
# Link the program and create a shared library.
CC $CFLAGS_64 -G -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -ldb2apie
# Copy the shared library to the sqllib/function subdirectory.
# Note: the user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

CC C++ 컴파일러.

$CFLAGS_64

'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.

-Kpic 공유 라이브러리에 대한 위치 독립적(position-independent) 코드를 생성합니다.

-I$DB2PATH/include

DB2 포함 파일의 위치를 지정합니다.

예를 들면, `$HOME/sqllib/include`와 같습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 스크립트 파일은 서로 다른 컴파일 및 링크 단계를 갖고 있습니다.

bldudf에 대한 컴파일 및 링크 옵션
링크 옵션:
CC 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다.
$CFLAGS_64 'BUILD_64BIT=true'가 주석 처리되지 않으면 "-xarch=v9" 값을 포함하고, 그렇지 않으면 어떠한 값도 포함하지 않습니다.
-G 공유 라이브러리를 생성합니다.
-o $1 실행 파일을 지정합니다.
$1.o 프로그램 오브젝트 파일을 포함합니다.
-L$DB2PATH/lib 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 /usr/lib:/lib가 가정됩니다.
-R$DB2PATH/lib 런타임시에 DB2 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다.
-ldb2 DB2 라이브러리로 링크합니다.
-ldb2apie LOB 위치 지정자를 사용할 수 있도록 DB2 API 엔진 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) 프로그램 udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldudf udfsrv
```

스크립트 파일은 UDF를 sqllib/function 디렉토리로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 UDF에 대해 파일 모드를 설정하십시오.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 스크립트 파일 bldapp를 사용하여 sqllib/samples/cpp에 있

는 udfcli.sqC 소스 파일에서 udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 378 페이지의『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 호출하려면 다음 실행 파일 이름을 입력하여 샘플 호출 응용프로그램을 수행하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv 라이브러리에서 ScalarUDF 함수를 호출합니다.

멀티스레드 응용프로그램

Solaris에서 Forte/WorkShop C++를 사용하는 멀티스레드 응용프로그램은 -mt로 컴파일 및 링크되어야 합니다. 이 옵션은 -D_REENTRANT를 선행 처리기에 전달하고 -lthread를 링커에 전달합니다. POSIX 스레드는 링커에 전달될 때 -lpthread도 필요합니다. 그 밖에 컴파일러 옵션 -D_POSIX_PTHREAD_SEMANTICS를 사용하면 getpwnam_r()와 같은 함수의 POSIX 가변을 허용합니다.

sqllib/samples/cpp에 있는 스크립트 파일 bldmt에 Embedded SQL 멀티스레드 프로그램을 빌드하기 위한 명령이 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하고 $4는 암호를 지정합니다. 첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름, 사용자 ID 및 암호는 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

```
#! /bin/ksh
# bldmt script file -- Solaris
# Builds a C++ multi-threaded embedded SQL program
# Usage:  bldmt <prog_name> [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
  embprep $1 $2 $3 $4
# To compile 64 bit programs, uncomment the following line.
# BUILD_64BIT=true
if [ "$BUILD_64BIT" != "" ]
then
  CFLAGS_64=-xarch=v9
```

```
else
  CFLAGS_64=
fi
# Compile the program.
CC $CFLAGS_64 -mt -D_POSIX_PTHREAD_SEMANTICS -I$DB2PATH/include -c $1.C
# Link the program.
CC $CFLAGS_64 -mt -o $1 $1.o -L$DB2PATH/lib -R$DB2PATH/lib -ldb2 -lpthread
```

위에서 언급한 -mt, -D_POSIX_PTHREAD_SEMANTICS와 -lpthread 옵션 및 링크된 유틸리티 파일이 없을 때 외에도 다른 컴파일 및 링크 옵션이 Embedded SQL 스크립트 파일 bldapp에 사용된 것과 동일합니다. 이들 옵션에 대한 자세한 내용은 378 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

소스 파일 thdsrver.sqC에서 샘플 프로그램 thdsrver를 빌드하려면 다음을 입력하십시오.

```
bldmt thdsrver
```

그 결과 실행 파일 thdsrver이 작성됩니다. sample 데이터베이스에 대해서 실행 파일을 수행하려면 다음을 입력하십시오.

```
thdsrver
```

연결 수가 공정한 멀티스레드 프로그램의 경우, 다음 커널 매개변수는 기본값을 초과하여 설정해야 할 수도 있습니다. 이들 매개변수는 이를 요구하는 멀티스레드 프로그램이 지역 응용프로그램일 경우에만 재설정해야 합니다.

semsys:seminfo_semume
: 하나의 프로세스에서 사용할 수 있는 세마포어 실행 취소 구조의 한계

shmsys:shminfo_shmseg
: 하나의 프로세스가 작성할 수 있는 공유 메모리 세그먼트 수의 한계

이들 매개변수는 /etc/system 파일에 설정되어 있습니다. 값을 설정하기 위한 지침으로 다음을 사용하십시오. 각 지역 연결마다 DB2는 하나의 세마포어와 하나의 공유 메모리를 사용하여 통신합니다. 멀티스레드 응용프로그램이 지역 응용프로그램이고 DB2에 대해 X개의 연결을 가지고 있는 것으로 가정한 경우, 그 응용프로그램(프로세스)에는 DB2와 통신하기 위해 X개의 공유 메모리 세그먼트와 X개의 세마포어 실행 취소 구조가 필요합니다. 그러므로 두 커널 매개변수의 값은 X + 10(+ 10은 안전을 위한 여분임)으로 설정해야 합니다.

Micro Focus COBOL

이 절에서는 다음 주제에 대해 설명합니다.

- 컴파일러 사용
- DB2 API 및 DB2 Embedded 응용프로그램
- Embedded SQL 저장 프로시듀어

컴파일러 사용

Embedded SQL 및 DB2 API 호출을 포함하는 응용프로그램을 개발하며 Micro Focus COBOL 컴파일러를 사용 중이면 다음 사항에 유의하십시오.

- 명령행 처리기 명령 db2 prep를 사용하여 응용프로그램을 사전 처리 컴파일할 때 target mfcob 옵션(기본값)을 사용하십시오.
- DB2 COBOL COPY 파일 디렉토리를 Micro Focus COBOL 환경 변수 COBCPY에 포함시켜야 합니다. COBCPY 환경 변수는 COPY 파일의 위치를 지정합니다. Micro Focus COBOL용 DB2 COPY 파일은 데이터베이스 인스턴스 디렉토리의의 sqllib/include/cobol_mf에 있습니다.

  이 디렉토리를 포함시키려면 다음을 입력하십시오.

  ```
  export COBCPY=$COBCPY:$HOME/sqllib/include/cobol_mf
  ```

  주: .profile 파일에서 COBCPY를 설정하려 할 수 있습니다.

DB2 API 및 Embedded SQL 응용프로그램

sqllib/samples/cobol_mf에 있는 스크립트 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 $3은 데이터베이스에 대한 사용자 ID를 지정하며 $4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
#! /bin/ksh
# bldapp script file -- Solaris
# Builds a Micro Focus COBOL application program
# Usage: bldapp [ <db_name> [ <userid> <password> ]]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# If an embedded SQL program, precompile and bind it.
if [[ -f $1".sqb" ]]
then
embprep $1 $2 $3 $4
fi
# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$DB2PATH/include/cobol_mf:$COBCPY

# Compile the checkerr.cbl error-checking utility.
cob -cx checkerr.cbl

# Compile the program.
cob -cx $1.cbl

# Link the program.
cob -x $1.o checkerr.o -L$DB2PATH/lib -ldb2 -ldb2gmf
```

bldapp에 대한 컴파일 및 링크 옵션	
컴파일 옵션:	
cob	Micro Focus COBOL 컴파일러.
-cx	오브젝트 모듈로 컴파일합니다.

bldapp에 대한 컴파일 및 링크 옵션
링크 옵션: **cob** 링커에 대한 프론트 엔드로서 컴파일러를 사용합니다. **-x** 실행 프로그램을 지정합니다. **$1.o** 프로그램 오브젝트 파일을 포함합니다. **checkerr.o** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **-L$DB2PATH/lib** 링크시에 DB2 정적 및 공유 라이브러리의 위치를 지정합니다. 예를 들면, $HOME/sqllib/lib와 같습니다. **-ldb2** DB2 라이브러리로 링크합니다. **-ldb2gmf** Micro Focus COBOL용 DB2 예외 핸들러 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.cbl에서 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client가 작성됩니다. 다음을 입력하여 sample 데이터베이스에 대해서 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqb에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

주:

1. Solaris에서 Micro Focus 저장 프로시듀어를 빌드하기 전에 다음 명령을 수행하십시오.

   ```
   db2stop
   db2set DB2LIBPATH=$LD_LIBRARY_PATH
   db2set DB2ENVLIST="COBDIR LD_LIBRARY_PATH"
   db2set
   db2start
   ```

   db2stop이 데이터베이스를 중지했는지 확인하십시오. 마지막 db2set 명령이 발행되어 설정 값을 확인합니다. DB2LIBPATH 및 DB2ENVLIST가 제대로 설정되어 있는지 확인하십시오.

2. Solaris에서 사용되는, 일부 최신 버전의 Micro Focus COBOL 컴파일러는 정적으로 링크된 저장 프로시듀어를 작성하는 데 사용할 수 없습니다. 따라서 makefile과 스크립트 파일 bldsrv가 동적으로 링크된 저장 프로시듀어의 작성을 허용하도록 수정되었습니다.

   원격 클라이언트 응용프로그램이 이러한 동적으로 링크된 정적 프로시듀어를 정상적으로 호출하려면 저장 프로시듀어를 실행하기 직전에 저장 프로시듀어가 상주하는 서버에서 Micro Focus COBOL 루틴 cobinit()를 호출해야 합니다. 이를 수행하는 랩퍼 프로그램이 makefile 또는 스크립트 파일 bldsrv의 실행 중에 작성됩니다. 그런 다음 저장 프로시듀어 코드로 링크되어 저장 프로시듀어 공유 라이브러리를 형성합니다. 이러한 랩퍼 프로그램의 사용으로 인해 클라이언트 응용프로그램이 x라는 저장 프로시듀어를 호출하려면 x 대신 x_wrap를 호출해야 합니다.

   랩퍼 프로그램의 세부사항을 이 절의 뒷 부분에서 설명합니다.

sqllib/samples/cobol_mf에 있는 스크립트 파일 bldsrv에 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 이 스크립트 파일은 클라이언트 응용프로그램에 의해 호출될 수 있는 저장 프로시듀어를 공유 라이브러리에 컴파일합니다.

첫 번째 매개변수 $1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 $2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

스크립트 파일은 공유 라이브러리 이름에 대한 소스 파일 이름 $1을 사용합니다.

```
#! /bin/ksh
# bldsrv script file -- Solaris
# Builds a Micro Focus COBOL stored procedure
# Usage:  bldsrv <prog_name> [ <db_name> ]
# Set DB2PATH to where DB2 will be accessed.
# The default is the standard instance path.
DB2PATH=$HOME/sqllib
# Precompile and bind the program.
embprep $1 $2
# Set COBCPY to include the DB2 COPY files directory.
export COBCPY=$DB2PATH/include/cobol_mf:$COBCPY
```

```
# Compile the program.
cob -cx $1.cbl
# Create the wrapper program for the stored procedure.
wrapsrv $1

# Link the program creating shared library $1 with main entry point ${1}_wrap
cob -x -o $1 ${1}_wrap.c $1.o -Q -G -L$DB2PATH/lib -ldb2 -ldb2gmf
# Copy the shared library to the sqllib/function subdirectory.
# The user must have write permission to this directory.
rm -f $DB2PATH/function/$1
cp $1 $DB2PATH/function
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cob COBOL 컴파일러.

-cx 오브젝트 모듈로 컴파일합니다.

링크 옵션:

cob 편집기를 링크하려면 컴파일러를 사용합니다.

-x 실행 프로그램을 만듭니다.

-o $1 실행 프로그램을 지정합니다.

${1}_wrap.c
랩퍼 프로그램을 지정합니다.

$1.o 프로그램 오브젝트 파일을 지정합니다.

-Q

-G

-L$DB2PATH/lib
DB2 런타임 공유 라이브러리의 위치를 지정합니다.
예를 들면, $HOME/sqllib/lib와 같습니다. -L 옵션을 지정하지 않으면 컴파일러는 /usr/lib:/lib 경로로 지정합니다.

-ldb2 DB2 라이브러리로 링크합니다.

-ldb2gmf
Micro Focus COBOL용 DB2 예외 핸들러 라이브러리로 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

랩퍼 프로그램 wrapsrv로 인해 저장 프로시듀어가 실행되기 직전에 Micro Focus COBOL 루틴 cobinit()가 호출됩니다. 내용은 다음과 같습니다.

```
#! /bin/ksh
# wrapsrv script file
# Creates the wrapper program for Micro Focus COBOL stored procedures
# Usage: wrapsrv <stored_proc>
# Note: The client program calls "<stored_proc>_wrap" not "<stored_proc>"

# Create the wrapper program for the stored procedure.
cat << WRAPPER_CODE > ${1}_wrap.c
#include <stdio.h>
void cobinit(void);
int $1(void *p0, void *p1, void *p2, void *p3);
int main(void)
{
  return 0;
}
int ${1}_wrap(void *p0, void *p1, void *p2, void *p3)
{
  cobinit();
  return $1(p0, p1, p2, p3);
}
WRAPPER_CODE
```

샘플 프로그램에 연결하고 있는 경우, 소스 파일 outsrv.sqb에서 샘플 프로그램 outsrv를 빌드하려면 다음을 입력하십시오.

```
bldsrv outsrv
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv outsrv database
```

스크립트 파일은 저장 프로시듀어를 경로 sqllib/function에 있는 서버로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 저장 프로시듀어에 대해 파일 모드를 설정하십시오.

일단 저장 프로시듀어 outsrv를 빌드하면 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 outcli를 빌드할 수 있습니다 스크립트 파일 bldapp를 사용하여 outcli를 빌드할 수 있습니다. 자세한 내용은 390 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

```
outcli database userid password
```

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 저장 프로시듀어 라이브러리 outsrv에 액세스하고 서버 데이터베이스에서 동일한 이름의 저장 프로시듀어 함수를 실행한 다음 출력을 클라이언트 응용프로그램으로 리턴합니다.

저장 프로시듀어 나감

저장 프로시듀어를 개발할 때 다음 명령문을 사용하여 저장 프로시듀어를 나가십시오.

```
move SQLZ-HOLD-PROC to return-code.
```

이 명령문을 통해 저장 프로시듀어가 클라이언트 응용프로그램으로 올바로 리턴합니다. 이것은 저장 프로시듀어가 지역 COBOL 클라이언트 응용프로그램에 의해 호출될 때 특히 중요합니다.

제13장 Windows 32비트 운영 체제용 응용프로그램 빌드

이 장에서는 Windows 32비트 운영 체제에서의 응용프로그램 빌드에 대한 자세한 정보를 제공합니다. 배치 파일에서 db2로 시작되는 명령은 명령행 처리기(CLP) 명령입니다. DB2 명령에 대한 자세한 내용은 *Command Reference*를 참조하십시오.

Windows 32비트 운영 체제에 대한 최신 DB2 응용프로그램 개발 갱신사항에 대한 자세한 내용은 다음 웹 페이지를 참조하십시오.

```
http://www.ibm.com/software/data/db2/udb/ad
```

주:

1. Windows 32비트 운영 체제의 모든 응용프로그램, Embedded SQL 및 비 Embedded SQL은 DB2 명령 창에서 빌드되어야 하며 운영 체제 명령 프롬프트에서 빌드되지 않아야 합니다.
2. 버전 7.1을 위한 Windows 32비트 운영 체제의 DB2에 대한 기본 설치가 공백을 포함하는 `\Program Files\sqllib`일 때 `%DB2PATH%` 변수를 포함하는 프로그램에서 사용되는 경로 이름은 `"%DB2PATH%\function"`처럼 따옴표로 묶어야 합니다. 따옴표를 사용하지 않으면 "명령 구문이 올바르지 않음"과 같은 오류가 발생할 수 있습니다. 이 장에서는 이러한 경로 이름이 명령 또는 코드 예의 일부로 제공된 경우에만 따옴표로 묶어서 표시합니다.

WCHARTYPE CONVERT 사전 처리 컴파일 옵션

WCHARTYPE 사전 처리 컴파일 옵션은 `wchar_t` 데이터 유형을 사용하여 복수 바이트 형식이나 와이드 문자 형식의 그래픽 데이터를 처리합니다. 이 옵션에 대한 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.

Windows 32비트 운영 체제에 대한 DB의 경우, Microsoft Visual C++ 컴파일러로 컴파일된 응용프로그램에 대해 WCHARTYPE CONVERT 옵션이 지원됩니다. 단, 응용프로그램이 데이터베이스 코드 페이지와 다른 코드 페이지의 DB2 데이터베이스에 데이터를 삽입하는 경우, CONVERT 옵션을 이 컴파일러에서 사용하지 마십시오. 이 경우 DB2는 일반적으로 코드 페이지 변환을 수행합니다. 그러나 Microsoft C 런타임 환경은 특정한 2바이트 문자에 대해서는 대체 문자를 처리하지 않습니다. 이는 런타임 변환 오류 결과로 나타날 수 있습니다.

WCHARTYPE CONVERT 옵션은 IBM VisualAge C++ 컴파일러로 컴파일된 응용프로그램에 대해서는 지원되지 않습니다. 이 컴파일러의 경우에는 WCHARTYPE에 대한 기본 NOCONVERT 옵션을 사용하십시오. NOCONVERT 옵션을 사용할 경우, 응용프로그램과 데이터베이스 관리자 간에 내재된 문자 변환이 발생하지 않습니다. 그래픽 호스트 변수에 있는 데이터는 변경되지 않은 2바이트 문자 세트(DBCS) 문자로 데이터베이스 관리자와 전송 및 수신됩니다.

그래픽 데이터를 와이드 문자 형식에서 복수 바이트 형식으로 변환해야 할 경우 wcstombs() 함수를 사용하십시오. 예를 들면, 다음과 같습니다.

```
wchar_t widechar[200];
wchar_t mb[200];
wcstombs((char *)mb,widechar,200);
EXEC SQL INSERT INTO TABLENAME VALUES(:mb);
```

마찬가지로 mbstowcs() 함수를 사용하여 복수 바이트 형식에서 와이드 문자 형식으로 변환할 수 있습니다.

응용프로그램을 C 런타임 라이브러리에 정적으로 바인드된 경우, C 런타임 변환 오류를 일으킬 수 있으므로 응용프로그램에서 setlocale() 호출을 발행하지 마십시오. 응용프로그램이 C 런타임 라이브러리에 동적으로 바인드된 경우, setlocale()의 사용은 문제가 되지 않습니다. 저장 프로시듀어에 대한 경우도 마찬가지입니다.

OLE DB 테이블 함수

DB2는 OLE DB 테이블 함수를 지원합니다. 이들 함수에 대해서는 CREATE FUNCTION DDL을 작성하는 것 외에 응용프로그램을 빌드할 필요가 없습니다. OLE DB 테이블 함수 샘플 파일은 %DB2PATH%\samples\oledb 디렉토리에서 DB2에 의해 제공됩니다. 이들 파일은 명령행 처리기(CLP) 파일입니다. 다음 단계를 통해 빌드할 수 있습니다.

1. db2 connect to database_name
2. db2 -t -v -f file_name.db2
3. db2 terminate

여기서 database_name는 연결 중인 데이터베이스이고 file_name은 .db2 확장자를 갖는 CLP 파일의 이름입니다.

OLE DB 테이블 함수에 대한 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.

Microsoft Visual Basic

주: Windows 32비트 운영 체제용 DB2 AD Client는 Microsoft Visual Basic에 대한 사전 처리 컴파일러를 제공하지 않습니다.

이 절에서는 다음 주제에 대해 설명합니다.

- ADO(ActiveX Data Objects)
- RDO(Remote Data Objects)
- OLE(Object Linking and Embedding) 자동화

ADO(ActiveX Data Objects)

ADO를 사용하여 OLE DB 제공업체를 통해 데이터베이스 서버에 있는 데이터에 액세스하고 조작하는 응용프로그램을 작성할 수 있습니다. ADO의 기본적인 이점은 고속, 사용의 용이성, 낮은 메모리 오버헤드 및 작은 디스크 자국(footprint)입니다.

Microsoft Visual Basic에서 ADO를 사용하려면 ADO 유형 라이브러리에 대한 참조를 설정해야 합니다. 다음을 수행하십시오.

1. 프로젝트 메뉴에서 "참조"를 선택하십시오.
2. "Microsoft ActiveX Data Objects <version_number> 라이브러리"를 선택하십시오.
3. "확인"을 선택하십시오.

여기서 <version_number>는 현재 ADO 라이브러리의 버전입니다.

이를 완료하면 VBA 오브젝트 브라우저 및 IDE 편집기를 통해 ADO 오브젝트, 메소드 및 등록 정보에 액세스할 수 있게 됩니다.

전체 Visual Basic 프로그램은 양식 및 기타 그래픽 요소를 포함하며, 사용자는 Visual Basic 환경에서 이를 보아야 합니다. 다음은 Visual Basic 명령으로, ODBC에서 카탈로그화된 DB2 `sample` 데이터베이스에 액세스하기 위한 프로그램의 일부입니다.

연결을 설정하십시오.

```
Dim db As Connection
Set db = New Connection
```

지역 커서 라이브러리에 의해 제공되는 클라이언트측 커서를 설정하십시오.

```
db.CursorLocation = adUseClient
```

ADO가 Microsoft ODBC Driver를 사용하도록 제공업체를 설정한 다음 사용자 ID/암호를 사용하지 않고 데이터베이스 ″sample″을 여십시오. 즉 다음과 같이 현재 사용자를 사용하십시오.

```
db.Open "SAMPLE"
```

다음 레코드 세트를 작성하십시오.

```
Set adoPrimaryRS = New Recordset
```

다음과 같이 레코드 세트를 채우기 위해 select문을 사용하십시오.

```
adoPrimaryRS.Open "select EMPNO,LASTNAME,FIRSTNME,MIDINIT,EDLEVEL,JOB
from EMPLOYEE Order by EMPNO", db
```

여기서 프로그래머는 다음 레코드 세트로 이동하는 것과 같이 ADO 메소드를 사용하여 데이터에 액세스할 수 있습니다.

```
adoPrimaryRS.MoveNext
```

레코드 세트에 있는 현재 레코드 삭제:

```
adoPrimaryRS.Delete
```

또한 프로그래머는 각 필드에 액세스하기 위해 다음을 수행할 수 있습니다.

```
Dim Text1 as String
Text1 = adoPrimaryRS!LASTNAME
```

DB2는 %DB2PATH%\samples\ADO\VB 디렉토리에서 Visual Basic ADO 샘플 프로그램을 제공합니다.

RDO(Remote Data Objects)

RDO는 ODBC를 통해 원격 데이터 소스에 액세스하기 위한 정보 모델을 제공합니다. RDO는 데이터베이스에 대한 연결, 조회 및 저장 프로시듀어 실행, 결과 조작 및 서버에 대한 변경사항 확약을 쉽게 해주는 오브젝트 세트를 제공합니다. 원

격 ODBC 관계형 데이터 소스에 액세스하도록 설계되었으며, 복잡한 응용프로그램 코드를 사용하지 않고도 보다 쉽게 ODBC를 사용하게 하며, ODBC 드라이버로 표시되는 관계형 데이터베이스에 액세스하는 1차적인 방법입니다. RDO는 연결을 설정하는 ODBC API와 드라이버 관리자에 thin 코드 계층을 구현하고 결과 세트와 커서를 작성하며 최소한의 워크스테이션 자원을 사용하여 복잡한 프로시듀어를 실행합니다.

Microsoft Visual Basic에서 RDO를 사용하려면 Visual Basic 프로젝트에 대한 참조를 설정해야 합니다. 다음을 수행하십시오.

1. 프로젝트 메뉴에서 "참조"를 선택하십시오.
2. "Microsoft Remote Data Object <Version Number>"를 선택하십시오.
3. "확인"을 선택하십시오.

여기서 <version_number>는 현재 RDO 버전입니다.

전체 Visual Basic 프로그램은 양식 및 기타 그래픽 요소를 포함하며, 사용자는 Visual Basic 환경에서 이를 보아야 합니다. 다음은 Visual Basic 명령으로, sample 데이터베이스에 연결하여 EMPLOYEE 테이블에서 모든 컬럼을 선택하는 레코드 세트를 열고, 사원 이름을 하나씩 메시지 창에 표시하는 DB2 프로그램의 일부입니다.

```
Dim rdoEn As rdoEngine
Dim rdoEv As rdoEnvironment
Dim rdoCn As rdoConnection
Dim Cnct$
Dim rdoRS As rdoResultset
Dim SQLQueryDB As String
```

연결 문자열을 지정하십시오.

```
Cnct$ = "DSN=SAMPLE;UID=;PWD=;"
```

RDO 환경을 설정하십시오.

```
Set rdoEn = rdoEngine
Set rdoEv = rdoEn.rdoEnvironments(0)
```

데이터베이스에 연결하십시오.

```
Set rdoCn = rdoEv.OpenConnection("", , , Cnct$)
```

레코드 세트에 대해 SELECT문을 지정하십시오.

```
SQLQueryDB = "SELECT * FROM EMPLOYEE"
```

레코드 세트를 열고 조회를 실행하십시오.

```
Set rdoRS = rdoCn.OpenResultset(SQLQueryDB)
```

레코드 세트의 맨 끝에 오기 전에 한 번에 사원 한 명씩 테이블에서 LASTNAME과 FIRSTNME이 있는 메시지 상자를 표시하십시오.

```
While Not rdoRS.EOF
MsgBox rdoRS!LASTNAME &", " &rdoRS!FIRSTNME
```

레코드 세트에서 다음 행으로 이동하십시오.

```
rdoRS.MoveNext
Wend
```

프로그램을 닫으십시오.

```
rdoRS.Close
rdoCn.Close
rdoEv.Close
```

DB2는 %DB2PATH%\samples\RDO 디렉토리에서 Visual Basic RDO 샘플 프로그램을 제공합니다.

OLE(Object Linking and Embedding) 자동화

이 절에서는 저장 프로시듀어에 대한 샘플 OLE 자동화 제어기에 대한 액세스뿐 아니라 Microsoft Visual Basic의 OLE 자동화 UDF에 대해 설명합니다.

OLE는 언어에 대해 독립적이므로 OLE 자동화 서버의 메소드를 표시하고 이 메소드를 DB2의 UDF로 등록하여 OLE 자동화 UDF와 저장 프로시듀어를 어떤 언어로든 구현할 수 있습니다. OLE 자동화 서버의 개발을 지원하는 응용프로그램 개발 환경은 Microsoft Visual Basic, Microsoft Visual C++, Microsoft Visual J++, Microsoft FoxPro, Borland Delphi, Powersoft PowerBuilder 및 Micro Focus COBOL의 버전을 포함합니다. 또한 OLE에 대해(예를 들어, Microsoft Visual J++를 사용하여) 적절히 랩핑된 Java Bean은 OLE 자동화를 통해 액세스할 수 있습니다.

OLE 자동화 서버 개발에 대한 자세한 내용은 해당 응용프로그램 개발 환경의 문서를 참조해야 합니다. OLE 자동화를 사용하는 DB2 프로그래밍에 대한 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.

OLE 자동화 UDF 및 저장 프로시듀어

Microsoft Visual Basic은 OLE 자동화 서버의 작성을 지원합니다. Visual Basic 프로젝트에 클래스 모듈을 추가하여 새로운 종류의 오브젝트가 Visual Basic에 작성됩니다. 메소드는 일반(public) 서브 프로시듀어를 클래스 모듈에 추가하여 작성됩니다. 이 일반 프로시듀어는 DB2에 OLE 자동화 UDF 및 저장 프로시듀어로서 등록될 수 있습니다. OLE 서버 작성 및 빌드에 대한 자세한 내용은 Microsoft Visual Basic 매뉴얼, *OLE 서버 작성, Microsoft Corporation, 1995* 및 Microsoft Visual Basic이 제공하는 OLE 샘플을 참조하십시오.

DB2는 `%DB2PATH%\samples\ole\msvb` 디렉토리에서 자체적으로 보유하고 있는 Microsoft Visual Basic의 OLE 자동화 UDF 및 저장 프로시듀어의 샘플을 제공합니다. OLE 자동화 UDF 샘플 및 저장 프로시듀어 샘플의 빌드와 수행에 대한 자세한 내용은 `%DB2PATH%\samples\ole`에 있는 `README` 파일을 참조하십시오.

Microsoft Visual C++

이 절에서는 다음 주제에 대해 설명합니다.

- ADO(ActiveX Data Objects)
- OLE(Object Linking and Embedding) 자동화
- DB2 CLI 응용프로그램
- DB2 CLI 저장 프로시듀어
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)

주: Visual C++ 컴파일러는 `%DB2PATH%\samples\c` 및 `%DB2PATH%\samples\cpp` 디렉토리에 제공되는 C 및 C++ 샘플 프로그램에 모두 사용됩니다. 이 두 디렉토리에는 동일한 배치 파일이 있습니다. 이 두 디렉토리에는 파일 확장자에 따라 C 또는 C++ 소스 파일을 승인하기 위한 명령이 들어 있습니다. 배치 파

일은 처음 두 항목 ″ADO(ActiveX Data Objects)″와 ″OLE(Object Linking and Embedding) 자동화″를 제외하고 프로그램 빌드를 설명하기 위해 이 절에서 사용됩니다.

ADO(ActiveX Data Objects)

다음과 같이 변경하면 Visual C++를 사용하는 DB2 ADO 프로그램은 일반 C++ 프로그램과 동일하게 컴파일될 수 있습니다.

사용자의 C++ 소스 프로그램을 ADO 프로그램처럼 수행되게 하기 위해 다음의 Import문을 소스 프로그램 파일의 맨 위에 삽입할 수 있습니다.

```
#import "C:\program files\common files\system\ado\msado<VERSION NUMBER>.dll" \
no_namespace \
rename( "EOF", "adoEOF")
```

여기서 <VERSION NUMBER>는 ADO 라이브러리의 버전입니다.

프로그램이 컴파일되면 msado<VERSION NUMBER>.dll이 지정된 경로에 있는지 검증해야 합니다. 또 다른 방법은 C:\program files\common files\system\ado를 환경 변수 LIBPATH에 추가하고 다음의 짧은 Import문을 소스 파일에서 사용하는 것입니다.

```
#import <msado<VERSION NUMBER>.dll> \
no_namespace \
rename( "EOF", "adoEOF")
```

이는 DB2 샘플 프로그램 BLOBAccess.dsp에서 사용되는 메소드입니다.

이 IMPORT문을 사용하여 DB2 프로그램이 ADO 라이브러리에 대한 액세스 권한을 갖게 됩니다. 이제 다른 프로그램처럼 Visual C++ 프로그램을 컴파일할 수 있게 되었습니다. DB2 API 또는 DB2 CLI와 같은 다른 프로그래밍 인터페이스도 사용하고 있는 경우, 프로그램 빌드에 대한 자세한 내용은 이 장의 해당 절을 참조하십시오.

DB2는 %DB2PATH%\samples\ADO\VC 디렉토리에서 Visual C++ ADO 샘플 프로그램을 제공합니다.

OLE(Object Linking and Embedding) 자동화

이 절에서는 저장 프로시듀어에 대한 샘플 OLE 자동화 제어기뿐 아니라 Microsoft Visual C++의 OLE 자동화 UDF에 대해서도 설명합니다.

OLE는 언어에 대해 독립적이므로 OLE 자동화 서버의 메소드를 표시하고 이 메소드를 DB2의 UDF로 등록하여 OLE 자동화 UDF와 저장 프로시듀어를 어떤 언어로든 구현할 수 있습니다. OLE 자동화 서버의 개발을 지원하는 응용프로그램 개발 환경은 Microsoft Visual Basic, Microsoft Visual C++, Microsoft Visual J++, Microsoft FoxPro, Borland Delphi, Powersoft PowerBuilder 및 Micro Focus COBOL의 버전을 포함합니다. 또한 OLE에 대해(예를 들어, Microsoft Visual J++를 사용하여) 적절히 랩핑된 Java Bean은 OLE 자동화를 통해 액세스할 수 있습니다.

OLE 자동화 서버 개발에 대한 자세한 내용은 해당 응용프로그램 개발 환경의 문서를 참조해야 합니다. OLE 자동화를 사용하는 DB2 프로그래밍에 대한 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.

OLE 자동화 UDF 및 저장 프로시듀어

Microsoft Visual C++는 OLE 자동화 서버의 작성을 지원합니다. 서버는 Microsoft Foundation Classes 및 Microsoft Foundation Class 응용프로그램 마법사를 사용하여 구현되거나 Win32 응용프로그램으로 구현될 수 있습니다. 서버는 DLL 또는 EXE가 될 수 있습니다. 자세한 내용은 Microsoft Visual C++ 문서 및 Microsoft Visual C++가 제공하는 OLE 샘플을 참조하십시오. DB2용 Visual C++ UDF 빌드에 대한 자세한 내용은 423 페이지의 『사용자 정의 함수(UDF)』를 참조하십시오. DB2 CLI를 사용한 Visual C++ 저장 프로시듀어 빌드에 대한 자세한 내용은 412 페이지의 『DB2 CLI 저장 프로시듀어』를 참조하십시오. DB2용 Visual C++ Embedded SQL 저장 프로시듀어 빌드에 대한 자세한 내용은 419 페이지의 『Embedded SQL 저장 프로시듀어』를 참조하십시오.

DB2는 `%DB2PATH%\samples\ole\msvc` 디렉토리에서 자체적으로 보유하고 있는 Microsoft Visual C++의 OLE 자동화 UDF 및 저장 프로시듀어의 샘플을 제공합니다. OLE 자동화 UDF 샘플 및 저장 프로시듀어 샘플의 빌드와 수행에 대한 정보는 `%DB2PATH%\samples\ole`에 있는 `README` 파일을 참조하십시오.

DB2 CLI 응용프로그램

%DB2PATH%\samples\cli에 있는 배치 파일 bldmcli.bat에 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다.

매개변수 %1은 소스 파일의 이름을 지정합니다.

이것은 유일한 필수 매개변수로 Embedded SQL이 없는 CLI 프로그램에 필요한 유일한 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하며 %4는 암호를 지정합니다.

프로그램에 .sqc 또는 .sqx 확장자로 표시되는 Embedded SQL이 들어 있는 경우, 프로그램을 사전 처리 컴파일하기 위해 embprep 배치 파일이 호출되고 각각 .c 또는 .cxx 확장자를 가진 프로그램 파일을 생성합니다.

```
@echo off
rem  bldmcli batch file - Windows 32-bit Operating Systems
rem  Builds a CLI program with Microsoft Visual C++.
rem  Usage: bldmcli prog_name [ db_name [ userid password ]]

if exist "%1.sqc" call embprep %1 %2 %3 %4
if exist "%1.sqx" call embprep %1 %2 %3 %4
rem Compile the error-checking utility.
cl -Z7 -Od -c -W1 -D_X86=1 -DWIN32 utilcli.c

rem  Compile the program.
if exist "%1.sqx" goto cpp
cl -Z7 -Od -c -W1 -D_X86=1 -DWIN32 %1.c
goto link_step
:cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx

rem  Link the program.
:link_step
link -debug:full -debugtype:cv -OUT:%1.exe %1.obj utilcli.obj db2cli.lib
@echo on
```

bldmcli에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cl Microsoft Visual C++ 컴파일러.

-Z7 생성된 C7 스타일 CodeView 정보.

-Od 최적화를 사용하지 않습니다. 최적화를 사용하지 않고 디버거를 사용하는 것이 더 쉽습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

-W1 경고 레벨을 설정합니다.

-D_X86_=1
Intel 기반 컴퓨터에서 수행하기 위해 Windows 32비트 운영 체제에 필요한 컴파일러 옵션.

-DWIN32
Windows 32비트 운영 체제에 필요한 컴파일러 옵션.

링크 옵션:

link 편집기를 링크하기 위해 32비트 링커를 사용합니다.

-debug:full
디버깅 정보를 포함시킵니다.

-debugtype:cv
디버거 유형을 나타냅니다.

-OUT:%1.exe
실행 파일을 지정합니다.

%1.obj 오브젝트 파일을 포함시킵니다.

utilcli.obj
오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다.

db2cli.lib
DB2 CLI 라이브러리로 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 tbinfo.c에서 샘플 프로그램 tbinfo를 빌드하려면 다음을 입력하십시오.

```
bldmcli tbinfo
```

그 결과 실행 파일 tbinfo가 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
tbinfo
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 dbusemx.sqc에서 Embedded SQL 응용프로그램 dbusemx를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldmcli dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldmcli dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldmcli dbusemx database userid password
   ```

그 결과 실행 파일 dbusemx가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   dbusemx database userid password
   ```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다.

sqllib/samples/cli에 있는 스크립트 파일 bldmapi에 DB2 API가 있는 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 이 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 bldmcli 배치 파일 간의 유일한 차이점입니다. bldmapi 및 bldmcli에 공통되는 컴파일 및 링크 옵션에 대한 자세한 내용은 409 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 dbmconn.c에서 샘플 프로그램 dbmconn을 빌드하려면 다음을 입력하십시오.

```
bldmapi dbmconn
```

그 결과 실행 파일 dbmconn이 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
dbmconn
```

DB2 CLI 저장 프로시듀어

%DB2PATH%\samples\cli에 있는 배치 파일 bldmclis.bat에 CLI 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 배치 파일은 저장 프로시듀어를 서버의 DLL로 빌드합니다.

매개변수 %1은 소스 파일의 이름을 지정합니다. 배치 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldmclis.bat file - Windows 32-bit Operating Systems
rem Builds a CLI stored procedure using the Microsoft Visual C++ compiler.
rem Usage: bldmclis prog_name

if "%1" == "" goto error
rem  Compile the program.
if exist "%1.cxx" goto cpp
```

```
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c  utilcli.c
goto link_step
:cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx  utilcli.c

rem  Link the program.
:link_step
link -debug:full -debugtype:cv -dll -out:%1.dll %1.obj utilcli.
obj db2cli.lib -def:%1.def
rem Copy the stored procedure DLL to the 'function' directory
copy %1.dll "%DB2PATH%\function"

goto exit
:error
echo Usage: bldmclis prog_name
:exit
@echo on
```

bldmclis에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cl Microsoft Visual C++ 컴파일러.

-Z7 생성된 C7 스타일 CodeView 정보.

-Od 최적화를 사용하지 않습니다. 최적화를 사용하지 않고 디버거를 사용하는 것이 더 쉽습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

-W2 경고 레벨을 설정합니다.

-D_X86_=1
Intel 기반 컴퓨터에서 수행하기 위해 Windows 32비트 운영 체제에 필요한 컴파일러 옵션.

-DWIN32
Windows 32비트 운영 체제에 필요한 컴파일러 옵션.

bldmclis에 대한 컴파일 및 링크 옵션
링크 옵션: **link** 편집기를 링크하기 위해 32비트 링커를 사용합니다. **-debug:full** 디버깅 정보를 포함시킵니다. **-debugtype:cv** 디버거 유형을 나타냅니다. **-OUT:%1.dll** .DLL 파일을 빌드합니다. **%1.obj** 오브젝트 파일을 포함시킵니다. **utilcli.obj** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **db2cli.lib** DB2 CLI 라이브러리로 링크합니다. **-def:%1.def** 모듈 정의 파일을 사용합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 spserver.c에서 spserver 저장 프로시듀어를 빌드하려면 다음을 입력하십시오.

```
bldmclis spserver
```

배치 파일은 CLI 샘플 프로그램과 동일한 디렉토리에 있는 모듈 정의 파일 spserver.def를 사용하여 저장 프로시듀어를 빌드합니다. 배치 파일은 저장 프로시듀어 DLL spserver.dll을 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 저장 프로시듀어 spserver를 빌드하면 이 저장 프로시듀어를 호출하는 CLI 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldmcli를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 409 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 저장 프로시듀어 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

DB2 API 및 Embedded SQL 응용프로그램

%DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 배치 파일 bldmapp.bat에 Embedded SQL 프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하며 %4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldmapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
@echo off
rem bldmapp.bat -- Windows 32-bit operating systems
rem Builds a Microsoft Visual C++ application program
rem Usage: bldmapp prog_name [ db_name [ userid password ]]
if exist "%1.sqx" goto embedded
if exist "%1.sqc" goto embedded
goto non_embedded
:embedded
rem Precompile and bind the program.
call embprep %1 %2 %3 %4
rem  Compile the program.
if exist "%1.cxx" goto cpp_emb
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c utilemb.c
goto link_embedded
:cpp_emb
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx utilemb.cxx
rem  Link the program.
:link_embedded
link -debug:full -debugtype:cv -out:%1.exe %1.obj utilemb.obj db2api.lib
goto exit
:non_embedded
rem  Compile the program.
if exist "%1.cxx" goto cpp_non
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c utilapi.c
goto link_non_embedded
:cpp_non
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx utilapi.cxx
rem  Link the program.
```

```
:link_non_embedded
link -debug:full -debugtype:cv -out:%1.exe %1.obj utilapi.obj db2api.lib
:exit
@echo on
```

bldmapp에 대한 컴파일 및 링크 옵션
컴파일 옵션:
cl Microsoft Visual C++ 컴파일러.
-Z7 생성된 C7 스타일 CodeView 정보.
-Od 최적화를 사용하지 않습니다. 최적화를 사용하지 않고 디버거를 사용하는 것이 더 쉽습니다.
-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.
-W2 경고 레벨을 설정합니다.
-D_X86_=1 Intel 기반 컴퓨터에서 수행하기 위해 Windows 32비트 운영 체제에 필요한 컴파일러 옵션.
-DWIN32 Windows 32비트 운영 체제에 필요한 컴파일러 옵션.

bldmapp에 대한 컴파일 및 링크 옵션
링크 옵션: **link** 편집기를 링크하려면 32비트 링커를 사용합니다. **-debug:full** 디버깅 정보를 포함시킵니다. **-debugtype:cv** 디버거 유형을 나타냅니다. **-out:%1.exe** 파일 이름을 지정합니다. **%1.obj** 오브젝트 파일을 포함시킵니다. **utilemb.obj** Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다. **utilapi.obj** Embedded SQL 프로그램이 아닌 경우 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다. **db2api.lib** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

%DB2PATH%\samples\c에 있는 소스 파일 client.c 또는 %DB2PATH%\samples\cpp에 있는 소스 파일 client.cxx에서 DB2 API 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldmapp client
```

그 결과 실행 파일 client.exe가 작성됩니다. 다음과 같이 명령행에서 확장자 없이 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

%DB2PATH%\samples\c에 있는 C 소스 파일 updat.sqc 또는 %DB2PATH%\samples\cpp에 있는 C++ 소스 파일 updat.sqx에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldmapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldmapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldmapp updat database userid password
   ```

그 결과 실행 파일 updat.exe가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

%DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 배치 파일 bldmsrv.bat에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 배치 파일은 저장 프로시듀어를 서버의 DLL로 빌드합니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 필요 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

배치 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldmsrv.bat -- Windows 32-bit operating systems
rem Builds a Microsoft Visual C++ stored procedure
rem Usage: bldmsrv prog_name [ db_name ]
rem Precompile and bind the program.
call embprep %1 %2

rem  Compile the program.
if exist "%1.cxx" goto cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c
goto link_step
:cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx

:link_step
rem  Link the program.
link -debug:full -debugtype:cv -out:%1.dll -dll %1.obj db2api.lib -def:%1.def

rem Copy the stored procedure DLL to the 'function' directory
copy %1.dll "%DB2PATH%\function"
@echo on
```

bldmsrv에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cl** Microsoft Visual C++ 컴파일러. **-Z7** 생성된 C7 스타일 CodeView 정보. **-Od** 최적화를 사용하지 않습니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다. **-W2** 출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지. **-D_X86_=1** Intel 기반 컴퓨터에서 수행하기 위해 Windows 32비트 운영 체제에 필요한 컴파일러 옵션. **-DWIN32** Windows 32비트 운영 체제에 필요한 컴파일러 옵션.
링크 옵션: **link** 편집기를 링크하려면 링커를 사용합니다. **-debug:full** 디버깅 정보를 포함시킵니다. **-debugtype:cv** 디버거 유형을 나타냅니다. **-out:%1.dll** .DLL 파일을 빌드합니다. **%1.obj** 오브젝트 파일을 포함시킵니다. **db2api.lib** DB2 라이브러리로 링크합니다. **-def:%1.def** 모듈 정의 파일. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

C 소스 파일 spserver.sqc 또는 C++ 소스 파일 spserver.sqx에서 spserver 저장 프로시듀어 DLL을 빌드하려면 다음을 입력하십시오.

```
bldmsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldmsrv spserver database
```

배치 파일은 샘플 프로그램과 동일한 디렉토리에 있는 모듈 정의 파일 spserver.def를 사용하여 DLL을 빌드합니다. 배치 파일은 DLL spserver.dll을 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 저장 프로시듀어 DLL spserver를 빌드하면 이 DLL을 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldmapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 416 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

```
spclient database userid password
```

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 저장 프로시듀어 DLL spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

사용자 정의 함수(UDF)

%DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 배치 파일 bldmudf에 UDF를 빌드하기 위한 명령이 들어 있습니다.

UDF는 Embedded SQL문을 포함할 수 없습니다. 그러므로 UDF 프로그램을 빌드하기 위해 데이터베이스에 연결하여 프로그램을 사전 처리 컴파일하고 바인드할 필요가 없습니다.

배치 파일은 소스 파일의 이름을 나타내는 매개변수 %1을 가집니다. 배치 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldmudf.bat -- Windows 32-bit operating systems
rem Builds a Microsoft Visual C++ user-defined function (UDF).
rem Usage: bldmudf udf_prog_name

if "%1" == "" goto error
rem  Compile the program.
if exist "%1.cxx" goto cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.c
goto link_step
:cpp
cl -Z7 -Od -c -W2 -D_X86_=1 -DWIN32 %1.cxx

:link_step
rem  Link the program.
link -debug:full -debugtype:cv -dll -out:%1.dll %1.obj db2api.
lib db2apie.lib -def:%1.def
rem Copy the UDF DLL to the 'function' directory
```

```
copy %1.dll "%DB2PATH%\function"
goto exit
:error
echo Usage: bldmudf prog_name
:exit
@echo on
```

bldmudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cl Microsoft Visual C++ 컴파일러.

-Z7 생성된 C7 스타일 CodeView 정보.

-Od 최적화를 사용하지 않습니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

-W2 출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지.

-D_X86_=1
Intel 기반 컴퓨터에서 수행하기 위해 Windows 32비트 운영 체제에 필요한 컴파일러 옵션.

-DWIN32
Windows 32비트 운영 체제에 필요한 컴파일러 옵션.

bldmudf에 대한 컴파일 및 링크 옵션
링크 옵션: **link** 편집기를 링크하기 위해 링커를 사용합니다. **-debug:full** 디버깅 정보를 포함시킵니다. **-debugtype:cv** 디버거 유형을 나타냅니다. **-dll** DLL을 작성합니다. **-out:%1.dll** .DLL 파일을 빌드합니다. **%1.obj** 오브젝트 파일을 포함시킵니다. **db2api.lib** DB2 라이브러리로 링크합니다. **db2apie.lib** DB2 API 엔진 라이브러리로 링크합니다. **-def:%1.def** 모듈 정의 파일. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udfsrv.c에서 사용자 정의 함수(UDF) udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldmudf udfsrv
```

배치 파일은 샘플 프로그램과 동일한 디렉토리에 있는 모듈 정의 파일 udfsrv.def를 사용하여 사용자 정의 함수(UDF)를 빌드합니다. 배치 파일은 사용자 정의 함수(UDF) DLL udfsrv.dll을 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 이 프로그램의 Embedded SQL C 및 C++ 버전뿐 아니라 DB2 CLI도 제공됩니다.

배치 파일 bldmcli를 사용하여 %DB2PATH%\samples\cli에 있는 udfcli.c 소스 파일에서 DB2 CLI udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 409 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

배치 파일 bldmapp를 사용하여 %DB2PATH%\samples\c에 있는 udfcli.sqc 소스 파일에서 Embedded SQL C udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 416 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

배치 파일 bldmapp를 사용하여 %DB2PATH%\samples\cpp에 있는 udfcli.sqx 소스 파일에서 Embedded SQL C++ udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 416 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 실행하려면 다음을 입력하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv DLL에서 ScalarUDF 함수를 호출합니다.

IBM VisualAge C++ 버전 3.5

이 절에서는 다음 주제에 대해 설명합니다.

- DB2 CLI 응용프로그램
- DB2 CLI 저장 프로시듀어
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어
- 사용자 정의 함수(UDF)

주: VisualAge C++ 컴파일러는 %DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp 디렉토리에 제공되는 C 및 C++ 샘플 프로그램에 모두 사용됩니다. 이 두 디렉토리에는 동일한 배치 파일이 있습니다. 이 두 디렉토리에는 파일 확장자에 따라 C 또는 C++ 소스 파일을 승인하기 위한 명령이 들어 있습니다.

DB2 CLI 응용프로그램

%DB2PATH%\samples\cli에 있는 배치 파일 bldvcli.bat에 IBM VisualAge C++의 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다.

매개변수 %1은 소스 파일의 이름을 지정합니다.

이것은 유일한 필수 매개변수로 Embedded SQL이 없는 CLI 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하며 %4는 암호를 지정합니다.

프로그램에 .sqc 또는 .sqx 확장자로 표시되는 Embedded SQL이 들어 있는 경우, 프로그램을 사전 처리 컴파일하기 위해 embprep 배치 파일이 호출되고 각각 .c 또는 .cxx 확장자를 가진 프로그램 파일을 생성합니다.

```
@echo off
rem  bldvcli batch file - Windows 32-bit Operating Systems
rem  Builds a CLI program with IBM VisualAge C++.
rem  Usage: bldvcli prog_name
if exist "%1.sqc" call embprep %1 %2 %3 %4
if exist "%1.sqx" call embprep %1 %2 %3 %4

rem Compile the error-checking utility.
icc -c -Ti -W1 /I"%DB2PATH%\include" utilcli.c

rem  Compile the program.
if exist "%1.sqx" goto cpp
icc -c -Ti -W1 /I"%DB2PATH%\include" %1.c
goto link_step
:cpp
icc -c -Ti -W1 /I"%DB2PATH%\include" %1.cxx

rem  Link the program.
:link_step
ilink /MAP /DEBUG /ST:32000 /PM:VIO %1.obj utilcli.obj db2cli.lib
@echo on
```

bldvcli에 대한 컴파일 및 링크 옵션
컴파일 옵션: **icc** IBM VisualAge C++ 컴파일러. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다. **-Ti** 디버거 정보를 작성합니다. **-W1** 출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지.
링크 옵션: **ilink** 편집기를 링크하기 위해 자원 링커를 사용합니다. **/MAP** 맵 파일을 작성합니다. **/DEBUG** 디버깅 정보를 포함시킵니다. **/ST:32000** 최소한 32 000의 스택 크기를 지정합니다. **/PM:VIO** 프로그램이 창이나 전체 화면에서 수행할 수 있게 합니다. **%1.obj** 오브젝트 파일을 포함시킵니다. **utilcli.obj** 오류 체크를 위한 유틸리티 오브젝트 파일을 포함시킵니다. **db2cli.lib** DB2 CLI 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 tbinfo.c에서 샘플 프로그램 tbinfo를 빌드하려면 다음을 입력하십시오.

```
bldvcli tbinfo
```

그 결과 실행 파일 tbinfo가 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
tbinfo
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 dbusemx.sqc에서 Embedded SQL 응용프로그램 dbusemx를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldvcli dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldvcli dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldvcli dbusemx database userid password
   ```

그 결과 실행 파일 dbusemx가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   dbusemx
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   dbusemx database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   dbusemx database userid password
   ```

DB2 API가 있는 DB2 CLI 응용프로그램

DB2에는 둘 이상의 데이터베이스에서 CLI 함수를 사용하는 방법을 보여주기 위해 DB2 API를 사용하여 데이터베이스를 작성 및 삭제하는 CLI 샘플 프로그램이 들어 있습니다. 29 페이지의 표7에 있는 CLI 샘플 프로그램에 대한 설명은 DB2 API를 사용하는 샘플을 나타냅니다.

sqllib/samples/cli에 있는 스크립트 파일 bldvapi에 DB2 API가 있는 DB2 CLI 프로그램을 빌드하기 위한 명령이 들어 있습니다. 이 파일은 데이터베이스를 작성 및 삭제하기 위한 DB2 API가 들어 있는 utilapi 유틸리티 파일에서 컴파일되고 링크됩니다. 이 점이 바로 이 파일과 bldvcli 배치 파일 간의 유일한 차이점입니다. bldvapi 및 bldvcli에 공통되는 컴파일 및 링크 옵션에 대한 자세한 내용은 427 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

소스 파일 dbmconn.c에서 샘플 프로그램 dbmconn을 빌드하려면 다음을 입력하십시오.

```
bldvapi dbmconn
```

그 결과 실행 파일 dbmconn이 작성됩니다. 다음 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
dbmconn
```

DB2 CLI 저장 프로시듀어

%DB2PATH%\samples\cli에 있는 배치 파일 bldvclis.bat에 CLI 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 배치 파일은 저장 프로시듀어를 서버의 DLL로 빌드합니다.

매개변수 %1은 소스 파일의 이름을 지정합니다. 배치 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldvclis.bat file - Windows 32-bit Operating Systems
rem Builds a CLI stored procedure using the IBM VisualAge C++ compiler
rem Usage: bldvclis prog_name

if "%1" == "" goto error

rem  Compile the program.
if exist "%1.cxx" goto cpp
icc -c+ -Ti -Ge- -Gm+ -W1 %1.c  utilcli.c
goto link_step
:cpp
icc -c+ -Ti -Ge- -Gm+ -W1 %1.cxx  utilcli.c
:link_step
rem Import the library and create an export file.
rem The function name in the .def file must be decorated to be consistent
rem with the function name in the .map file. Typically, this is done by
rem prepending "_" and appending "@" and the number of bytes of arguments,
rem as in: "@16". In spserverva.def, for example, the IBM VisualAge C++
rem compiler requires "EXPORTS _outlanguage@16" and not "EXPORTS outlanguage".
```

```
ilib /GI %1va.def

rem Link the program and produce a DLL.
ilink /ST:64000 /PM:VIO /MAP /DLL %1.obj utilcli.obj %1va.exp db2cli.lib

rem Copy the stored procedure DLL to the 'function' directory
copy %1.dll "%DB2PATH%\function"
goto exit
:error
echo Usage: bldvclis prog_name
:exit
@echo on
```

bldvclis에 대한 컴파일 및 링크 옵션

컴파일 옵션:

icc IBM VisualAge C++ 컴파일러.

-c+ 컴파일만 수행하고 링크는 수행하지 않습니다. 이 배치 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.

-Ti 디버거 정보를 작성합니다.

-Ge- .DLL 파일을 빌드합니다. 정적으로 링크된 런타임 라이브러리 버전을 포함합니다.

-Gm+ 멀티타스킹 라이브러리로 링크합니다.

-W1 출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지.

링크 옵션:

ilink 편집기를 링크하기 위해 자원 링커를 사용합니다.

/ST:64000
최소한 64 000의 스택 크기를 지정합니다.

/PM:VIO
프로그램이 창이나 전체 화면에서 수행할 수 있게 합니다.

/MAP 맵 파일을 작성합니다.

/DLL .DLL 파일을 빌드합니다.

%1.obj 오브젝트 파일을 포함시킵니다.

%1.exp VisualAge의 내보낼 파일을 포함시킵니다.

db2cli.lib
DB2 CLI 라이브러리로 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 spserver.c에서 spserver 저장 프로시듀어를 빌드하려면 다음을 입력하십시오.

```
bldvclis spserver
```

배치 파일은 CLI 샘플 프로그램과 동일한 디렉토리에 있는 모듈 정의 파일 spserverva.def를 사용하여 저장 프로시듀어를 빌드합니다. 배치 파일은 저장 프로시듀어 DLL spserver.dll을 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 저장 프로시듀어 spserver를 빌드하면 이 저장 프로시듀어를 호출하는 CLI 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

배치 파일 bldvcli를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 427 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 공유 라이브러리 spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

DB2 API 및 Embedded SQL 응용프로그램

%DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 배치 파일 bldvapp.bat에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하며 %4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldvapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
@echo off
rem bldvapp.bat -- Windows 32-bit operating systems
rem Builds a VisualAge C++ application program
rem Usage: bldvapp prog_name [ db_name [  userid password ]]

if exist "%1.sqx" goto embedded
if exist "%1.sqc" goto embedded
goto non_embedded
:embedded
rem Precompile and bind the program.
```

```
call embprep %1 %2 %3 %4
rem  Compile the program.
if exist "%1.cxx" goto cpp_emb
icc -c -Ti -W1 %1.c utilemb.c
goto link_embedded
:cpp_emb
icc -c -Ti -W1 %1.cxx utilemb.cxx
rem  Link the program.
:link_embedded
ilink /MAP /DEBUG /ST:32000 /PM:VIO %1.obj utilemb.obj db2api.lib
goto exit
:non_embedded
rem  Compile the program.
if exist "%1.cxx" goto cpp_non
icc -c -Ti -W1 %1.c utilapi.c
goto link_non_embedded
:cpp_non
icc -c -Ti -W1 %1.cxx utilapi.cxx
rem  Link the program.
:link_non_embedded
ilink /MAP /DEBUG /ST:32000 /PM:VIO %1.obj utilapi.obj db2api.lib
:exit
@echo on
```

bldvapp에 대한 컴파일 및 링크 옵션

컴파일 옵션:

icc IBM VisualAge C++ 컴파일러.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

-Ti 디버거 정보를 작성합니다.

-W1 출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지.

bldvapp에 대한 컴파일 및 링크 옵션
링크 옵션: **ilink** 편집기를 링크하기 위해 자원 링커를 사용합니다. **/MAP** 맵 파일을 작성합니다. **/DEBUG** 디버깅 정보를 포함시킵니다. **/ST:32000** 최소한 32 000의 스택 크기를 지정합니다. **/PM:VIO** 프로그램이 창이나 전체 화면에서 수행할 수 있게 합니다. **%1.obj** 오브젝트 파일을 포함시킵니다. **utilemb.obj** Embedded SQL 프로그램의 경우, 오류 체크를 위한 Embedded SQL 유틸리티 오브젝트 파일을 포함합니다. **utilapi.obj** Embedded SQL 프로그램이 아닌 경우, 오류 체크를 위한 DB2 API 유틸리티 오브젝트 파일을 포함합니다. **db2api.lib** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

%DB2PATH%\samples\c에 있는 소스 파일 client.c 또는 %DB2PATH%\samples\cpp에 있는 소스 파일 client.cxx에서 DB2 API 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldvapp client
```

그 결과 실행 파일 client.exe가 작성됩니다. 다음과 같이 명령행에서 확장자 없이 실행 파일 이름을 입력하여 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

%DB2PATH%\samples\c에 있는 C 소스 파일 updat.sqc 또는 %DB2PATH%\samples\cpp에 있는 C++ 소스 파일 updat.sqx에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldvapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldvapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldvapp updat database userid password
   ```

그 결과 실행 파일 updat.exe가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

%DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 배치 파일 bldvsrv.bat에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 배치 파일은 저장 프로시듀어를 DLL에 컴파일하고 이를 서버에 저장합니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

배치 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldvsrv.bat -- Windows 32-bit operating systems
rem Builds a VisualAge C++ stored procedure
rem Usage: bldvsrv prog_name [ db_name ]

rem Precompile and bind the program.
call embprep %1 %2
rem  Compile the program.
if exist "%1.cxx" goto cpp
icc -c+ -Ti -Ge- -Gm+ -W1 %1.c
goto link_step
:cpp
icc -c+ -Ti -Ge- -Gm+ -W1 %1.cxx
:link_step
rem Import the library and create a definition file.
rem The function name in the .def file must be decorated to be consistent
rem with the function name in the .map file. Typically, this is done by
rem prepending "_" and appending "@" and the number of bytes of arguments,
rem for example, "@16". In spserverva.def, the IBM VisualAge C++ compiler requires
rem "EXPORTS _outlanguage@16" and not "EXPORTS outlanguage".
ilib /GI %1va.def

rem Link the program and produce a DLL.
ilink /ST:64000 /PM:VIO /MAP /DLL %1.obj %1va.exp db2api.lib

rem Copy the Stored Procedure DLL to the 'function' directory.
copy %1.dll "%DB2PATH%\function"
@echo on
```

bldvsrv에 대한 컴파일 및 링크 옵션	
컴파일 옵션:	
icc	IBM VisualAge C++ 컴파일러.
-c+	컴파일만 수행하고 링크는 수행하지 않습니다. 이 배치 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.
-Ti	디버거 정보를 작성합니다.
-Ge-	.DLL 파일을 빌드합니다. 정적으로 링크된 런타임 라이브러리 버전을 사용합니다.
-Gm+	멀티타스킹 라이브러리로 링크합니다.
-W1	출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지.
링크 옵션:	
ilink	편집기를 링크하기 위해 자원 링커를 사용합니다.
/ST:64000	최소한 64,000의 스택 크기를 지정합니다.
/PM:VIO	프로그램이 창이나 전체 화면에서 수행할 수 있게 합니다.
/MAP	맵 파일을 작성합니다.
/DLL	.DLL 파일을 빌드합니다.
%1.obj	오브젝트 파일을 포함시킵니다.
%1va.exp	VisualAge의 내보낼 파일.
db2api.lib	DB2 라이브러리로 링크합니다.
추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.	

C 소스 파일 spserver.sqc 또는 C++ 소스 파일 spserver.sqx에서 spserver 저장 프로시듀어 DLL을 빌드하려면 다음을 입력하십시오.

```
bldvsrv spserver
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldmsrv spserver database
```

배치 파일은 샘플 프로그램과 동일한 디렉토리에 있는 모듈 정의 파일 spserverva.def를 사용하여 저장 프로시듀어를 빌드합니다. 배치 파일은 저장 프로시듀어 DLL spserver.dll을 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

다음에는 서버에서 spcreate.db2 스크립트를 수행하여 저장 프로시듀어를 카탈로그화하십시오. 먼저, 다음의 데이터베이스에 연결하십시오.

```
db2 connect to sample
```

저장 프로시듀어가 이전에 카탈로그화되었으면 다음 명령을 사용하여 삭제할 수 있습니다.

```
db2 -td@ -vf spdrop.db2
```

그런 다음, 다음 명령을 사용하여 카탈로그화하십시오.

```
db2 -td@ -vf spcreate.db2
```

데이터베이스를 중지한 다음 재시작하여 새로운 공유 라이브러리가 인식될 수 있게 하십시오. 필요하면 DB2 인스턴스가 액세스할 수 있도록 공유 라이브러리에 대해 파일 모드를 설정하십시오.

일단 저장 프로시듀어 DLL spserver를 빌드하면 이 DLL을 호출하는 클라이언트 응용프로그램 spclient를 빌드할 수 있습니다.

스크립트 파일 bldvapp를 사용하여 spclient를 빌드할 수 있습니다. 자세한 내용은 433 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

spclient *database userid password*

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 데이터베이스 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 저장 프로시듀어 DLL spserver에 액세스하고 서버 데이터베이스에서 많은 수의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

사용자 정의 함수(UDF)

%DB2PATH%\samples\c 및 %DB2PATH%\samples\cpp에 있는 배치 파일 bldvudf에 UDF를 빌드하기 위한 명령이 들어 있습니다.

UDF는 Embedded SQL문을 포함할 수 없습니다. 따라서 UDF 프로그램을 빌드하기 위해 데이터베이스를 연결하거나 프로그램을 사전 처리 컴파일하고 바인드할 필요가 없습니다.

매개변수 %1은 소스 파일의 이름을 지정합니다. 배치 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldvudf.bat -- Windows 32-bit operating systems
rem Builds a VisualAge C++ user-defined function (UDF)
rem Usage: bldvudf program_name

if "%1" == "" goto error
rem  Compile the program.
if exist "%1.cxx" goto cpp
icc -Ti -c+ -Ge- -Gm+ -W1 %1.c
goto link_step
:cpp
icc -Ti -c+ -Ge- -Gm+ -W1 %1.cxx

:link_step
rem Generate an import library and export file using a definition file.
rem Function(s) in the .def file are prepended with an underscore, and
rem appended with the @ sign and number of bytes of arguments (in decimal).
rem Parameters of less than four bytes are rounded up to four bytes.
rem Structure size is rounded up to a multiple of four bytes.
rem For example, function fred prototyped as: "int fred(int, int, short);"
rem would appear as:  "_fred@12" in the .def file.
rem These decorated function names can also be found in %1.map
rem after running the following ilink command without %1va.exp.
ilib /gi %1va.def

rem Link the program to a dynamic link library
ilink /ST:64000 /PM:VIO /MAP /DLL %1.obj %1va.exp db2api.lib db2apie.lib
rem Copy the UDF DLL to the 'function' directory.
```

```
copy %1.dll "%DB2PATH%\function"
goto exit
:error
echo Usage: bldvudf prog_name
:exit
@echo on
```

bldvudf에 대한 컴파일 및 링크 옵션

컴파일 옵션:

icc IBM VisualAge C++ 컴파일러.

-Ti 디버거 정보를 작성합니다.

-c+ 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다.

-Ge- .DLL 파일을 빌드합니다. 정적으로 링크된 런타임 라이브러리 버전을 포함합니다.

-Gm+ 멀티타스킹 라이브러리로 링크합니다.

-W1 출력 경고, 오류 및 심각하고 복구할 수 없는 오류 메시지.

링크 옵션:

ilink 편집기를 링크하려면 자원 링커를 사용합니다.

/ST:64000
최소한 64000의 스택 크기를 지정합니다.

/PM:VIO
프로그램이 창이나 전체 화면에서 수행할 수 있게 합니다.

/MAP 맵 파일을 작성합니다.

/DLL .DLL 파일을 빌드합니다.

%1.obj 오브젝트 파일을 포함시킵니다.

%1va.exp
VisualAge의 내보낼 파일을 포함시킵니다.

db2api.lib
DB2 라이브러리로 링크합니다.

db2apie.lib
DB2 API 엔진 라이브러리로 링크합니다.

추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 udf.c에서 사용자 정의 함수(UDF) udfsrv를 빌드하려면 다음을 입력하십시오.

```
bldvudf udfsrv
```

배치 파일은 샘플 프로그램과 동일한 디렉토리에 있는 모듈 정의 파일 udfsrv.def를 사용하여 사용자 정의 함수(UDF)를 빌드합니다. 배치 파일은 사용자 정의 함수(UDF) DLL udfsrv.dll을 경로 %DB2PATH%\function에 있는 서버로 복사합니다.

일단 udfsrv를 빌드하면 이를 호출하는 클라이언트 응용프로그램 udfcli를 빌드할 수 있습니다. 이 프로그램의 Embedded SQL C 및 C++ 버전뿐 아니라 DB2 CLI도 제공됩니다.

배치 파일 bldvcli를 사용하여 %DB2PATH%\samples\cli에 있는 udfcli.c 소스 파일에서 DB2 CLI udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 427 페이지의 『DB2 CLI 응용프로그램』을 참조하십시오.

배치 파일 bldvapp를 사용하여 %DB2PATH%\samples\c에 있는 udfcli.sqc 소스 파일에서 Embedded SQL C udfcli 프로그램을 빌드할 수 있습니다. 자세한 내용은 433 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

배치 파일 bldvapp를 사용하여 %DB2PATH%\samples\cpp에 있는 udfcli.sqx 소스 파일에서 Embedded SQL C++ udfcli 프로그램을 빌드할 수있습니다. 자세한 내용은 433 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

UDF를 실행하려면 다음을 입력하십시오.

```
udfcli
```

호출 응용프로그램은 udfsrv DLL에서 ScalarUDF 함수를 호출합니다.

IBM VisualAge C++ 버전 4.0

VisualAge C++ 버전 4 컴파일러에 대한 응용프로그램 빌드 정보는 AIX, OS/2 및 Windows 32비트 운영 체제에서 공통됩니다. 이 정보에 대해서는 172 페이지의 『VisualAge C++ 버전 4.0』을 참조하십시오.

IBM VisualAge COBOL

이 절에서는 다음 주제에 대해 설명합니다.

- 컴파일러 사용
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어

컴파일러 사용

Embedded SQL 및 DB2 API 호출을 포함하는 응용프로그램을 개발하며 IBM VisualAge COBOL 컴파일러를 사용 중인 경우, 다음 사항에 유의하십시오.

- 명령행 처리기 명령 `db2 prep`를 사용하여 응용프로그램을 사전 처리 컴파일할 때 `target ibmcob` 옵션(기본값)을 사용하십시오.
- 소스 파일에서 탭 문자를 사용하지 마십시오.
- 소스 파일에서 `PROCESS` 및 `CBL` 키워드를 사용하여 컴파일 옵션을 설정할 수 있습니다. 키워드를 8-72 컬럼에만 배치하십시오.
- 응용프로그램에 Embedded SQL만 들어 있고 DB2 API 호출은 들어 있지 않을 경우, `pgmname(mixed)` 컴파일 옵션을 사용할 필요가 없습니다. DB2 API 호출을 사용할 경우에는 `pgmname(mixed)` 컴파일 옵션을 사용해야 합니다.
- IBM VisualAge COBOL 컴파일러의 "System/390 호스트 데이터 유형 지원" 기능을 사용 중인 경우, 응용프로그램에 대한 DB2 포함 파일은 다음 디렉토리에 있습니다.

  ```
  %DB2PATH%\include\cobol_i
  ```

  제공된 배치 파일을 사용하여 DB2 샘플 프로그램을 빌드 중인 경우, 배치 파일에 지정된 포함 파일 경로를 `cobol_a` 디렉토리가 아닌 `cobol_i` 디렉토리를 가리키도록 변경해야 합니다.

IBM VisualAge COBOL 컴파일러의 "System/390 호스트 데이터 유형 지원" 기능을 사용하지 않거나 해당 컴파일러의 이전 버전을 사용 중인 경우, 응용프로그램에 대한 DB2 포함 파일은 다음 디렉토리에 있습니다.

```
%DB2PATH%\include\cobol_a
```

다음과 같이 COPY 파일 이름이 .cbl 확장자를 포함하도록 지정하십시오.

```
COPY "sql.cbl".
```

DB2 API 및 Embedded SQL 응용프로그램

%DB2PATH%\samples\cobol에 있는 배치 파일 bldapp.bat에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하며 %4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 전달합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
@echo off
rem bldapp.bat  -- Windows 32-bit operating systems
rem Builds a VisualAge COBOL application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]
rem If an embedded SQL program, precompile and bind it.
if not exist "%1.sqb" goto compile_step
call embprep %1 %2 %3 %4
:compile_step
rem Compile the error-checking utility.
cob2 -qpgmname(mixed) -c -qlib -I"%DB2PATH%\include\cobol_a" checkerr.cbl

rem  Compile the program.
cob2 -qpgmname(mixed) -c -qlib -I"%DB2PATH%\include\cobol_a" %1.cbl

rem  Link the program.
cob2 %1.obj checkerr.obj db2api.lib
@echo on
```

bldapp에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cob2** IBM VisualAge COBOL 컴파일러. **-qpgmname(mixed)** 컴파일러에 대소문자가 혼합된 이름으로 라이브러리 시작점에 대한 CALL을 허용하도록 지시합니다. **-c** 컴파일만 수행하고 링크는 수행하지 않습니다. 이 책은 컴파일과 링크가 서로 다른 단계로 간주합니다. **-qlib** COPY문을 처리할 컴파일러를 지시합니다. **-I***path* DB2 포함 파일의 위치를 지정합니다. 예를 들면, -I"%DB2PATH%\include\cobol_a"입니다. **checkerr.cbl** 오류 체크 유틸리티를 컴파일합니다.
링크 옵션: **cob2** 편집기를 링크하기 위해 컴파일러를 사용합니다. **checkerr.obj** 오류 체크 유틸리티 오브젝트 파일을 포함시킵니다. **db2api.lib** DB2 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.cbl에서 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client.exe가 작성됩니다. 다음과 같이 확장자 없이 실행 파일 이름을 입력하여 sample 데이터베이스에 대해 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqb에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

%DB2PATH%\samples\cobol에 있는 배치 파일 bldsrv.bat에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 배치 파일은 저장 프로시듀어를 서버의 DLL에 컴파일합니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 필요 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

배치 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldsrv.bat  -- Windows 32-bit operating systems
rem Builds a VisualAge COBOL stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
call embprep %1 %2
rem  Compile the stored procedure.
cob2 -qpgmname(mixed) -c -qlib -I"%DB2PATH%\include\cobol_a" %1.cbl

rem  Link the stored procedure and create a shared library.
ilib /nol /gi:%1 %1.obj
ilink /free /nol /dll db2api.lib %1.exp %1.obj iwzrwin3.obj

rem Copy stored procedure to the %DB2PATH%\function directory.
copy %1.dll "%DB2PATH%\function"
@echo on
```

bldsrv에 대한 컴파일 및 링크 옵션

컴파일 옵션:

cob2 IBM VisualAge COBOL 컴파일러.

-qpgmname(mixed)
컴파일러에 대소문자가 혼합된 이름으로 라이브러리 시작점에 대한 CALL을 허용하도록 지시합니다.

-c 컴파일만 수행하고 링크는 수행하지 않습니다. 이 배치 파일은 서로 다른 컴파일 및 링크 단계를 가지고 있습니다.

-qlib COPY문을 처리할 컴파일러를 지시합니다.

-I*path* DB2 포함 파일의 위치를 지정합니다. 예를 들면, -I"%DB2PATH%\include\cobol_a"와 같습니다.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션: **ilink** IBM VisualAge COBOL 링커를 사용합니다. **/free** Free format. **/nol** 로고가 없습니다. **/dll** 소스 프로그램 이름으로 DLL을 작성합니다. **db2api.lib** DB2 라이브러리로 링크합니다. **%1.exp** 내보낼 파일을 포함시킵니다. **%1.obj** 프로그램 오브젝트 파일을 포함합니다. **iwzrwin3.obj** IBM VisualAge COBOL이 제공하는 오브젝트 파일을 포함시킵니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

샘플 데이터베이스에 연결하여 소스 파일 outsrv.sqb에서 샘플 프로그램 outsrv를 빌드하려면 다음을 입력하십시오.

```
bldsrv outsrv
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 포함시키십시오.

```
bldsrv outsrv database
```

스크립트 파일은 저장 프로시듀어를 경로 sqllib/function에 있는 서버로 복사합니다.

필요하면 DB2 인스턴스가 실행할 수 있도록 저장 프로시듀어에 대한 파일 모드를 설정하십시오.

일단 저장 프로시듀어 outsrv를 빌드하면 저장 프로시듀어를 호출하는 클라이언트 응용프로그램 outcli를 빌드할 수 있습니다. 배치 파일 bldapp를 사용하여 outcli를 빌드할 수 있습니다. 자세한 내용은 444 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

```
outcli database userid password
```

여기서

database
연결하려는 데이터베이스의 이름입니다. 이름은 sample, 원격 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password
유효한 암호입니다.

클라이언트 응용프로그램은 저장 프로시듀어 라이브러리 outsrv에 액세스하고 서버 데이터베이스에서 동일한 이름의 저장 프로시듀어 함수를 실행한 다음 출력을 클라이언트 응용프로그램으로 리턴합니다.

Micro Focus COBOL

이 절에서는 다음 주제에 대해 설명합니다.

- 컴파일러 사용
- DB2 API 및 Embedded SQL 응용프로그램
- Embedded SQL 저장 프로시듀어

컴파일러 사용

Embedded SQL 및 DB2 API 호출을 포함하는 응용프로그램을 개발하며 Micro Focus 컴파일러를 사용 중인 경우 다음 사항에 유의하십시오.

- 명령행 처리기 명령 db2 prep를 사용하여 응용프로그램을 사전 처리 컴파일할 때 기본값인 target mfcob 옵션을 사용하십시오.
- LIB 환경 변수가 다음과 같이 %DB2PATH%\lib를 가리키는지 확인하십시오.

  ```
  set LIB="%DB2PATH%\lib;%LIB%"
  ```

- Micro Focus COBOL용 DB2 COPY 파일은 %DB2PATH%\include\cobol_mf에 있습니다. 다음 디렉토리를 포함시키도록 COBCPY 환경 변수를 설정하십시오.

  ```
  set COBCPY="%DB2PATH%\include\cobol_mf;%COBCPY%"
  ```

모든 DB2 API에 대한 호출은 호출 규칙 74를 사용하여 이루어져야 합니다. DB2 COBOL 사전 처리 컴파일러는 자동으로 CALL-CONVENTION 절을 SPECIAL-NAMES 단락에 추가합니다. SPECIAL-NAMES 단락이 없을 경우, DB2 COBOL 사전 처리 컴파일러가 다음과 같이 이를 작성합니다.

```
Identification Division
Program-ID. "static".
special-names.
    call-convention 74 is DB2API.
```

또한 사전 처리 컴파일러는 DB2 API가 호출될 때마다 ″call″ 키워드 다음에 호출 규칙을 식별하는 데 사용되는 기호 DB2API를 배치합니다. 이것은, 예를 들어, 사전 처리 컴파일러가 Embedded SQL문으로부터 DB2 API 런타임 호출을 생성할 때마다 발생합니다.

사전 처리 컴파일되지 않은 응용프로그램에서 DB2 API에 대한 호출이 이루어지면 사용자는 수동으로 위에 제공된 것과 비슷하게 응용프로그램에 SPECIAL-NAMES 단락을 작성해야 합니다. DB2 API를 직접 호출하는 경우에는 수동으로 DB2API 기호를 ″call″ 키워드 다음에 추가해야 합니다.

DB2 API 및 Embedded SQL 응용프로그램

%DB2PATH%\samples\cobol_mf에 있는 배치 파일 bldapp에 DB2 응용프로그램을 빌드하기 위한 명령이 들어 있습니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 이것은 Embedded SQL이 없는 프로그램에 대해 유일한 필수 매개변수입니다. Embedded SQL 프로그램을 빌드할 때는 데이터베이스에 연결해야 하므로 세 개의 선택적 매개변수가 함께 제공되어야 합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정하고 세 번째 매개변수 %3은 데이터베이스에 대한 사용자 ID를 지정하며 %4는 암호를 지정합니다.

Embedded SQL 프로그램의 경우, bldapp는 사전 처리 컴파일 및 바인드 파일 embprep에 매개변수를 제공합니다. 데이터베이스 이름이 제공되지 않으면 기본 sample 데이터베이스가 사용됩니다. 사용자 ID와 암호 매개변수는 프로그램이 빌드된 인스턴스가 데이터베이스가 있는 인스턴스와 다른 경우에만 필요합니다.

```
@echo off
rem bldapp.bat  -- Windows 32-bit operating systems
rem Builds a Micro Focus Cobol application program
rem Usage: bldapp <prog_name> [ <db_name> [ <userid> <password> ]]

rem If an embedded SQL program, precompile and bind it.
if not exist "%1.sqb" goto compile_step
call embprep %1 %2 %3 %4
:compile_step
rem Compile the error-checking utility.
cobol checkerr.cbl;

rem  Compile the program.
cobol %1.cbl;

rem  Link the program.
cbllink -l %1.obj checkerr.obj db2api.lib
@echo on
```

bldapp에 대한 컴파일 및 링크 옵션
컴파일 옵션: **cobol** Micro Focus COBOL 컴파일러.
링크 옵션: **cbllink** 편집기를 링크하기 위해 링커를 사용합니다. **-l** lcobol 라이브러리로 링크합니다. **checkerr.obj** 오류 체크 유틸리티 오브젝트 파일로 링크합니다. **db2api.lib** DB2 API 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

소스 파일 client.cbl에서 비 Embedded SQL 샘플 프로그램 client를 빌드하려면 다음을 입력하십시오.

```
bldapp client
```

그 결과 실행 파일 client.exe가 작성됩니다. 다음과 같이 확장자 없이 실행 파일 이름을 입력하여 sample 데이터베이스에 대해 실행 파일을 수행할 수 있습니다.

```
client
```

Embedded SQL 응용프로그램 빌드 및 수행

소스 파일 updat.sqb에서 Embedded SQL 응용프로그램 updat를 빌드하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 같은 데이터베이스에 연결하고 있으면 다음을 입력하십시오.

   ```
   bldapp updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

   ```
   bldapp updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 연결하고 있으면 데이터베이스 인스턴스의 사용자 ID와 암호도 입력하십시오.

   ```
   bldapp updat database userid password
   ```

그 결과 실행 파일 updat.exe가 작성됩니다.

이 Embedded SQL 응용프로그램을 수행하는 방법에는 세 가지가 있습니다.

1. 같은 인스턴스에서 sample 데이터베이스에 액세스하고 있으면 확장자 없이 실행 파일 이름만 입력하십시오.

   ```
   updat
   ```

2. 같은 인스턴스에서 다른 데이터베이스에 액세스하고 있으면 실행 파일 이름과 데이터베이스 이름을 입력하십시오.

   ```
   updat database
   ```

3. 다른 인스턴스에서 데이터베이스에 액세스하고 있으면 실행 파일 이름, 데이터베이스 이름 및 데이터베이스 인스턴스의 사용자 ID와 암호를 입력하십시오.

   ```
   updat database userid password
   ```

Embedded SQL 저장 프로시듀어

%DB2PATH%\samples\cobol_mf에 있는 명령 파일 bldsrv에 Embedded SQL 저장 프로시듀어를 빌드하기 위한 명령이 들어 있습니다. 배치 파일은 저장 프로시듀어를 서버의 DLL에 컴파일합니다.

첫 번째 매개변수 %1은 소스 파일의 이름을 지정합니다. 두 번째 매개변수 %2는 연결하려는 데이터베이스의 이름을 지정합니다. 이 저장 프로시듀어는 데이터베이스가 상주하는 동일한 인스턴스에서 빌드해야 되므로 사용자 ID와 암호에 대한 매개변수가 필요 없습니다.

첫 번째 매개변수만 소스 파일 이름이 필요합니다. 데이터베이스 이름은 선택적입니다. 데이터베이스 이름이 제공되지 않으면 프로그램은 기본 sample 데이터베이스를 사용합니다.

배치 파일은 DLL 이름에 대한 소스 파일 이름 %1을 사용합니다.

```
@echo off
rem bldsrv.bat  -- Windows 32-bit operating systems
rem Builds a Micro Focus Cobol stored procedure
rem Usage: bldsrv <prog_name> [ <db_name> ]

rem Precompile and bind the program.
call embprep %1 %2
rem  Compile the stored procedure.
cobol %1.cbl /case;

rem  Link the stored procedure and create a shared library.
cbllink /d %1.obj db2api.lib

rem Copy the stored procedure to the %DB2PATH%\function directory.
copy %1.dll "%DB2PATH%\function"
@echo on
```

bldsrv에 대한 컴파일 및 링크 옵션
컴파일 옵션:
cobol Micro Focus COBOL 컴파일러.
/case 외부 기호가 대문자로 변환되지 못하게 합니다.

bldsrv에 대한 컴파일 및 링크 옵션
링크 옵션: **cbllink** 편집기에 링크하기 위해 Micro Focus COBOL 링커를 사용합니다. **/d** .dll 파일을 작성합니다. **db2api.lib** DB2 API 라이브러리로 링크합니다. 추가 컴파일러 옵션에 대한 자세한 내용은 해당 컴파일러의 문서를 참조하십시오.

샘플 프로그램에 연결하고 있는 경우, 소스 파일 outsrv.sqb에서 샘플 프로그램 outsrv를 빌드하려면 다음을 입력하십시오.

```
bldsrv outsrv
```

다른 데이터베이스에 연결하고 있으면 데이터베이스 이름도 입력하십시오.

```
bldsrv outsrv database
```

스크립트 파일은 DLL을 경로 sqllib/function에 있는 서버로 복사합니다.

필요하면 클라이언트 응용프로그램이 액세스할 수 있도록 DLL에 대한 파일 모드를 설정하십시오.

일단 DLL outsrv를 빌드하면 DLL을 호출하는 클라이언트 응용프로그램 outcli를 빌드할 수 있습니다. 배치 파일 bldapp를 사용하여 outcli를 빌드할 수 있습니다. 자세한 내용은 450 페이지의 『DB2 API 및 Embedded SQL 응용프로그램』을 참조하십시오.

저장 프로시듀어를 호출하려면 다음을 입력하여 샘플 클라이언트 응용프로그램을 수행하십시오.

```
outcli database userid password
```

여기서

database

연결하려는 데이터베이스의 이름입니다. 이름은 sample, 별명 또는 다른 이름이 될 수 있습니다.

userid 유효한 사용자 ID입니다.

password

유효한 암호입니다.

클라이언트 응용프로그램은 DLL, outsrv에 액세스하고 서버 데이터베이스에서 동일한 이름의 저장 프로시듀어 함수를 실행합니다. 출력은 클라이언트 응용프로그램으로 리턴됩니다.

오브젝트 REXX

오브젝트 REXX는 REXX 언어의 객체 지향 버전입니다. 객체 지향 확장자가 classic REXX에 추가되었으나 REXX의 기존 함수와 명령은 변경되지 않았습니다. 오브젝트 REXX 인터프리터는 다음의 추가 지원을 포함하는 해당 선임자 버전에 대한 확장 버전입니다.

- 클래스, 오브젝트 및 메소드
- 메시지 기능 및 다형성(polymorphism)
- 단일 및 복수 계승

오브젝트 REXX는 classic REXX와 완전한 호환성을 갖습니다. 이 절에서 REXX를 언급할 때마다 이는 오브젝트 REXX를 포함하는 모든 버전의 REXX를 지칭하는 것입니다.

REXX 프로그램을 사전 처리 컴파일하거나 바인드하지 마십시오.

Windows NT에서 REXX 프로그램은 주석과 함께 시작할 필요가 없습니다. 단, 이식성을 이유로 각 REXX 프로그램을 처음 행의 첫 번째 컬럼에서 시작되는 주석과 함께 시작하는 것이 좋습니다. 이는 프로그램이 다른 플랫폼에서의 배치 명령과 구별될 수 있게 합니다.

```
/* Any comment will do. */
```

REXX 샘플 프로그램은 %DB2PATH%\samples\rexx 디렉토리에 있습니다. 샘플 REXX 프로그램 updat를 수행하려면 다음을 수행하십시오.

1. 서버에서 데이터베이스 관리자가 이미 수행되고 있지 않을 경우, 다음을 입력하여 이를 시작하십시오.

   ```
   db2start
   ```

2. 다음을 입력하십시오.

   ```
   rexx updat.cmd
   ```

REXX 및 DB2에 대한 자세한 내용은 *응용프로그램 개발 안내서*의 "REXX 프로그래밍"을 참조하십시오.

부록A. 데이터베이스 관리자 인스턴스 정보

DB2는 동일한 버전에서 여러 데이터베이스 관리자 인스턴스를 지원합니다. 데이터베이스 관리자 인스턴스는 자체 구성 파일과, 디렉토리 및 데이터베이스를 가집니다.

각 데이터베이스 관리자 인스턴스는 여러 데이터베이스를 관리할 수 있습니다. 그러나 지정된 데이터베이스는 한 개의 인스턴스에만 해당됩니다. 그림1에서 이러한 관계를 보여줍니다.

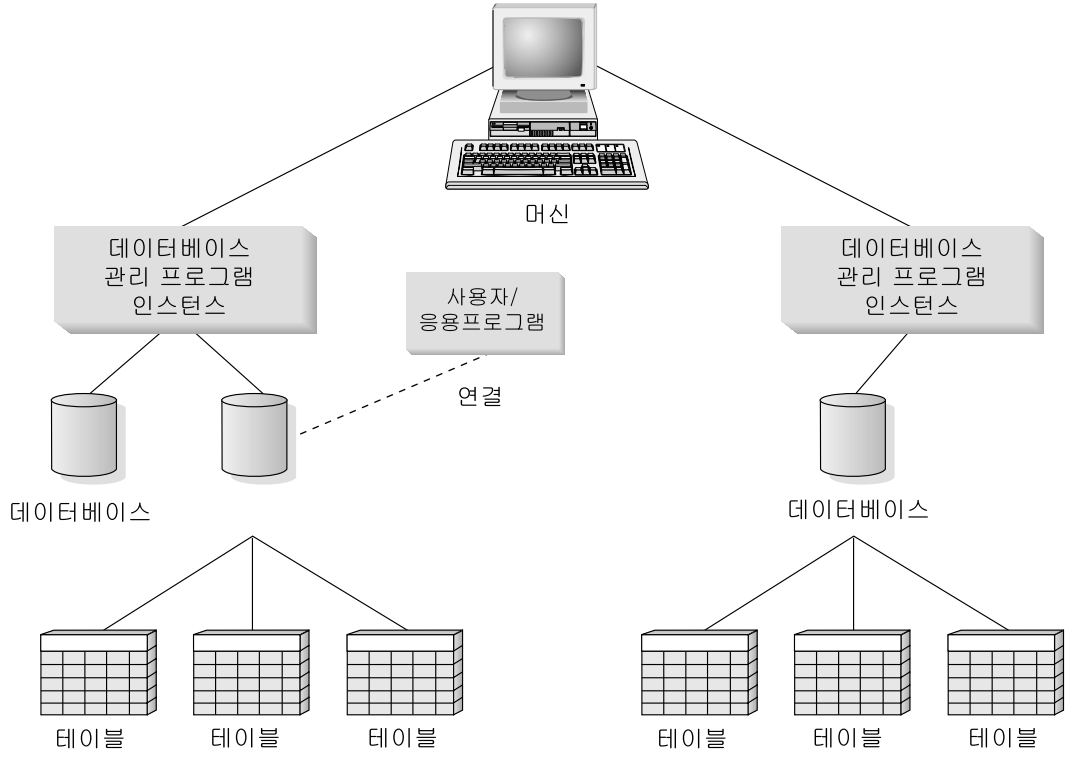

그림 1. 데이터베이스 관리자 인스턴스

데이터베이스 관리자 인스턴스는 사용자에게 동일한 머신에서 여러 데이터베이스 환경을 가질 수 있는 융통성을 제공합니다. 예를 들어, 하나의 데이터베이스 관리자 인스턴스를 개발용으로, 그리고 또 다른 데이터베이스 관리자 인스턴스를 생산용으로 가질 수 있습니다.

UNIX 서버를 사용한 경우, 다른 데이터베이스 관리자 인스턴스에 다른 DB2 버전을 가질 수 있습니다. 예를 들어, DB2 Universal Database 버전 6.1을 수행하는 데이터베이스 관리자 인스턴스와 DB2 Universal Database 버전 7.1을 수행하는 또 다른 데이터베이스 관리자 인스턴스가 있을 수 있습니다. 그러나 버전 레벨에서는 하나의 릴리스 및 수정 레벨만 지원됩니다. 예를 들어, DB2 버전 5.0과 DB2 버전 5.2는 UNIX 서버에서 공존할 수 없습니다.

OS/2, Windows NT 및 Windows 2000 서버를 사용할 경우, 각 데이터베이스 관리자 인스턴스에 대해 동일한 DB2 버전, 릴리스 및 수정 레벨을 가져야 합니다. DB2 Universal Database 버전 6.1을 수행하는 하나의 데이터베이스 관리자 인스턴스와 DB2 Universal Database 버전 7.1을 수행하는 또 다른 데이터베이스 관리자 인스턴스를 가질 수 없습니다.

사용 중인 각 인스턴스에 대해 다음 정보를 알아 두어야 합니다.

인스턴스 이름

UNIX 플랫폼의 경우, 이는 데이터베이스 관리자 인스턴스를 작성할 때 지정하는 유효한 사용자 이름입니다.

OS/2, Windows NT 및 Windows 2000의 경우, 이것은 최대 8자의 영숫자 문자열입니다. 설치하는 동안 "DB2"라는 인스턴스가 작성됩니다.

인스턴스 디렉토리

인스턴스가 위치한 홈 디렉토리입니다.

UNIX 플랫폼의 경우, 인스턴스 디렉토리는 `$HOME/sqllib`이며, 여기서 `$HOME`은 인스턴스 소유자의 홈 디렉토리입니다.

OS/2, Windows NT 및 Windows 2000의 경우, 인스턴스 디렉토리는 `%DB2PATH%\`*instance_name*입니다. `%DB2PATH%` 변수가 DB2가 설치된 위치를 결정합니다. DB2가 설치되어 있는 드라이브에 따라 `%DB2PATH%`는 *drive*`:\sqllib`를 가리킵니다.

OS/2, Windows NT 및 Windows 2000의 인스턴스 경로는 다음의 두 가지 기준에 따라 작성됩니다.

`%DB2PATH%\%DB2INSTANCE%`(예: `C:\SQLLIB\DB2`)

또는 DB2INSTPROF가 정의되어 있는 경우,

%DB2INSTPROF%\%DB2INSTANCE%(예: C:\PROFILES\DB2)

DB2INSTPROF 환경 변수는 클라이언트 머신이 읽기 액세스 권한만 있는 네트워크 드라이브에서 DB2를 수행하도록 지원하기 위해 OS/2, Windows NT 및 Windows 2000에서 사용됩니다. 이 경우, DB2는 *drive*:\sqllib를 가리키도록 설정되고, DB2INSTPROF는 카탈로그 및 구성과 같은 모든 인스턴스 특정 정보를 포함하는 지역 경로(예: C:\PROFILES)를 가리키도록 설정됩니다. 이는 DB2가 이들 파일에 대해 갱신 액세스 권한이 있어야 하기 때문입니다.

데이터베이스 관리자 인스턴스 작성 및 관리에 대한 자세한 내용은 사용자 플랫폼에 대한 *빠른 시작*을 참조하십시오.

부록B. 응용프로그램 이주

DB2의 버전 2 이상을 설치한 후 DB2 Universal Database 버전 7로 업그레이드할 때 DB2 Client Application Enabler 또는 DB2 Software Developer's Kit, 데이터베이스 및 노드 디렉토리는 자동으로 이주됩니다. DB2 버전 1에서 이주하려면 먼저 DB2 Universal Database 버전 5로 이주해야 합니다. 그런 다음 버전 5에서 버전 7로 이주할 수 있습니다. 기존 데이터베이스를 이주하려면 *관리 안내서*에 설명된 도구를 사용하십시오.

주:

1. **DB2 버전 7.1 및 7.2**. 이들 버전은 UNIX 시스템에서 동일한 디렉토리에 설치됩니다. 예를 들어, AIX에서 DB2 버전 7.1이나 DB2 버전 7.2에 사용되는 설치 경로는 `/usr/lpp/db2_07_01/lib`입니다. 이 책에서 DB2 버전 7을 언급할 경우, 이는 DB2 버전 7.1과 DB2 버전 7.2를 모두 지칭합니다.
2. **HP-UX**. HP-UX 버전 10 이하에서 HP-UX 버전 11로 DB2를 이주할 경우, DB2 프로그램은 HP-UX 버전 11에 있는 DB2에서 다시 사전 처리 컴파일된 다음(Embedded SQL을 포함하는 경우) 다시 컴파일되어야 합니다. 모든 DB2 응용프로그램, 저장 프로시듀어, 사용자 정의 함수(UDF) 및 User Exit 프로그램을 포함합니다. 또한 HP-UX 버전 11에서 컴파일된 DB2 프로그램은 HP-UX 버전 10 이하에서는 수행되지 않을 수 있습니다. HP-UX 버전 10에서 컴파일되고 수행되는 DB2 프로그램은 HP-UX 버전 11 서버에 원격으로 연결할 수 있습니다.
3. **Linux**. DB2는 Lunux 버전 5.2(Beta)용 DB2 Universal Database에서의 이주를 지원하지 않습니다.
4. **Micro Focus COBOL**. DB2 버전 2.1.1 이하에서 사전 처리 컴파일되고 Micro Focus COBOL에서 컴파일된 기존 응용프로그램은 현재 DB2 버전에서 다시 사전 처리 컴파일된 다음 Micro Focus COBOL에서 다시 컴파일되어야 합니다. IBM 사전 처리 컴파일러의 초기 버전에서 빌드된 이들 응용프로그램은 다시 사전 처리 컴파일하지 않으면 비정상 종료가 발생할 경우에 데이터베이스가 손상될 가능성이 있습니다.

5. 이주에 대한 *관리 안내서*의 '부록 D. 릴리스 간 비호환성'과 사용하는 플랫폼에 대한 *빠른 시작*의 '설치 계획' 및 'DB2 사후 설치 이주 타스크' 부분을 참조하십시오.

주: "질문" 및 "조건" 절에서와 마찬가지로 다음은 UNIX 플랫폼에만 적용됩니다.

DB2 버전 1, DB2 버전 2, DB2 버전 5 또는 DB2 버전 6.1의 응용프로그램을 가지고 있으며 동일한 머신에서 DB2 버전 7 인스턴스뿐 아니라 이전 버전의 데이터베이스 인스턴스에서 이들 응용프로그램을 수행하려면 환경을 일부 변경할 필요가 있습니다. 변경사항을 결정하려면 다음 질문에 응답한 다음 "조건" 절을 검토하여 사용자의 상황에 적용되는 조건이 있는지 확인하십시오.

제기된 문제를 설명하기 위해 AIX 시스템이 사용됩니다. 다른 UNIX 플랫폼에도 동일한 개념이 적용되지만 환경 변수나 특정한 명령과 같이 세부사항은 다를 수 있습니다. 운영 체제에 대한 이러한 세부사항에 대해 잘 알고 있지 못한 경우, *관리 안내서* 또는 *UNIX용 DB2 빠른 시작* 책의 "설치 계획" 장에서 "이전 버전의 DB2에서 이주" 절을 참조하십시오.

질문

질문 1: 이전 DB2 버전의 응용프로그램이 어떤 방법으로 DB2 클라이언트 런타임 라이브러리(예: AIX의 libdb2.a)에 링크되었습니까?

실행 파일에 대한 포함된 공유 라이브러리 검색 경로를 결정하려면 다음 시스템 명령 중 하나를 사용하십시오.

AIX /usr/bin/dump -H *executable_filename*

HP-UX
/usr/bin/chatr *executable_filename*

Linux /usr/bin/objdump -p *executable_filename*

PTX /usr/bin/dump -Lv *executable_filename*

Silicon Graphics IRIX
/bin/elfdump -Lv *executable_filename*

Solaris

/usr/bin/dump -Lv *executable_filename*

여기서 *executable_filename*은 응용프로그램에 대한 실행 파일 이름입니다.

다음은 AIX용 DB2 버전 1 응용프로그램에서 나열되는 샘플 덤프 파일입니다.

```
dbcat:
  ***Loader Section***
                      Loader Header Information
VERSION#         #SYMtableENT     #RELOCent        LENidSTR
0x00000001       0x00000012       0x00000029       0x00000064
#IMPfilID        OFFidSTR         LENstrTBL        OFFstrTBL
0x00000004       0x000003bc       0x00000077       0x00000420
  ***Import File Strings***
INDEX  PATH                              BASE                MEMBER
0      /usr/lpp/db2_01_01_0000/lib:/usr/lpp/xlC/lib:/usr/lib:/lib
1                                        libc.a              shr.o
2                                        libC.a              shr.o
3                                        libdb2.a            shr.o
```

0행은 실행 파일이 링크된 공유 라이브러리를 찾기 위해 검색하는 디렉토리 경로를 나타냅니다. 1, 2 및 3행은 응용프로그램이 링크된 공유 라이브러리를 나타냅니다.

응용프로그램이 빌드된 방법에 따라 `/usr/lpp/db2_01_01_0000/lib`, `INSTHOME/sqllib/lib`(여기서 `INSTHOME`은 데이터베이스 인스턴스 소유자의 홈 디렉토리임) 경로 또는 `/usr/lib:/lib` 조합 경로가 나타날 수 있습니다.

질문 2: DB2 런타임 라이브러리가 시스템에서 어떻게 구성되었습니까?

DB2 버전 1, 2, 5, 6.1 또는 7이 설치되어 있을 경우, 시스템 기본 공유 라이브러리 경로 `/usr/lib`에서 DB2 클라이언트 런타임 라이브러리가 들어 있는 DB2 설치 경로로 기호 링크를 작성하는 선택적 단계가 있습니다.

다른 DB2 버전에 대한 설치 경로는 다음과 같습니다.

버전 1 `/usr/lpp/db2_01_01_0000/lib`

버전 2 `/usr/lpp/db2_02_01/lib`

버전 5 /usr/lpp/db2_05_00/lib

버전 6.1
/usr/lpp/db2_06_01/lib

버전 7 /usr/lpp/db2_07_01/lib

모든 경우에 런타임 공유 라이브러리의 이름은 libdb2.a입니다.

이들 라이브러리 중 한 버전만 기본값이 될 수 있습니다. DB2는 이 기본값을 제공하여 사용자가 응용프로그램을 빌드할 때 특정 버전의 DB2에 종속되지 않도록 합니다.

질문 3: 사용자 환경에서 다른 검색 경로를 지정하고 있습니까?

AIX의 경우 LIBPATH 환경 변수, HP-UX의 경우 SHLIB_PATH, 그리고 Linux, PTX, Silicon Graphics IRIX 및 Solaris의 경우 LD_LIBRARY_PATH를 사용하여 사용자의 응용프로그램에 코딩된 공유 라이브러리 검색 경로를 겹쳐쓸 수 있습니다.

주: Silicon Graphics IRIX의 n32 오브젝트 유형 응용프로그램의 경우, LD_LIBRARYN32_PATH 환경 변수를 사용하십시오.

질문 1에 대한 응답으로 지정한 플랫폼에 대한 적절한 시스템 명령을 사용하여 라이브러리 검색 경로를 볼 수 있습니다.

조건

일단 위의 질문에 대해 응답하면 사용자 환경을 변경해야 합니다. 아래에 나열된 조건들을 읽어 보십시오. 사용자의 상황에 해당하는 조건이 있으면 필요한 변경을 수행하십시오.

조건 1: 버전 6.1 응용프로그램이 AIX 기본 공유 라이브러리 경로 /usr/lib/libdb2.a로부터 공유 라이브러리를 로드할 경우 및

- /usr/lib/libdb2.a에서 /usr/lpp/db2_06_01/lib/libdb2.a로 기호 링크가 있으며 데이터베이스 서버가 AIX용 DB2 Universal Database 버전 7일 경우, 다음 중 하나를 수행하십시오.

– 다음을 가리키도록 기호 링크를 변경하십시오.

```
/usr/lpp/db2_07_01/lib/libdb2.a
```

*UNIX용 DB2 빠른 시작*에는 라이브러리 간 링크 설정에 대한 정보가 있습니다. 루트로서 다음과 같이 "db2ln" 명령을 사용하여 링크를 변경할 수 있습니다.

```
/usr/lpp/db2_07_01/cfg/db2ln
```

– LIBPATH 환경 변수를 /usr/lpp/db2_07_01/lib 또는 INSTHOME/sqllib/lib를 가리키도록 설정하십시오. 여기서 INSTHOME은 버전 7 DB2 인스턴스 소유자의 홈 디렉토리입니다.

– TCP/IP 연결을 응용프로그램(클라이언트) 인스턴스에서 서버 인스턴스로 구성하십시오. TCP/IP 구성에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

- /usr/lib/libdb2.a에서 /usr/lpp/db2_07_01/lib/libdb2.a로 기호 링크가 있으며 데이터베이스 서버가 DB2 버전 6.1일 경우, TCP/IP 연결을 응용프로그램(클라이언트) 인스턴스에서 서버 인스턴스로 구성하십시오. TCP/IP 구성에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

조건 2: 버전 6.1 응용프로그램이 DB2 버전 6.1 인스턴스 소유자의 $HOME 경로($HOME/sqllib/lib/libdb2.a)로부터 공유 라이브러리를 로드하고 데이터베이스 서버가 AIX용 DB2 Universal Database 버전 7일 경우, 다음 중 하나를 수행하십시오.

- 응용프로그램 인스턴스를 데이터베이스 서버 인스턴스와 같은 버전으로 이주하십시오.
- LIBPATH 환경 변수를 /usr/lpp/db2_07_01/lib 또는 INSTHOME/sqllib/lib를 가리키도록 설정하십시오. 여기서 INSTHOME은 버전 7 인스턴스 소유자의 홈 디렉토리입니다.
- TCP/IP 연결을 응용프로그램(클라이언트) 인스턴스에서 서버 인스턴스로 구성하십시오. TCP/IP 구성에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

조건 3: 버전 6.1 응용프로그램이 DB2 버전 6.1 설치 경로(/usr/lpp/db2_06_00/lib/libdb2.a)로부터 공유 라이브러리를 로드하고 데이터베이스 서버가 AIX용 DB2 Universal Database 버전 7일 경우, 다음 중 하나를 수행하십시오.

- LIBPATH 환경 변수를 /usr/lpp/db2_07_01/lib 또는 INSTHOME/sqllib/lib를 가리키도록 설정하십시오. 여기서 INSTHOME은 데이터베이스 인스턴스 소유자의 홈 디렉토리입니다.
- TCP/IP 연결을 응용프로그램(클라이언트) 인스턴스에서 서버 인스턴스로 구성하십시오. TCP/IP 구성에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

조건 4: 버전 6.1 응용프로그램이 AIX용 DB2 Universal Database 버전 7 설치 경로(/usr/lpp/db2_07_01/lib/libdb2.a)로부터 공유 라이브러리를 로드하고 데이터베이스 서버가 DB2 버전 6.1일 경우, TCP/IP 연결을 응용프로그램(클라이언트) 인스턴스에서 서버 인스턴스로 구성하십시오. TCP/IP 구성에 대한 자세한 내용은 *설치 및 구성 보충 설명서*를 참조하십시오.

기타 이주 고려사항

응용프로그램을 개발할 때 다음 사항을 고려하십시오. 다음과 같이 하면 응용프로그램 이식성을 높이는 데 도움이 됩니다.

- UNIX의 경우, 응용프로그램에서 기본 경로 /usr/lib:/lib만 사용하십시오. OS/2 및 Windows 32비트 운영 체제에서는 다음을 사용하여 LIB 환경 변수가 %DB2PATH%\lib를 가리키도록 하십시오.

  ```
  set LIB=%DB2PATH%\lib;%LIB%
  ```

  또한 기본 경로와 사용 중인 DB2 버전 간의 기호 링크를 작성하십시오. 이 링크가 응용프로그램에 필요한 최소 레벨의 DB2에 대한 것인지 확인하십시오. 링크 설정에 대한 자세한 내용은 사용자의 플랫폼에 대한 *빠른 시작* 책을 참조하십시오.

- 응용프로그램에 특정 버전의 DB2가 필요한 경우, 응용프로그램의 DB2 버전을 지정하는 경로를 코딩하십시오. 예를 들어, AIX 응용프로그램에 DB2 버전 5가 필요한 경우, /usr/lpp/db2_05_00/lib를 코딩하십시오. 대개는 이를 수행할 필요가 없습니다.
- 내부 개발이 아닌 생산을 위해 응용프로그램을 빌드하는 경우, 응용프로그램의 경로는 UNIX의 `sqllib/lib` 디렉토리 또는 OS/2 및 Windows 32비트 운영 체제의 `%DB2PATH%\lib` 디렉토리의 인스턴스 소유자 사본을 가리키지 않아야 합니다. 이는 응용프로그램의 특정 사용자 이름과 환경에 대한 의존성을 높여 줍니다.
- 일반적으로, 특정 환경에서 검색 경로를 변경하려면 LIBPATH 환경 변수 또는 Windows 32비트 운영 체제의 LIB 환경 변수를 사용하지 마십시오. 이 변수는 해당 환경에서 수행 중인 응용프로그램에 지정되어 있는 검색 경로를 겹쳐 쓰게 됩니다. 응용프로그램은 응용프로그램에 필요한 라이브러리나 파일을 찾지 못할 수도 있습니다.
- DB2 Universal Database 버전 6.1 및 7에서 문자열 의미를 갖는 모든 문자 배열 항목은 부호 없는 char과 같은 다른 변형체 대신 유형 char을 가집니다. DB2 Universal Database 버전 6.1 또는 7로 코딩한 모든 응용프로그램은 이 사항을 준수해야 합니다.

  부호 없는 char을 사용하는 DB2 버전 1 응용프로그램이 있을 경우, 컴파일러는 버전 1 응용프로그램의 부호 없는 char과 버전 6.1 또는 7 함수 프로토타입의 char 간에 유형 충돌이 일어나므로 경고 또는 오류를 발생할 수 있습니다. 이 경우에는 컴파일러 옵션 `-DSQLOLDCHAR`을 사용하여 문제를 제거하십시오.
- DB2 Universal Database 버전 7과 이전 버전의 DB2 간의 비호환성 목록에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오. DB2 Universal Database 버전 7과 이전 버전의 DB2 간의 API 비호환성 목록에 대한 자세한 내용은 *Administrative API Reference*를 참조하십시오.

부록C. 문제점 판별

응용프로그램을 빌드하거나 수행할 때 다음과 같은 종류의 문제점이 발생할 수 있습니다.

- 응용프로그램을 빌드 또는 수행 중 데이터베이스에 연결하는 데 실패하는 것과 같은 클라이언트 또는 서버 문제점.
- 빌드 중에 파일을 찾을 수 없는 것과 같은 운영 체제 문제점.
- 빌드 중에 발생하는 컴파일러 옵션 문제점.
- 응용프로그램을 빌드 또는 수행 중 발생하는 구문 및 코딩 문제점.

다음 소스의 정보를 사용하여 이들 문제점을 해결할 수 있습니다.

빌드 파일

데이터베이스 연결, 사전 처리 컴파일, 컴파일, 링크 및 바인드와 같은 빌드 문제점의 경우, 이 책에 있는 빌드 파일을 사용하여 작동하는 명령행 처리기 명령 및 컴파일러 옵션을 찾을 수 있습니다.

컴파일러 문서

빌드 스크립트 파일이 포함하지 않는 컴파일러 옵션 문제점에 대한 것입니다.

응용프로그램 개발 안내서

구문 및 기타 코딩 문제점에 대한 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.

CLI Guide and Reference

구문, CLI 추적 기능, 구성 키워드 및 CLI 프로그램에 관련된 코딩 문제점에 대한 자세한 내용은 *CLI Guide and Reference*를 참조하십시오.

SQL 참조서

SQL문 및 함수의 구문에 대한 자세한 내용은 *SQL 참조서*를 참조하십시오.

SQLCA 데이터 구조

응용프로그램이 SQL문을 발행하거나 데이터베이스 관리자 API를 호출할 경우, SQLCA 데이터 구조를 조사하여 오류 조건을 확인해야 합니다.

SQLCA 데이터 구조는 SQLCODE 및 SQLSTATE 필드에 오류 정보를 리턴합니다. 데이터베이스 관리자는 모든 SQL문이 실행된 다음과 대부분의 데이터베이스 관리자 API가 호출한 다음에 구조를 갱신합니다.

응용프로그램은 오류 정보를 검색하여 인쇄하거나 화면에 표시할 수 있습니다. 자세한 내용은 *응용프로그램 개발 안내서*를 참조하십시오.

온라인 오류 메시지

데이터베이스 관리자, 데이터베이스 관리 유틸리티, 설치 및 구성 프로세스, 그리고 명령행 처리기를 포함하는 다양한 DB2 구성요소는 온라인 오류 메시지를 생성합니다. 이들 각 메시지는 고유 접두부를 가지며 이 접두부 뒤에 4내지 5자리의 메시지 번호를 가집니다. 메시지 번호 뒤에는 오류의 심각도를 나타내는 단일 문자가 표시됩니다.

다음을 입력한 다음 명령행 처리기를 사용하여 이 메시지에 대한 도움말을 볼 수 있습니다.

```
db2 "? xxxnnnn"
```

여기서 xxx는 메시지 접두부이고 nnnn는 메시지 번호입니다. 따옴표를 포함하십시오.

DB2 오류 메시지의 전체 목록과 설명에 대한 자세한 내용은 *메시지 참조서*를 참조하십시오.

진단 도구 및 오류 로그

다른 소스 정보를 사용하여 해결할 수 없는 빌드 또는 런타임 문제점을 위해 제공됩니다. 진단 도구는 추적 기능, 시스템 로그 및 메시지 로그를 포함합니다. DB2는 우선순위와 원인에 따라 오류와 경고 조건을 오류 로그에 기록합니다. 자세한 내용은 *문제점 해결 안내서*를 참조하십시오. 특별히 CLI 프로그램을 디버깅하기 위한 CLI 추적 기능도 있습니다. 자세한 내용은 *CLI Guide and Reference*를 참조하십시오.

부록D. DB2 라이브러리 사용

DB2 Universal Database 라이브러리는 온라인 도움말, 책(PDF 및 HTML) 및 샘플 프로그램이 HTML 형식으로 구성됩니다. 이 절에서는 제공되는 정보 및 액세스하는 방법에 대해 설명합니다.

제품 정보에 온라인으로 액세스하려면 정보 센터를 이용할 수 있습니다. 자세한 내용은 486 페이지의 『정보 센터로 정보에 액세스』를 참조하십시오. 웹에서 타스크 정보, DB2 책, 문제점 해결 정보, 샘플 프로그램 및 DB2 정보를 볼 수 있습니다.

DB2 PDF 파일 및 인쇄된 책

DB2 정보

다음의 표는 DB2 책을 네 개의 범주로 나눕니다.

DB2 안내 및 참조 정보

이 책에는 모든 플랫폼에 공통적인 DB2 정보가 들어 있습니다.

DB2 설치 및 구성 정보

이들 책은 특정 플랫폼에서의 DB2에 대한 것입니다. 예를 들어, OS/2, Windows 및 UNIX 기반 플랫폼에서의 DB2용으로 각각 다른 *빠른 시작* 책이 있습니다.

HTML 형식의 크로스 플랫폼 샘플 프로그램

이들 샘플은 응용프로그램 개발 클라이언트와 함께 설치된 샘플 프로그램의 HTML 버전입니다. 이들은 단지 정보용으로서 실제 프로그램을 바꾸지는 않습니다.

릴리스 정보

이러한 파일에는 DB2 책에 포함되지 않은 최신 정보가 포함되어 있습니다.

설치 매뉴얼, 릴리스 정보 및 자습서는 제품 CD-ROM의 HTML 디렉토리에서 볼 수 있습니다. 대부분의 책은 단지 보기용으로 제품 CD-ROM에서 HTML 형식으로 제공되고 보기와 인쇄용으로 DB2 책 CD-ROM에서 Adobe Acrobat(PDF) 형식으로 제공됩니다. 또한 IBM에서 인쇄된 책을 주문할 수 있습니다. 481 페이지의 『인쇄된 책 주문』을 참조하십시오. 다음 표에는 주문할 수 있는 책을 보여줍니다.

OS/2 및 Windows 플랫폼에서는 sqllib\doc\html 디렉토리에 HTML 파일을 설치할 수 있습니다. DB2 정보는 여러 언어로 번역되었습니다. 그러나 모든 정보가 모든 언어로 번역된 것은 아닙니다. 정보를 특정 언어로 사용할 수 없을 경우에는 영문으로 제공됩니다.

UNIX 플랫폼에서는 doc/*%L*/html 디렉토리에 여러 언어 버전의 HTML 파일을 설치할 수 있습니다. 여기서 *%L*은 로케일을 나타냅니다. 자세한 내용은 *빠른 시작* 책을 참조하십시오.

다음의 여러 가지 방법으로 DB2 책을 구하고 정보에 액세스할 수 있습니다.

- 485 페이지의 『온라인 정보 보기』
- 490 페이지의 『온라인 정보 검색』
- 481 페이지의 『인쇄된 책 주문』
- 481 페이지의 『PDF 책 인쇄』

표 18. DB2 정보

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
	DB2 안내 및 참조 정보		
관리 안내서	*관리 안내서: 계획*에서는 데이터베이스의 개념에 대한 개요, 논리적 또는 물리적인 데이터베이스 설계와 같은 설계에 대한 정보, 그리고 고가용성에 대한 정보를 제공합니다. *관리 안내서: 구현*에서는 사용자의 설계 구현, 데이터베이스 액세스, 감사, 백업 및 복구와 같은 구현에 대한 정보를 제공합니다. *관리 안내서: 성능*에서는 데이터베이스의 환경, 응용프로그램 성능 평가 및 조정에 대한 정보를 제공합니다. 사용자는 문서 번호 SBOF-8934를 사용하여 세 권으로 된 *관리 안내서* 책을 주문할 수 있습니다.	SA30-0990 db2d1x70 SA30-0988 db2d2x70 SA30-0989 db2d3x70	db2d0
Administrative API Reference	데이터베이스를 관리하는 데 사용할 수 있는 DB2 API와 데이터 구조에 대해 설명합니다. 또한 응용프로그램에서 API를 호출하는 방법에 대해 설명합니다.	SC09-2947 db2b0x70	db2b0
응용프로그램 빌드 안내서	환경 설정 정보와 Windows, OS/2 및 UNIX 기반 플랫폼에서 DB2 응용프로그램을 컴파일, 링크 및 수행하는 방법에 대한 단계별 지침을 제공합니다.	SA30-0991 db2axx70	db2ax
APPC, CPI-C, and SNA Sense Codes	DB2 Universal Database 제품을 사용할 때 발생할 수 있는 APPC, CPI-C 및 SNA 감지 코드에 대한 일반 정보를 제공합니다. HTML 형식으로만 사용할 수 있습니다.	문서 번호 없음 db2apx70	db2ap
응용프로그램 개발 안내서	Embedded SQL 또는 Java(JDBC 및 SQLJ)를 사용하여 DB2 데이터베이스에 액세스하는 응용프로그램을 개발하는 방법에 대해 설명합니다. 저장 프로시듀어 작성, 사용자 정의 함수(UDF) 작성, 사용자 정의 유형 작성, 트리거 사용, 파티션된 환경 또는 연합 시스템에서 응용프로그램을 개발하는 등의 다양한 주제가 다루어집니다.	SA30-0992 db2a0x70	db2a0

표 18. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
CLI Guide and Reference	Microsoft ODBC 스펙과 호환 가능한 DB2 콜 레벨 인터페이스 및 호출 가능 SQL 인터페이스를 사용하여 DB2 데이터베이스에 액세스하는 응용프로그램을 개발하는 방법에 대해 설명합니다.	SC09-2950 db2l0x70	db2l0
Command Reference	명령행 처리기를 사용하는 방법 및 데이터베이스를 관리하기 위해 사용할 수 있는 DB2 명령에 대해 설명합니다.	SC09-2951 db2n0x70	db2n0
연결성 보충 설명서	AS/400용 DB2, OS/390용 DB2, MVS용 DB2 또는 VM용 DB2를 DB2 Universal Database 서버와의 DRDA 응용프로그램 리퀘스터(AR)로 사용하는 방법에 대한 참조 정보 및 설치 정보를 제공합니다. 또한 DB2 Connect 응용프로그램 리퀘스터(AR)와 함께 DRDA 응용프로그램 서버를 사용하는 방법에 대해서도 상세히 설명합니다. HTML 및 PDF 형식으로만 사용할 수 있습니다.	문서 번호 없음 db2h1x70	db2h1
데이터 이동 유틸리티 안내 및 참조서	가져오기, 내보내기, 로드, 자동 로드 프로그램 및 DPROP와 같이 데이터 이동을 용이하게 해 주는 DB2 UDB 유틸리티의 사용 방법에 대해 설명합니다.	SA30-0994 db2dmx70	db2dm
Data Warehouse Center 관리 안내서	Data Warehouse Center를 사용하여 데이터 웨어하우스를 빌드 및 유지보수하는 방법에 대한 정보를 제공합니다.	SA30-1000 db2ddx70	db2dd
Data Warehouse Center 응용프로그램 통합 안내서	프로그래머들이 Data Warehouse Center 및 Information Catalog Manager를 응용프로그램과 통합하는 데 도움을 주는 정보를 제공합니다.	SA30-1001 db2adx70	db2ad
DB2 Connect 사용자 안내서	DB2 Connect 제품에 대한 개념, 프로그래밍 및 일반 사용 정보를 제공합니다.	SA30-0993 db2c0x70	db2c0
DB2 Query Patroller Administration Guide	DB2 Query Patroller 시스템의 조작 개요, 특정 조작 및 관리 정보, 관리 그래픽 사용자 인터페이스 유틸리티에 대한 타스크 정보를 제공합니다.	SC09-2958 db2dwx70	db2dw

표 18. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
DB2 Query Patroller User's Guide	DB2 Query Patroller의 도구 및 함수를 사용하는 방법에 대해 설명합니다.	SC09-2960 db2wwx70	db2ww
용어집	DB2에서 사용되는 용어와 그 구성요소에 대한 정의를 제공합니다. HTML 형식과 *SQL 참조서*에서 사용할 수 있습니다.	문서 번호 없음 db2t0x70	db2t0
Image, Audio, and Video Extenders 관리 및 프로그래밍	DB2 Extender에 대한 일반 정보와, 이미지, 오디오 및 비디오(IAV) Extenders 관리 및 구성에 대한 정보, 그리고 IAV Extenders를 사용한 프로그래밍에 대한 정보를 제공합니다. 여기에는 참조 정보, 진단 정보(메시지 포함) 및 샘플도 들어 있습니다.	SA30-1043 dmbu7x70	dmbu7
Information Catalog Manager Administration Guide	정보 카탈로그 관리에 대한 지침을 제공합니다.	SC26-9995 db2dix70	db2di
Information Catalog Manager Programming Guide and Reference	Information Catalog Manager에 대한 아키텍처 인터페이스에 대한 정의를 제공합니다.	SC26-9997 db2bix70	db2bi
Information Catalog Manager 사용자 안내서	Information Catalog Manager 사용자 인터페이스 사용에 대한 정보를 제공합니다.	SA30-1002 db2aix70	db2ai
설치 및 구성 보충 설명서	플랫폼 특정 DB2 클라이언트의 플랜, 설치 및 설정에 대해 설명합니다. 또한 바인딩, 클라이언트 및 서버 통신의 설정, DB2 GUI 도구, DRDA AS, 분산 설치 및 분산 요청(DR)의 구성 및 이기종 데이터 소스에 액세스 등에 대한 정보가 들어 있습니다.	GA30-0975 db2iyx70	db2iy
메시지 참조서	DB2, Information Catalog Manager 및 Data Warehouse Center가 발행하는 메시지와 코드를 나열하고 수행해야 할 조치에 대해 설명합니다. 문서 번호(SBOF-8932)를 사용하여 두 권으로 된 메시지 참조서 책을 모두 주문할 수 있습니다.	볼륨 1 GA30-0986 db2m1x70 볼륨 2 GA30-0987 db2m2x70	db2m0

표 18. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
OLAP Integration Server Administration Guide	OLAP Integration Server의 관리 프로그램 구성요소를 사용하는 방법에 대해 설명합니다.	SC27-0787 db2dpx70	n/a
OLAP Integration Server Metaoutline User's Guide	표준 OLAP Metaoutline 인터페이스(Metaoutline Assistant가 아닌)를 사용하여 OLAP Metaoutlines를 작성하고 처리하는 방법에 대해 설명합니다.	SC27-0784 db2upx70	n/a
OLAP Integration Server Model User's Guide	표준 OLAP 모델 인터페이스(Model Assistant가 아닌)를 사용하여 OLAP 모델을 작성하는 방법에 대해 설명합니다.	SC27-0783 db2lpx70	n/a
OLAP 설치 및 사용자 안내서	OLAP Starter Kit에 대한 구성 및 설치 정보를 제공합니다.	SA30-1074 db2ipx70	db2ip
Excel용 OLAP Spreadsheet Add-in 사용자 안내서	Excel 스프레드시트 프로그램을 사용하여 OLAP 데이터를 분석하는 방법에 대해 설명합니다.	SA30-1094 db2epx70	db2ep
Lotus 1-2-3용 OLAP Spreadsheet Add-in 사용자 안내서	Lotus 1-2-3 스프레드시트 프로그램을 사용하여 OLAP 데이터를 분석하는 방법에 대해 설명합니다.	SA30-1093 db2tpx70	db2tp
복제 안내 및 참조서	DB2와 함께 제공된 IBM 복제 도구에 관한 플랜, 구성, 관리 및 사용 정보를 제공합니다.	SA30-1003 db2e0x70	db2e0
Spatial Extender 사용자 안내 및 참조서	Spatial Extender 설치, 구성, 관리, 프로그래밍 및 문제점 해결에 대한 정보를 제공합니다. 또한 공간 데이터 개념에 대한 설명을 제공하고 Spatial Extender에 대한 특정 참조 정보(메시지 및 SQL)를 제공합니다.	SA30-1045 db2sbx70	db2sb
SQL 시작하기	SQL 개념을 소개하고 많은 구조와 타스크에 관한 예를 제공합니다.	SA30-0996 db2y0x70	db2y0
SQL 참조서, 볼륨 1 및 볼륨 2	SQL 구문, 의미 및 규칙에 대해 설명합니다. 또한 릴리스 간 비호환성, 제품 제한사항 및 카탈로그 뷰에 대한 정보도 들어 있습니다. SBOF-8933 문서 번호를 사용하여 *SQL 참조서*를 주문할 수 있습니다.	볼륨 1 SA30-0997 db2s1x70 볼륨 2 SA30-0998 db2s2x70	db2s0

표 18. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
시스템 모니터 안내 및 참조서	데이터베이스와 데이터베이스 관리자에 대한 다른 종류의 정보를 수집하는 방법에 대해 설명합니다. 이 책은 데이터베이스 활동을 이해하고 성능을 향상시키며 문제점의 원인을 판별하기 위한 정보를 사용하는 방법에 대해 설명합니다.	SA30-0995 db2f0x70	db2f0
Text Extender 관리 및 프로그래밍	DB2 Extenders에 대한 일반 정보, Text Extenders 관리 및 구성에 관한 정보, Text Extenders를 사용한 프로그래밍에 대한 정보를 제공합니다. 여기에는 참조 정보, 진단 정보(메시지 포함) 및 샘플도 들어 있습니다.	SA30-1044 desu9x70	desu9
문제점 해결 안내서	오류의 소스를 판별하고 문제점으로부터 복구하며 DB2 고객 서비스와 상담하여 진단 도구를 사용하는 것을 도와줍니다.	GA30-0704 db2p0x70	db2p0
새로운 기능	DB2 Universal Database 버전 7의 새로운 특성, 기능 및 향상된 내용에 대해 설명합니다.	SA30-0999 db2q0x70	db2q0
	DB2 설치 및 구성 정보		
OS/2 및 Windows용 DB2 Connect Enterprise Edition 빠른 시작	OS/2 및 Windows 32비트 운영 체제에서 DB2 Connect Enterprise Edition에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0974 db2c6x70	db2c6
UNIX용 DB2 Connect Enterprise Edition 빠른 시작	UNIX 기반 플랫폼에서의 DB2 Connect Enterprise Edition에 대한 플랜, 이주, 설치, 구성 및 타스크 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0973 db2cyx70	db2cy
DB2 Connect Personal Edition 빠른 시작	OS/2 및 Windows 32비트 운영 체제에서 DB2 Connect Personal Edition에 관한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 모든 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0981 db2c1x70	db2c1
DB2 Connect Personal Edition Quick Beginnings for Linux	지원되는 모든 Linux 분산에서 DB2 Connect Personal Edition에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다.	GC09-2962 db2c4x70	db2c4

표 18. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
DB2 Data Links Manager 빠른 시작	AIX 및 Windows 32비트 운영 체제용 DB2 Data Links Manager에 대한 플랜, 설치, 구성 및 타스크 정보를 제공합니다.	GA30-0980 db2z6x70	db2z6
UNIX용 DB2 Enterprise - Extended Edition 빠른 시작	UNIX 기반 플랫폼에서의 DB2 Enterprise - Extended Edition 플랜, 설치 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0978 db2v3x70	db2v3
Windows용 DB2 Enterprise - Extended Edition 빠른 시작	Windows 32비트 운영 체제용 DB2 Enterprise - Extended Edition에 대한 플랜, 설치 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0977 db2v6x70	db2v6
OS/2용 DB2 빠른 시작	OS/2 운영 체제에서의 DB2 Universal Database에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0982 db2i2x70	db2i2
UNIX용 DB2 빠른 시작	UNIX 기반 플랫폼에서의 DB2 Universal Database에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0984 db2ixx70	db2ix
Windows용 DB2 빠른 시작	Windows 32비트 운영 체제에서의 DB2 Universal Database에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다. 또한 지원되는 많은 클라이언트에 대한 설치 및 설정 정보도 들어 있습니다.	GA30-0985 db2i6x70	db2i6
DB2 Personal Edition 빠른 시작	OS/2 및 Windows 32비트 운영 체제에서의 DB2 Universal Database Personal Edition에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다.	GA30-0983 db2i1x70	db2i1
DB2 Personal Edition Quick Beginnings for Linux	지원되는 모든 Linux 분산에서 DB2 Universal Database Personal Edition에 대한 플랜, 설치, 이주 및 구성 정보를 제공합니다.	GC09-2972 db2i4x70	db2i4
DB2 Query Patroller 설치 안내서	DB2 Query Patroller에 대한 설치 정보를 제공합니다.	GA30-0976 db2iwx70	db2iw

표 18. DB2 정보 (계속)

이름	설명	문서 번호 **PDF** 파일 이름	**HTML** 디렉토리
DB2 Warehouse Manager 설치 안내서	웨어하우스 에이전트, 웨어하우스 변환기 및 Information Catalog Manager에 대한 설치 정보를 제공합니다.	GA30-1027 db2idx70	db2id
HTML 형식의 크로스 플랫폼 샘플 프로그램			
HTML 형식의 샘플 프로그램	DB2가 지원하는 모든 플랫폼에서 프로그래밍 언어에 대한 샘플 프로그램이 HTML 형식으로 제공됩니다. 이 샘플 프로그램은 정보용으로만 제공됩니다. 모든 샘플을 모든 프로그래밍 언어로 사용할 수 있는 것은 아닙니다. HTML 샘플은 DB2 응용프로그램 개발 클라이언트가 설치될 때만 사용할 수 있습니다. 프로그램에 대한 자세한 내용은 *응용프로그램 빌드 안내서*를 참조하십시오.	문서 번호 없음	db2hs
릴리스 정보			
DB2 Connect 릴리스 정보	DB2 Connect 책에는 포함될 수 없었던 최신 정보를 제공합니다.	#2를 참조하십시오.	db2cr
DB2 설치 정보	DB2 책에는 포함될 수 없었던 최신 설치 특정 정보를 제공합니다.	제품 CD-ROM에서만 사용할 수 있습니다.	
DB2 릴리스 정보	DB2 책에는 포함될 수 없었던 모든 DB2 제품 및 기능에 대한 최신 정보를 제공합니다.	#2를 참조하십시오.	db2ir

주:

1. 파일 이름의 6번째 자리에 있는 문자 *x*는 책의 언어 버전을 나타냅니다. 예를 들어, 파일 이름 `db2d0e70`은 *관리 안내서* 책의 영문 버전을 나타내며 `db2d0k70`은 같은 책의 한글 버전을 나타냅니다. 다음 문자는 파일 이름의 6번째 자리에 사용되어 언어 버전을 나타냅니다.

언어	식별자
브라질 포르투갈어	b
불가리아어	u
체코어	x
덴마크어	d
네덜란드어	q
영어	e

핀란드어	y
프랑스어	f
독일어	g
그리스어	a
헝가리어	h
이탈리아어	i
일본어	j
한국어	k
노르웨이어	n
폴란드어	p
포르투갈어	v
러시아어	r
중국어	c
슬로베니아어	l
스페인어	z
스웨덴어	s
대만어	t
터키어	m

2. DB2 책에 포함되어 있지 않을 수 있는 최신 정보는 릴리스 정보에서 HTML 형식과 ASCII 파일로 사용할 수 있습니다. HTML 버전은 정보 센터와 제품 CD-ROM에서 사용할 수 있습니다. ASCII 파일을 보려면 다음을 수행하십시오.
   - UNIX 기반 플랫폼의 경우에는 Release.Notes 파일을 참조하십시오. 이 파일은 DB2DIR/Readme/*%L* 디렉토리에 있으며, 여기서 *%L*은 로케일 이름을 나타내고 DB2DIR은 다음과 같습니다.
     - AIX에서는 /usr/lpp/db2_07_01
     - HP-UX, PTX, Solaris 및 Silicon Graphics IRIX에서는 /opt/IBMdb2/V7.1
     - Linux에서는 /usr/IBMdb2/V7.1
   - 다른 플랫폼의 경우에는 RELEASE.TXT 파일을 참조하십시오. 이 파일은 제품이 설치된 디렉토리에 있습니다. OS/2 플랫폼에서는 **IBM DB2** 폴더를 더블 클릭한 다음 **릴리스 정보** 아이콘을 더블 클릭할 수 있습니다.

PDF 책 인쇄

인쇄된 책의 사본을 원하는 경우, DB2 책 CD-ROM에 있는 PDF 파일을 인쇄할 수 있습니다. Adobe Acrobat Reader를 사용하여 책 전체나 특정 페이지를 인쇄할 수 있습니다. 라이브러리에 있는 각 책의 파일 이름에 대한 자세한 내용은 473 페이지의 표18을 참조하십시오.

Adobe 웹 사이트 http://www.adobe.com에서 Adobe Acrobat Reader의 최신 버전을 얻을 수 있습니다.

PDF 파일은 파일 확장자가 PDF인 DB2 책 CD-ROM에 들어 있습니다. PDF 파일에 액세스하려면 다음을 수행하십시오.

1. DB2 책 CD-ROM을 삽입하십시오. UNIX 기반의 플랫폼에서는 DB2 책 CD-ROM을 마운트합니다. 마운트 프로시듀어에 대한 자세한 내용은 *빠른 시작* 책을 참조하십시오.
2. Acrobat Reader를 시작하십시오.
3. 다음 위치 중 하나에서 원하는 PDF 파일을 여십시오.
   - OS/2 및 Windows 플랫폼에서

     x:\doc*language* 디렉토리. 여기서 *x*는 CD-ROM 드라이브를 나타내며 *language*는 사용자 언어를 나타내는 2문자 국가 코드를 나타냅니다. 예를 들어, 영문인 경우에는 EN입니다.
   - UNIX 기반 플랫폼에서

     /cdrom/doc/*%L* 디렉토리. 여기서 */cdrom*은 CD-ROM의 마운트 지점을 나타내고 *%L*은 원하는 로케일의 이름을 나타냅니다.

또한 PDF 파일을 CD-ROM에서 지역이나 네트워크 드라이브로 파일을 복사하고 거기서 읽을 수도 있습니다.

인쇄된 책 주문

인쇄된 DB2 책은 책 주문 번호(SBOF)를 사용하여 세트나 낱권으로 주문할 수 있습니다. 인쇄본을 주문하려면 IBM 협력업체 또는 영업 대표에게 문의하십시오. 또한 웹 페이지 http://www.elink.ibmlink.ibm.com/pbl/pbl에서도 책을 주문할 수 있습니다.

두 종류의 책 세트를 사용할 수 있습니다. SBOF-8935는 DB2 Warehouse Manager에 대한 참조 및 사용에 관한 정보를 제공합니다. SBOF-8931은 다른 모든 DB2 Universal Database 제품과 기능에 대한 참조 및 사용 정보를 제공합니다. 각 SBOF의 내용은 다음 표에 나열되어 있습니다.

표 19. 인쇄된 책 주문

SBOF 번호	**포함된 책**	
SBOF-8931	• 관리 안내서: 계획 • 관리 안내서: 구현 • 관리 안내서: 성능 • Administrative API Reference • 응용프로그램 빌드 안내서 • 응용프로그램 개발 안내서 • CLI Guide and Reference • Command Reference • 데이터 이동 유틸리티 안내 및 참조서 • Data Warehouse Center 관리 안내서 • Data Warehouse Center 응용프로그램 통합 안내서 • DB2 Connect 사용자 안내서 • 설치 및 구성 보충 설명서 • Image, Audio, and Video Extenders 관리 및 프로그래밍 • 메시지 참조서, 볼륨 1 및 2	• OLAP Integration Server Administration Guide • OLAP Integration Server Metaoutline User's Guide • OLAP Integration Server Model User's Guide • OLAP Integration Server User's Guide • OLAP 설치 및 사용자 안내서 • Excel용 OLAP Spreadsheet Add-in 사용자 안내서 • Lotus 1-2-3용 OLAP Spreadsheet Add-in 사용자 안내서 • 복제 안내 및 참조서 • Spatial Extender Administration and Programming Guide • SQL 시작하기 • SQL 참조서, 볼륨 1 및 2 • 시스템 모니터 안내 및 참조서 • Text Extender 관리 및 프로그래밍 • 문제점 해결 안내서 • 새로운 기능

표 19. 인쇄된 책 주문 (계속)

SBOF 번호	포함된 책	
SBOF-8935	• Information Catalog Manager Administration Guide • Information Catalog Manager 사용자 안내서 • Information Catalog Manager Programming Guide and Reference	• Query Patroller Administration Guide • Query Patroller User's Guide

DB2 온라인 문서

온라인 도움말 액세스

온라인 도움말은 모든 DB2 구성요소에서 사용 가능합니다. 다음의 표에서는 다양한 도움말 유형에 대해 설명합니다.

도움말 유형	내용	액세스하는 방법
명령 도움말	명령행 처리기의 명령 구문에 대해 설명합니다.	대화식 모드의 명령행 처리기에서 다음을 입력하십시오. `? command` 여기서 *command*는 키워드이거나 전체 명령입니다. 예를 들어, `? catalog`는 모든 CATALOG 명령에 대한 도움말을 표시하고, `? catalog database`는 CATALOG DATABASE 명령에 대한 도움말을 표시합니다.

도움말 유형	내용	액세스하는 방법
클라이언트 구성 지원 프로그램 도움말 명령 센터 도움말 제어 센터 도움말 Data Warehouse Center 도움말 이벤트 분석기 도움말 Information Catalog Manager 도움말 위성 관리 센터 도움말 스크립트 센터 도움말	창 또는 노트북에서 수행할 수 있는 타스크에 대해 설명합니다. 도움말은 알아야 할 개요와 전제조건 정보를 포함하고, 창 또는 노트북 제어를 사용하는 방법에 대해 설명합니다.	창이나 노트북에서 **도움말** 버튼을 누르거나 **F1** 키를 누르십시오.

메시지 도움말	메시지의 원인과 사용자가 취해야 할 조치에 대해 설명합니다.	대화식 모드의 명령행 처리기에서 다음을 입력하십시오.
		`? XXXnnnnn`
		여기서 *XXXnnnnn*은 유효한 메시지 식별자입니다.
		예를 들어, `? SQL30081`은 SQL30081 메시지에 대한 도움말을 표시합니다.
		한 번에 한 화면씩 메시지 도움말을 보려면 다음을 입력하십시오.
		`? XXXnnnnn | more`
		파일에 메시지 도움말을 저장하려면 다음을 입력하십시오.
		`? XXXnnnnn > filename.ext`
		여기서 *filename.ext*는 메시지 도움말을 저장하려는 파일입니다.

도움말 유형	내용	액세스하는 방법
SQL 도움말	SQL문의 구문에 대해 설명합니다.	대화식 모드의 명령행 처리기에서 다음을 입력하십시오. `help statement` 여기서 *statement*는 SQL문입니다. 예를 들어, help SELECT는 SELECT문에 대한 도움말을 표시합니다. **주:** SQL 도움말은 UNIX 기반 플랫폼에서 사용할 수 없습니다.
SQLSTATE 도움말	SQL 상태와 클래스 코드에 대해 설명합니다.	대화식 모드의 명령행 처리기에서 다음을 입력하십시오. `? sqlstate` 또는 `? class code` 여기서 *sqlstate*는 유효한 5자리 숫자로 된 SQL 상태이고 *class code*는 SQL 상태의 처음 2자리 숫자입니다. 예를 들어, `? 08003`은 08003 SQL 상태에 대한 도움말을 표시하고, `? 08`은 08 클래스 코드에 대한 도움말을 표시합니다.

온라인 정보 보기

이 제품에 들어 있는 책은 HTML(Hypertext Markup Language) 소프트카피 형식으로 제공됩니다. 소프트카피 형식은 정보를 검색할 수 있게 하고 관련된 정보로 링크하는 하이퍼텍스트를 제공합니다. 또한 사이트에서 라이브러리를 공유하는 것도 더 쉬워집니다.

HTML 버전 3.2 스펙을 따르는 브라우저로 온라인 책 또는 샘플 프로그램을 볼 수 있습니다.

온라인 책 또는 샘플 프로그램을 보려면 다음을 수행하십시오.

- DB2 관리 도구를 수행할 경우, 정보 센터를 사용하십시오.
- 브라우저에서 **파일 —> 페이지 열기**를 누르십시오. 열린 페이지에 DB2 정보에 대한 설명과 링크가 들어 있습니다.
  - UNIX 기반 플랫폼의 경우, 다음 페이지를 여십시오.

    INSTHOME/sqllib/doc/%L/html/index.htm

여기서 *%L*은 로케일 이름입니다.

– 기타 플랫폼에서는 다음 페이지를 여십시오.

```
sqllib\doc\html\index.htm
```

이 경로는 DB2가 설치된 드라이브에 있습니다.

정보 센터를 설치하지 않은 경우, **DB2 정보** 아이콘을 더블 클릭하여 페이지를 열 수 있습니다. 사용하는 시스템에 따라 주 제품 폴더나 Windows 시작 메뉴에 아이콘이 있습니다.

Netscape 브라우저 설치

웹 브라우저를 설치하지 않은 경우, 제품 상자에 있는 Netscape CD-ROM에서 Netscape를 설치할 수 있습니다. 설치하는 방법에 대한 자세한 지침을 보려면 다음을 수행하십시오.

1. Netscape CD-ROM을 삽입하십시오.
2. UNIX 기반 플랫폼에서는 CD-ROM을 마운트하십시오. 마운트 프로시듀어에 대한 자세한 내용은 *빠른 시작* 책을 참조하십시오.
3. 설치 지침의 경우에는 CDNAV*nn*.txt 파일을 참조하십시오. 여기서 *nn*은 2문자로 된 언어 식별자입니다. 파일은 CD-ROM의 루트 디렉토리에 있습니다.

정보 센터로 정보에 액세스

정보 센터는 DB2 제품 정보에 대한 빠른 액세스를 제공합니다. 정보 센터는 DB2 관리 도구를 사용할 수 있는 모든 플랫폼에서 사용할 수 있습니다.

정보 센터 아이콘을 더블 클릭하여 정보 센터를 열 수 있습니다. 사용하는 시스템에 따라 아이콘은 주 제품 폴더나 Windows **시작** 메뉴의 정보 폴더에 있습니다.

또한 DB2 Windows 플랫폼에서 도구 모음이나 **도움말** 메뉴를 사용하여 정보 센터에 액세스할 수 있습니다.

정보 센터는 6개 유형의 정보를 제공합니다. 적절한 탭을 눌러 해당 유형에 제공되는 주제를 보십시오.

타스크 DB2를 사용하여 수행할 수 있는 주요 타스크.

참조 키워드, 명령 및 API와 같은 DB2 참조 정보.

책 DB2 책.

문제점 해결 오류 메시지의 범주와 복구 조치.

샘플 프로그램 DB2 응용프로그램 개발 클라이언트와 함께 제공되는 샘플 프로그램. DB2 응용프로그램 개발 클라이언트를 설치하지 않은 경우, 이 탭은 표시되지 않습니다.

웹 월드 와이드 웹에서의 DB2 정보. 이 정보에 액세스하려면 시스템에서 웹으로의 연결을 갖고 있어야 합니다.

목록 중 하나에서 항목을 선택하면 정보 센터가 표시기를 시작하여 정보를 표시합니다. 표시기는 사용자가 선택하는 정보의 종류에 따라 시스템 도움말 표시기, 편집기 또는 웹브라우저가 될 수 있습니다.

정보 센터는 찾기 기능을 제공하므로 목록을 찾지 않고도 특정 주제를 찾을 수 있습니다.

전체 텍스트 검색의 경우, **DB2 온라인 정보 검색** 검색 양식으로 연결된 정보 센터의 하이퍼텍스트 링크를 따라 검색하십시오.

HTML 검색 서버는 보통 자동으로 시작됩니다. HTML 정보에서 검색 기능이 작동하지 않으면 다음 방법 중 하나를 사용하여 검색 서버를 시작할 수 있습니다.

Windows의 경우:

시작을 누르고 **프로그램 —> IBM DB2 —> 정보 —> HTML 검색 서버 시작**을 선택하십시오.

OS/2의 경우

OS/2용 DB2 폴더를 더블 클릭한 다음 **HTML 검색 서버 시작** 아이콘을 더블 클릭하십시오.

HTML 정보 검색시 그 외의 다른 문제가 발생한 경우에는 릴리스 정보를 참조하십시오.

주: 검색 기능은 Linux, PTX 및 Silicon Graphics IRIX 환경에서는 사용할 수 없습니다.

DB2 마법사 사용

마법사는 한 번에 한 단계씩 각 타스크를 수행하게 함으로써 특정 관리 타스크를 완료하는 데 도움을 줍니다. 마법사는 제어 센터 및 클라이언트 구성 지원 프로그램을 통해 사용할 수 있습니다. 다음 표에서는 마법사를 나열하고 해당 기능에 대해 설명합니다.

주: 데이터베이스 작성, 색인 작성, 다중 사이트 갱신 구성 및 성능 구성 마법사는 파티션된 데이터베이스 환경에서 사용할 수 있습니다.

마법사	도움 내용	액세스하는 방법
데이터베이스 추가	클라이언트 워크스테이션의 데이터베이스를 카탈로그화합니다.	클라이언트 구성 지원 프로그램에서 **추가**를 누르십시오.
데이터베이스 백업	백업 플랜의 결정, 작성 및 스케줄합니다.	제어 센터에서 백업하려는 데이터베이스를 마우스 오른쪽 단추로 누른 다음 **백업 —> 마법사를 사용한 데이터베이스**를 선택하십시오.
다중 사이트 갱신 구성	다중 사이트 갱신, 분산 트랜잭션 또는 2단계 확약을 구성합니다.	제어 센터에서 **데이터베이스** 폴더를 마우스 오른쪽 단추로 누른 다음 **다중 사이트 갱신**을 선택하십시오.
데이터베이스 작성	데이터베이스 작성 및 일부 기본 구성 타스크를 수행합니다.	제어 센터에서 **데이터베이스** 폴더를 마우스 오른쪽 단추로 누른 다음 **작성 —> 마법사를 사용한 데이터베이스**를 선택하십시오.
테이블 작성	기본 데이터 유형의 선택 및 테이블의 기본 키를 작성합니다.	제어 센터에서 **테이블** 아이콘을 마우스 오른쪽 단추로 누른 다음 **작성 —> 마법사를 사용한 테이블**을 선택하십시오.
테이블 공간 작성	새로운 테이블 공간을 작성합니다.	제어 센터에서 **테이블 공간** 아이콘을 마우스 오른쪽 단추로 선택한 다음 **작성 —> 마법사를 사용한 테이블 공간**을 선택하십시오.
색인 작성	사용자의 모든 조회를 작성하고 삭제하기 위해 색인화합니다.	제어 센터에서 **색인** 아이콘을 마우스 오른쪽 단추로 누른 다음 **작성 —> 마법사를 사용한 색인**을 선택하십시오.

마법사	도움 내용	액세스하는 방법
성능 구성	비지니스 요구에 맞게 구성 매개변수를 갱신하여 데이터베이스의 성능을 조정합니다.	제어 센터에서 조정하려는 데이터베이스를 마우스 오른쪽 단추로 누른 다음 **마법사를 사용한 성능 구성**을 선택하십시오. 파티션된 데이터베이스 환경에 대해 데이터베이스 파티션 뷰에서 조정하려는 첫 번째 데이터베이스 파티션을 마우스 오른쪽 단추로 누른 다음 **마법사를 사용한 성능 구성**을 선택하십시오.
데이터베이스 복원	실패 후에 데이터베이스를 복구합니다. 사용할 백업 종류 및 재작동할 로그를 이해하는 데 도움을 줍니다.	제어 센터에서 복원하려는 데이터베이스를 마우스 오른쪽 단추로 누른 다음 **복원 —> 마법사를 사용한 데이터베이스**를 선택하십시오.

문서 서버 설정

기본값으로 DB2 정보는 지역 시스템에 설치됩니다. 이는 DB2 정보에 액세스해야 하는 모든 사람이 동일한 파일을 설치해야 함을 의미합니다. DB2 정보를 단일 위치에 저장하려면 다음 단계를 수행하십시오.

1. 모든 파일과 서브디렉토리를 지역 시스템의 \sqllib\doc\html에서 웹 서버로 복사하십시오. 각 책은 책을 구성하는 데 필요한 모든 HTML 및 GIF 파일이 들어 있는 서브디렉토리를 가집니다. 디렉토리 구조가 변경되지 않게 하십시오.
2. 새로운 위치에 있는 파일을 찾도록 웹 서버를 구성하십시오. 자세한 내용은 *설치 및 구성 보충 설명서*의 부록 NetQuestion을 참조하십시오.
3. Java 버전의 정보 센터를 사용 중인 경우, 모든 HTML 파일에 대한 기본 URL을 지정할 수 있습니다. 책 목록에 대한 URL을 사용해야 합니다.
4. 책 파일을 볼 수 있게 되면 다음과 같이 자주 보는 주제 항목에 대해서는 책갈피를 설정할 수 있습니다. 다음 페이지를 책갈피로 설정해 두면 도움이 됩니다.
   - 책 목록
   - 자주 사용되는 책의 목차
   - ALTER TABLE 주제와 같은 자주 참조하는 항목
   - 검색 양식

DB2 Universal Database 온라인 문서 파일을 중앙 머신에서 제공하는 방법에 대한 자세한 내용은 *설치 및 구성 보충 설명서*의 부록 NetQuestion을 참조하십시오.

온라인 정보 검색

HTML 파일에서 정보를 찾으려면 다음 방법 중 하나를 사용하십시오.

- 맨 위 프레임에서 **검색**을 누르십시오. 특정 주제를 찾으려면 검색 양식을 사용하십시오. 이 기능은 Linux, PTX 또는 Silicon Graphics IRIX 환경에서는 사용할 수 없습니다.
- 맨 위 프레임에서 **색인**을 누르십시오. 책에서 특정 주제를 찾으려면 색인을 사용하십시오.
- 책에서 특정 주제를 찾으려면 목차나 도움말의 색인 또는 HTML 책을 표시하고 웹 브라우저의 찾기 기능을 사용하십시오.
- 특정 주제로 빨리 리턴하려면 웹 브라우저의 책갈피 기능을 사용하십시오.
- 특정 주제를 찾으려면 정보 센터의 검색 기능을 사용하십시오. 자세한 내용은 486 페이지의 『정보 센터로 정보에 액세스』를 참조하십시오.

부록E. 주의사항

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 사용권까지 부여하는 것은 아닙니다. 사용권에 대한 의문사항은 다음으로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2 바이트(DBCS) 정보에 관한 사용권 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이 책을『현상태대로』제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 이 변경사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통고 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트의 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독립적으로 작성된 프로그램 및 기타 프로그램(이 프로그램 포함)간의 정보 교환 (ii) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 사용권자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

이러한 정보는 해당 조항 및 조건(예를 들면, 사용권 지불 포함)에 따라 사용할 수 있습니다.

이 정보에 기술된 사용권 프로그램 및 사용가능한 모든 사용권 자료는 IBM이 IBM 기본 계약, IBM 프로그램 사용권 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

여기에 있는 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서, 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 측정치는 개발 레벨 시스템에서 작성되었을 수 있으며, 이러한 측정치가 일반적으로 사용가능한 시스템에서도 동일하다고는 보장하지 않습니다. 더우기, 일부 측정치는 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 본인의 특정 환경에 적용할 수 있는 데이터를 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 배상 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지없이 변경될 수 있습니다.

이 정보에는 일상의 업무에서 사용되는 자료와 보고의 예제가 포함되어 있을 수 있습니다. 가능한 완벽하게 설명하기 위해 개인, 회사, 상표 및 제품의 이름이 예제에 들어 있습니다. 이들 이름은 모두 가공의 것이며, 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권:

이 정보에는 여러 가지 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 포함되어 있을 수 있습니다. 샘플 응용프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스에 부합하는 응용프로그램을 개발, 사용, 마케팅 또는 배포를 목적으로 이들 샘플 프로그램을 복사, 수정 및 배포할 수 있으며 IBM에 대한 지불 의무는 없습니다. 이들 예제 프로그램은 모든 조건에서 철저하게 검사된 것은 아닙니다. 따라서, IBM은 이들 프로그램의 신뢰성, 서비스 가능성 또는 기능에 대해 어떠한 보증도 하지 않습니다.

각 사본이나 이들 샘플 프로그램의 일부 또는 파생본에는 다음과 같은 저작권 주의사항을 포함시켜야 합니다.

상표

별표(*)로 표시된 다음의 용어는 전세계에서 IBM의 상표입니다.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 ConnectDB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational
Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

다음 용어는 기타 회사의 상표 또는 등록상표입니다.

Microsoft, Windows 및 Windows NT는 Microsoft Corporation의 상표 또는 등록상표입니다.

Java 또는 모든 Java 관련 상표와 로고 및 Solaris는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.

Tivoli 및 NetView는 미국 또는 기타 국가에서 사용되는 Tivoli Systems Inc.의 상표입니다.

UNIX는 미국 또는 기타 국가에서 X/Open Company Limited가 독점적인 사용권을 가진 등록상표입니다.

두 개의 별표(**)가 붙은 기타 회사 이름, 제품 이름 또는 서비스 이름은 해당 회사의 상표이거나 서비스표입니다.

색인

[아]

[자]

[차]

[카]

[타]

[파]

[하]

A

B

C

D

E

F

H

J

L

M

N

O

P

R

S

U

W

IBM에 문의

기술적인 문제가 발생한 경우에는 DB2 고객 지원 센터에 문의하기 전에 *문제점 해결 안내서*에서 제안한 조치를 검토하고 실행해 보십시오. 이것은 DB2 고객 지원 부서로 하여금 사용자를 보다 더 잘 지원할 수 있도록 사용자가 모을 수 있는 정보를 제공합니다.

DB2 Universal Database 제품에 대한 정보나 주문은 지방 사무소의 IBM 담당자나 공인 IBM 소프트웨어 재판매업자에게 문의하십시오.

미국에 사시는 분은 다음 번호 중 하나를 선택하여 전화하십시오.

- 고객 지원을 받으려면, 1-800-237-5511.
- 사용 가능한 서비스 옵션을 알려면, 1-888-426-4343.

제품 정보

미국에 사시는 분은 다음 번호 중 하나를 선택하여 전화하십시오.

- 제품 주문이나 일반 정보를 얻으려면 1-800-IBM-CALL(1-800-426-2255) 또는 1-800-3IBM-OS2(1-800-342-6672).
- 책에 대한 주문은, 1-800-879-2755.

http://www.ibm.com/software/data/

DB2 World Wide Web 페이지에서는 새로운 소식, 제품 설명, 교육 일정 등에 관한 현재 DB2 정보를 제공합니다.

http://www.ibm.com/software/data/db2/library/

DB2 제품 및 서비스 기술 라이브러리는 빈도 높은 질문(FAQ), 수정사항(fixes), 책 및 최신 DB2 기술 정보에 대한 액세스를 제공합니다.

주: 이러한 정보는 영어로만 제공됩니다.

http://www.elink.ibmlink.ibm.com/pbl/pbl/

여기에서는 책을 웹 사이트에서 주문할 수 있는 방법을 제공합니다.

http://www.ibm.com/education/certify/

IBM 웹 사이트에서 기술 전문 인증 프로그램은 DB2를 포함하여 다른 IBM 제품의 기술 전문 인증 테스트 정보를 제공합니다.

ftp.software.ibm.com

anonymous로 로그온하십시오. /ps/products/db2 디렉토리에서 DB2와 많은 관련 제품에 관한 데이터, 수정사항, 도구 등을 찾을 수 있습니다.

comp.databases.ibm-db2, bit.listserv.db2-l

이러한 인터넷 뉴스 그룹으로 사용자는 DB2 제품에 대한 자신의 사용 경험을 토론할 수 있습니다.

Compuserve에서 GO IBMDB2

이 명령을 입력하여 IBM DB2 계열 포럼을 액세스하십시오. 모든 DB2 제품이 이러한 포럼을 통해 지원됩니다.

미국 외 지역에서 IBM에 연락하는 방법에 관한 정보는 IBM Software Support Handbook의 Appendix A를 참조하십시오. 이 문서에 액세스하려면, 웹 사이트 http://www.ibm.com/support/로 가서 페이지 맨 밑에 있는 IBM Software Support Handbook 링크를 누르십시오.

주: 일부 국가에서는 IBM 공인 딜러는 IBM 지원 센터 대신 해당 딜러 지원 부서에 연락해야 합니다.

IBM 한글 지원에 관한 설문

FAX : (02) 3787-0123

보내 주시는 의견은 더 나은 고객 지원 체제를 위한 귀중한 자료가 됩니다.
독자 여러분의 좋은 의견을 기다립니다.

책 제목: IBM® DB2® Universal Database
응용프로그램 빌드 안내서
버전 7

책 번호: SA30-0991-01

성 명		직위/담당업무	
회 사 명		부 서 명	
주 소			
전화번호		팩스번호	
전자우편 주소			
사용중인 시스템	o 중대형 서버 o UNIX 서버 o PC 및 PC 서버		

1. IBM에서 제공하는 한글 책자와 영문 책자 중 어느 것을 더 좋아하십니까? 그 이유는 무엇입니까?
   ☐ 한글 책자 ☐ 영문 책자
   (이유:)

2. 본 책자와 해당 소프트웨어에서 사용된 한글 용어에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

3. 본 책자와 해당 소프트웨어에서 번역 품질에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

4. 본 책자의 인쇄 상태에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

5. 한글 소프트웨어 및 책자가 지원되는 분야에 대해 귀하는 어떻게 생각하십니까?
   ☐ 한글 책자를 늘려야 함 ☐ 현재 수준으로 만족
   ☐ 그다지 필요성을 느끼지 않음

6. IBM은 인쇄물 형식(hardcopy)과 화면 형식(softcopy)의 두 종류로 책자를 제공합니다. 어느 형식을 더 좋아하십니까?
   ☐ 인쇄물 형식(hardcopy) ☐ 화면 형식(softcopy) ☐ 둘 다

☜ IBM 한글 지원 서비스에 대해 기타 제안사항이 있으시면 적어주십시오.

☺ 설문에 답해 주셔서 감사합니다.
귀하의 의견은 저희에게 매우 소중한 것이며, 고객 여러분들께 보다 좋은 제품을 제공해 드리기 위해 최선을 다하겠습니다.

SA30-0991-01