IBM® DB2® Universal Database

XML Extender 관리 및 프로그래밍

버전 7

IBM® DB2® Universal Database

XML Extender 관리 및 프로그래밍

버전 7

이 책과 이 책이 지원하는 제품을 사용하기 전에 319 페이지의 『주의사항』의 일반 정보를 반드시 읽으십시오.

목차

표

이 책에 관하여

이 절에서는 다음 정보에 대해 설명합니다.

- 『이 책의 사용자』
- x 페이지의 『이 책의 사용법』
- xii 페이지의 『강조표시 규약』
- xiii 페이지의 『구문 도표 읽는 방법』
- xv 페이지의 『관련 정보』

이 책의 사용자

이 책은 다음과 같은 독자들을 위한 책입니다.

- DB2 응용프로그램에서 XML 데이터에 대해 작업하고 XML 개념에 익숙한 독자. 이 책의 독자는 XML 및 DB2에 대한 일반 지식을 가지고 있어야 합니다. XML 및 관련 주제에 대해 좀 더 알려면, 다음 웹 사이트를 참조하십시오.

  http://www.w3c.org/XML

  DB2에 대해 좀더 알려면 다음 웹 사이트를 참조하십시오.

  http://www.ibm.com/software/data/db2/library
- DB2 관리 개념, 도구 및 기술에 익숙한 DB2 데이터베이스 관리자.
- SQL 및 DB2 응용프로그램에 사용될 수 있는 하나 이상의 프로그래밍 언어에 익숙한 DB2 응용프로그램 프로그래머.

이 책의 현재 버전 구입 방법

XML Extender 웹 사이트에서 이 책의 최신 버전을 구할 수 있습니다.

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

이 책의 사용법

이 책은 다음과 같이 구성되어 있습니다.

제1부. 소개

이 부분에서는 XML Extender 개요 및 비지니스 응용프로그램에서 이를 사용하는 방법을 제공합니다. 구동하고 실행하는 데 도움이 되는 시작 시나리오가 들어 있습니다.

제2부. 관리

이 부분에서는 XML 데이터용으로 DB2 데이터베이스를 준비하고 유지보수하는 방법에 대해 설명합니다. XML 데이터가 들어있는 DB2 데이터베이스를 관리해야 하는 경우 이 부분을 읽으십시오.

제3부. 프로그래밍

이 부분에서는 XML 데이터 관리 방법에 대해 설명합니다. XML 응용프로그램에서 XML 데이터에 액세스하고 이에 대해 조작해야 하는 경우 이 부분을 읽으십시오.

제4부. 참조

이 부분에서는 XML Extender 관리 명령, 사용자 정의 유형, 사용자 정의 함수 및 저장 프로시듀어의 사용 방법에 대해 설명합니다. 또한 XML Extender가 실행하는 메시지 및 코드도 나열합니다. XML Extender 개념 및 타스크에 익숙하지만, UDT(사용자 정의 유형), UDF(사용자 정의 함수), 명령, 메시지, 메타데이터 테이블, 제어 테이블 또는 코드에 대한 정보가 필요하면 이 부분을 읽으십시오.

제5부. 부록

부록에서는 문서 액세스 정의를 위한 DTD, 예제 샘플과 시작 시나리오, 그리고 기타 IBM XML 제품에 대해 설명합니다.

DB2 UDB 버전 Fixpak 2의 신규 내용

이 문서에는 다음에 대한 신규 또는 재기술된 정보가 포함되어 있습니다.

- 관리 마법사 설정 및 시작
- 갱신 UDF가 XML 문서를 수정하는 방법
- XMLClob UDF가 결과를 리턴하는 방법
- 저장 프로시듀어에 대해 CLOB 한계를 증가시키는 방법
- DAD 요구사항 변경:
  - 신규 XML 문서 작성시 스타일 시트 지시사항 처리 포함.
  - 하나의 테이블만 지정한 경우 첫 번째 RDB 노드에 대해 누락되거나 빈 조건 사용함.
- JDBC를 사용하여 매개변수 표시문자를 유형 변환하는 방법
- 코드 페이지에 대한 작업:
  - 법적 코드화 선언
  - 변환 가정
  - 변환 시나리오
- XML Extender 매개변수 한계 부록

변경된 정보는 수직 변경 막대, 『I』로 표시됩니다.

이번 fixpak에 대한 모든 결함 수정에 대해 readme 파일을 참조하십시오.

일반 질문에 대한 수정 사항 및 해결책등의 추가 정보에 대한 FAQ 및 알려진 문제점 목록에 대해 다음의 XML Extender 웹 사이트 지원 페이지를 참조하십시오. http://www.ibm.com/software/data/db2/extenders/xmlext/support.html

DB2 UDB 버전 7 Information Center에 이 책 포함

XML Extender 문서는 다음 단계를 거쳐 DB2 Information Center에 포함됩니다.

- 사용자 언어로 된 XML 제품을 설치하기 위해 DOC 서브디렉토리에서 이 책에 대한 HTML 파일을 복사하여 DB2 UDB 문서 디렉토리의 해당 언어 서브디렉토리에 설치하십시오.

  Windows의 경우, Information Center 디렉토리는 sqllib\doc\html\db2sx입니다.

  UNIX의 경우, Information Center 디렉토리는 *install directory*/doc/html/db2sx입니다.
- Information Center를 재시작하면 이 책이 **Books** 탭에 포함됩니다.

강조표시 규약

이 책에서는 다음 규약을 사용합니다.

굵은체	굵은 텍스트는 다음을 나타냅니다. • 명령 • 필드 이름 • 메뉴 이름 • 누름 버튼
기울임꼴	기울임꼴 텍스트는 다음을 나타냅니다. • 값으로 대체될 변수 매개변수 • 강조된 단어 • 용어집 용어의 처음 사용
대문자	대문자는 다음을 나타냅니다. • 데이터 유형 • 컬럼 이름 • 테이블 이름
예	예 텍스트는 다음을 나타냅니다. • 시스템 메시지 • 입력한 값 • 코딩 예

- 디렉토리 이름
- 파일 이름
- 경로 이름

구문 도표 읽는 방법

이 책 전체에서, 명령 및 SQL문의 구문은 구문 도표를 사용하여 설명합니다.

다음과 같이 구문 도표를 읽으십시오.

- 구문 도표는 왼쪽에서 오른쪽으로, 위에서 아래로, 줄의 경로를 따라 읽습니다.

  ▶▶── 기호는 명령문 시작을 나타냅니다.

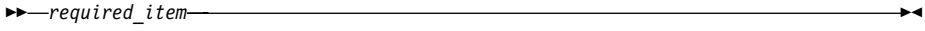
  ──▶ 기호는 명령문 구문이 다음 줄에서 계속되는 것을 나타냅니다.

  ▶── 기호는 명령문이 이전 줄로부터 계속되는 것을 나타냅니다.

  ──▶◀ 기호는 명령문 끝을 나타냅니다.

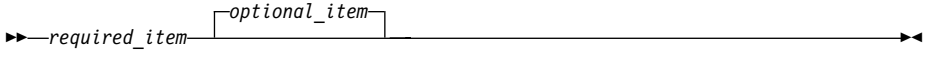
  완전한 문장이 아닌 구문 단위 도표는 ▶── 기호로 시작되고 ──▶ 기호로 끝납니다.

- 필수 항목은 가로줄에 나타납니다(주요 경로).



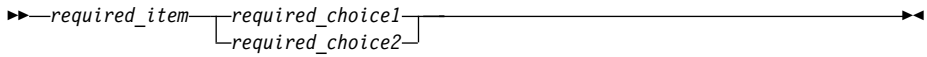
- 옵션 항목은 주요 경로 아래에 나타납니다.

  선택적 항목이 기본 경로 위에 표시되면, 그 항목은 명령문 실행에 아무런 영향이 없으며 단지 읽기 쉽도록 하기 위해서만 사용됩니다.



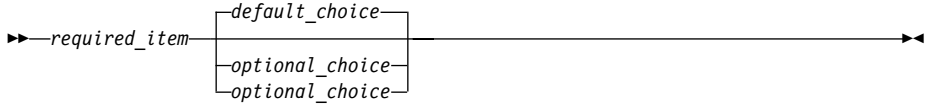
- 둘 이상의 항목에서 선택할 수 있으면, 이들은 수직의 스택으로 나타납니다.

  한 항목을 *반드시* 선택해야 하는 경우, 스택의 한 항목이 주요 경로에 나타납니다.



항목 중 하나를 선택하는 것이 선택적인 경우, 전체 스택이 기본 경로 아래에 나타납니다.

항목 중 하나가 기본값이면, 이것은 기본 경로 위에 나타나고, 나머지는 아래에 표시됩니다.



- 기본 행 위에서 왼쪽으로 리턴되는 화살표는 반복될 수 있는 항목을 나타냅니다.

반복 화살표에 구두점이 포함되면, 반복되는 항목을 지정된 구두점으로 분리해야 합니다.

스택 위에 있는 반복 화살표는 스택에서 항목을 반복할 수 있음을 나타냅니다.

- 키워드는 대문자로 나타납니다(예를 들면 FROM). XML Extender에서, 키워드는 어느 문자로든 표시될 수 있습니다. 키워드가 아닌 용어는 소문자로 표시됩니다(예를 들면, *column-name*). 이들은 사용자 정의 이름이나 값을 나타냅니다.
- 구두점, 괄호, 산술 연산자 또는 기타 다른 기호가 나타나면, 구문의 일부로 이것을 입력해야 합니다.

관련 정보

다음 문서는 XML Extender 및 관련 제품을 사용할 때 유용할 수 있습니다.

문서	주문 번호	설명
Call Level Interface Guide and Reference	SC09-2950	이 책에서는 CLI를 사용하여 DB2 서버에 액세스하는 응용프로그램 작성법에 대해 설명합니다.
DB2 응용프로그램 개발 안내서	SA30-0992	이 책에서는 응용프로그램 개발 프로세스와 Embedded SQL 및 API를 사용하여 데이터베이스에 액세스하는 응용프로그램의 코딩, 컴파일 및 실행 방법에 대해 설명합니다.
DB2 Extender 페이지	N/A	이 페이지에는 Extender에 해당하는 기술뿐만 아니라, DB2 Extenders에 대한 정보도 들어 있습니다. DB2 Extenders 페이지의 웹 주소는 http://www.software.ibm.com/ data/db2/extenders입니다.
DB2 Universal Database SQL 참조서, 볼륨 1 및 2	• 볼륨 1: SA30-0997 • 볼륨 2: SA30-0998	이 책에서는 SQL 구문, 의미 및 언어 규칙에 대해 설명합니다. 또한 릴리스간 비호환성, 제품 한계 및 카탈로그 뷰에 대한 정보도 포함됩니다.
• *DB2 Universal Database 관리 안내서: 구현* • *DB2 Universal Database 관리 안내서: 성능* • *DB2 Universal Database 관리 안내서: 계획*	• 구현: SA30-0988-00 • 성능: SA30-0989-00 • 계획: SA30-0990-00	이 책에서는 DB2 데이터베이스 디자인, 구현 및 유지보수 방법에 대해 설명합니다.

문서	주문 번호	설명
DB2 Universal Database Image, Audio, and Video Extenders 관리 및 프로그래밍	SA30-1043	이 책에서는 이미지, 오디오 및 비디오 데이터에 대한 DB2 데이터베이스를 관리하는 방법을 설명합니다. 또한 Extender가 제공하는 API를 사용하여 이러한 유형의 데이터에 액세스하고 작업하는 방법에 대해 설명합니다.
DB2 Universal Database Text Extender 관리 및 프로그래밍	SA30-1044	이 책에서는 텍스트 데이터를 위한 DB2 데이터베이스 관리법에 대해 설명합니다. 또한 Extender가 제공하는 API를 사용하여 이러한 유형의 데이터에 액세스하고 작업하는 방법에 대해 설명합니다.

제1부 소개

이 부분에서는 XML Extender 개요 및 비지니스 응용프로그램에서 이를 사용하는 방법을 제공합니다.

제1장 XML Extender 소개

IBM® DB2® Extenders™ 계열에서는 일반 및 특수 데이터를 처리하기 위해 데이터 및 메타데이터 관리 솔루션을 제공합니다. XML Extender를 사용하면 IBM DB2 Universal Database(DB2 UDB)™의 강력한 기능과 XML의 유연성을 통합할 수 있습니다.

DB2의 XML Extender는 XML 문서를 저장 및 액세스하고, 기존의 관계형 데이터로부터 XML 문서를 생성하고, XML 문서를 관계형 데이터로 나누는(분해하여 태그가 해제된 요소 또는 속성 내용을 저장) 기능을 제공합니다. XML Extender는 새 데이터 유형, 기능 및 저장 프로시듀어를 제공하여 DB2에 있는 XML 데이터를 관리합니다.

XML Extender는 다음의 운영 체제에서도 사용 가능합니다.

- Windows NT
- AIX
- Sun Solaris
- Linux
- NUMA-Q

XML 문서

컴퓨터 업계에는 많은 응용프로그램이 있고, 이들 각각에는 장점과 단점이 있습니다. 오늘날 사용자에게는 자신의 특별한 타스크 요구에 맞는 최적의 응용프로그램을 선택할 수 있는 기회가 주어 집니다. 그러나 사용자는 여러 응용프로그램 사이에서 데이터를 공유하는 경향이 있으므로, 다른 응용프로그램으로 가져올 수 있는 다른 형식으로 데이터 복제, 변환, 가져가기 또는 저장하는 데 계속해서 어려움이 있습니다. 이것은 이러한 다수의 변환 프로세스가 데이터의 일부를 제거하는 경향이 있거나 최소한 사용자로 하여금 데이터가 일치하는지 확인하는 지루한 프로세스를 거치도록 요구하기 때문이며, 비지니스 응용프로그램에 있어서 심각한 문제점이 될 수 있습니다. 이것은 시간과 비용 모두를 낭비하는 일입니다.

오늘날, 이 문제점을 해결하는 방법중 하나는 응용프로그램 개발자가 데이터베이스 관리 시스템에 데이터를 저장하도록 *ODBC(Open Database Connectivity)* 응용프로그램을 작성하는 것입니다. 이 때부터 데이터는 다른 응용프로그램에 필요한 양식으로 조작되거나 표시될 수 있습니다. 데이터베이스 응용프로그램은 응용프로그램에 필요한 양식으로 데이터를 변환하도록 작성되어야 합니다. 그러나 응용프로그램은 빠르게 변경되며 구식이 됩니다. 데이터를 HTML로 변환하는 응용프로그램은 프리젠테이션 솔루션을 제공하지만, 표시된 데이터는 다른 목적에 실제로 사용될 수 없습니다. 프리젠테이션에서 데이터를 분할하는 다른 방법이 있는 경우, 이 방법은 응용프로그램 간 상호교환의 실제 양식으로 사용될 수 있을 것입니다.

XML은 이 문제점을 지적하기 위해 사용합니다. XML은 *eXtensible Markup Language*의 약자입니다. 언어 자체가 기업의 필요에 따라 자신의 언어를 작성할 수 있도록 하는 메타언어인 점에서 발전된 것입니다. XML을 사용하여 특별한 응용프로그램의 데이터뿐만 아니라 데이터 구조도 캡쳐합니다. XML은 단순한 교환 형식이 아닙니다. 그러나 XML은 데이터 교환을 위한 승인된 표준으로 사용됩니다. 이 표준에 따라 응용프로그램은 독점적인 형식을 사용하여 데이터를 변환할 필요없이 궁극적으로 데이터를 공유할 수 있습니다.

XML 응용프로그램

XML은 이제 데이터 교환을 위한 승인된 표준이므로, 많은 응용프로그램이 병합되어 이 장점을 이용하게 됩니다.

사용자가 특별한 프로젝트 관리 응용프로그램을 사용하고 있고 그 데이터 일부를 캘린더 응용프로그램과 공유하려 한다고 가정합니다. XML을 사용하면 쉽게 해결할 수 있습니다. 오늘날 서로 얽혀있는 세상에서, 응용프로그램 벤더는 응용프로그램에 빌드된 XML 교환 유틸리티를 제공하지 않으면 경쟁할 수 없습니다. 따라서 이 예에서, 프로젝트 관리 응용프로그램은 XML로 된 타스크를 가져갈 수 있으며, 이 타스크는 있는 그대로 캘린더 응용프로그램에 가져올 수 있습니다(정보가 승인된 DTD에 일치할 경우).

XML 및 DB2를 사용하는 이유

XML이 데이터 교환을 위한 표준 포맷을 제공하여 많은 문제점을 해결하는 경우라도, 여전히 극복해야 하는 많은 다른 문제점이 있습니다. 엔터프라이즈 데이터 응용프로그램을 빌드할 때, 다음과 같은 질문에 응답해야 합니다.

- 데이터를 복제하는 빈도
- 응용프로그램들이 공유해야 하는 정보 유형
- 필요한 정보를 찾는 데 소요되는 시간
- 새 항목 추가와 같이 모든 응용프로그램 사이에서 자동 데이터 교환을 트리거하는 특별한 조치를 수행하는 방법

이러한 종류의 문제점은 데이터베이스 관리 시스템에서만 지적될 수 있습니다. XML 정보와 메타정보를 직접 데이터베이스에 병합시켜, 특별한 목적을 위해 다른 응용프로그램이 필요로 하는 XML 결과를 좀 더 직접(보다 신속하게) 구할 수 있습니다. 이것이 XML Extender가 사용자에게 도움을 줄 수 있는 분야입니다. XML Extender를 사용하면 여러 XML 응용프로그램에서 DB2 기능의 장점을 이용할 수 있습니다.

DB2 데이터베이스에서 구조화된 XML 문서의 내용을 이용하면, 구조화된 XML 정보를 기존의 관계형 데이터와 결합할 수 있습니다. 응용프로그램에 따라, 전체 XML 문서들을 DB2에서 특수 사용자 정의 데이터 유형으로 저장할 것인지 여부를 선택할 수 있으며, 그렇지 않으면 XML 내용을 관계형 테이블에서 일반 데이터로서 맵핑할 수 있습니다. 특수 XML 데이터 유형의 경우, XML Extender는 DB2 UDB Text Extender에서 제공하는 구조화된 텍스트 검색 외에도, 다양한 데이터 유형의 XML 요소 또는 속성 값을 검색하는 기능을 추가합니다.

XML Extender를 사용하여 응용프로그램은 다음을 수행할 수 있습니다.

- 검색을 위해 원하는 XML 요소나 속성 값을 *부가 테이블*로 추출하는 동안 전체 XML 문서를 응용프로그램 테이블의 *컬럼 데이터*에 저장하거나 외부적으로는 지역 파일에 저장합니다. XML 컬럼 메소드를 사용하여 다음을 수행할 수 있습니다.
  - 부가 테이블로 추출되었거나 색인화된 SQL 일반 *데이터 유형*의 XML 요소나 속성에 대해 빠른 검색을 수행합니다.

  - *XML 요소*의 내용이나 *XML 속성*의 값을 갱신합니다.
  - SQL 조회를 사용하여 XML 요소나 속성을 동적으로 추출합니다.
  - 삽입 및 갱신하는 동안 XML 문서를 확인합니다.
  - Text Extender를 사용하여 구조화된 텍스트 검색을 수행합니다.
- XML 콜렉션 저장영역 및 액세스 메소드를 사용하여 하나 이상의 관계형 테이블을 사용하는 XML 문서의 내용을 작성하거나 분해합니다.

XML을 DB2에 통합

XML Extender가 제공하는 다음 기능은 사용자가 DB2를 사용하여 XML 데이터를 관리하고 이용하는 데 도움을 줍니다.

- 관계형 테이블에 있는 XML 데이터의 통합을 관리하는 데 도움을 주는 관리 도구.
- XML 데이터에 대한 저장영역 및 사용 메소드.
- XML 데이터의 유효성을 검증하는 데 사용된 DTD를 저장하기 위한 DTD 저장소: DTD_REF
- XML 문서를 관계형 데이터에 맵핑하기 위한 DAD(문서 액세스 정의) 파일이라는 맵핑 스키마

관리 도구

XML Extender 관리 도구를 사용하면 XML용 데이터베이스 및 테이블 컬럼을 사용할 수 있으며, XML 데이터를 DB2 관계형 구조로 맵핑할 수 있습니다. XML Extender는 관리 타스크를 수행할 응용프로그램을 개발하려는지 여부 또는 단순히 마법사를 사용하려는지 여부에 따라 사용자 편의를 위해 여러 관리 도구를 제공합니다. 다음 도구를 사용하여 XML Extender에 대한 관리 타스크를 완료할 수 있습니다.

- XML Extender 관리 마법사는 관리 타스크에 대한 그래픽 사용자 인터페이스를 제공합니다.
- **dxxadm** 명령은 관리 타스크에 대한 명령행 옵션을 제공합니다.
- XML Extender 관리 저장 프로시듀어는 관리 타스크에 대한 응용프로그램 개발 옵션을 제공합니다.

저장영역 및 액세스 메소드

XML Extender는 XML 문서를 DB2로 통합하기 위한 두 가지 저장영역 및 액세스 메소드, 즉 XML 컬럼 및 XML 콜렉션을 제공합니다. 이들 메소드는 매우 다른 용도로 쓰이지만 같은 응용프로그램 내에서 사용할 수 있습니다.

XML 컬럼

이 메소드는 본래의 XML 문서를 DB2에 저장하는 데 도움을 줍니다. XML 컬럼은 문서를 아카이브하는 데 유용합니다. 문서들은 XML에 사용할 수 있으며, 갱신, 검색 및 탐색될 수 있는 컬럼에 삽입됩니다. 요소 및 속성 데이터는 DB2 테이블(부가 테이블)에 맵핑될 수 있으며, 이는 구조화된 빠른 검색을 위해 색인화될 수 있습니다.

XML 콜렉션

이 메소드는 XML 문서 구조를 DB2 테이블에 맵핑하여 기존 DB2 데이터로부터 XML 문서를 작성하거나 XML 문서를 DB2 데이터에 분해(태그가 해제된 요소 또는 속성 내용을 저장)할 수 있게 도와줍니다. 이 메소드는 데이터 교환 응용프로그램에 유용하며, 특히 XML 문서의 내용이 자주 갱 신될 경우 특히 유용합니다.

DTD 저장소

XML Extender는 XML *DTD(문서 유형 정의) 저장*소를 제공하며, 이 저장소는 XML 요소 및 속성에 대한 일련의 선언입니다. 데이터베이스를 XML에 *사용할 수 있게* 되면, *DTD 참조 테이블(DTD_REF)*이 작성됩니다. 이 테이블의 각 행은 DTD와 함께 추가 메타데이터 정보를 나타냅니다. 사용자는 이 테이블에 액세스하여 자신의 DTD를 삽입할 수 있습니다. DTD_REF 테이블의 *DTD*는 XML 문서의 유효성을 검증하는 데 사용됩니다.

문서 액세스 정의(DAD)

사용자는 구조화된 XML 문서가 *문서 액세스 정의(DAD)*에서 처리되는 방법을 지정합니다. DAD 자체는 XML 형식화된 문서입니다. XML 컬럼이나 XML 콜렉션을 사용할 때 XML 문서 구조를 DB2 데이터베이스에 연결합니다. XML 콜렉션과는 대조적으로 XML컬럼 정의시 DAD의 구조가 다릅니다.

DAD 파일은 XML_USAGE 테이블을 사용하여 관리되고, XML에 대해 데이터베이스를 사용할 수 있는 경우에 작성됩니다.

XML 컬럼: 구조화된 문서 저장영역 및 검색

XML에는 문서 세트를 작성하는 데 필요한 모든 정보가 포함되어 있으므로, 현재 그대로 문서 구조를 저장하고 유지보수해야 하는 경우가 여러 번 있게 됩니다.

예를 들어 웹을 통해 기사를 제공하는 신문사라면, 발표된 기사의 아카이브를 유지하려 할 수 있습니다. 그러한 시나리오에서, XML Extender를 사용하면 XML 기사 전부 또는 일부를 DB2 테이블 컬럼에 저장할 수 있습니다. 이러한 유형의 XML 문서 저장영역을 그림1에서처럼 *XML 컬럼*이라고 합니다.

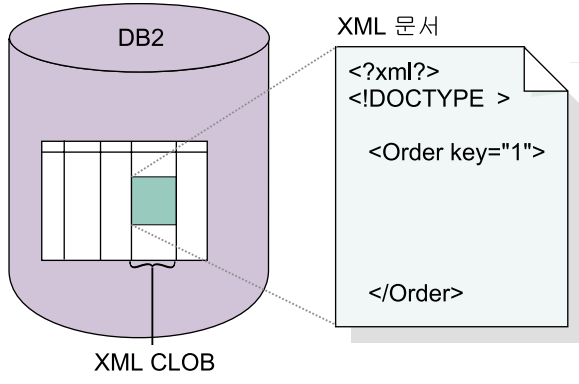


그림 1. 구조화된 XML 문서를 DB2 테이블 컬럼에 저장

XML Extender는 XML 컬럼에 사용할 사용자 정의 유형(UDT)을 다음과 같이 제공합니다.

- XMLVarchar
- XMLCLOB
- XMLFILE

모든 XML Extender UDT의 접두어는 db2xml이고, 이것은 DB2 XML Extender UDT의 *스키마 이름*입니다. 이러한 데이터 유형은 응용프로그램 테이블에서 XML 문서의 저장영역 유형을 식별하는 데 사용됩니다. XML Extender는 *계승*(legacy) 플랫 파일을 지원하므로, XML 문서를 DB2 내에 저장할 필요가 없습니다. XML 문서를 지역 파일 이름에 의해 지정된 대로, *지역 파일 시스템*에 파일로 저장할 수 있습니다.

DB2 XML Extender는 XML 요소 또는 속성값을 추출할 뿐만 아니라, XML 컬럼에서 XML 문서를 저장하고 검색하는 강력한 *사용자 정의 기능(UDF)*을 제공합니다. UDF는 데이터베이스 관리 시스템에 정의되어 SQL 조회시 참조될 수 있는 기능입니다. XML Extender는 다음과 같은 UDF 유형을 제공합니다.

- 저장영역: 본래의 XML 문서를 XML 데이터 유형의 XML 사용 컬럼에 저장합니다.
- 추출: XML 문서를 추출하거나 요소 및 속성이 기본 데이터 유형으로 지정된 값을 추출합니다.
- 갱신: 전체 XML 문서 또는 지정된 요소 및 속성 값을 갱신합니다.

추출 기능을 사용하면 일반 SQL 데이터 유형에 대해 강력한 검색을 수행할 수 있습니다. 이 외에도 XML Extender와 함께 DB2 UDB Text Extender를 사용하여 XML 문서에 있는 텍스트에 대해 구조 및 *전체 텍스트 검색*을 수행할 수 있습니다. 이러한 강력한 검색 기능은 예를 들어 뉴스 기사나 *EDI(Electronic Data Interchange)* 응용프로그램처럼 많은 양의 읽을 수 있는 텍스트를 발표하는 웹 사이트의 이용도를 향상시키는 데 사용될 수 있으며, 이 응용프로그램에는 자주 검색할 수 있는 요소 또는 속성이 있습니다.

모든 XML Extender의 UDF 접두어는 db2xml이며, 이것은 DB2 XML Extender UDF의 스키마 이름입니다. UDF는 XML UDT에 적용되며, 주로 XML 컬럼에 사용됩니다.

위치 경로

*위치 경로*는 XML 요소 또는 속성을 식별하는 일련의 *XML 태그*입니다. XML Extender는 위치 경로를 사용하여 XML 문서의 구조를 나타내며, 이것은 요소나 속성의 문맥을 나타냅니다. 단일 슬래시(/) 경로는 문맥이 전체 문서임을 나타냅니다. 위치 경로는 다음 경우에 사용됩니다.

- UDF를 추출할 때, 추출할 요소 및 속성을 식별하기 위해
- XML 컬럼에 대해 DAD에 있는 색인화 스킴을 정의할 때, XML 요소 또는 속성과 DB2 컬럼 사이의 맵핑 파일을 지정하기 위해
- 구조적 텍스트 탐색을 위해 Text Extender를 사용하는 경우 XML 요소나 속성을 식별하기 위해

10 페이지의 그림2에서는 XML 문서 구조의 위치 경로 및 그 관계 예를 보여줍니다.

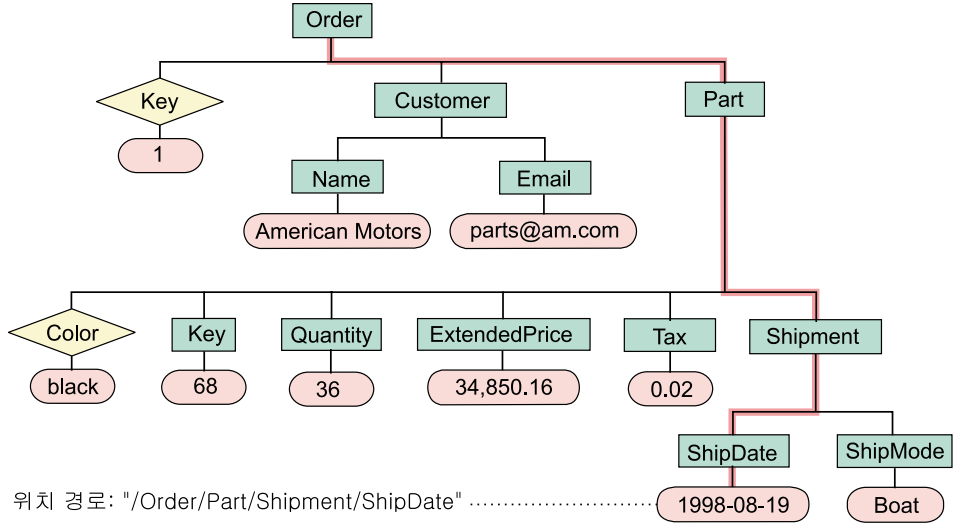


그림 2. 구조화된 XML 문서로서 DB2 테이블 컬럼에 문서 저장

위치 경로를 지정하기 위해 XML Extender는 *XSLT(XML Stylesheet Language Transformation)* 및 *XPath(XML Path Language)*의 서브세트를 사용합니다. 이 책에서는 *위치 경로*라는 용어를 사용하며, 이 용어는 XPath의 스펙에 정의됩니다. 위치 경로는 XML 요소 또는 속성을 나타내는 일련의 XML 태그입니다. 이 책에서도 *절대 위치 경로*의 XSLT 또는 XPath 축약 구문을 사용하며, 이 구문은 XPath 스펙에 지정됩니다. 절대 위치 경로는 오브젝트의 전체 경로 이름입니다.

XSLT는 XML 문서를 다른 XML 문서로 변환시키는 언어입니다. *XSL(XML Stylesheet Language)*의 일부로 사용되도록 설계되었으며, XML용 스타일 시트 언어입니다. XSLT 뿐만 아니라, XSL에는 포맷을 지정하기 위한 XML 용어집이 포함됩니다. XSL은 XSLT를 사용하며, 문서가 포맷팅 용어집을 사용하는 다른 XML 문서로 변환되는 방식에 대해 설명함으로써 XML 문서의 스타일을 지정합니다.

XPath는 XML 문서의 주소지정 파트를 위한 언어이며, XSLT가 사용하도록 디자인되었습니다. 모든 위치 경로는 XPath에 대해 정의된 구문을 사용하여 표현될 수 있습니다.

XSLT 및 XPath에 대해서는 다음 웹 페이지를 참조하십시오.

- XSLT의 경우, http://www.w3.org/TR/WD-xslt
- XPath의 경우, http://www.w3.org/TR/xpath

구문 및 제한사항에 대해서는 58 페이지의 『위치 경로』를 참조하십시오.

XML 컬럼 용어

이 절에서는 이 책에서 참조하는 XML 개념과 용어에 대해 설명합니다.

문서 액세스 정의(DAD)

XML 컬럼의 경우, 구조적 조회용으로 색인화된 DB2 부가 테이블에 대한 XML 문서 구조의 맵핑.

DXX_INSTALL

XML Extender 설치 디렉토리.

부가 테이블

XML Extender가 XML 컬럼에서 요소나 속성을 검색할 때 성능을 향상시키기 위해 작성하는 추가 테이블.

XML 컬럼

XML 데이터 유형의 DB2 컬럼을 사용하고 본래의 XML 문서를 사용 가능한 컬럼에 저장함으로써 XML 문서를 저장하고 액세스하는 메소드. 관리 도구 중 하나를 사용하여 XML에 사용 가능한 컬럼을 참조하기도 합니다.

XML 테이블

관리 도구 중 하나를 사용하여 XML에 사용한 하나 이상의 컬럼이 포함된 응용프로그램 테이블.

XML UDF

XML Extender가 제공하는 DB2 사용자 정의 함수.

XML UDT

XML Extender가 제공하는 DB2 사용자 정의 유형.

XML 콜렉션: 통합된 데이터 관리

일반 SQL 데이터는 가져오는 XML 문서에서 *분해*되거나, 나가는 XML 문서를 *작성*하는 데 사용됩니다. 데이터를 다른 응용프로그램과 공유해야 하는 경우, 입력 및 출력 XML 문서를 작성하고 분해하며, DB2의 관계 기능을 이용하는 데 필요한 만큼 데이터를 관리하고자 할 수 있습니다. 이러한 유형의 XML 문서 저장 영역을 *XML 콜렉션*이라고 합니다.

그림3에 XML 콜렉션의 예가 나와 있습니다.

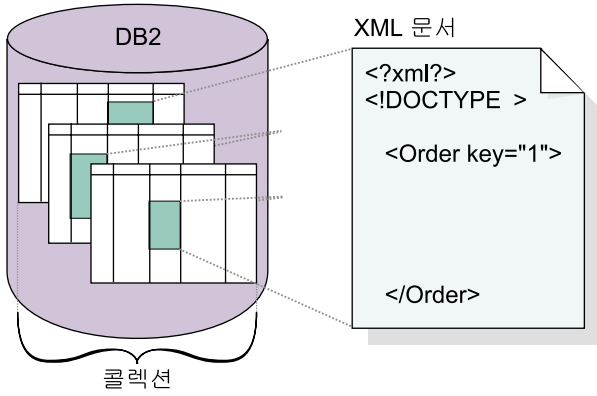


그림 3. DB2 테이블에 태그가 없는 데이터로 문서 저장

XML 콜렉션은 DAD 파일에 정의되며, 이는 요소 및 속성이 하나 이상의 관계형 테이블에 맵핑되는 방법을 지정합니다. 이를 사용 가능하게 하여 콜렉션 이름을 정의할 수 있으며, 저장 프로시듀어에 사용하여 XML 문서를 작성하거나 분해할 수 있습니다.

DAD 파일에서 콜렉션을 정의할 경우에는 SQL 맵핑 또는 RDB_node 맵핑의 두 가지 유형의 맵핑 스킴 중 하나를 사용합니다. SQL 맵핑은 SQL SELECT문을 사용하여 콜렉션에 대한 DB2 테이블과 조건을 정의합니다. RDB_node 맵핑은 XPath 기반 RDB_node를 사용하여 테이블, 컬럼 및 조건을 정의합니다.

저장 프로시듀어는 XML 문서를 작성하고 분해하기 위해 제공됩니다. 저장 프로시듀어는 db2xml 접두어를 사용하며, 이 접두어는 XML Extender의 *스키마 이름*입니다. XML 콜렉션에 대해 다음 저장 프로시듀어를 사용하십시오.

- 작성:
  - dxxGenXML(): XML 콜렉션에 DAD 파일을 사용하여 XML 문서를 작성합니다.
  - dxxRetrieveXML(): 사용 가능한 XML 콜렉션을 사용하여 XML 문서를 작성합니다.
- 분해:
  - dxxShredXML(): XML 콜렉션에 DAD 파일을 사용하여 XML 문서를 분해합니다.
  - dxxInsertXML(): 사용 가능한 XML 콜렉션을 사용하여 XML 문서를 분해합니다.

XML 콜렉션 용어

다음 용어는 XML Extender에 고유하며, 이 책에서 자주 사용되는 것입니다.

작성 DAD 파일에서 정의된 대로 기존의 관계형 데이터에서 XML 문서를 생성합니다.

분해 DAD 파일에서 정의된 대로 XML 문서를 태그가 없는 관계형 데이터로 저장합니다.

문서 액세스 정의(DAD)
XML 콜렉션의 경우, XML 문서 작성 또는 분해를 위해 XML 문서 구조를 DB2 데이터 구조에 맵핑

DXX_INSTALL
XML Extender 설치 디렉토리.

XML 콜렉션
관계형 테이블 세트를 사용하여 XML 데이터를 저장하고 액세스하는 메소드. 태그가 없는 데이터는 XML 문서로 작성되거나 XML 문서에서 분해될 수 있습니다. 또한 XML 문서가 작성 또는 분해되는 테이블 세트를 말합니다.

XML 저장 프로시듀어
XML 문서를 작성하거나 분해하기 위한 저장 프로시듀어.

제2장 XML Extender 시작하기

이 장에서는 XML Extender를 사용하여 응용프로그램의 XML 데이터를 액세스하고 수정 과정을 시작하는 방법을 보여줍니다. 제공된 자습서 연습에 따라, 제공된 샘플 데이터를 사용하여 데이터베이스를 설정하고, SQL 데이터를 XML 문서로 맵핑하고, XML 문서를 데이터베이스에 저장한 다음, XML 문서에서 데이터를 검색하고 추출할 수 있습니다.

관리 연습에서는, 관리 명령이 있는 DB2 명령 창을 사용합니다. 이러한 타스크는 XML Extender 관리 마법사를 사용하여 수행할 수 있으며, 이 책에서도 이에 대해 설명합니다. XML 데이터 관리 연습에서는, XML Extender 제공 UDF 및 저장 프로시듀어를 사용합니다. 이 책 나머지에 있는 예제 중 대부분은 이 장에서 사용된 샘플 데이터에 대해 설명하는 것입니다.

필수: 이 장의 연습을 완료하려면, DB2 UDB 버전 6.1 이상이 설치되어 있어야 합니다. 버전 6.1을 사용하는 경우에는 Fixpak 2를 반드시 설치해야 합니다. 또한, 이 연습의 모든 단계는 Windows NT®의 사용을 전제로 합니다.

연습은 다음과 같습니다.

- 본래의 XML 문서를 DB2 테이블 컬럼에 저장
  - 문서 및 자주 검색될 XML 요소와 속성을 저장할 XML UDT 플랜
  - 데이터베이스 및 테이블 설정
  - XML에 사용할 수 있도록 데이터베이스 설정
  - DTD를 DTD 저장소 테이블로 삽입
  - XML 컬럼용 DAD 준비
  - XML 유형의 기존 테이블로 컬럼 추가
  - XML에 사용할 수 있도록 새 컬럼 설정
  - 부가 테이블에 색인 작성
  - XML 컬럼에 XML 문서 저장
  - XML Extender UDF를 사용하여 XML 컬럼 검색

- 기존 데이터에서 XML 문서를 작성
  - XML 문서의 데이터 구조 플랜
  - 데이터베이스 및 테이블 설정
  - XML에 사용할 수 있도록 데이터베이스 설정
  - XML 콜렉션용 문서 액세스 정의(DAD) 파일 준비
  - 기존 데이터에서 XML 문서 작성
  - 데이터베이스에서 XML 문서 검색
- 데이터베이스 정리

연습에 대한 시나리오

이 연습에서, 사용자는 자동차와 트럭을 자동차 대리점에 판매하는 ACME Auto Direct사에서 일합니다. 사용자에게는 두 가지 타스크가 제공됩니다. 첫 번째, 사용자는 판매 부서에 의한 조회를 위해 SALES_DB 데이터베이스에 주문을 아카이브할 수 있는 시스템을 설정합니다. 두 번째, 기존의 구매 주문 데이터베이스 SALES_DB에서 정보를 확보하고 이로부터 XML 문서에 저장되는 키 정보를 추출합니다.

연습: XML 컬럼에 XML 문서 저장

시나리오

서비스 부서의 판매 데이터를 아카이브하는 타스크가 주어졌습니다. 이 데이터는 같은 DTD를 사용하는 XML 문서에 저장되어 있습니다. 서비스 부서는 고객의 요청이나 불만을 처리할 때 이들 XML 문서를 사용합니다.

서비스 부서는 XML 문서에 대해 권장되는 구조를 제공하며, 가장 자주 조회될 것으로 예상되는 요소 데이터를 지정합니다. 이 부서는 XML 문서를 SALES_DB 데이터베이스 내의 SALES_TAB 테이블에 저장하고, 이를 신속하게 검색할 수 있기를 바랄 것입니다. SALES_DB 테이블은 각 판매에 대한 데이터가 있는 두 개의 컬럼을 포함하고, XML 문서를 포함할 세번째 컬럼을 포함합니다. 이 컬럼을 ORDER라고 합니다.

자주 조회될 XML 요소 및 속성과 XML 문서를 저장할 XML 데이터 유형도 결정합니다. 그런 다음, XML용 SALES_DB 데이터베이스를 설정하고, SALES_TAB 테이블을 작성하며, ORDER 컬럼을 사용 가능하게 하여 본래의 문서를 DB2에 저장할 수 있도록 합니다. 또한 유효성 검증을 위해 XML 문서의 DTD를 삽입한 후, 이 문서를 XMLVARCHAR 데이터 유형으로 저장합니다. 이 컬럼이 사용 가능하면, 문서의 구조적 검색을 위해 색인화될 부가 테이블을 문서 액세스 정의(DAD) 파일, 즉, 부가 테이블의 구조를 지정하는 XML 문서에 정의합니다. DAD 파일, DTD 및 XML 문서의 샘플을 보려면 295 페이지의 『부록B. 샘플』을 참조하십시오.

SALES_TAB은 표1에 설명되어 있습니다.

표 1. SALES_TAB 테이블

컬럼 이름	데이터 유형
INVOICE_NUM	CHAR(6) NOT NULL PRIMARY KEY
SALES_PERSON	VARCHAR(20)
ORDER	XMLVARCHAR

계획

문서를 저장하기 위해 XML Extender에 대한 작업을 시작하기 전에, 문서 검색 방법을 결정할 수 있도록 먼저 XML 문서의 구조를 이해해야 합니다. 문서 검색 방법을 계획할 때는 다음을 결정해야 합니다.

- XML 문서를 저장할 XML 사용자 정의 유형
- 성능 향상을 위해 색인화될 수 있도록, 서비스 부서에서 자주 검색할 XML 요소 및 속성

다음 절에서는 이러한 의사 결정 방법에 대해 설명합니다.

XML 문서 구조

이 연습에 대한 XML 문서 구조는 주문 번호가 최상위 레벨로 구조화되고 고객, 파트 및 운송 정보가 그 다음 레벨로 구조화된 특정 주문에 대한 정보를 사용합니다. XML 문서는 18 페이지의 그림4에 설명되어 있습니다.

이 연습은 또한 XML 문서 구조를 이해하고 유효성을 검증하는 데 사용할 수 있는 샘플 DTD도 제공합니다. 295 페이지의 『부록B. 샘플』에서 DTD 파일을 볼 수 있습니다. 그림4에 있는 구조와 일치합니다.

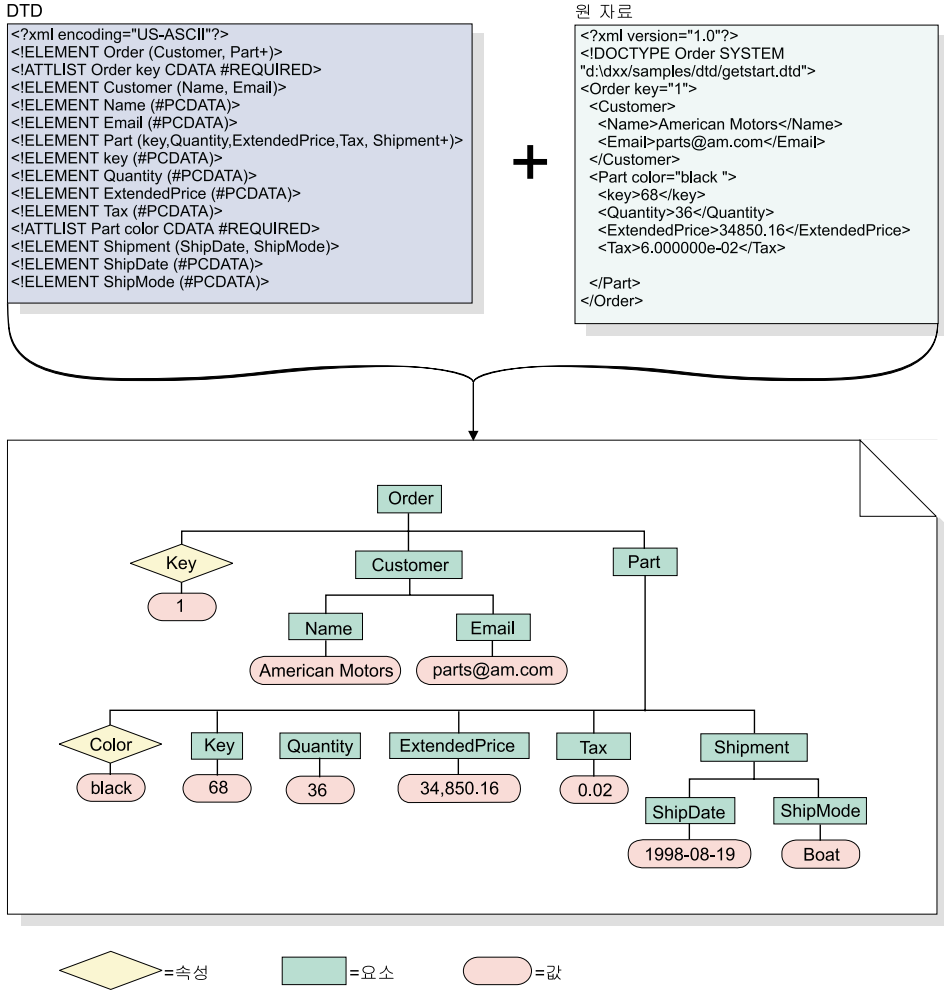


그림 4. DTD 및 XML 문서의 계층적 구조

XML 컬럼의 XML 데이터 유형 판별

XML Extender는 사용자가 XML 문서를 보유하기 위한 컬럼을 정의하는 XML 사용자 데이터 유형을 제공합니다. 이들 데이터 유형은 다음과 같습니다.

- XMLVarchar: DB2에 저장되는 작은 문서용
- XMLCLOB: DB2에 저장되는 큰 문서용
- XMLFILE: DB2 외부에 저장되는 문서용

이 연습에서는 작은 문서를 DB2에 저장하므로, XMLVarchar 데이터 유형을 사용합니다.

검색할 요소 및 속성 판별

XML 문서 구조 및 응용프로그램의 요구사항을 이해하면, 검색할 요소 및 속성을 판별할 수 있으며, 가장 자주 검색되거나 추출되며 또는 조회 비용이 가장 많이 드는 요소 및 속성을 판별할 수 있습니다. 서비스 부서에서는 주문 번호, 고객 이름, 가격 및 주문의 운송 일자를 가장 자주 조회한다고 나타내며, 이 검색의 신속한 성능을 필요로 합니다. 이 정보는 XML 문서 구조의 요소 및 속성에 들어 있습니다. 표2는 각 요소 및 속성의 위치 경로를 설명합니다.

표 2. 검색할 요소 및 속성 판별

데이터	위치 경로
주문 키	/Order/@key
고객	/Order/Customer/Name
가격	/Order/Part/ExtendedPrice
운송 일자	/Order/Part/Shipment/ShipDate

부가 테이블에 XML 문서 맵핑

이 자습서에서는 XML 문서를 DB2에 저장하는 데 사용되는 XML 컬럼에 대한 DAD 파일을 작성하게 됩니다. 또한 XML 요소 및 속성 내용을 검색 성능을 향상시키는 색인화에 사용되는 DB2 부가 테이블에 맵핑합니다. 마지막 절에는 검색될 요소와 속성이 나와 있습니다. 이 절에는 이들 요소 및 속성 값을 색인화될 수 있는 DB2 테이블에 맵핑하는 방법에 대해 자세히 나와 있습니다.

검색할 요소 및 속성을 식별한 후, 이들이 부가 테이블에 구성되는 방법과 몇 개의 테이블이 있어야 하며, 어느 컬럼이 어느 테이블에 있어야 하는지를 결정합니다. 일반적으로, 유사한 정보를 같은 테이블에 넣음으로써 부가 테이블을 구성합니다. 이 구조는 요소의 위치 경로가 문서 내에서 두 번 이상 반복될 수 있는지에 따라서도 결정됩니다. 예를 들어, 문서에서 파트 요소는 여러 번 반복될 수 있으므

로, 가격과 날짜 요소도 여러 번 발생할 수 있습니다. 여러 번 발생할 수 있는 요소는 그 고유 테이블 내에 있어야 합니다.

이 외에도 요소나 속성 값이 사용할 DB2 기본 유형을 판별해야 합니다. 일반적으로 이 유형은 데이터 형식으로 쉽게 판별됩니다. 데이터가 텍스트인 경우에는 VARCHAR를 선택하십시오. 데이터가 정수인 경우에는 INTEGER를 선택하십시오. 또는, 데이터가 날짜이고 범위 검색을 수행하는 경우에는 DATE를 선택하십시오.

이 자습서에서, 요소 및 속성은 다음의 부가 테이블에 맵핑되어 있습니다.

ORDER_SIDE_TAB

컬럼 이름	데이터 유형	위치 경로	다중 발생 여부
ORDER_KEY	INTEGER	/Order/@key	아니오
CUSTOMER	VARCHAR(16)	/Order/Customer/Name	아니오

PART_SIDE_TAB

컬럼 이름	데이터 유형	위치 경로	다중 발생 여부
PRICE	DECIMAL(10,2)	/Order/Part/ExtendedPrice	예

SHIP_SIDE_TAB

컬럼 이름	데이터 유형	위치 경로	다중 발생 여부
DATE	DATE	/Order/Part/Shipment/ShipDate	예

이 연습에서는 사용자 환경을 설정하기 위해 사용하는 스크립트 세트를 제공합니다. 이러한 스크립트는 *DXX_INSTALL*\samples\cmd 디렉토리(여기서 *DXX_INSTALL*은 XML Extender가 설치되어 있는 드라이브 및 디렉토리입니다. 예: c:\dxx\samples\cmd)에 있으며 다음과 같습니다.

getstart_db.cmd

데이터베이스를 작성하고 4개의 테이블을 채웁니다.

getstart_prep.cmd

XML Extender 저장 프로시듀어 및 DB2 CLI에 데이터베이스를 바인드합니다.

getstart_insertDTD.cmd

XML 문서의 유효성을 검증하는 데 사용되는 DTD를 XML 컬럼에 삽입합니다.

getstart_createTabCol.cmd

XML 사용 컬럼이 있는 응용프로그램 테이블을 작성합니다.

getstart_alterTabCol.cmd

XML에 사용 가능한 컬럼을 추가하여 응용프로그램 테이블을 변경합니다.

getstart_enableCol.cmd

XML 컬럼을 사용 가능하게 합니다.

getstart_createIndex.cmd

XML 컬럼에 대해 부가 테이블상에 색인을 작성합니다.

getstart_insertXML.cmd

XML 문서를 XML 컬럼에 삽입합니다.

getstart_queryCol.cmd

응용프로그램 테이블상에서 SELECT문을 수행하고 XML 문서를 리턴합니다.

getstart_clean.cmd

자습서 환경을 정리합니다.

설정

이 절에서는 XML Extender와 함께 사용할 데이터베이스를 준비합니다. 다음을 수행하게 됩니다.

1. 데이터베이스를 작성합니다.
2. 데이터베이스를 사용 가능하게 합니다.

데이터베이스 작성

이 절에서, 데이터베이스를 설정하는 명령을 사용합니다. 이 명령은 샘플 데이터베이스를 작성하고, 이에 연결한 다음 데이터가 들어갈 테이블을 작성한 뒤 데이터를 삽입합니다.

데이터베이스를 작성하려면, 다음과 같이 하십시오.

1. *DXX_INSTALL*\samples\cmd 디렉토리로 변경하십시오. 여기서 *DXX_INSTALL*은 XML Extender가 설치된 드라이브 및 디렉토리입니다. 이 연습에서는 c:\dxx 디렉토리가 사용됩니다. 드라이브와 디렉토리가 다르면 이들 값을 변경하십시오.
2. Windows NT 시작 메뉴에서 DB2 명령 창을 열거나 Windows NT 명령 프롬프트에서 다음 명령을 입력하십시오.

   ```
   DB2CMD
   ```

3. DB2 명령 창에서 다음 명령을 실행하십시오.

   ```
   getstart_db.cmd
   ```

데이터베이스 사용 가능

데이터베이스에 XML 정보를 저장하려면, XML Extender용으로 이를 사용할 수 있어야 합니다. XML에 사용할 수 있도록 데이터베이스를 설정하면, XML Extender는 다음을 수행합니다.

- 사용자 정의 유형(UDT) 및 사용자 정의 기능(UDF) 모두를 작성합니다.
- 제어 테이블을 작성하고 XML Extender에 필요한 메타데이터로 채웁니다.
- db2xml 스키마를 작성하고 필요한 특권을 지정합니다.

XML용으로 데이터베이스를 사용 가능하게 하려면, 다음과 같이 하십시오.

DB2 명령 창에서 다음 스크립트를 실행하여 SALES_DB 데이터베이스를 사용 가능하게 하십시오.

```
getstart_prep.cmd
```

이 스크립트는 데이터베이스를 XML Extender 저장 프로시듀어 및 DB2 CLI에 바인드합니다. 또한 데이터베이스를 사용 가능하게 하는 **dxxadm** 명령 옵션을 실행합니다.

```
dxxadm enable_db SALES_DB
```

XML 컬럼 작성

XML Extender는 전체 XML 문서를 XML 컬럼이라는 데이터베이스에 저장 및 액세스하는 메소드를 제공합니다. XML 컬럼 메소드를 사용하여 XMLFILE 유형을 사용하는 문서를 저장하고, 부가 테이블에서 컬럼을 색인화 및 XML 문서를

조회하거나 검색할 수 있습니다. 이 저장영역 메소드는 특히 문서가 자주 갱신되지 않는 아카이브 응용프로그램에 유용합니다. 이 자습서의 목적은 제공된 XML 문서를 XML 컬럼에 저장하는 것입니다.

이 연습에서는 SALES_TAB 테이블에 문서를 저장합니다. 문서를 저장하기 위해 다음을 수행합니다.

1. XML 문서용 DTD를 DTD 참조 테이블 DTD_REF에 삽입하십시오.
2. XML 문서 위치 및 구조적인 검색을 위한 부가 테이블을 지정하는 DAD 파일을 준비하십시오.
3. SALES_TAB 테이블에 XML 사용자 정의 유형이 XMLVARCHAR인 컬럼을 추가합니다.
4. XML에 사용 가능하도록 컬럼을 설정합니다.
5. 구조적 검색을 위해 부가 테이블 색인화하십시오.
6. 사용자 정의 함수를 사용하여 문서를 저장합니다. 이 함수는 XML Extender가 제공합니다.

DTD 저장소에 DTD 저장

XML 컬럼에 있는 XML 데이터의 유효성을 검증하기 위해 DTD를 사용할 수 있습니다. XML Extender는 XML 사용 데이터베이스에서 DTD_REF라는 테이블을 작성합니다. 이 테이블은 DTD 참조라고 하며, DTD를 저장하는 데 사용할 수 있습니다. XML 문서의 유효성을 검증하기로 결정한 경우, 이 DTD를 이 저장소에 저장해야 합니다. 이 자습서에 대한 DTD는 `c:\dxx\samples\dtd\getstart.dtd`입니다.

DTD를 삽입하려면, 다음과 같이 하십시오.

DB2 명령 창에서 다음의 SQL INSERT 명령을 하나의 행으로 입력하십시오.

```
DB2 CONNECT TO SALES_DB
DB2 INSERT into db2xml.dtd_ref values('c:\dxx\samples\dtd\getstart.dtd',
    db2xml.XMLClobFromFile('c:\dxx\samples\dtd\getstart.dtd'), 0, 'user1',
    'user1', 'user1')
```

다음 명령 파일을 실행하여 DTD를 삽입할 수도 있습니다.

```
getstart_insertDTD.cmd
```

DAD 파일 준비

XML 컬럼에 대한 DAD 파일의 구조는 간단합니다. 저장영역 모드가 XML 컬럼이라고 지정하고, 색인 지정을 위해 테이블과 컬럼을 정의합니다.

다음 단계에서, DAD에 있는 요소는 *태그*를 말하며, XML 문서 구조의 요소는 요소를 말합니다. 작성할 DAD 파일과 유사한 DAD 파일 샘플이 `c:\dxx\samples\dad\getstart_xcolumn.dad`에 있습니다. 다음 단계에서 생성되는 파일과는 약간의 차이가 있습니다. 연습용으로 이를 사용하는 경우에는 파일 경로가 사용자 환경의 파일 경로와 다를 수 있습니다. <validation> 값이 YES가 아닌 NO로 설정되어 있는지 확인하십시오.

DAD 파일을 준비하려면, 다음과 같이 하십시오.

1. 텍스트 편집기를 열고 파일 이름을 `getstart_xcolumn.dad`로 지정하십시오.

   DAD 파일에서 사용되는 모든 태그는 대소문자를 구별한다는 점에 유의하십시오.

2. XML 및 Doctype 선언을 사용하여 DAD 머리글을 작성하십시오.

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
   ```

   DAD 파일은 XML 문서이며 XML 선언이 필요합니다.

3. 열기 및 닫기 <DAD></DAD> 태그를 삽입하십시오. 기타 모든 태그들은 이 태그 안에 있습니다.

4. 열기 및 닫기 <DTDID></DTDID> 태그를 삽입하여 DAD를 XML 문서 DTD와 연결하는 DTD ID 식별자를 지정하고, 클라이언트에서 DTD 위치를 지정할 수 있습니다.

   ```
   <dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   ```

   이 문자열이 23 페이지의 『DTD 저장소에 DTD 저장』에서 DTD를 DTD 참조 테이블에 삽입할 때 최초 매개변수 값으로 사용된 값과 일치하는지 확인하십시오. 예를 들어, 사용자가 다른 시스템 드라이브에서 작업하는 경우 DTDID에 사용된 경로는 위의 문자열과 다를 수 있습니다.

5. XML Extender가 DTD 저장소 테이블에 삽입했던 DTD를 사용하여 XML 문서 구조의 유효성을 검증하는 것을 나타내려면 열기 및 닫기 <validation> </validation> 태그를 지정하십시오.

```
<validation>YES</validation>
```

<validation> 값은 대문자여야 합니다.

6. 저장영역 메소드를 XML 컬럼으로 정의하려면 열기 및 닫기 <Xcolumn></Xcolumn> 태그를 삽입하십시오. 메소드는 XML 데이터가 XML 컬럼에 저장되도록 정의합니다.

```
<Xcolumn>
   </Xcolumn>
```

7. 생성될 각 부가 테이블에 대해 열기 및 닫기 <table></table> 태그를 삽입하십시오.

```
<Xcolumn>
<table name="order_side_tab">
     </table>
<table name="part_side_tab">
     </table>
<table name="ship_side_tab">
     </table>
   </Xcolumn>
```

8. 부가 테이블에 포함시킬 각 컬럼에 대해 열기 및 닫기 <column></column> 태그를 삽입하십시오. 각각의 <column> 태그에는 4개의 속성이 있습니다.
   - **name:** 컬럼 이름
   - **type:** 컬럼의 데이터 유형
   - **path:** XPath 구문을 사용하는, XML 문서에 있는 해당 요소의 위치 경로. 위치 경로 구문에 대해서는 58 페이지의 『위치 경로』를 참조하십시오.
   - **multi-occurrence:** 요소의 위치 경로가 XML 문서 구조에서 두 번 이상 발생할 수 있는지 표시.

```
<Xcolumn>
<table name="order_side_tab">
  <column name="order_key"
            type="integer"
           path="/Order/@key"
         multi_occurrence="NO"/>
  <column name="customer"
```

```
            type="varchar(50)"
             path="/Order/Customer/Name"
            multi_occurrence="NO"/>
        </table>
    <table name="part_side_tab">
      <column name="price"
            type="decimal(10,2)"
            path="/Order/Part/ExtendedPrice"
            multi_occurrence="YES"/>
        </table>
    <table name="ship_side_tab">
      <column name="date"
            type="DATE"
            path="/Order/Part/Shipment/ShipDate"
            multi_occurrence="YES"/>
        </table>
     </Xcolumn>
```

9. 마지막 </table> 태그 다음에 종료 </Xcolumn>이 있는지 확인하십시오.
10. 마지막 </Xcolumn> 태그 다음에 종료 </DAD>가 있는지 확인하십시오.
11. 파일을 getstart_xcolumn.dad로 저장하십시오.

방금 작성한 파일을 샘플 파일 c:\dxx\samples\dad\getstart_xcolumn.dad와 비교할 수 있습니다. 이 파일은 XML 컬럼을 사용 가능하게 하고 부가 테이블을 작성하는 데 필요한 DAD 파일의 작업 사본입니다. 샘플 파일에는 정상적으로 실행되도록 환경에 맞게 변경해야 하는 경로 명령문이 들어 있습니다.

SALES_TAB 테이블 작성

이 절에서는 SALES_TAB 테이블을 작성합니다. 초기에, 여기에는 주문을 위한 영업 정보와 함께 두 개의 컬럼이 있습니다.

테이블을 작성하려면, 다음과 같이 하십시오.

DB2 명령 창에서 다음 CREATE TABLE 문을 입력하십시오.

```
DB2 CONNECT TO SALES_DB
DB2 CREATE TABLE SALES_TAB(INVOICE_NUM CHAR(6) NOT NULL PRIMARY KEY,
        SALES_PERSON VARCHAR(20))
```

또는, 다음 명령 파일을 실행하여 테이블을 작성할 수 있습니다.

```
getstart_createTabCol.cmd
```

XML 유형의 컬럼 추가

이제, 새 컬럼을 SALES_TAB 테이블에 추가하십시오. 이 컬럼에는 이전에 생성되었던 원래의 XML 문서가 들어가며, XML UDT의 것이어야 합니다. XML Extender는 187 페이지의 『제8장 XML Extender 사용자 정의 유형』에서 설명한 것처럼 다중 데이터 유형을 제공합니다. 이 자습서에서는 문서를 XMLVARCHAR로 저장합니다.

XML 유형의 컬럼을 추가하려면, 다음과 같이 하십시오.

DB2 명령 창에서 다음 SQL 문을 입력하십시오.

```
DB2 ALTER TABLE SALES_TAB ADD ORDER DB2XML.XMLVARCHAR
```

또는, 다음 명령 파일을 실행하여 테이블을 변경할 수 있습니다.

```
getstart_alterTabCol.cmd
```

XML 컬럼 사용 가능

XML 유형의 컬럼을 작성한 뒤, 이를 XML Extender용으로 사용 가능하게 합니다. 컬럼이 사용 가능하면, XML Extender는 DAD 파일을 읽고 부가 테이블을 작성합니다. 컬럼이 사용 가능하기 전에 다음을 수행해야 합니다.

- XML 문서를 포함하는 XML 컬럼과 부가 테이블 컬럼의 기본 뷰를 작성할지 여부를 결정하십시오. XML 문서를 조회할 때 기본 뷰를 지정할 수 있습니다. 이 연습에서는, `-v` 매개변수를 사용하여 뷰를 지정합니다.
- 기본 키를 *ROOT ID*, 응용프로그램 테이블의 기본 키의 컬럼 이름 및 모든 부가 테이블을 응용프로그램 테이블과 연결하는 고유 식별자로 지정하려는지 여부를 결정하십시오. 기본 키를 지정하지 않으면, XML Extender가 DXXROOT_ID 컬럼을 응용프로그램 테이블과 부가 테이블에 추가합니다. ROOT_ID 컬럼은 응용프로그램과 부가 테이블을 함께 묶어, XML 문서가 갱신될 때 XML Extender가 자동으로 부가 테이블을 갱신할 수 있게 합니다. 이 연습에서는, `-r` 매개변수로 명령의 기본 키 이름(INVOICE_NUM)을 지정합니다. XML Extender는 지정된 컬럼을 ROOT_ID로 사용하고, 컬럼을 부가 테이블에 추가합니다.
- 테이블 공간을 지정할 것인지 기본 테이블 공간을 사용할 것인지 결정하십시오. 이 연습에서는 기본 테이블 공간을 사용합니다.

XML에 컬럼을 사용할 수 있게 하려면, 다음과 같이 하십시오.

DB2 명령 창에서 다음 명령을 입력하십시오.

```
dxxadm enable_column SALES_DB SALES_TAB ORDER GETSTART_XCOLUMN.DAD
        -v SALES_ORDER_VIEW -r INVOICE_NUM
```

또는, 다음 명령 파일을 실행하여 XML용으로 컬럼을 사용 가능하게 할 수 있습니다.

```
getstart_enableCol.cmd
```

XML Extender는 INVOICE_NUM 컬럼이 있는 부가 테이블을 작성하고 기본 뷰를 작성합니다.

중요사항: 어떤 방법으로든 부가 테이블을 수정하지 마십시오. XML Extender가 제공하는 UDF를 사용하여 XML 문서를 갱신해야만 합니다. XML Extender는 사용자가 XML 컬럼에 있는 XML 문서를 갱신할 때 자동으로 부가 테이블을 갱신합니다.

컬럼 및 부가 테이블 뷰

XML 컬럼을 사용 가능하게 설정할 때, XML 컬럼과 부가 테이블의 뷰를 작성했습니다. XML 컬럼에 대해 작업할 때 이 뷰를 사용할 수 있습니다.

XML 컬럼 및 부가 테이블 컬럼을 보려면 다음과 같이 하십시오.

DB2 명령 창에서 다음 SQL SELECT 문을 입력하십시오.

```
DB2 SELECT * FROM SALES_ORDER_VIEW
```

이 뷰는 getstart_xcolumn.dad 파일에 지정된 대로 부가 테이블에 있는 컬럼을 표시합니다.

부가 테이블에서 색인 작성

부가 테이블에서 색인을 작성하면 XML 문서의 신속한 구조적 검색을 수행할 수 있습니다. 이 단계에서는 XML 컬럼, ORDER를 사용 가능하게 할 때 작성된 부가 테이블에 있는 키 컬럼에서 색인을 작성합니다. 서비스 부서에서는 부서원이 가장 자주 조회할 것 같은 컬럼을 지정했습니다. 29 페이지의 표3은 색인화할 컬럼을 설명합니다.

표 3. 색인화될 부가 테이블 컬럼

컬럼	부가 테이블
ORDER_KEY	ORDER_SIDE_TAB
CUSTOMER	ORDER_SIDE_TAB
PRICE	PART_SIDE_TAB
DATE	SHIP_SIDE_TAB

부가 테이블을 색인화하려면, 다음과 같이 하십시오.

DB2 명령 창에서 다음 SQL 명령을 입력하십시오.

```
DB2 CREATE INDEX KEY_IDX
      ON ORDER_SIDE_TAB(ORDER_KEY)
DB2 CREATE INDEX CUSTOMER_IDX
      ON ORDER_SIDE_TAB(CUSTOMER)
DB2 CREATE INDEX PRICE_IDX
      ON PART_SIDE_TAB(PRICE)
DB2 CREATE INDEX DATE_IDX
      ON SHIP_SIDE_TAB(DATE)
```

또는, 다음 명령 파일을 실행하여 색인을 작성할 수 있습니다.

```
getstart_createIndex.cmd
```

XML 문서 저장

이제 XML 문서를 포함할 수 있는 컬럼을 사용할 수 있으므로, XML Extender가 제공하는 기능을 사용하여 문서를 저장할 수 있습니다. 데이터를 XML 컬럼에 저장할 때, 기본 유형 변환 기능이나 XML Extender UDF를 사용합니다. 기본 유형 VARCHAR인 오브젝트를 XML UDT XMLVARCHAR 컬럼에 저장하게 되므로, 기본 유형변환 기능을 사용합니다. 기본 유형 변환 기능 및 XML Extender 제공 UDF에 대해서는 134 페이지의 『데이터 저장』을 참조하십시오.

XML 문서를 저장하려면, 다음과 같이 하십시오.

중요사항: XML 문서 `c:\dxx\samples\xml\getstart.xml`을 여십시오. DTD를 DTD 저장소에 삽입할 때, DOCTYPE의 파일 경로가 DAD에 지정된 DTD ID와 일치하는지 확인하십시오. db2xml.DTD_REF 테이블을 조회하고 DAD 파일

의 DTDID 요소를 점검하여 일치하는지 검증할 수 있습니다. 기본값과 다른 드라이브 및 디렉토리 구조를 사용하고 있다면, DOCTYPE 선언의 경로를 변경해야 할 수도 있습니다.

DB2 명령 창에서 다음 SQL INSERT 명령을 입력하십시오.

```
DB2 INSERT INTO SALES_TAB (INVOICE_NUM, SALES_PERSON, ORDER) VALUES('123456',
   'Sriram Srinivasan', db2xml.XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

XML 문서를 저장할 때, XML Extender는 자동으로 부가 테이블을 갱신합니다.

또는, 다음 명령 파일을 실행하여 문서를 저장할 수 있습니다.

```
getstart_insertXML.cmd
```

테이블이 갱신되었는지 확인하려면, DB2 명령 창에서 테이블에 대해 다음 SELECT 문을 실행하십시오.

```
DB2 SELECT * FROM SALES_TAB
DB2 SELECT * FROM PART_SIDE_TAB
DB2 SELECT * FROM ORDER_SIDE_TAB
DB2 SELECT * FROM SHIP_SIDE_TAB
```

XML 문서 검색

부가 테이블에 대한 직접 조회로 XML 문서를 검색할 수 있습니다. 이 단계에서, 가격이 2500.00 이상인 모든 주문을 검색합니다.

부가 테이블을 조회하려면, 다음과 같이 하십시오.

DB2 명령 창에서 다음 SELECT 문을 입력하십시오.

```
DB2 "SELECT DISTINCT SALES_PERSON FROM SALES_TAB S, PART_SIDE_TAB P
   WHERE PRICE > 2500.00 AND
   S.INVOICE_NUM=P.INVOICE_NUM"
```

결과 세트에서는 가격이 2500.00 이상인 항목을 판매한 영업 사원의 이름을 표시해야 합니다.

또는, 다음 명령 파일을 실행하여 문서를 검색할 수 있습니다.

```
getstart_queryCol.cmd
```

XML 문서를 DB2 테이블에 저장하는 데 대한 시작하기 자습서를 완료했습니다. 이 책에 있는 많은 예들은 이 연습을 기초로 합니다.

연습: XML 문서 작성

연습 시나리오

기존의 구매 주문 데이터베이스인 SALES_DB에 있는 정보를 사용하여 XML 문서에 저장할 키 정보를 추출하는 타스크가 주어집니다. 그러면 서비스 부서에서는 고객 요청 및 불만을 처리할 때 이들 XML 문서를 사용합니다. 서비스 부서는 포함시킬 특정 데이터를 요청하고, XML 문서에 대한 권장 구조를 제공합니다.

기존 데이터를 사용하여 이 테이블에 있는 데이터로부터 XML 문서 `getstart.xml`을 작성합니다.

또한 관련 테이블의 컬럼을, 구매 주문 레코드를 제공하는 XML 문서 구조로 맵핑하는 DAD 파일을 계획하고 작성합니다. 이 문서는 여러 테이블에서 작성되었으므로, 이러한 테이블을 XML 구조 및 DTD와 연결시켜 XML 콜렉션을 작성합니다. 이 DTD를 사용하여 XML 문서의 구조를 정의합니다. 이 DTD를 사용하여 응용프로그램에 작성된 XML 문서의 유효성을 검증할 수도 있습니다.

XML 문서에 대한 기존 데이터베이스 데이터에 대해서는 다음 테이블에서 설명합니다. *이탤릭체*로 된 컬럼 이름은 XML 문서 구조에서 서비스 부서가 요청한 컬럼입니다.

ORDER_TAB

컬럼 이름	데이터 유형
ORDER_KEY	INTEGER
CUSTOMER	VARCHAR(16)
CUSTOMER_NAME	VARCHAR(16)
CUSTOMER_EMAIL	VARCHAR(16)

PART_TAB

컬럼 이름	데이터 유형
PART_KEY	INTEGER
COLOR	CHAR(6)
QUANTITY	INTEGER
PRICE	DECIMAL(10,2)
TAX	REAL
ORDER_KEY	INTEGER

SHIP_TAB

컬럼 이름	데이터 유형
DATE	DATE
MODE	CHAR(6)
COMMENT	VARCHAR(128)
PART_KEY	INTEGER

계획

XML Extender에 대해 작업하여 문서를 작성하기 전에, XML 문서 구조와, 이들이 데이터베이스 데이터 구조에 해당되는 방식을 결정해야 합니다. 이 절에서는 서비스 부서가 요청한 XML 문서 구조 및 XML 문서의 구조를 정의하는 데 사용하는 DTD의 개요와 이 문서를 문서 처리에 사용된 데이터가 들어 있는 컬럼으로 맵핑하는 방법을 제공합니다.

문서 구조 판별

XML 문서 구조는 여러 테이블에서 특정 주문 정보를 선택하여 주문을 위한 XML 문서를 작성합니다. 이러한 테이블 각각에는 주문에 관련된 정보가 들어 있으며, 키 컬럼을 바탕으로 조인될 수 있습니다. 서비스 부서에서는 주문 번호를 최상위 레벨로 구성한 후 고객, 파트 및 운송 정보로 구조화된 문서가 필요합니다. 문서 구조는 직관적이고 유연성이 있어야 하며, 요소가 문서 구조보다는 데이터에 대해 설명해야 합니다(예를 들어 고객 이름은 문단이 아닌 『customer』라는 요소에 있어야 합니다). 이러한 요청에 따라 DTD 및 XML 문서의 계층적 구조는 33 페이지의 그림5에 설명된 구조와 같아야 합니다.

문서 구조를 설계한 후, XML 문서의 구조를 설명하는 DTD를 작성해야 합니다. 이 연습에서는 XML 문서 및 DTD를 제공합니다. 295 페이지의 『부록B. 샘플』에서 DTD 파일을 볼 수 있습니다. 그림5에 있는 구조와 일치하는지 볼 수 있습니다.

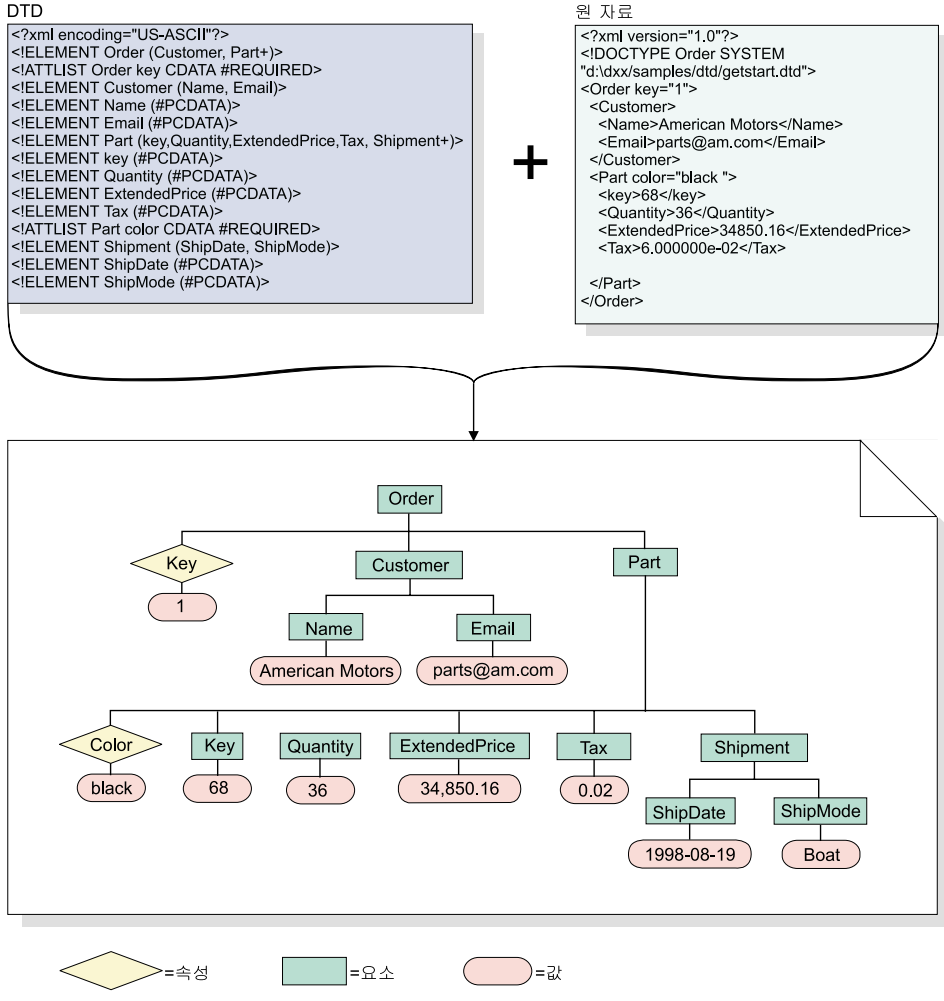


그림 5. DTD 및 XML 문서의 계층적 구조

XML 문서 및 데이터베이스 관계의 맵핑

구조를 디자인하고 DTD를 작성하고 난 뒤에는, 문서 구조를 요소 및 속성을 채우는 데 사용할 DB2 테이블과 연관시키는 방법에 대해 표시해야 합니다. 그림6에서처럼 계층적 구조를 관계형 테이블에 있는 특정 컬럼으로 맵핑할 수 있습니다.

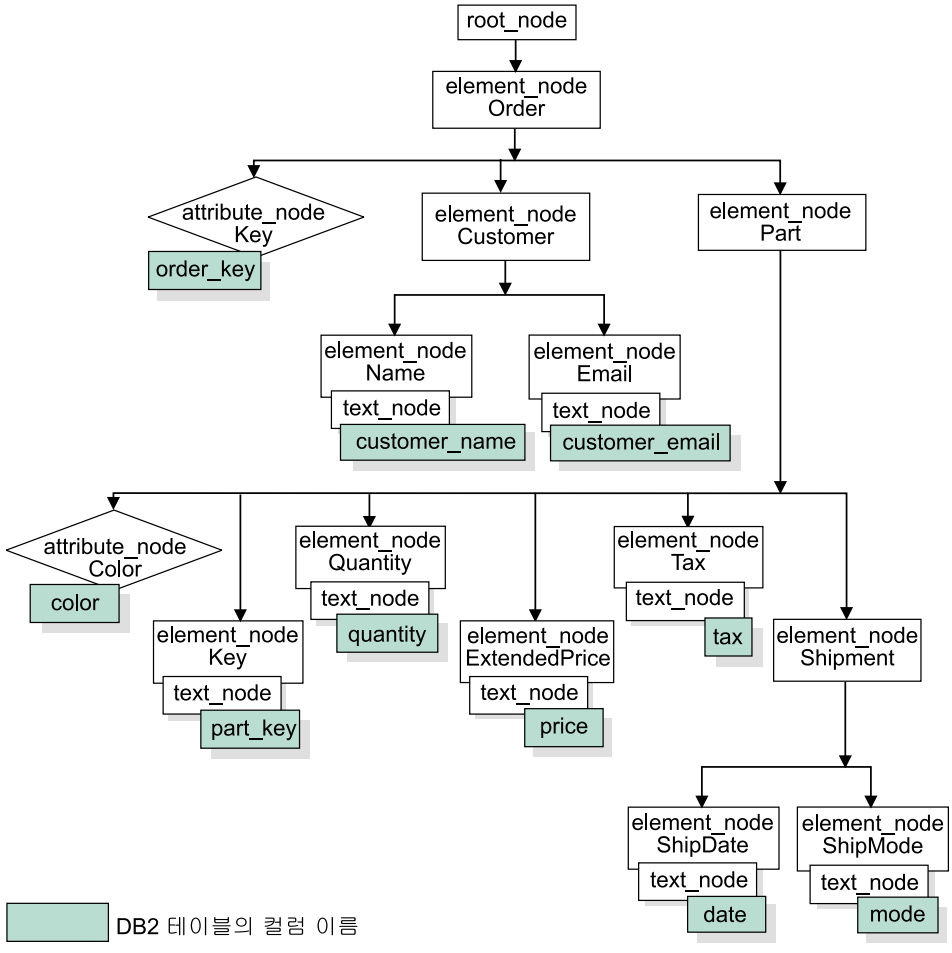


그림 6. 관계형 테이블 컬럼으로 맵핑된 XML 문서

이 관계 설명을 사용하여 관계형 데이터와 XML 문서 구조간의 관계를 정의하는 DAD 파일을 작성하십시오.

XML 콜렉션 DAD 파일을 작성하려면, 34 페이지의 그림6에서 설명한 것처럼 XML 문서가 데이터베이스 구조에 해당되는 방법을 이해해야 하며, 그렇게 되면 XML 문서 구조가 요소 및 속성 데이터를 추출하는 테이블 및 컬럼에 대해 설명할 수 있습니다. 이 정보를 사용하여 XML 콜렉션용 DAD 파일을 작성합니다.

이 연습에서는 사용자 환경을 설정하기 위해 사용하는 스크립트 세트를 제공합니다. 이러한 스크립트는 *DXX_INSTALL*\samples\cmd 디렉토리(여기서 *DXX_INSTALL*은 XML Extender가 설치되어 있는 드라이브 및 디렉토리입니다. 예: c:\dxx\samples\cmd)에 있으며 다음과 같습니다.

getstart_db.cmd
데이터베이스를 작성하고 4개의 테이블을 채웁니다.

getstart_prep.cmd
XML Extender 저장 프로시듀어 및 DB2 CLI에 데이터베이스를 바인드합니다.

getstart_stp.cmd
XML 콜렉션을 구성할 저장 프로시듀어를 실행합니다.

getstart_exportXML.cmd
응용프로그램에서 사용할 데이터베이스에서 XML 문서를 내보냅니다.

getstart_clean.cmd
자습서 환경을 정리합니다.

설정

데이터베이스 작성

이 절에서, 데이터베이스를 설정하는 명령을 사용합니다. 이 명령은 샘플 데이터베이스를 작성하고, 이에 연결한 다음 데이터가 들어갈 테이블을 작성한 뒤 데이터를 삽입합니다.

중요사항: XML 컬럼 연습을 완료했으나 환경을 정리하지 않은 경우, 이 단계를 건너 뛸 수 있습니다. SALES_DB 데이터베이스가 있는지 확인하십시오.

데이터베이스를 작성하려면, 다음과 같이 하십시오.

1. *DXX_INSTALL*\samples\cmd 디렉토리로 변경하십시오. 여기서 *DXX_INSTALL*은 XML Extender가 설치된 드라이브 및 디렉토리입니다. 이 연습에서는 c:\dxx 디렉토리가 사용됩니다. 드라이브와 디렉토리가 다르면 이들 값을 변경하십시오.
2. Windows NT 시작 메뉴에서 DB2 명령 창을 열거나 Windows NT 명령 프롬프트에서 다음 명령을 입력하십시오.

   ```
   DB2CMD
   ```

3. DB2 명령 창에서 다음 명령을 실행하십시오.

   ```
   getstart_db.cmd
   ```

데이터베이스 사용 가능

데이터베이스에 XML 정보를 저장하려면, XML Extender용으로 이를 사용할 수 있어야 합니다. XML에 사용할 수 있도록 데이터베이스를 설정하면, XML Extender는 다음을 수행합니다.

- 사용자 정의 유형(UDT) 및 사용자 정의 기능(UDF) 모두를 작성합니다.
- 제어 테이블을 작성하고 XML Extender에 필요한 메타데이터로 채웁니다.
- db2xml 스키마를 작성하고 필요한 특권을 지정합니다.

중요사항: XML 컬럼 연습을 완료했으나 환경을 정리하지 않은 경우, 이 단계를 건너 뛸 수 있습니다.

XML용으로 데이터베이스를 사용 가능하게 하려면, 다음과 같이 하십시오.

DB2 명령 창에서 다음 스크립트를 실행하여 SALES_DB 데이터베이스를 사용 가능하게 하십시오.

```
getstart_prep.cmd
```

이 스크립트는 데이터베이스를 XML Extender 저장 프로시듀어 및 DB2 CLI에 바인드합니다. 또한 데이터베이스를 사용 가능하게 하는 **dxxadm** 명령 옵션을 실행합니다.

```
dxxadm enable_db SALES_DB
```

XML 콜렉션 작성: DAD 파일 준비

데이터가 이미 여러 테이블에 있으므로, 이 테이블을 XML 문서와 연결하는 XML 콜렉션을 작성합니다. XML 콜렉션을 작성하려면 DAD 파일을 준비하여 콜렉션을 정의합니다.

32 페이지의 『계획』에서는 관계형 데이터베이스에서 데이터가 있는 컬럼과 테이블 데이터가 XML 문서로 구조화되는 방식을 판별했습니다. 이 절에서는, 테이블과 XML 문서 구조 사이의 관계를 지정하는 맵핑 스킴을 맵핑 파일에서 작성합니다.

다음 단계에서, DAD에 있는 요소는 *태그*를 말하며, XML 문서 구조의 요소는 요소를 말합니다. 작성된 DAD 파일과 유사한 DAD 파일 샘플이 `c:\dxx\samples\dad\getstart_xcollection.dad`에 있습니다. 이것은 다음 단계에서 생성하는 파일과 약간의 차이가 있습니다. 연습에 이를 사용할 경우, 파일 경로가 사용자 환경에 있는 경로와 다를 수 있습니다.

XML 문서를 작성하기 위해 DAD 파일을 작성하려면 다음과 같이 하십시오.

1. `c:\dxx\samples\cmd` 디렉토리에서 텍스트 편집기를 열고, `getstart_xcollection.dad` 파일을 작성하십시오.
2. 다음 텍스트를 사용하여 DAD 머리글을 작성하십시오.

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
   ```

   XML Extender에서는 사용자가 제품을 `c:\dxx`에 설치한 것으로 가정합니다. 이것이 올바르지 않으면, 이 값을 이 단계 및 다음 단계에서 이 제품을 설치하는 동안 지정했던 드라이브와 디렉토리로 변경하십시오.
3. <DAD></DAD> 태그를 삽입하십시오. 기타 모든 태그들은 이 태그 안에 있습니다.
4. XML Extender가 DTD 저장소 테이블에 삽입했던 DTD를 사용하여 XML 문서 구조를 검증하는지 여부를 나타내도록 <validation> </validation> 태그를 지정하십시오.

   ```
   <validation>NO</validation>
   ```

5. <Xcollection> </Xcollection> 태그를 사용하여 액세스 및 저장영역 메소드를 XML 콜렉션으로 정의하십시오. 액세스 및 저장영역 메소드는 XML 데이터가 DB2 테이블 콜렉션에 저장되도록 정의합니다.

```
<Xcollection>
  </Xcollection>
```

6. XML 콜렉션에 사용되는 테이블과 컬럼을 지정하는 SQL문을 지정합니다. 이 메소드를 SQL 맵핑이라고 하며, XML 문서 구조에 관계형 데이터를 맵핑시키는 두 가지 방법 중 하나입니다(맵핑 스킴에 대해서는 67 페이지의 『맵핑 스킴 유형』 참조). 다음 명령문을 입력하십시오.

```
    <SQL_stmt>
SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
price, tax, ship_id, date, mode from order_tab o, part_tab p,
table (select substr(char(timestamp(generate_unique())),16)
  as ship_id, date, mode, part_key from ship_tab) s
     WHERE o.order_key = 1 and
               p.price > 20000 and
           p.order_key = o.order_key and
           s.part_key = p.part_key
     ORDER BY order_key, part_key, ship_id
   </SQL_stmt>
```

이 SQL문은 SQL 맵핑을 사용할 때 다음 지침을 사용합니다. 문서 구조에 대해서는 34 페이지의 그림6을 참조하십시오.

- 컬럼은 XML 문서 구조의 계층 구조로, 하향식 순서로 지정됩니다. 예를 들어, 주문 및 고객 요소에 대한 컬럼은 첫번째이고 파트 요소는 두번째이며 운송은 세번째입니다.
- 엔터티에 대한 컬럼은 함께 그룹화되며, 각 그룹에는 오브젝트 ID 컬럼, ORDER_KEY, PART_KEY 및 SHIP_ID가 있습니다.
- 오브젝트 ID 컬럼은 각 그룹의 첫번째 컬럼입니다. 예를 들어, O.ORDER_KEY는 키 속성에 관한 컬럼에 우선하며, p.PART_KEY는 파트 요소에 대한 컬럼에 우선합니다.
- SHIP_TAB 테이블에는 단일 키 조건 컬럼이 없으므로, SHIP_ID 컬럼을 생성할 때 generate_unique DB2 내장 함수가 사용됩니다.
- 오브젝트 ID 컬럼은 ORDER BY문에 있는 내림차순으로 나열됩니다. ORDER BY에 있는 컬럼은 스키마 및 테이블 이름으로 규정되지 않아야 하며, SELECT절에 있는 컬럼 이름과 일치해야 합니다.

SQL문 작성시 요구사항은 69 페이지의 『맵핑 스킴 요구사항』을 참조하십시오.

7. 작성된 XML 문서에서 사용할 다음 프롤로그 정보를 추가하십시오.

```
<prolog>?xml version="1.0"?</prolog>
```

   모든 DAD 파일에서는 이 텍스트 그대로가 필요합니다.

8. 작성 중인 XML 문서에서 사용할 <doctype></doctype> 태그를 추가하십시오. <doctype> 태그에는 클라이언트에 저장된 DTD에 대한 경로가 포함됩니다.

```
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

9. <root_node></root_node> 태그를 사용하여 XML 문서의 루트 요소를 정의하십시오. root_node 내부에서는 XML 문서를 구성하는 요소와 속성을 지정합니다.

10. 다음 세 유형의 노드를 사용하여 XML 문서 구조를 DB2 관계형 테이블 구조로 맵핑하십시오.

    element_node

    XML 문서에 있는 요소를 지정합니다. Element_node에는 하위 element_node가 있을 수 있습니다.

    attribute_node

    XML 문서에서 요소의 속성을 지정합니다.

    text_node

    관계형 테이블에서 최하위 레벨 element_node에 있는 요소의 텍스트 내용과 컬럼 데이터를 지정합니다.

    이들 노드에 대해서는 63 페이지의 『DAD 파일』을 참조하십시오. 34 페이지의 그림6에서는 XML 문서 및 DB2 테이블 컬럼의 계층적 구조를 보여주며, 사용되는 노드 종류를 나타냅니다. 음영처리된 상자는 XML 문서를 작성하기 위해 데이터를 추출할 DB2 테이블 컬럼 이름을 나타냅니다.

    다음 단계에서는 한번에 한 유형씩 각 유형의 노드를 추가합니다.

    a. XML 문서에서 각 요소에 대해 <element_node> 태그를 정의합니다.

```
      <root_node>
         <element_node name="Order">
  <element_node name="Customer">
    <element_node name="Name">
     </element_node>
     <element_node name="Email">
     </element_node>
              </element_node>
  <element_node name="Part">
     <element_node name="key">
     </element_node>
    <element_node name="Quantity">
              </element_node>
     <element_node name="ExtendedPrice">
              </element_node>
     <element_node name="Tax">
              </element_node>
     <element_node name="Shipment" multi_occurrence="YES">
        <element_node name="ShipDate">
        </element_node>
        <element_node name="ShipMode">
        </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

<Shipment> 하위 요소가 multi_occurrence=″YES″ 속성을 지니는지 확인하십시오. 이 속성은 속성 없는 요소에 사용되며, 문서에서 반복됩니다. <Part> 요소는 이를 고유하게 만드는 색상 속성을 보유하므로, 다중 발생 속성을 사용하지 않습니다.

b. XML 문서에 있는 각 속성에 대해 <attribute_node> 태그를 정의합니다. 이러한 속성들은 element_node 내에 포함됩니다. 추가된 attribute_node는 굵은체로 강조표시됩니다.

```
      <root_node>
         <element_node name="Order">
  <attribute_node name="key">
  </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
     </element_node>
     <element_node names"Email">
     </element_node>
              </element_node>
```

```
  <element_node name="Part">
    <attribute_node name="color">
    </attribute_node>
     <element_node name="key">
     </element_node>
    <element_node name="Quantity">
              </element_node>
...
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

c. 최하위 레벨 element_node마다 <text_node> 태그를 정의하여, XML 요소에 문서 작성시 DB2에서 추출할 문자 데이터가 들어 있다는 것을 알립니다.

```
     <root_node>
        <element_node name="Order">
  <attribute_node name="key">
  </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
        <text_node>
      </text_node>
     </element_node>
    <element_node name="Email">
        <text_node>
      </text_node>
     </element_node>
              </element_node>
  <element_node name="Part">
    <attribute_node name="color">
    </attribute_node>
     <element_node name="key">
       <text_node>
       </text_node>
     </element_node>
    <element_node name="Quantity">
       <text_node>
      </text_node>
              </element_node>
    <element_node name="ExtendedPrice">
       <text_node>
      </text_node>
              </element_node>
    <element_node name="Tax">
        <text_node>
      </text_node>
```

```
            </element_node>
    <element_node name="Shipment" multi-occurrence="YES">
            <element_node name="ShipDate">
        <text_node>
        </text_node>
            </element_node>
            <element_node name="ShipMode">
        <text_node>
        </text_node>
            </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

d. 최하위 element_node의 경우 <column> 태그를 정의하십시오. 이러한 태그들은 XML 문서를 작성할 때 데이터를 추출할 컬럼을 지정하며, 보통 <attribute_node> 또는 <text_node> 태그 안에 있습니다. 여기에 정의된 컬럼들은 <SQL_stmt> SELECT절 내에 있어야 함을 기억하십시오.

```
     <root_node>
        <element_node name="Order">
  <attribute_node name="key">
    <column name="order_key"/>
  </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
      <text_node>
        <column name="customer_name"/>
      </text_node>
            </element_node>
    <element_node name="Email">
      <text_node>
        <column name="customer_email"/>
     </text_node>
            </element_node>
            </element_node>
  <element_node name="Part">
    <attribute_node name="color">
       <column name="color"/>
    </attribute_node>
     <element_node name="key">
      <text_node>
        <column name="part_key"/>
      </text_node>
    <element_node name="Quantity">
      <text_node>
```

```
        <column name="quantity"/>
      </text_node>
              </element_node>
    <element_node name="ExtendedPrice">
      <text_node>
        <column name="price"/>
      </text_node>
              </element_node>
    <element_node name="Tax">
      <text_node>
        <column name="tax"/>
      </text_node>
              </element_node>
    <element_node name="Shipment" multi-occurrence="YES">
              <element_node name="ShipDate">
        <text_node>
          <column name="date"/>
                 </text_node>
              </element_node>
              <element_node name="ShipMode">
        <text_node>
          <column name="mode"/>
                 </text_node>
              </element_node>
    </element_node> <!-- end Shipment -->
  </element_node> <!-- end Part -->
</element_node> <!-- end Order -->
</root_node>
```

11. 마지막 </element_node> 태그 다음에 종료 </root_node> 태그가 있는지 확인하십시오.
12. </root_node> 태그 다음에 종료 </Xcollection> 태그가 있는지 확인하십시오.
13. </Xcollection> 태그 다음에 종료 </DAD> 태그가 있는지 확인하십시오.
14. 파일을 getstart_xcollection.dad로 저장하십시오.

방금 작성한 파일을 샘플 파일 c:\dxx\samples\dad\getstart_xcollection.dad와 비교할 수 있습니다. 이 파일은 XML 문서를 작성하는 데 필요한 DAD 파일의 작업 사본입니다. 샘플 파일에는 정상적으로 실행되도록 환경에 맞게 변경해야 하는 경로 명령문이 들어 있습니다.

응용프로그램에서 문서를 작성하기 위해 XML 콜렉션을 자주 사용하는 경우, 콜렉션을 사용 가능하게 하여 콜렉션 이름을 정의할 수 있습니다. 콜렉션을 사용 가능하게 하면 이를 XML_USAGE 테이블에 등록하고 저장 프로시듀어 실행시 콜렉션 이름을 지정할 때(DAD 파일 이름이 아님) 성능을 향상시킬 수 있습니다. 이 연습에서는 콜렉션을 사용 가능하게 하지 않습니다. 콜렉션 사용에 대한 자세한 내용은 125 페이지의 『XML 콜렉션 사용』을 참조하십시오.

XML 문서 작성

이 단계에서는 dxxGenXML() 저장 프로시듀어를 사용하여 DAD 파일에서 지정한 XML 문서를 작성합니다. 이 저장 프로시듀어는 문서를 XMLVARCHAR UDT로서 리턴합니다.

XML 문서를 작성하려면 다음과 같이 하십시오.

1. DB2 명령 창에서, 다음 명령을 입력하여 저장 프로시듀어를 실행하십시오.

   ```
   getstart_stp.cmd
   ```

   XML 문서가 작성되어 RESULT_TAB 테이블에 저장됩니다.

   이 단계에서 사용할 수 있는 샘플 저장 프로시듀어를 다음 파일에서 볼 수 있습니다.

   - `c:\dxx\samples\c\tests2x.sqc`는 Embedded SQL을 사용하여 저장 프로시듀어를 호출하는 방법을 보여주고, `texts2x` 실행 파일을 생성합니다. 이 파일은 `getstart_stp.cmd`에서 사용됩니다.
   - `c:\dxx\samples\cli\sql2xml.c`는 CLI를 사용하여 저장 프로시듀어를 호출하는 방법을 보여줍니다.

2. XML Extender 검색 함수 Content()를 사용하여 테이블에서 파일로 XML 문서를 내보내십시오.

   ```
   DB2 CONNECT TO SALES_DB
   DB2 SELECT db2xml.Content(db2xml.xmlvarchar(doc),
      'c:\dxx\samples\cmd\getstart.xml') FROM RESULT_TAB
   ```

   또는, 다음 명령을 실행하여 파일을 내보내십시오.

   ```
   getstart_exportXML.cmd
   ```

이 연습은 DB2 저장 프로시듀어의 결과 세트 기능을 사용하여 하나 이상의 작성된 XML 문서를 가져오는 방법을 알려줌으로써, 사용자가 각 행을 사전 추출하여 각 문서를 가져올 수 있도록 합니다. 문서의 각 행을 가져옴으로써 이를 파일에 내보낼 수 있는 데, 이는 이 기능을 예시하는 가장 간단한 방법입니다. 데이터 사전 추출의 가장 효율적인 방법에 대해서는 c:\dxx\samples\cli의 CLI 예를 참조하십시오.

자습서 환경 정리

자습서 환경을 정리하려는 경우, getstart_clean.cmd 파일을 실행할 수 있습니다. 이 파일은

- XML 컬럼, ORDER를 사용할 수 없게 합니다.
- 자습서에서 작성된 테이블을 제거합니다.
- DTD 참조 테이블에서 DTD를 제거합니다.

이 명령 파일은 SALES_DT 데이터베이스를 사용 불가능하게 하거나 제거하지 않습니다. 데이터베이스는 계속 XML Extender에서 사용할 수 있습니다. 이 장에 있는 두 연습을 모두 완료하지 않으면 오류 메시지를 수신할 수 있습니다. 이들 오류는 무시할 수 있습니다.

자습서 환경을 정리하려면, 다음과 같이 하십시오.

1. DB2 명령 창에서 다음 명령을 실행하십시오.

   ```
   getstart_clean.cmd
   ```

2. 데이터베이스를 사용할 수 없게 하려면, DB2 명령 창에서 다음 XML Extender 명령을 실행하십시오.

   ```
   dxxadm disable_db SALES_DB
   ```

   이 명령은 관리 제어 테이블 DTD_REF 및 XML_USAGE를 제거하고, XML Extender가 제공한 사용자 정의 유형 및 함수도 제거합니다.

3. 데이터베이스를 제거하려면, DB2 명령 창에서 다음 명령을 실행할 수 있습니다.

   ```
   db2 drop database SALES_DB
   ```

이 명령은 SALES_DB를 제거합니다.

제2부 관리

이 부분에서는 XML Extender에 대한 관리 타스크를 수행하는 방법에 대해 설명합니다.

제3장 XML Extender 사용 준비: 관리

이 장에서는 XML Extender의 설정 및 플랜에 필요한 요구사항에 대해 설명합니다.

설정 요구사항

다음 절에서는 XML Extender의 설정 요구사항에 대해 설명합니다.

소프트웨어 요구사항

XML Extender는 AIX, Windows NT 및 Sun Solaris에서 사용할 수 있습니다.

필수 소프트웨어: XML Extender에서는 DB2 Universal Database 버전 7.1 이상을 요구합니다.

선택적인 소프트웨어:

- 구조적 텍스트 검색의 경우, DB2 Universal Database Text Extender 버전 7.1 이상
- XML Extender 관리 도구용
  - DB2 UDB JDBC(DB2 UDB 버전 7.1 이상과 함께 사용 가능한)
  - JDK 1.1.7 또는 JRE 1.1.7(DB2 UDB Control Center와 함께 사용 가능함)
  - Swing 1.1을 갖춘 JFC 1.1(DB2 UDB Control Center와 함께 사용 가능함)

설치 요구사항

다음 타스크의 운영 체제에 대해서는 README 파일을 참조하십시오.

- XML Extender와 DB2 UDB 데이터베이스 바인딩.

보안상의 이유로 XML Extender를 각 데이터베이스에 바인드해야 합니다. 바인드를 완료하는 방법에 대한 자세한 내용은 233 페이지의 『시작하기 전에』를 참조하고, 예제는 다음을 참조하십시오.

DXX_INSTALL\samples\cmd\getstart_prep.cmd

- UNIX의 설정 명령 보기.
- XML 액세스용 데이터베이스 작성.

권한 부여 요구사항

관리 타스크를 수행하려면 DB2ADM 권한이 필요합니다.

관리 도구

XML Extender는 관리를 위한 세 가지 방법 즉, XML Extender 관리 마법사, XML Extender 관리 명령 및 XML Extender *저장 프로시듀어*를 제공합니다.

- 관리 마법사는 관리 타스크를 통해 프롬프트를 표시하며, 관리를 위해 권장되는 방법이기도 합니다. 이 도구 사용에 대해서는 75 페이지의 『제4장 XML 데이터 관리』에 있는 관리 타스크에서 설명합니다.
- 관리 명령, **dxxadm**은 다양한 관리 타스크의 옵션을 제공합니다. 이 명령 사용에 대해서는 75 페이지의 『제4장 XML 데이터 관리』 및 173 페이지의 『제7장 XML Extender 관리 명령: **dxxadm**』에 있는 관리 타스크에서 설명합니다.
- 관리 저장 프로시듀어는 여러 관리 타스크에 대한 옵션도 제공합니다. 이러한 저장 프로시듀어는 234 페이지의 『관리 저장 프로시듀어』에 설명되어 있습니다.

관리 플랜

결정해야 합니다.

- 데이터베이스에 있는 데이터로 XML 문서를 작성할 것인지 여부
- 이전에 있었던 XML 문서를 저장하는지, 이들을 원래의 XML 문서로서 컬럼에 저장하거나 일반 DB2 데이터로 해체할지 여부.

이것을 결정한 다음에는 나머지 관리 타스크를 계획할 수 있습니다.

- XML 문서를 검증하려는지

- 빠른 탐색(Search) 및 검색(Retrieve)을 위해 XML 컬럼에 색인을 지정하려는지
- XML 문서 구조를 DB2 관계형 테이블에 맵핑하는 방법

XML Extender 사용법은 응용프로그램에 필요한 내용에 따라 달라집니다. 3 페이지의 『제1장 XML Extender 소개』에서 설명한 것처럼, 기존 DB2 데이터에서 XML 문서를 작성하고, DB2에 XML 문서를 문서 그대로 또는 DB2 데이터로서 저장할 수 있습니다. 이러한 각 저장영역 및 액세스 메소드에는 서로 다른 플랜 요구사항이 있습니다. 다음 절에서는 이러한 플랜 고려사항에 대해 설명합니다.

액세스 및 저장영역 메소드 선택

XML Extender는 XML 저장소로서 DB2를 사용하기 위해 2개의 액세스 및 저장영역 메소드 즉, XML 컬럼과 XML 콜렉션을 제공합니다. 먼저 XML 데이터에 액세스하고 조작하기 위해 응용프로그램에 필요한 최상의 방법을 결정해야 합니다.

XML 컬럼

XML 문서 전체를 DB2 컬럼 데이터로서 저장하고 검색합니다. XML 데이터는 XML 컬럼으로 표현됩니다.

XML 콜렉션

XML 문서를 관계형 테이블 콜렉션으로 분해하거나, 관계형 테이블 콜렉션에서 XML 문서를 작성합니다.

응용프로그램 특징으로 사용할 액세스 및 저장영역 방법과, XML 데이터 구성 방법이 결정됩니다. 다음 시나리오에서는 각 액세스 및 저장영역 메소드가 가장 적합한 상황에 대해 설명합니다.

XML 컬럼을 사용하는 경우

다음의 경우 XML 컬럼을 사용합니다.

- XML 문서가 이미 있거나 외부 소스에서 추출한 것이고 문서를 원래 XML 형식으로 저장하려 합니다. 통합 및 아카이브와 감사 목적으로 DB2에 이들을 저장하려 합니다.
- 보통 XML 문서를 읽지만 갱신하지 않습니다.

- 파일 이름 데이터 유형을 사용하여 XML 문서를 외부의 지역 또는 원격 파일 시스템에 있는 DB2에 저장하고, 관리 및 검색 조작에 DB2를 사용하려 합니다.
- XML 요소 또는 검색 값을 기초로 한 범위 검색이 필요하며, 자주 검색 인수가 되는 요소나 속성들을 알고 있습니다.
- 문서에는 대형 텍스트 블록이 있는 요소가 있으며, 전체 문서를 원래 그대로 유지하는 동안 구조적인 텍스트 검색을 위해 DB2 Text Extender를 사용하려 합니다.

XML 콜렉션을 사용하는 경우

다음의 경우 XML 콜렉션을 사용합니다.

- 기존 관계형 테이블에 데이터가 있고, 일정한 DTD를 기초로 XML 문서를 작성하고자 합니다.
- 관계형 테이블에 맵핑되는 데이터 콜렉션으로 XML 문서가 저장되어야 합니다.
- 다른 맵핑 스킴을 사용하여 관계형 데이터의 다른 뷰를 작성하고자 합니다.
- 다른 데이터 소스에 있는 XML 문서가 있습니다. 데이터에 대해서는 주의를 기울이지만 태그는 관계하지 않으며, 순수한 데이터를 데이터베이스에 저장하려 합니다. 데이터를 기존의 테이블 저장할지 또는 새 테이블에 저장할지를 유연하게 결정하려 합니다.
- XML 문서의 일부 부분을 자주 갱신해야 하며, 갱신 성능은 중요합니다.
- 들어오는 XML 문서의 전체 데이터를 저장해야 하지만 이에 대한 부분 집합을 자주 검색하려 합니다.
- XML 문서가 2GB를 초과하므로 이 문서를 분해해야 합니다.

문서 액세스 정의(DAD) 파일을 사용하여 이러한 두 개의 액세스 및 저장영역 메소드을 통해 XML 데이터를 DB2 테이블과 연결합니다. 53 페이지의 그림7에서는 DAD가 액세스 및 저장영역 메소드를 지정하는 방법을 보여줍니다.

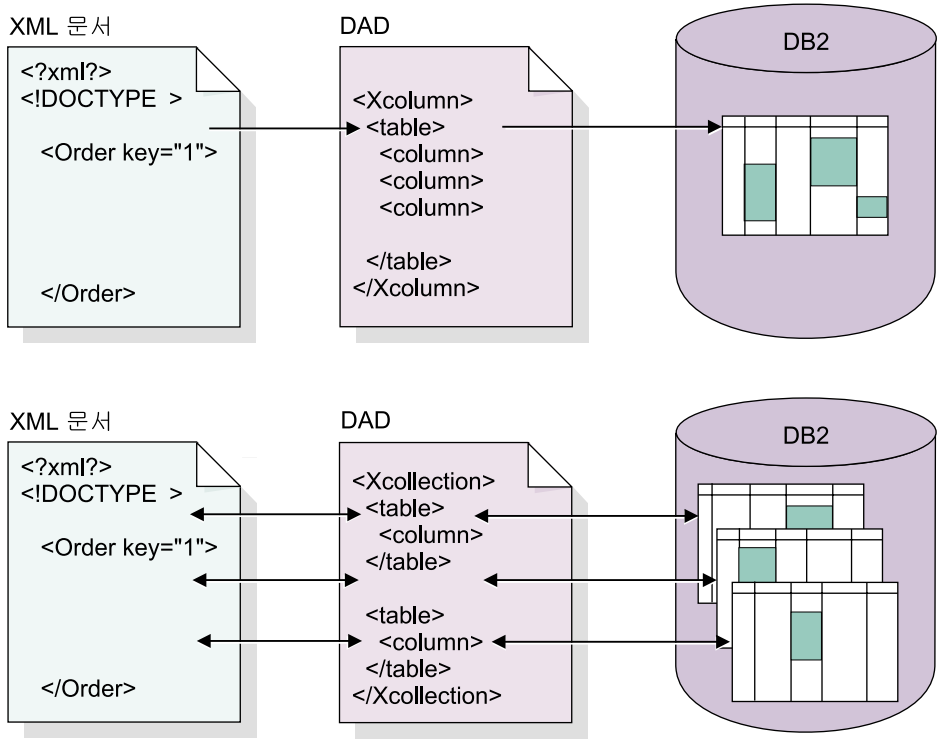


그림 7. DAD 파일은 XML 문서 구조를 DB2에 맵핑하고, 액세스 및 저장영역 메소드를 지정합니다.

DAD 파일은 XML Extender를 관리하는 중요한 부분입니다. DTD와 같은 키 파일의 위치를 정의하며, XML 문서 구조가 DB2 데이터와 연결되는 방식을 지정합니다. 가장 중요한 점은 응용프로그램에서 사용하는 액세스 및 저장영역 메소드를 정의하는 것입니다.

XML 컬럼 플랜

다음 절에서는 XML 컬럼의 플랜 타스크에 대해 설명합니다.

유효성 검증

액세스 및 저장영역 메소드를 선택한 후에, 데이터를 *유효한지* 판별할 수 있습니다. DTD를 사용하여 XML 데이터를 검증하는 경우에는 XML 문서가 유효한지 확인할 수 있으며, XML 데이터에서 구조화된 검색을 수행할 수 있습니다. DTD는 DTD 저장소에 저장되거나, DB2 서버가 액세스하는 파일 시스템에 저장될 수 있습니다.

다른 DTD를 사용하여 같은 XML 컬럼에서 문서의 유효성을 검증할 수 있습니다. 다시 말해, 구조와 요소 및 속성은 비슷하지만 다른 DTD를 호출하는 문서가 있을 수 있습니다. 여러 DTD를 참조하기 위해 다음 지침을 사용하십시오.

- DOCTYPE 정의에 있는 XML 문서의 시스템 ID는 전체 경로 이름을 사용하여 DTD 파일을 지정해야 합니다.
- DAD 파일에서 유효성 검증에 예를 지정해야 합니다.
- 적어도 하나의 DTD가 DTD_REF 테이블에 저장되어야 합니다. 모든 DTD가 이 테이블에 저장될 수 있습니다.
- DTD는 일반 구조여야 하며, 서브 요소에만 차이가 있습니다.
- DAD 파일은 이 컬럼에서 문서가 참조하는 모든 DTD에 공통적인 요소나 속성을 지정해야 합니다.

중요사항: XML 데이터를 DB2에 삽입하기 전에 유효성 검증을 수행할 것인지 결정하십시오. XML Extender는 이미 DB2에 삽입된 데이터의 유효성 검증은 지원하지 않습니다.

고려사항:

- 아카이브 목적으로 XML 문서를 저장하지 않는 경우에는 DTD를 사용하여 XML 데이터를 검증하도록 권장합니다. 검증하려면, XML Extender 저장소에 DTD가 있어야 합니다. DTD를 저장소에 삽입하는 방법에 대해서는 81 페이지의 『DTD 저장소에 DTD 저장』을 참조하십시오.
- XML 문서를 저장하거나 아카이브하기 위해 DTD가 필요하지 않습니다.
- XML 데이터의 유효성 검증을 수행하면 성능에 영향을 줄 수도 있습니다.
- 여러 DTD를 사용할 수 있지만, 공통 요소와 속성만을 색인화할 수 있습니다.
- 문서의 유효성을 검증하기로 선택하지 않은 경우에는, XML 문서에 의해 지정된 DTD는 처리되지 않습니다. 검증될 수 없는 문서 조각을 처리하는 경우일지라도 엔터티와 속성 기본값을 분석하도록 DTD를 처리하는 것이 중요합니다.

XML 사용자 정의 유형

XML 문서를 XML 컬럼에 UDT로서 저장합니다. 사용 가능한 UDT에 대해서는 55 페이지의 표4를 참조하십시오.

표 4. XML Extender UDT

사용자 정의 유형 컬럼	소스 데이터 유형	사용법 설명
XMLVARCHAR	VARCHAR(*varchar_len*)	DB2에서 전체 XML 문서를 VARCHAR로서 저장합니다.
XMLCLOB	CLOB(*clob_len*)	DB2에서 전체 XML 문서를 *CLOB*로서 저장합니다.
XMLFILE	VARCHAR(1024)	DB2에 XML 문서의 파일 이름을 저장하며, DB2 서버에 로컬인 파일에 XML 문서를 저장합니다.

부가 테이블

부가 테이블을 계획할 때, 이 테이블을 구성하는 방법과 작성할 테이블의 수 및 부가 테이블에 대한 기본 뷰 작성 여부를 고려해야 합니다. 이에 대한 결정은 요소와 속성이 여러 번 발생할 수 있는지 여부 및 조회 성능에 대한 요구사항과 같은 몇 가지 문제에 따라 일부 결정됩니다.

다중 발생: 문서에 여러 번 발생하는 위치 경로가 있으면, XML Extender는 두 번 이상 발생하는 요소의 순서를 기록하기 위해 각 부가 테이블에 INTEGER 유형의 DXX_SEQNO 컬럼을 추가합니다. DXX_SEQNO를 사용하여, SQL 조회에 ORDER BY DXX_SEQNO를 지정함으로써 원래 XML 문서와 같은 순서를 사용하여 요소의 목록을 검색할 수 있습니다.

기본 뷰 및 조회 성능: XML 컬럼을 사용 가능하게 하면, 응용프로그램 테이블을 ROOT ID라는 고유 ID를 사용하는 부가 테이블과 조인하는 기본적인 읽기 전용 뷰를 지정할 수 있습니다. 이 기본 뷰로 부가 테이블을 조회하여 XML 문서를 검색할 수 있습니다. 예를 들어, 응용프로그램 테이블 SALES_TAB과 부가 테이블 ORDER_TAB, PART_TAB 및 SHIP_TAB이 있는 경우에는 다음과 같습니다.

```
SELECT sales_person FROM sales_order_view
   WHERE price > 2500.00
```

SQL문은 ORDER 컬럼에 저장된 주문서를 가지며, 그 PRICE가 2500.00을 초과하는 SALES_TAB에 있는 해당 영업 사원의 이름을 리턴합니다.

기본 뷰 조회의 장점은 응용프로그램 테이블과 부가 테이블의 가상적인 단일 뷰를 제공하는 것입니다. 그러나, 부가 테이블을 작성하면 할수록 조회시 비용이 추가됩니다.따라서, 기본 뷰는 부가 테이블의 총 컬럼 수가 작을 때만 작성할 것을 권장합니다. 응용프로그램은 중요한 부가 테이블 컬럼을 조인하여 그 고유의 뷰를 작성할 수 있습니다.

XML 컬럼 데이터의 색인

중요한 플랜 결정은 XML 컬럼 문서를 색인화할 것인지 여부를 결정하는 것입니다. 이것은 데이터에 액세스하는 빈도와, 구조적인 검색을 수행하는 동안 성능의 중요도를 기초로 결정되어야 합니다.

전체 XML 문서가 포함된 XML 컬럼을 사용하는 경우, XML 요소나 속성값의 컬럼이 들어갈 부가 테이블을 작성한 뒤 이들 컬럼에 대해 색인을 지정할 수 있습니다. 색인을 작성해야 하는 요소와 속성을 결정해야 합니다.

XML 컬럼에 색인을 지정하면 자주 조회되는 일반 데이터 유형(예를 들면 정수, 십진수 또는 날짜)의 데이터에 데이터베이스 엔진에서 고유한 DB2 색인 지원을 사용하여 색인을 지정할 수 있습니다. XML Extender는 XML 문서에서 XML 요소 또는 속성값을 추출하고 이들을 *부가 테이블*에 저장하여, 이들 부가 테이블에서 색인을 작성할 수 있도록 합니다.

XML 요소나 속성 및 SQL 데이터 유형을 식별하는 위치 경로로 부가 테이블의 각 컬럼을 지정할 수 있습니다. 57 페이지의 그림8에서는 부가 테이블이 있는 XML 컬럼을 보여줍니다.

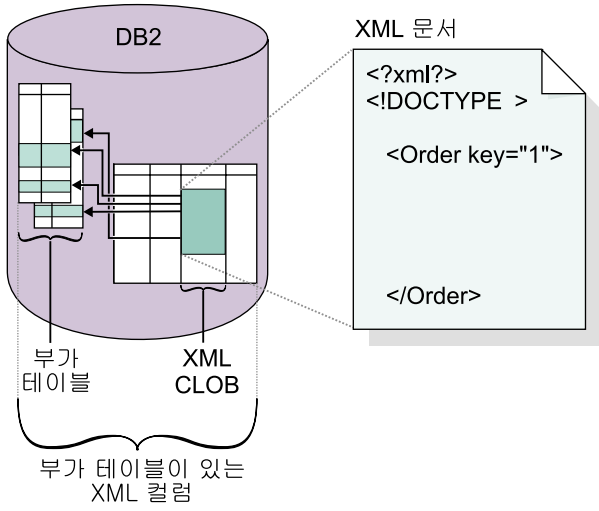


그림 8. 부가 테이블이 있는 XML 컬럼

XML Extender는 XML 컬럼에 XML 문서를 저장할 때 자동으로 부가 테이블을 채웁니다.

빠른 검색을 위해, DB2 *B-트리 색인화* 기법을 사용하여 이들 컬럼에 색인을 작성합니다. 색인을 작성하는 데 사용되는 메소드는 여러 운영 체제에 따라 다르며, XML Extender는 이들 메소드를 지원합니다.

고려사항:

- XML 문서에서 *여러 번 발생하는* 요소나 속성의 경우, XML 문서의 복잡한 구조로 인해 여러 번 발생하는 XML 요소나 속성 각각에 대해 별도의 부가 테이블을 작성해야 합니다.

  예를 들어 `/Order/Part/ExtendedPrice`에 대해 색인을 작성하고 `/Order/Part/ExtendedPrice`의 데이터 유형이 REAL이 되도록 지정하려 합니다. 이 경우, XML Extender는 `/Order/Part/ExtendedPrice`의 값을 부가 테이블에 있는 PRICE 컬럼에 저장합니다.
- 하나의 XML 컬럼에 여러 색인을 작성할 수 있습니다. 이전 예를 사용하여, 두 개의 부가 테이블에서 두 컬럼을 작성할 수 있으며, 하나는 `ExtendedPrice`를 위한 것이고 다른 하나는 `ShipDate`를 위한 것입니다.
- 응용프로그램 테이블에 있는 기본 키의 컬럼 이름 및 모든 부가 테이블을 응용프로그램과 연결하는 고유 식별자인 ROOT ID를 사용하여 부가 테이블을 응용프로그램 테이블

에 연결할 수 있습니다. 응용프로그램 테이블의 기본 키가 복합 키가 될 수 없더라도, 이를 ROOT ID가 되도록 할 것인지를 결정할 수 있습니다. 이 방법이 좋습니다.

응용프로그램 테이블에 하나의 기본 키가 존재하지 않거나 어떤 이유로 이를 사용하지 않으려는 경우, XML Extender는 삽입할 때 작성했던 고유한 ID를 저장하는 컬럼 DXXROOT_ID를 추가하도록 응용프로그램을 변경합니다. 모든 부가 테이블에는 고유한 ID와 함께 DXXROOT_ID 컬럼이 있습니다. 기본 키가 ROOT ID로 사용되면, 모든 부가 테이블에는 응용프로그램에 있는 기본 키 컬럼과 같은 이름과 유형을 갖는 컬럼이 있으며, 이 기본 키의 값이 저장됩니다.

- DB2 Text Extender에 대해 XML 컬럼을 사용할 수 있게 되면, Text Extender의 구조적인 텍스트 기능을 사용할 수도 있습니다. Text Extender에는 *"섹션 검색"* 지원이 있어, 위치 경로에서 지정된 특정 문서 문맥에서 일치되는 단어를 검색하도록 함으로써 편리한 전체 텍스트 검색 기능을 확장합니다. *구조적인 텍스트 색인*은 일반 SQL 데이터 유형에 대해 XML Extender의 색인 기능과 함께 사용될 수 있습니다.

위치 경로

*위치 경로*는 XML 요소 또는 속성을 식별하는 일련의 *XML 태그*입니다. XML Extender는 다음의 상황에서 위치 경로를 사용합니다.

- UDF를 추출하여 추출할 요소 및 속성을 식별하는 경우
- XML 컬럼에 대해 DAD에 있는 색인화 스킴을 정의할 때, XML 요소 또는 속성과 DB2 컬럼 사이의 맵핑 파일을 지정하기 위해
- 구조적인 텍스트 검색을 위해 Text Extender에 의해

59 페이지의 그림9에서는 XML 문서 구조의 위치 경로 및 그 관계 예를 보여줍니다.

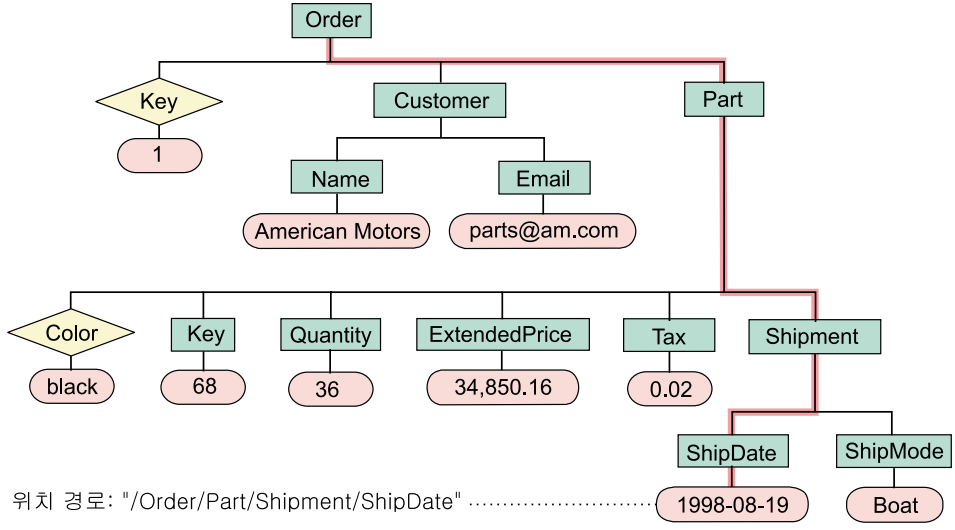


그림 9. 구조화된 XML 문서로서 DB2 테이블 컬럼에 문서 저장

위치 경로 구문: 다음 목록은 XML Extender가 지원하는 위치 경로 구문을 설명합니다. 단일 슬래시(/) 경로는 문맥이 전체 문서임을 나타냅니다.

1. **/** XML 루트 요소를 나타냅니다.

2. */tag1*

루트 아래의 *tag1* 요소를 나타냅니다.

3. */tag1/tag2/..../tagn*

루트, *tag1*, *tag2* - *tagn*-1까지의 하향 체인의 하위로서, 이름이 *tagn*인 요소를 나타냅니다.

4. *//tagn*

이름이 *tagn*인 요소를 나타내며, 더블슬래시(//)는 0개 또는 그 이상의 임의 태그를 나타냅니다.

5. */tag1//tagn*

이름이 *tagn*인 요소를 나타내며, 이것은 루트 아래에 이름이 *tag1*인 요소의 하위입니다. 여기서 더블슬래시(//)는 0개 또는 그 이상의 이름의 태그를 나타냅니다.

6. */tag1/tag2/@attr1*

이름이 *tag2*인 요소의 속성 *attr1*을 나타내며, 이것은 루트 아래에 있는 *tag1* 요소의 하위입니다.

7. */tag1/tag2[@attr1="5"]*

속성 *attr1*의 값이 5인, 이름이 *tag2*인 요소를 나타냅니다. *tag2*는 루트 아래에 이름이 *tag1*인 요소의 하위입니다.

8. */tag1/tag2[@attr1="5"]/.../tagn*

이름이 *tagn*인 요소를 나타내며, 이것은 루트에서 부터 *tag1*, *tag2* - *tagn*-1 까지 체인의 하위입니다. 여기서 *tag2*의 속성 *attr1* 값은 5입니다.

와일드 카드: 임의 문자열과 일치시킬 위치 경로에 있는 요소에 별표를 대체시킬 수 있습니다.

단순 위치 경로: *단순 위치 경로*는 XML 컬럼 DAD 파일에 정의된, 부가 테이블에 대한 요소 및 속성을 지정하는데 사용되는 위치 경로 구문입니다. 단순 위치 경로는 단일 슬래시(/)에 의해 연결되는 일련의 요소 유형 이름으로 표현됩니다. 속성값은 대괄호로 묶여지고 뒤에 요소 유형이 옵니다. 표5는 단순 위치 경로의 구문을 요약합니다.

표 5. 단순 위치 경로 구문

주제	위치 경로	설명
XML 요소	*/tag1/tag2/..../tagn-1/tagn*	이름이 *tagn*인 요소와 그 상위를 식별하는 요소 내용
XML 속성	*/tag_1/tag_2/..../tag_n-1/tag_n/@attr1*	*tagn* 및 그 상위가 식별하는 요소의, 이름이 *attr1*인 속성

XML Extender 제한사항: XML Extender에는 DAD에서 요소 또는 속성을 정의할 때의 위치 경로 사용에 관한 제한사항이 있습니다. XML Extender가 요소나 속성, 그리고 DB2 컬럼간에 일대일 맵핑을 사용하므로, 이는 DAD 파일이나 함수에서 허용되는 구문 규칙을 제한합니다. 61 페이지의 표6에서는 위치 경로에 대한 제한사항에 대해 설명합니다. 위치 경로를 지원하는 컬럼에서 지정된 번호는 59 페이지의 『위치 경로 구문』에 있는 구문 표현을 나타냅니다.

표 6. 위치 경로를 사용한 XML Extender의 제한사항

위치 경로 사용	지원되는 위치 경로
부가 테이블에 대한 XML 컬럼 DAD 맵핑의 요소	3, 6(단순 위치 경로에 대해서는 60 페이지의 표 5에서 설명됨)
추출 UDF	1-8[1]
UDF 갱신	1-8[1]
Text Extender의 검색 UDF	1-8

[1] 추출 및 갱신 UDF는 요소가 아닌 속성을 지닌 술어를 보유하는 위치 경로를 지원합니다.

DAD 파일

XML 컬럼의 경우, DAD는 주로 XML 컬럼에 저장된 문서에 색인이 지정되는 방식을 지정합니다. DAD는 XML 형식으로 되어있는 문서이며, 클라이언트에 있습니다. DTD로 XML 문서의 유효성 검증을 수행하도록 결정하면, DAD 파일은 그 DTD와 연결될 수 있습니다. DAD 파일의 데이터 유형은 CLOB입니다. 이 파일은 최대 100 KB 입니다.

XML 컬럼에 대한 DAD 파일에는 XML 머리글이 있으며, 여기에서는 클라이언트에서 DAD 파일 및 DTD에 대한 디렉토리 경로를 지정하고, 색인화를 위해 부가 테이블에 저장할 XML 데이터의 맵을 제공합니다.

XML 컬럼 액세스 및 저장영역 메소드를 지정하기 위해, DAD 파일에서 다음 태그를 사용합니다.

<Xcolumn>

XML 데이터용으로 사용할 수 있는 DB2 컬럼에서 XML 데이터가 전체 XML 문서로서 저장되고 검색되도록 지정합니다.

XML에 사용 가능한 컬럼은 XML Extender UDT의 것입니다. 응용프로그램에는 모든 *사용자 테이블*에 있는 컬럼이 포함될 수 있습니다. 주로 SQL문 및 XML Extender의 UDF를 통해 XML 컬럼에 액세스합니다.

XML Extender 관리 마법사 또는 편집기를 사용하여 DAD를 작성하고 갱신할 수 있습니다.

XML 콜렉션 플랜

XML 콜렉션을 계획할 때, DB2 데이터에서 문서 작성, XML 문서를 DB2 데이터로 분해 또는 이 둘 모두에 대해 각기 다른 고려사항이 있습니다. 다음 절에서는 XML 콜렉션에 대한 플랜과 작성 및 분해 고려사항을 설명합니다.

유효성 검증

액세스 및 저장영역 메소드를 선택한 다음에는 데이터에 대해 유효성 검증을 수행할 것인지 결정할 수 있습니다. DTD를 사용하여 XML 데이터의 유효성을 검증합니다. DTD를 사용하여 XML 문서가 유효한지 확인하고, XML 데이터에 대해 구조적인 검색을 수행할 수 있습니다. DTD는 DTD 저장소에 저장됩니다.

권장사항: DTD로 XML 데이터의 유효성을 검증하십시오. 검증하려면, XML Extender 저장소에 DTD가 있어야 합니다. DTD를 저장소에 삽입하는 방법에 대해서는 81 페이지의 『DTD 저장소에 DTD 저장』을 참조하십시오. DTD 요구사항은 XML 문서를 작성할 것인지 아니면 분해할 것인지 여부에 따라 다릅니다.

- 작성의 경우, 하나의 DTD에 대해 생성된 XML 문서의 유효성 검증을 수행할 수 있습니다. 사용할 DTD는 DAD 파일에서 지정됩니다.
- 분해의 경우, 다른 DTD를 사용하여 문서가 작성하기에 유효한지 검증할 수 있습니다. 즉, 같은 DAD 파일을 사용하지만 다른 DTD를 호출하는 문서를 분해할 수 있습니다. 여러 DTD를 참조하기 위해 다음 지침을 사용하십시오.
  - 적어도 하나의 DTD가 DTD_REF 테이블에 저장되어야 합니다. 모든 DTD가 이 테이블에 저장될 수 있습니다.
  - DTD는 일반 구조여야 하며, 하위 요소에 차이가 있습니다.
  - DAD 파일에서 유효성 검증을 지정해야 합니다.
  - XML 문서의 시스템 ID는 전체 경로 이름을 사용하여 DTD 파일을 지정해야 합니다.
  - DAD 파일에는 문서를 분해하는 방법에 대한 스펙이 들어 있으므로, 분해를 위해 공통적인 요소 및 속성만을 지정할 수 있습니다. DTD에 고유한 요소 및 속성들은 분해될 수 없습니다.

중요사항: XML 데이터를 DB2에 삽입하기 전에 XML 데이터에 대해 유효성 검증을 수행할 것인지 결정하십시오. XML Extender는 이미 DB2에 삽입된 데이터의 유효성 검증은 지원하지 않습니다.

고려사항:

- XML을 교환 포맷으로 사용할 때 DTD를 사용해야 합니다.
- XML 데이터의 유효성 검증을 수행하면 성능에 영향을 줄 수도 있습니다.
- 분해를 위해 여러 DTD를 사용하는 경우 공통적인 요소 및 속성만을 분해할 수 있습니다.
- 하나의 DTD를 사용할 경우 모든 요소 및 속성들을 분해할 수 있습니다.
- 작성을 위해 하나의 DTD만을 사용할 수 있습니다.

DAD 파일

XML 콜렉션의 경우, DAD 파일은 XML 문서 구조를 문서를 작성하거나 분해한 DB2 테이블로 맵핑합니다.

예를 들어 XML 문서에 <Tax> 요소가 있으면, <Tax>를 TAX 컬럼에 맵핑해야 합니다. XML 데이터와 DAD에 있는 관계형 데이터간의 관계를 정의합니다.

DAD 파일은 콜렉션이 사용 가능한 동안 또는 XML 콜렉션 *저장 프로시듀어*에서 DAD 파일을 사용할 때 지정됩니다. DAD는 XML 형식으로 되어있는 문서이며, 클라이언트에 있습니다. DTD로 XML 문서의 유효성 검증을 수행하도록 결정하면, DAD 파일은 그 DTD와 연결될 수 있습니다. XML Extender 저장 프로시듀어의 입력 매개변수로서 사용될 경우, DAD 파일의 데이터 유형은 CLOB입니다.이 파일은 최대 100 KB 입니다.

XML 콜렉션 액세스 및 저장영역 메소드를 지정하기 위해, DAD 파일에서 다음 태그를 사용합니다.

<Xcollection>

XML 데이터를 XML 문서에서 관계형 테이블 콜렉션으로 분해하거나, 관계형 테이블 콜렉션에서 XML 문서로 작성하도록 지정합니다.

XML 콜렉션은 XML 데이터가 들어 있는 관계형 테이블 세트의 가상 이름입니다. 응용프로그램은 사용자 테이블의 XML 콜렉션에 사용 가능합

니다. 이러한 사용자 테이블은 XML Extender가 최근 작성했던 테이블이나 계승(legacy) 비지니스 데이터의 기존 테이블이 될 수 있습니다. 주로 XML Extender가 제공하는 저장 프로시듀어를 통해 XML 콜렉션 데이터에 액세스합니다.

DAD 파일은 다음 종류의 노드를 사용하여 XML 문서 트리 구조를 정의합니다.

root_node
문서의 루트 요소를 지정합니다.

element_node
요소를 식별하며, 이것은 루트 요소나 하위 요소가 될 수 있습니다.

text_node
요소의 CDATA 텍스트를 나타냅니다.

attribute_node
요소의 속성을 나타냅니다.

65 페이지의 그림10에서는 DAD 파일에 사용된 맵핑 조각을 보여줍니다. 노드는 관계형 테이블에 있는 테이블 컬럼으로 XML 문서 내용을 맵핑합니다.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
  ...
<Xcollection>
    <SQL_stmt>
      ...
    </SQL_stmt>
      <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE DAD SYSTEM "c:\dxx\sample\dtd\getstart.dtd"</doctype>
      <root_node>
   <element_node name="Order">          --> <Order> 요소를 식별
     <attribute_node name="key">        --> "key" 속성을 식별
        <column name="order_key"/>      --> 요소와 속성이 맵핑되는 "order_key"
                                            컬럼 이름 정의
              </attribute_node>
     <element_node name="Customer">     --> <Order>의 하위 요소로
                                            <Customer> 식별
        <text_node>                     --> <Customer> 요소의
                                            <Customer> 식별
          <column name="customer">      --> 하위 요소가 맵핑되는 "customer"
                                            컬럼 이름 정의
                 </text_node>
              </element_node>
        ...
              </element_node>
      ...
      <root_node>
  </Xcollection>
</DAD>
```

그림 10. 노드 정의

이 예에서, SQL문에 있는 처음 두 컬럼에는 이에 맵핑된 요소와 속성이 있습니다.

XML Extender에서는 <stylesheet> 요소를 사용하여 스타일 시트에 대한 처리 지시사항도 지원합니다. 이는 DAD 파일의 루트 노드 내에 위치해야 하며, doctype 및 prolog가 XML 문서에 대해 정의되어야 합니다. 예를 들면 다음과 같습니다.

```
<Xcollection>
...
<prolog>...</prolog>
<doctype>...</doctype>
<stylesheet>?xml-stylesheet type="text/css" href="order.css"?</stylesheet>
<root_node>...</root_node>
...
  </Xcollection>
```

XML Extender 관리 마법사나 편집기를 사용하여 DAD 파일을 작성하고 갱신할 수 있습니다. <stylesheet> 요소는 현재 XML Extender 관리 마법사에서 지원하지 않습니다.

XML 콜렉션을 위한 맵핑 스킴

XML 콜렉션을 사용하는 경우, 관계형 데이터베이스에서 XML 데이터가 표현되는 방식을 정의하는 *맵핑 스킴*을 선택해야 합니다. XML 콜렉션은 XML 문서에 사용되는 계층형 구조를 관계형 구조와 대응시켜야 하므로, 두 구조를 비교하는 방법을 이해해야 합니다. 그림11에서는 계층형 구조가 관계형 테이블 컬럼에 맵핑되는 방식을 보여줍니다.

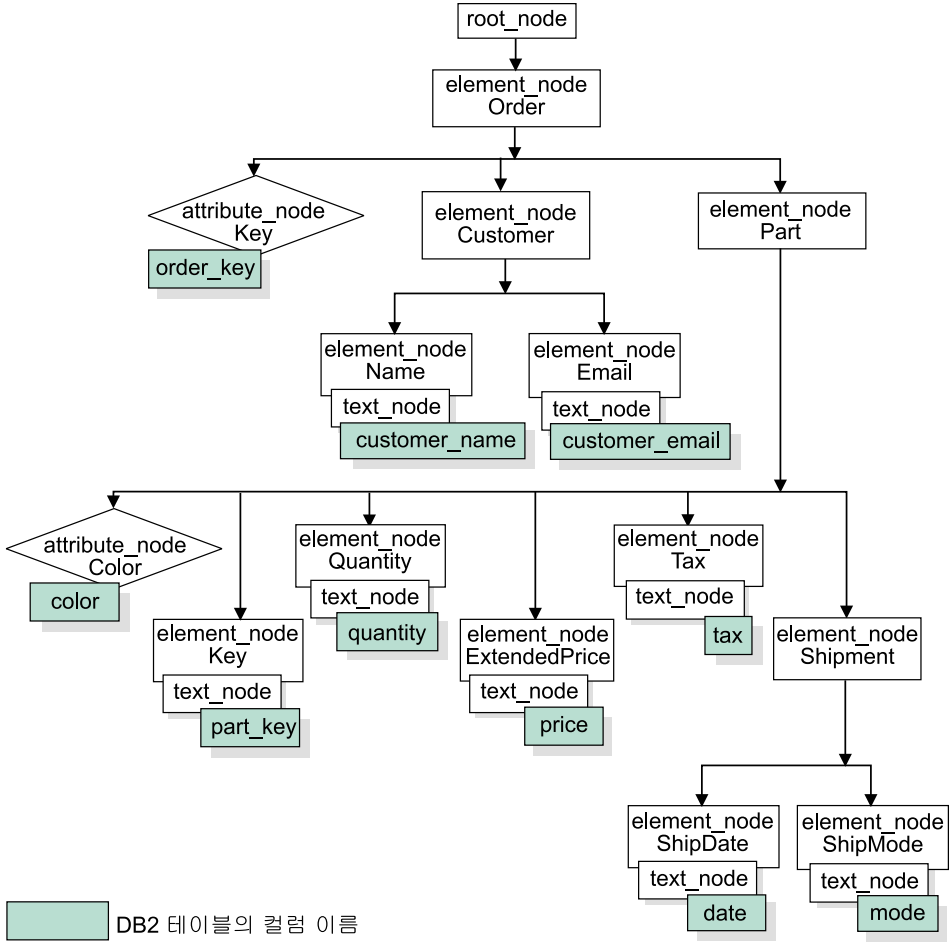


그림 11. 관계형 테이블 컬럼에 맵핑되도록 구조화된 XML 문서

XML Extender는 여러 관계형 테이블에 있는 XML 문서를 작성하거나 분해할 때 맵핑 스킴을 사용합니다. XML Extender는 DAD 파일을 작성할 때 도움이 되는 마법사를 제공합니다. 그러나 DAD 파일을 작성하기 전에, XML 데이터가 XML 콜렉션으로 맵핑되는 방식을 생각해야 합니다.

맵핑 스킴 유형: 맵핑 스킴은 DAD 파일에 있는 <Xcollection> 요소에서 지정됩니다. XML Extender는 두 가지 유형의 맵핑 스킴 즉, *SQL 맵핑* 및 *관계형 데이터베이스(RDB_node) 맵핑*을 제공합니다. 두 메소드 모두는 XSLT 모델을 사용하여 XML 문서의 계층을 정의합니다.

SQL 맵핑

하나의 SQL문과 *XSLT 데이터 모델*을 통해 관계형 데이터에서 XML 문서로의 단일 및 직접 맵핑을 허용합니다. SQL 맵핑은 작성에 사용되며, 분해에는 사용되지 않습니다. SQL 맵핑은 DAD 파일에서 SQL_stmt 요소로 정의됩니다. SQL_stmt 내용은 유효한 SQL문입니다. SQL_stmt는 SELECT 절에 있는 컬럼을 XML 문서에 사용된 XML 요소나 속성으로 맵핑합니다. XML 문서를 작성하기 위해 정의될 때, SQL문의 SELECT 절에 있는 컬럼 이름은 *attribute_node* 값이나 *text_node* 내용을 정의할 때 사용됩니다. FROM 절에서는 데이터가 들어있는 테이블을 정의하며, WHERE 절에서는 *조인* 및 *검색 조건*을 지정합니다.

SQL 맵핑을 사용하면 DB2 사용자가 SQL을 사용하여 데이터를 맵핑할 수 있습니다. SQL 맵핑 사용시, 하나의 SELECT 문에 있는 모든 테이블들을 조인하여 조회를 구성할 수 있어야 합니다. 하나의 SQL문으로 충분하지 않으면, RDB_node 맵핑 사용을 고려하십시오. 모든 테이블들을 함께 연결하기 위해, 이들 테이블에서 *기본 키* 및 *외부 키* 관계를 사용하는 것이 좋습니다.

RDB_node 맵핑

XML Extender가 XML 데이터를 저장하거나 검색하는 위치를 판별할 수 있도록 XML 요소의 내용 위치나 XML 속성값을 정의합니다.

*RDB_node*에는 테이블, 선택적인 컬럼 및 선택적인 조건을 위한 하나 이상의 노드 정의가 들어 있습니다. 테이블 및 컬럼은 XML 데이터가 데이

터베이스에서 저장되는 방식을 정의하는 데 사용됩니다. 조건에서는 XML 데이터를 선택하는 기준이나 XML 콜렉션 테이블을 조인하는 방식을 지정합니다.

맵핑 스킴을 정의하기 위해, <Xcollection> 요소로 DAD를 작성합니다. 그림12에서는 세 개의 관계형 테이블에 있는 데이터로 XML 문서 세트를 작성하는 XML 콜렉션 SQL 맵핑과 함께 샘플 DAD 파일 조각을 보여줍니다.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
  <dtdid>c:\dxx\samples\dad\getstart.dtd</dtdid>
   <validation>YES</validation>
  <Xcollection>
    <SQL_stmt>
       SELECT o.order_key, customer, p.part_key, quantity, price, tax, date,
              mode, comment
                FROM order_tab o, part_tab p,
               table(select substr(char(timestamp(generate_unique())),
               as ship_id, date, mode, from ship_tab) as s
        WHERE p.price > 2500.00 and s.date > "1996-06-01" AND
              p.order_key = o.order_key and s.part_key = p.part_key
    </SQL_stmt>
      <prolog>?xml version="1.0"?</prolog>
    <doctype>!DOCTYPE DAD SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
      <root_node>
           <element_node name="Order">
                  <attribute_node name="key">
                        <column_name="order_key"/>
                  </attribute_node>
                  <element_node name="Customer">
                          <text_node>
                                  <column name="customer"/>
                          </text_node>
                  <element_node>

                ...
                  </element_node><!-end Part->
           </element_node><!-end Order->
      </root_node>
  </Xcollection>
</DAD>
```

그림 12. SQL 맵핑 스킴

XML Extender는 XML 콜렉션에 있는 데이터를 관리하는 여러 저장 프로시듀어를 제공합니다. 이러한 저장 프로시듀어는 두 가지 유형 모두의 맵핑을 지원하지만, DAD 파일이 69 페이지의 『맵핑 스킴 요구사항』에서 설명했던 규칙을 준수해야 합니다.

맵핑 스킴 요구사항: 다음 절에서는 각 유형의 XML 콜렉션 맵핑 스킴에 대한 요구사항에 대해 설명합니다.

SQL 맵핑 사용시 요구사항

이 맵핑 스킴에서는 DAD <Xcollection> 요소에서 SQL_stmt 요소를 지정해야 합니다. SQL_stmt에는 여러 관계형 테이블을 조회 술어와 조인할 수 있는 하나의 SQL문이 포함되어야 합니다. 뿐만 아니라, 다음 절이 필요합니다.

- **SELECT 절**
  - 컬럼 이름이 고유한지 확인합니다. 두 테이블에 같은 이름의 컬럼이 있으면, AS 키워드를 사용하여 이들 중 하나에 대한 별명 이름을 작성하십시오.
  - 같은 테이블 컬럼을 함께 그룹 지정하고, 관계형 테이블의 논리 계층형 레벨을 사용하십시오. 이것은 XML 문서의 계층형 구조에 맵핑되면 중요도에 따라 테이블을 그룹화함을 의미합니다. SELECT 절에서, 상위 테이블의 컬럼은 하위 테이블의 컬럼보다 우선되어야 합니다. 다음 예에서는 테이블간의 계층적 관계에 대해 설명합니다.

    ```
    SELECT o.order_key, customer, p.part_key, quantity, price, tax,
           ship_id, date, mode
    ```

    이 예에서, ORDER_TAB 테이블의 order_key 및 customer는 XML 문서의 계층형 트리에서 상위에 있으므로 가장 높은 레벨에 있습니다. SHIP_TAB 테이블의 ship_id, date 및 mode는 가장 낮은 관계형 레벨에 있습니다.
  - 단일 컬럼 후보 키를 사용하여 각 레벨을 시작하십시오. 그러한 키를 테이블에서 사용할 수 없으면, 조회에서는 테이블 표현식과 내장 함수인 generate_unique()를 사용하여 그 테이블에 대해 키를 생성해야 합니다. 위 예에서, o.order_key는 ORDER_TAB의 기본 키이며, part_key는 PART_TAB의 기본 키입니다. 이들은 선택할 컬럼의 소유 그룹 앞에 표시됩니다. SHIP_TAB 테이블에는 기본 키가 없으므로 하나를 생성해야 합니다. 이 경우는 ship_id이며, 이것은 SHIP_TAB 테이블 그룹의 첫번째 컬럼으로 나열됩니다. FROM 절을 사용하여 다음 예에서처럼 기본 키 컬럼을 생성하십시오.

- **FROM 절**
  - 테이블 표현식 및 내장 함수 generate_unique()를 사용하여 1차 단일 키가 없는 테이블에 대해 단일 키를 생성하십시오. 예를 들면 다음과 같습니다.

```
FROM order_tab as o, part_tab as p,
       table(select substr(char(timestamp(generate_unique())),16) as
       ship_id, date, mode from ship_tab) as s
```

    이 예에서, generate_unique() 함수는 TIMESTAMP의 CHAR 데이터 유형으로 유형변환되며, 이는 ship_id 이름의 별명이 부여됩니다.
  - 컬럼을 별도로 작성해야 하는 경우 별명 이름을 사용하십시오. 예를 들어, o는 ORDER_TAB에, p는 PART_TAB에, 그리고 s는 SHIP_TAB에 사용할 수 있습니다.
- **Where 절**
  - 콜렉션에 있는 테이블들을 함께 연결하는 조인 조건으로서 기본키와 외부 키 관계를 지정하십시오. 예를 들면 다음과 같습니다.

```
WHERE p.price > 2500.00 AND s.date > "1996-06-01" AND
      p.order_key = o.order_key AND s.part_key = p.part_key
```

  - 술어에서 다른 검색 조건을 지정하십시오. 모든 올바른 술어를 사용할 수 있습니다.
- **ORDER BY 절**
  - SQL_stmt 끝에서 ORDER BY 절을 정의합니다.
  - 컬럼 이름이 SELECT 절에 있는 컬럼 이름과 일치하는지 확인하십시오.
  - 데이터베이스의 엔터티 관계 디자인에서 엔터티를 고유하게 식별하는 컬럼 이름이나 식별자를 지정합니다. 식별자는 테이블 표현식과 내장 함수 generate_unique 또는 사용자 정의 함수(UDF)를 사용하여 생성될 수 있습니다.

– 엔터티 계층 구조에서 위에서 아래로 순서를 유지보수하십시오. ORDER BY 절에서 지정된 컬럼은 각 엔터티에 나열된 첫번째 컬럼이 되어야 합니다. 순서를 보존하면 생성될 XML 문서에 틀린 복제본이 들어가지 않습니다.

– 스키마 또는 테이블 이름으로 ORDER BY 절에 있는 컬럼을 규정하지 마십시오.

SQL_stmt에 선행하는 요구사항이 있는 경우라도, 술어에 있는 표현식이 테이블에 있는 컬럼을 사용하는 동안은 WHERE 절에서 술어를 지정할 수 있습니다.

RDB_node 맵핑 사용시 요구사항

이 맵핑 메소드를 사용할 경우, DAD 파일의 <Xcollection> 요소에서 SQL_stmt 요소를 사용하지 마십시오. 대신, *element_node*의 맨 위 노드 각각에 대해, 그리고 각각의 attribute_node 및 text_node에서 RDB_node 요소를 사용하십시오.

- **맨 위 element_node의 RDB_node**

  DAD 파일에 있는 *맨 위 element_node*는 XML 문서의 루트 요소를 나타냅니다. 다음과 같이 맨 위 element_node에 RDB_node를 지정하십시오.

  – XML 문서와 연관된 모든 테이블을 지정하십시오. 예를 들어, 다음 맵핑에서는 element_node <Order>의 RDB_node에서 세 개의 테이블을 지정하며, 이것이 맨 위 element_node입니다.

```
        <element_node name="Order">
                    <RDB_node>
                    <table name="order_tab"/>
                    <table name="part_tab"/>
                       <table name="ship_tab"/>
                       <condition>
order_tab.order_key = part_tab.order_key AND
                    part_tab.part_key = ship_tab.part_key
                       </condition>
                    </RDB_node>
```

  콜렉션에 테이블이 하나만 존재하는 경우에는 조건 요소가 없거나 누락될 수 있습니다.

– 분해 또는 DAD 파일이 지정한 XML 콜렉션을 사용 가능화하는 중이라면, 각 테이블에 대해 기본 키를 지정해야 합니다. 기본 키는 복합 키라고 하며, 단일 컬럼이나 다중 컬럼으로 구성될 수 있습니다. 기본 키는 속성키를 RDB_node의 테이블 요소에 추가하여 지정됩니다. 복합 키가 제공되면, 키 속성은 공백으로 구분되는 키 컬럼 이름으로 지정됩니다. 예를 들면 다음과 같습니다.

```
<table name="part_tab" key="part_key, price"/>
```

분해를 위해 지정된 정보는 문서를 작성할 때는 무시됩니다.

– orderBy 속성을 사용하여, 여러 번 발생하는 요소나 속성이 포함된 XML 문서를 원래 구조로 다시 작성하십시오. 이 속성을 사용하면 문서 순서를 유지하기 위해 사용할 키가 될 컬럼 이름을 지정할 수 있습니다. orderBy 속성은 DAD 파일에 있는 테이블 요소의 일부이며, 선택적인 속성입니다.

테이블 이름과 컬럼 이름에서 명시적으로 판독해야 합니다.

• **각 attribute_node 및 text_node의 RDB_node**

이 맵핑 스킴에서, 데이터는 각 element_node의 attribute_node 및 text_node에 있습니다. 그러므로 XML Extender는 데이터베이스에서 데이터를 찾아야 할 위치를 알고 있어야 합니다. attribute_node 및 text_node 각각에 대해 RDB_node를 지정하여, 데이터를 구할 테이블, 컬럼 및 조회 조건을 저장 프로시듀어에 알려야 합니다. 테이블 및 컬럼 값을 지정해야 하며, 조건값은 선택적입니다.

– 컬럼 데이터가 들어있는 테이블 이름을 지정하십시오. 테이블 이름은 맨 위 element_node의 RDB_node에 포함되어야 합니다. 이 예에서, 요소 <Price>의 text_node의 경우 테이블은 PART_TAB으로 지정됩니다.

```
<element_node name="Price">
                  <text_node>
                     <RDB_node>
                     <table name="part_tab"/>
                        <column name="price"/>
                        <condition>
                           price > 2500.00
```

```
            </condition>
          </RDB_node>
        </text_node>
      </element_node>
```

- 요소 텍스트에 대한 데이터가 들어있는 컬럼 이름을 지정하십시오. 이전 예에서, 컬럼은 PRICE로 지정됩니다.
- XML 문서가 조회 조건을 사용하여 생성되게 하려면 조건을 지정하십시오. 위 예에서, 조건은 `price > 2500.00`으로 지정됩니다. 조건을 만족하는 데이터만이 생성된 XML 문서에 있게 됩니다. 이 조건은 올바른 WHERE 절이어야 합니다.
- 문서를 분해하거나, 또는 DAD 파일이 지정한 XML 콜렉션을 사용가능화하는 중이라면, 각 attribute_node와 text_node에 대해 컬럼 유형을 지정해야 합니다. 이렇게 하면 XML 콜렉션을 사용할 수 있는 동안 새 테이블이 작성될 때 각 컬럼에 대해 정확한 데이터 유형이 지정됩니다. 컬럼 유형은 컬럼 요소로 속성 유형을 추가하여 지정됩니다. 예를 들면 다음과 같습니다.

  ```
  <column name="order_key" type="integer"/>
  ```

  분해를 위해 지정된 정보는 문서를 작성할 때는 무시됩니다.

RDB_node 맵핑 접근 방식을 사용할 경우는 SQL문을 제공하지 않아도 됩니다. 그러나 RDB_node 요소에 복잡한 조회 조건을 삽입하면 좀 더 어려워질 수 있습니다. 예를 들어 *union*, 표현식 또는 피연산자를 사용하면 SQL-XML 접근 방식을 사용할 경우보다 다소 기능이 감소합니다.

분해 테이블 크기 요구사항

분해는 요소 및 속성 값을 테이블 행으로 추출함으로써 XML 문서가 DB2 테이블에 분해되는 방법을 지정하기 위해 RDB_node 맵핑을 사용합니다. 각 XML 문서의 값은 하나 이상의 DB2 테이블에 저장됩니다. 각 테이블에는 각 문서에서 분해된 최대 1024행이 있을 수 있습니다.

예를 들어, XML 문서가 5개의 테이블로 분해된 경우, 5개의 테이블 각각에는 특정 문서에 대해 최대 1024행이 있을 수 있습니다. 이 테이블에 여러 문서의 행이

들어 있는 경우, 테이블에는 각 문서마다 최대 1024행이 있을 수 있습니다. 테이블에 20개의 문서가 들어 있는 경우, 테이블에는 각 문서에 대해 1024행씩 모두 20,480행이 있을 수 있습니다.

다중 발생 요소(XML 구조에서 두 번 이상 발생할 수 있는 위치 경로가 있는 요소)는 행 수에 영향을 줍니다. 예를 들어, 20번 발생하는 요소 <Part>가 들어 있는 문서는 한 테이블에서 20행으로 분해될 수 있습니다. 다중 발생 요소를 사용한다면 테이블 *크기* 제한사항을 고려하십시오.

제4장 XML 데이터 관리

XML Extender 관리 타스크는 XML용으로 데이터베이스 및 테이블 컬럼을 사용하게 설정하고, XML 데이터를 DB2 관계형 구조로 맵핑하는 타스크로 구성됩니다. XML Extender는 관리 타스크를 수행할 응용프로그램을 개발하려는지 여부 또는 단순히 마법사를 사용하려는지 여부에 따라 사용자 편의를 위해 여러 관리 도구를 제공합니다. 다음 도구를 사용하여 XML Extender에 대한 관리 타스크를 완료할 수 있습니다.

- XML Extender 관리 마법사
- **dxxadm** 명령
- XML Extender 관리 저장 프로시듀어

이 장에서는 관리 마법사와 **dxxadm** 명령으로 연결되는 관리 타스크에 대해 설명합니다. 관리 저장 프로시듀어에 대해서는 234 페이지의 『관리 저장 프로시듀어』에서 설명합니다.

이 장에 있는 타스크를 완료하려면, 50 페이지의 『관리 플랜』에서 설명한 개념 및 타스크 계획에 익숙해야 합니다.

다음 절에서는 XML Extender 관리 타스크에 대해 설명합니다.

1. 『관리 마법사 시작』
2. 79 페이지의 『XML의 데이터베이스 사용』
3. 81 페이지의 『DTD 저장소에 DTD 저장』
4. 83 페이지의 『XML 컬럼 또는 콜렉션 정의』
5. 83 페이지의 『XML 컬럼에 대한 작업』
6. 97 페이지의 『XML 콜렉션에 대한 작업』

관리 마법사 시작

이 절에는 XML Extender 관리 마법사 설정 및 호출에 대한 정보가 들어 있습니다.

관리 마법사 설정

운영 체제의 readme 파일에 있는 관리 마법사의 설치 및 구성 단계를 준수하도록 하십시오. 여기에는 bind 명령문을 실행하고, CLASSPATH 명령문에 필수 소프트웨어를 포함시키는 타스크가 포함됩니다.

- 바인드 명령문은 마법사 readme 파일 및 시작하기 샘플 파일에서 제공됩니다.

  /dxx_install/samples/cmd/getstart_prep.cmd

- CLASSPATH 명령문은 다음과 유사합니다(편의상 행 분리가 되어 있음).

```
.;C:\java\db2java.zip;C:\java\runtime.zip;C:\java\sqlj.zip;
C:\dxx\dxxadmin\dxxadmin.jar;C:\dxx\dxxadmin\dxxadmin.cmd;
C:\dxx\dxxadmin\html\dxxahelp*.htm;C:\java\jdk\lib\classes.zip;
C:\java\swingall.jar
```

  중요사항: 마법사에는 공백이 없는 경로 이름이 필요합니다. IBM DB2 Universal Database V7.1 기본 설치를 사용하고 SQLLIB\java가 프로그램 파일 디렉토리 아래에 있는 경우, Java 코드를 보다 단순한 경로로 복사하십시오. Java 코드를 이동하거나 CLASSPATH를 변경하지 마십시오. 제어 센터는 설치시 지정된 CLASSPATH를 요구합니다.

XML Extender 관리 마법사는 클래스 파일을 사용합니다. 기본 XML Extender 관리의 전체 파일 이름은 다음과 같습니다.

```
com.ibm.dxx.admin.Admin.
```

시스템에 대한 이 파일을 수정하여 마법사를 호출하십시오.

- JDK를 사용하여 호출하려면 다음을 입력하십시오.

```
java -classpath classpath com.ibm.dxx.admin.Admin
```

- JRE를 사용하여 호출하려면 다음을 입력하십시오.

```
jre -classpath classpath com.ibm.dxx.admin.Admin
```

  여기에서 *classpath*는 다음 중 하나를 지정합니다.

  – 관리 마법사 클래스 파일의 위치를 지정하는 %CLASSPATH% 환경 변수. 이 옵션을 사용하는 경우에 시스템 CLASSPATH는 *dxx_install*/dxxadmin

디렉토리를 지시해야 합니다. 이 디렉토리에는 다음 파일이 포함되어 있습니다. dxxadmin.jar, xml4j.jar 및 db2java.zip. 예를 들면 다음과 같습니다.

```
java -classpath %CLASSPATH% com.ibm.dxx.admin.Admin
```

– XML Extender 관리 마법사를 실행하는 *dxx_install*/dxxadmin 디렉토리의 파일에 대한 포인터가 있는 %CLASSPATH% 환경 변수의 대체. 예를 들면 다음과 같습니다.

```
java -classpath dxxadmin.jar;xml4j.jar;db2java.zip com.ibm.dxx.admin.Admin url=jdbc:db2:mydb
userid=db2xml password=db2xml driver=COM.ibm.db2.jdbc.app.DB2Driver
```

선택적으로, 런타임시 다음 매개변수를 지정할 수 있습니다.

url 연결할 IBM DB2 UDB 데이터 소스의 완전한 URL 경로.
예: jdbc:db2://dxx.stl.ibm.com:8080/guidb. 마법사에서 『주소』로 레이블됨.

userid 위의 데이터 소스를 액세스하는데 사용되는 사용자 ID.
예: db2guest.

password
위의 사용자 ID에 대한 암호.
예: guest.

driver 위의 URL에 대한 JDBC 드라이버 이름.
기본값: COM.ibm.db2.jdbc.net.DB2Driver. 마법사에 『JDBC 드라이버』로 레이블됨.

이들 값에 대한 자세한 정보는 『관리 마법사 호출』의 내용을 참조하십시오.

관리 마법사 호출

다음 단계에 따라 XML Extender 관리 마법사를 호출하십시오.

1. 마법사를 호출하십시오.

   Windows NT의 경우:

   데스크탑에서 XML Extender 관리 마법사 아이콘을 두 번 누르십시오.

   AIX, Sun Solaris 및 Linux의 경우:

   dxxadmin 파일을 실행하십시오.

관리 마법사 로그온 창이 열립니다.

XML Extender 관리 마법사를 호출하면 로그온 창이 나타납니다. XML 데이터에 대해 작업할 때 사용할 데이터베이스에 로그인하십시오. XML Extender가 현재 인스턴스에 연결됩니다.

2. **주소** 필드에 연결하는 IBM DB2 UDB 데이터 소스의 완전한 JDBC URL을 입력하십시오. 주소의 구문은 다음과 같습니다.

   독립형 구성의 경우(권장):

   ```
   jdbc:db2:database_name
   ```

   여기서,

   database_name
   연결하고 XML 문서를 저장하는 데이터베이스.

   예를 들면 다음과 같습니다.

   ```
   jdbc:db2:sales_db
   ```

   네트워크 구성의 경우:

   ```
   jdbc:db2://server_name:port_number/database_name
   ```

   여기서,

   server_name
   XML Extender가 있는 서버 이름.

   port_number
   서버에 연결하는 데 사용되는 포트 번호. 포트 번호를 결정하려면, 서버 머신의 DB2 명령 행에서 다음 명령을 입력하십시오.

   ```
   db2jstrt port#
   ```

   Windows NT 사용자는 포트 번호에 대해 다음의 파일 \winnt\system32\driver\etc\services를 확인할 수 있습니다.

   database_name
   연결하고 XML 문서를 저장하는 데이터베이스.

예를 들면 다음과 같습니다.

```
jdbc:db2://host1.ibm.com:8080/sales_db
```

3. **사용자 ID** 및 **암호** 필드에서, 연결 중인 데이터베이스의 DB2 사용자 ID 및 암호를 입력하거나 확인하십시오.
4. **JDBC 드라이브** 필드에서는 다음 값을 사용하여 지정된 주소에 대한 JDBC 드라이버 이름을 확인하십시오.

   독립형 구성의 경우(기본 및 권장):

   ```
   COM.ibm.db2.jdbc.app.DB2DRIVER
   ```

   네트워크 구성의 경우:

   ```
   COM.ibm.db2.jdbc.net.DB2DRIVER
   ```

5. **완료**를 눌러 마법사에 연결하고, LaunchPad 창으로 진행하십시오.

LaunchPad 창은 5개의 관리 마법사에 대한 액세스를 제공합니다. 이러한 마법사를 사용하여 다음을 수행할 수 있습니다.

- 데이터베이스 사용
- DTD 저장소에 DTD 추가
- 다음에 대해 DAD 파일을 사용하여 작업할 수 있습니다.
  - XML 컬럼
  - XML 콜렉션
- XML 컬럼에 대한 작업
- XML 콜렉션에 대한 작업

XML의 데이터베이스 사용

XML Extender를 사용하여 DB2에서 XML 문서를 저장하거나 검색하려면, XML에 대해 데이터베이스를 사용할 수 있습니다. XML Extender는 현재 인스턴스를 사용하여, 연결한 데이터베이스를 사용할 수 있습니다.

XML에 사용할 수 있도록 데이터베이스를 설정하면, XML Extender는 다음을 수행합니다.

- 사용자 정의 유형(UDT) 및 사용자 정의 기능(UDF) 모두를 작성합니다.

- 제어 테이블을 작성하고 XML Extender에 필요한 메타데이터로 채웁니다.
- db2xml 스키마를 작성하고 필요한 특권을 지정합니다.

  XML 함수의 전체 이름은 *schema-name.function-name*이며, 여기서 *schema-name*은 SQl *오브젝트*에 대한 논리적 그룹화를 제공하는 식별자입니다. UDF 또는 UDT를 참조할 때는 언제나 전체 이름을 사용할 수 있습니다. 또한 UDF 또는 UDT를 참조할 때 스키마 이름을 생략할 수도 있으며, 이런 경우, DB2는 함수 경로를 사용하여 원하는 함수나 데이터 유형을 결정합니다.

관리 마법사 사용

다음 단계에서 XML 데이터에 대한 데이터베이스를 사용하게 하십시오.

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. LaunchPad 창에서 **데이터베이스 사용**을 눌러 현재 데이터베이스를 사용하십시오.

   데이터베이스가 이미 사용중이면, **데이터베이스 사용 안함**만을 선택할 수 있습니다.

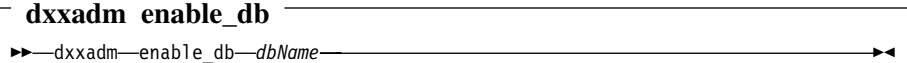
   데이터베이스가 사용 가능하면 LaunchPad 창으로 되돌아갑니다.

DB2 명령 쉘에서

명령행에서 **dxxadm**을 입력하여, 사용할 수 있도록 하려는 데이터베이스를 지정하십시오.

구문:

dxxadm enable_db



매개변수:

dbName

사용할 수 있도록 하려는 데이터베이스 이름.

예: 기존 데이터베이스 SALES_DB를 사용할 수 있도록 합니다.

```
dxxadm enable_db SALES_DB
```

DTD 저장소에 DTD 저장

DTD를 사용하여 XML 컬럼이나 XML 콜렉션에 있는 XML 데이터에 대해 유효성 검증을 수행할 수 있습니다. DTD는 XML 컬럼의 유효성을 검증하고, XML의 구조적인 검색에 또는 콜렉션 작성 및 분해에 사용되는 DAD 파일을 정의하는 데 사용됩니다.

모든 DTD는 DTD_REF라는 DB2 테이블인 DTD 저장소에 저장됩니다. 이것의 스키마 이름은 db2xml입니다. DTD_REF 테이블에 있는 각 DTD에는 고유한 ID가 있습니다. XML Extender는 XML용으로 데이터베이스를 사용할 수 있는 경우 DTD_REF 테이블을 작성합니다.

DTD 사용법에 대해서는 53 페이지의 『XML 컬럼 플랜』 및 62 페이지의 『XML 콜렉션 플랜』을 참조하십시오.

DB2 명령 쉘에서 또는 관리 마법사를 사용하여 DTD를 삽입할 수 있습니다.

관리 마법사 사용

다음 단계에서 DTD를 삽입하십시오.

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. 기존의 DTD 파일을 현재 데이터베이스의 DTD 저장소로 가져오려면, LaunchPad 창에서 **DTD 가져오기**를 누르십시오. DTD 가져오기 창이 표시됩니다.
3. **DTD 파일 이름** 필드에 DTD 파일 이름을 입력하거나, **...**을 눌러 기존 DTD 파일을 찾아보십시오.
4. **DTD ID** 필드에 DTD ID를 입력하십시오.

DTD ID는 DTD에 대한 식별자이며, 지역 시스템에서 DTD의 위치를 지정하는 경로가 될 수 있습니다. DTD ID는 <DTDID> 요소에 대해 DAD 파일에서 지정된 값과 일치해야 합니다.

5. 선택적으로, DTD의 작성자 이름을 **작성자** 필드에 입력하십시오.

   XML Extender는 작성자 이름이 DTD에서 지정되었으면 자동으로 그 이름을 표시합니다.

6. **완료**를 눌러 DTD를 DTD 저장소 테이블, DB2XML.DTD_REF에 삽입하고, LaunchPad 창으로 되돌아가십시오.

DB2 명령 쉘에서

표7에 있는 스키마를 사용하여 DTD_REF 테이블에 대해 SQL_INSERT 명령문을 실행하십시오.

표 7. DTD_REF DTD 테이블의 스키마

컬럼 이름	데이터 유형	설명
DTDID	VARCHAR(128)	기본 키(고유하고 널이 아님). 기본 키는 DTD를 식별하는 데 사용되며, 유효성 검증이 사용될 때 각 XML 문서의 DOCTYPE 행에 있는 SYSTEM ID와 같아야 합니다. 기본 키가 DAD 파일에서 지정되면, DAD 파일은 DTD에서 정의된 스키마에 따라야 합니다.
CONTENT	XMLCLOB	DTD 내용.
USAGE_COUNT	INTEGER	이 DTD를 사용하여 DAD를 정의하는, 데이터베이스에 있는 XML 컬럼 및 XML 콜렉션의 갯수.
AUTHOR	VARCHAR(128)	DTD 작성자, 사용자가 입력하는 선택적인 정보.
CREATOR	VARCHAR(128)	처음 삽입하는 사용자 ID.
UPDATOR	VARCHAR(128)	마지막으로 갱신하는 사용자 ID.

예를 들면 다음과 같습니다.

```
DB2 INSERT into db2xml.dtd_ref values('c:\dxx\samples\dtd\getstart.dtd',
    db2xml.XMLClobFromFile('c:\dxx\samples\dtd\getstart.dtd'), 0, 'user1',
    'user1', 'user1')
```

XML 콜렉션 중요사항: DTDID는 지역 시스템에서 DTD의 위치를 지정하는 경로입니다. DTD ID는 <DTDID> 요소에 대해 DAD 파일에서 지정된 값과 일치해야 합니다.

XML 컬럼 또는 콜렉션 정의

다음 절에서는 XML 컬럼 또는 콜렉션에 대한 데이터베이스를 설정하고 정의하는 방법과, 필요한 데이터 맵핑 스킴을 준비하는 방법에 대해 설명합니다.

이 절은 다음과 같습니다.

- 『XML 컬럼에 대한 작업』
- 97 페이지의 『XML 콜렉션에 대한 작업』

XML 컬럼에 대한 작업

XML 컬럼을 설정하려면, XML 데이터에 액세스하고 XML 테이블에서 XML 데이터에 대한 컬럼을 사용할 수 있도록 DAD 파일을 정의해야 합니다. DAD 작성 시 중요한 개념은, DAD는 사용자가 DB2 테이블에 색인화할 요소 및 속성 값을 맵핑하는 데 사용되므로 위치 경로 구문을 이해하는 것입니다. 위치 경로와 구문에 대한 자세한 정보는 58 페이지의 『위치 경로』를 참조하십시오.

DAD 파일 작성 또는 편집

DAD 파일을 지정할 때, 검색해야 하는 데이터의 속성과 키 요소를 정의합니다. XML Extender는 사용자가 데이터에 색인을 지정하여 신속하게 검색할 수 있도록, 이 정보를 사용하여 부가 테이블을 작성합니다. DAD 파일 작성에 대한 계획 사항에 대해서는 61 페이지의 『DAD 파일』을 참조하십시오.

시작하기 전에

- 색인 지정 및 빠른 검색을 위해 키 요소 및 속성을 정의할 수 있도록 XML 데이터의 계층적 구조를 이해하십시오.
- XML 문서의 DTD를 준비하고 DTD_REF 테이블로 삽입합니다. 이 단계는 유효성 검증에 필수입니다.

관리 마법사 사용

다음 단계에서 DAD 파일을 작성하십시오.

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.

2. XML DAD 파일을 편집하거나 작성하려면 LaunchPad 창에서 **DAD 파일에 대한 작업**을 누르십시오. DAD 파일 지정 창이 열립니다.
3. 기존 DAD 파일을 편집할 것인지 또는 새 DAD 파일을 작성할 것인지 선택하십시오.
   - **기존 DAD를 편집하려면, 다음과 같이 하십시오.**
     a. **...**을 눌러 풀다운 메뉴에서 기존 DAD 파일에 대해 찾아보거나 **파일 이름** 필드에 DAD 파일 이름을 입력하십시오.
     b. 마법사가 지정된 DAD 파일을 인식하는지 검증하십시오.
        – 마법사가 지정된 DAD 파일을 인식하면, **다음**을 선택할 수 있고, XML 컬럼이 **유형** 필드에 표시됩니다.
        – 마법사가 지정된 DAD 파일을 인식하지 않으면, **다음**을 선택할 수 없습니다. **파일 이름** 필드에 DAD 파일 이름을 다시 입력하거나 **열기**를 눌러 기존의 DAD 파일을 다시 찾아보십시오. **다음**을 사용할 수 있을 때까지 계속하십시오.
     c. **다음**을 누르십시오.
   - **새 DAD를 작성하려면, 다음과 같이 하십시오.**
     a. **파일 이름** 필드를 공백으로 두십시오.
     b. **유형** 메뉴에서 **XML 컬럼**을 누르십시오.
     c. **다음**을 누르십시오.
4. 유효성 검증 선택 창에서 DTD로 XML 문서를 유효성 검증하려는지 선택하십시오.
   - 유효성을 검증하려면, 다음과 같이 하십시오.
     a. **DAD로 XML 문서의 유효성 검증**을 누르십시오.
     b. **DTD ID** 메뉴에서 유효성 검증에 사용할 DTD를 선택하십시오.

     데이터베이스에 대한 DTD 저장소로 DTD를 가져오지 않았으면, XML 문서를 유효성 검증할 수 없습니다.
   - XML 문서의 유효성을 검증하지 않고 계속 수행하려면 **DAD로 XML 문서의 유효성을 검증하지 않음**을 누르십시오.
5. **다음**을 누르십시오.

6. 새 부가 테이블을 추가할 것인지, 기존의 부가 테이블을 편집할 것인지 또는 부가 테이블 창에서 기존의 부가 테이블을 제거하려는지 선택하십시오.
   - **새 부가 테이블 또는 부가 테이블 컬럼을 추가하려면, 다음과 같이 하십시오.**

     새 부가 테이블을 추가하려면 테이블에서 컬럼을 정의합니다. 부가 테이블의 각 컬럼마다 다음 단계를 완료하십시오.

     a. 부가 테이블 창의 **세부사항** 상자 필드를 채우십시오.

        1) **테이블 이름**: 컬럼이 들어 있는 테이블 이름을 입력합니다. 예를 들면 다음과 같습니다.

           ```
           ORDER_SIDE_TAB
           ```

        2) **컬럼 이름**: 컬럼 이름을 입력합니다. 예를 들면 다음과 같습니다.

           ```
           CUSTOMER_NAME
           ```

        3) **유형**: 메뉴에서 컬럼 유형을 선택합니다. 예를 들면 다음과 같습니다.

           ```
           XMLVARCHAR
           ```

        4) **길이(VARCHAR 유형만 해당)**: 최대 VARCHAR 글자를 입력합니다. 예를 들면 다음과 같습니다.

           ```
           30
           ```

        5) **경로**: 요소 또는 속성의 위치 경로를 입력하십시오. 예를 들면 다음과 같습니다.

           ```
           /ORDER/CUSTOMER/NAME
           ```

           위치 경로 구문에 대해서는 58 페이지의 『위치 경로』를 참조하십시오.

        6) **다중 발생**: 메뉴에서 **아니오** 또는 **예**를 선택합니다.

           이 요소 또는 속성의 위치 경로가 문서에서 두 번 이상 사용될 수 있는지 여부를 나타냅니다.

           중요사항: 컬럼에 대해 다중 발생을 지정할 경우, 컬럼이 포함된 부가 테이블에서 한 컬럼만 지정할 수 있습니다.

     b. **추가**를 눌러 컬럼을 추가하십시오.

c. 부가 테이블에 대해 컬럼 추가, 편집 또는 제거를 계속하거나 **다음**을 누르십시오.

- **기존 부가 테이블 컬럼을 편집하려면, 다음과 같이 하십시오.**

  기존 컬럼의 정의를 변경하여 부가 테이블을 갱신할 수 있습니다.

  a. 편집하려는 부가 테이블 및 컬럼 이름을 누르십시오.

  b. **세부사항** 상자의 필드를 편집하십시오.

  c. **변경**을 눌러 변경사항을 저장하십시오.

  d. 각 부가 테이블에 대해 컬럼 추가, 편집 또는 제거를 계속하거나, **다음**을 누르십시오.

- **기존 부가 테이블 컬럼을 제거하려면, 다음과 같이 하십시오.**

  a. 제거하려는 부가 테이블 및 컬럼을 누르십시오.

  b. **제거**를 누르십시오.

  c. 부가 테이블 컬럼을 추가, 편집 제거를 계속하거나, **다음**을 누르십시오.

- **기존 부가 테이블을 제거하려면, 다음과 같이 하십시오.**

  전체 부가 테이블을 제거하려면 테이블에 있는 각 컬럼을 삭제합니다.

  a. 제거하려는 테이블의 각 부가 테이블 컬럼을 누르십시오.

  b. **제거**를 누르십시오.

  c. 부가 테이블 컬럼 추가, 편집 또는 제거를 계속하거나, **다음**을 누르십시오.

7. DAD 지정 창의 **파일 이름** 필드에 수정된 DAD 파일의 출력 파일 이름을 입력하십시오.
8. **완료**를 눌러 DAD 파일을 저장하고 LaunchPad 창으로 되돌아가십시오.

DB2 명령 쉘에서

DAD 파일은 모든 텍스트 편집기에서 작성될 수 있는 XML 파일입니다.

다음 단계에서 DAD 파일을 작성하십시오.

1. 텍스트 편집기를 여십시오.
2. 다음 구문을 사용하여 DAD 파일 머리글을 작성하십시오.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "path\dtd\dad.dtd"> --> DAD 파일용 DTD의
                                            경로 및 파일 이름
```

3. <DAD></DAD> 태그를 삽입하십시오.
4. <DAD> 태그 안에서는, DAD 파일을 유효성 검증을 위해 XML 문서 DTD와 연결시키는 DTDID를 선택적으로 지정하십시오.

   ```
   <dtdid>path\dtd_name.dtd</dtdid> --> 사용자 응용프로그램에
                                        대한 DTD의
                                        DTD의 경로 및 파일 이름
   ```

   DTD ID는 유효성 검증을 위해 필요하며, DTD를 DTD 참조 테이블(db2xml.DTD_REF)에 삽입할 때 사용한 DTD ID 값과 일치해야 합니다.
5. 유효성 검증 여부(즉, DTD를 사용하여 XML 문서가 유효한 XML 문서인지 확인함)를 지정합니다. 예를 들면 다음과 같습니다.

   ```
   <validation>YES</validation>   --> YES 또는 NO를 지정하십시오.
   ```

   Yes를 지정하면, DTD를 DTD_REF 테이블에 삽입할 뿐만 아니라 이전 단계에서 DTDID를 지정했어야 합니다.
6. <Xcolumn> 요소를 사용하여 액세스 및 저장영역 메소드를 XML 컬럼으로 정의하십시오.

   ```
   <Xcolumn>
     </Xcolumn>
   ```

7. 구조적인 검색을 위해 색인을 지정하도록 각 부가 테이블 및 중요 요소와 속성을 정의하십시오. 각 테이블에 대해 다음 단계를 수행하십시오. 다음 단계에서는 297 페이지의 『DAD 파일: XML 컬럼』에서 표시된 샘플 DAD 파일에서 사용한 예를 사용합니다.
   a. <TABLE></TABLE> 태그와 이름 속성을 삽입하십시오.

      ```
      <table name="order_tab">
           </table>
      ```

   b. <TABLE> 태그 다음에 테이블에 있는 각 컬럼에 대해 <COLUMN> 태그와 그 속성을 삽입하십시오.
      - ***name***: 컬럼 이름
      - ***type***: 컬럼 유형

- ***path***: 요소 또는 속성의 위치 경로. 위치 경로 구문에 대해서는 58 페이지의 『위치 경로』를 참조하십시오.
- ***multi_occurrence***: 이 요소나 속성이 문서에서 한 번 이상 사용될 수 있는지 여부를 나타냅니다.

```
<table ...>
        <column name="order_key"
         type="integer"
         path="/Order/@key"
         multi_occurrence="NO"/>
        <column name="customer"
         type="varchar(50)"
         path="/Order/Customer/Name"
         multi_occurrence="NO"/>
    </table>
```

8. 마지막 컬럼 정의 다음에 종료 </TABLE> 태그가 있는지 확인하십시오.
9. 마지막 </TABLE> 태그 다음에 종료 </Xcolumn> 태그가 있는지 확인하십시오.
10. </Xcolumn> 태그 다음에 종료 </DAD> 태그가 있는지 확인하십시오.

XML 테이블 작성 또는 변경

본래 XML 문서를 테이블에 저장하려면, 테이블이 XML 사용자 정의 유형(UDT)의 컬럼을 포함하도록 테이블을 작성하거나 변경해야 합니다. 테이블은 *XML 테이블*이며, XML 문서가 포함된 테이블입니다. 테이블은 변경된 테이블이나 새 테이블이 될 수 있습니다. 테이블에 XML 유형의 컬럼이 있으면, XML용으로 컬럼을 사용할 수 있습니다.

관리 마법사를 사용하거나 DB2 명령 쉘을 사용하여 XML 유형의 컬럼이 있는 기존 테이블을 변경할 수 있습니다.

관리 마법사 사용

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. LaunchPad 창에서 **XML 컬럼에 대한 작업**을 누르십시오. 타스크 선택 창이 열립니다.
3. **XML 컬럼 추가**를 누르십시오. XML 컬럼 추가 창이 열립니다.

4. **테이블 이름** 풀다운 메뉴에서 테이블의 이름을 선택하거나, 변경하려는 테이블의 이름을 입력하십시오. 예를 들면 다음과 같습니다.

   SALES_DB

5. **컬럼 이름** 필드에서 테이블에 추가할 컬럼의 이름을 입력하십시오. 예를 들면 다음과 같습니다.

   ORDER

6. **컬럼 유형** 풀다운 메뉴에서 컬럼에 대한 UDT를 선택하십시오. 예를 들면 다음과 같습니다.

   XMLVARCHAR

7. **완료**를 눌러 XML 유형의 컬럼을 추가하십시오.

DB2 명령 쉘에서

CREATE TABLE 또는 ALTER TABLE 문의 컬럼 절에서 XML 유형의 컬럼이 있는 테이블을 작성하거나 변경하십시오.

예: 영업 응용프로그램에서, SALES_TAB 응용프로그램 테이블의 ORDER 컬럼에 XML 형식의 행 항목 주문을 저장하고자 합니다. 이 테이블에는 INVOICE_NUM 및 SALES_PERSON 컬럼도 있습니다. 작은 주문이므로, XMLVARCHAR 유형을 사용하여 이를 저장합니다. 기본 키는 INVOICE_NUM입니다. 다음 CREATE TABLE 명령문은 XML 유형의 컬럼이 있는 테이블을 작성합니다.

```
CREATE TABLE sales_tab(
             invoice_num  char(6) NOT NULL PRIMARY KEY,
             sales_person     varchar(20),
             order            XMLVarchar);
```

XML 컬럼 사용

DB2 데이터베이스에서 XML 문서를 저장하려면, XML용으로 컬럼을 사용할 수 있어야 합니다. 컬럼을 신속하게 검색할 수 있도록 색인지정하기 위해 컬럼을 사용하게 하는 것입니다. XML Extender 관리 마법사나 DB2 명령 쉘을 사용하여 컬럼을 사용할 수 있도록 합니다. 컬럼은 XML 유형의 것이어야 합니다.

XML Extender가 XML 컬럼을 사용할 수 있으면, 다음을 수행합니다.

- 선택적으로 DAD 파일을 읽습니다.

  - DAD 파일용 DTD에 대해 DAD 파일의 유효성을 검증합니다.
  - 지정되면 DTD_REF 테이블에서 DTDID를 검색합니다.
  - XML 컬럼에서 색인화할 부가 테이블을 작성합니다.
  - XML 데이터가 포함될 컬럼을 준비합니다.
- 선택적으로 XML 테이블 및 부가 테이블의 *기본 보기*를 작성합니다.
- ROOT ID 값이 지정되지 않았으면 이를 지정합니다.

XML 컬럼을 사용 가능하면, 다음을 수행할 수 있습니다.

- 부가 테이블에 색인 작성
- XML 문서를 XML 컬럼에 삽입
- XML 컬럼에 있는 XML 문서 조회, 갱신 또는 검색

시작하기 전에

XML UDT 컬럼이 있는 DB2 테이블을 작성하거나 변경하여 XML 테이블을 작성하십시오.

관리 마법사 사용

다음 단계에서 XML 컬럼을 사용 가능하게 하십시오.

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. XML Extender 컬럼 관련 타스크를 보려면 LaunchPad 창에서 **XML 컬럼에 대한 작업**을 누르십시오. 타스크 선택 창이 열립니다.
3. **컬럼 사용**을 누른 후, **다음**을 눌러 데이터베이스에 있는 기존 테이블 컬럼을 사용 가능하게 하십시오.
4. **테이블 이름** 필드에서 XML 컬럼이 포함된 테이블을 선택하십시오. 예를 들면 다음과 같습니다.

   ```
   SALES_TAB
   ```

5. **컬럼 이름** 필드에서 사용할 수 있는 컬럼을 선택하십시오. 예를 들면 다음과 같습니다.

   ```
   ORDER
   ```

   컬럼은 XML 유형으로 있어야 합니다.

6. DAD 경로 및 파일 이름을 **DAD 파일 이름** 필드에 입력하거나, **...**을 눌러 기존 DAD 파일을 찾아보십시오. 예를 들면 다음과 같습니다.

   ```
   c:\dxx\samples\dad\getstart.dad
   ```

7. 선택적으로, 기존의 테이블 공간의 이름을 **테이블 공간** 필드에 입력하십시오.

   테이블 공간에는 XML Extender가 작성했던 부가 테이블이 포함됩니다. 테이블 공간을 지정하면, 지정된 테이블 공간에 부가 테이블이 작성됩니다. 테이블 공간을 지정하지 않을 경우, 부가 테이블은 기본 테이블 공간에 작성됩니다.

8. 선택적으로 **기본 뷰** 필드에 기본 뷰의 이름을 입력하십시오.

   지정된 경우, 기본 뷰는 컬럼이 사용 가능할 때 자동으로 작성되고 XML 테이블과 모든 관련 부가 테이블을 조인합니다.

9. 선택적으로, **루트 ID** 필드에 응용프로그램 테이블에 있는 기본 키의 컬럼 이름을 입력하십시오. 입력하는 것이 좋습니다.

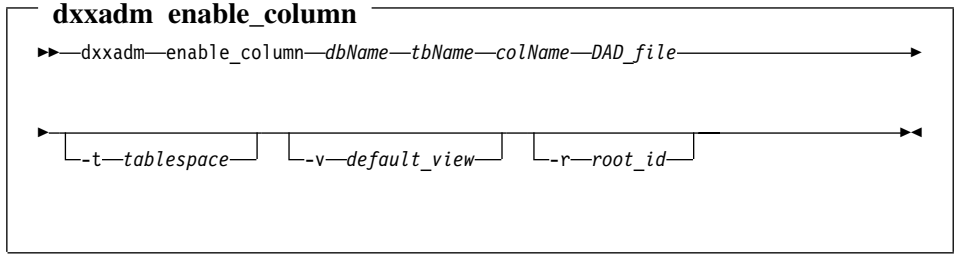
   XML Extender는 ROOT ID 값을 고유 식별자로 사용하여 모든 부가 테이블을 응용프로그램 테이블과 연결합니다. 지정되지 않았으면, XML Extender는 DXXROOT_ID 컬럼을 응용프로그램 테이블에 추가하고, 식별자를 생성합니다.

10. **완료**를 눌러 XML 컬럼을 사용 가능하게 하고, 부가 테이블을 작성한 후, LaunchPad 창으로 되돌아가십시오.
    - 컬럼을 사용할 수 있게 되면, 컬럼 사용 성공 메시지가 표시됩니다.
    - 컬럼을 사용할 수 없게 되면, 오류 상자가 표시됩니다. 컬럼을 정상적으로 사용할 수 있을 때까지 입력 필드의 값을 정정하십시오.

DB2 명령 쉘에서

XML 컬럼을 사용 가능하게 하려면 다음 명령을 입력하십시오.

구문:



매개변수:

dbName

데이터베이스 이름.

tbName

사용할 수 있게 될 컬럼이 들어있는 테이블 이름.

colName

사용할 수 있게 될 XML 컬럼 이름.

DAD_file

문서 액세스 정의(DAD)가 포함되는 파일 이름.

tablespace

XML Extender가 작성했던 부가 테이블이 포함된, 이전에 작성했던 테이블 공간. 지정되지 않았으면 기본 테이블 공간이 사용됩니다.

default_view

선택적. 응용프로그램 테이블 및 관련된 모든 부가 테이블을 조인하기 위해 XML Extender가 작성했던 기본 뷰의 이름.

root_id

선택적. 응용프로그램 테이블에 있는 기본 키의 컬럼 이름 및 모든 부가 테이블을 응용프로그램 테이블과 연결하는 고유 식별자. XML Extender는 *root_id*를 고유 식별자로서 사용하여, 모든 부가 테이블을 응용프로그램 테이블과 연결합니다. ROOT ID를 지정하는 것이 바람직합니다. ROOT ID를 지정하지 않으면, XML Extender는 DXXROOT_ID 컬럼을 응용프로그램 테이블에 추가하고 식별자를 생성합니다.

제한사항: 응용프로그램 테이블에 DXXROOT_ID의 컬럼 이름이 있지만 이 컬럼에 *root_id* 값이 들어있지 않은 경우, *root_id* 매개변수를 지정해야 합니다. 그렇지 않으면 오류가 발생합니다.

예: 다음 예에서는 DB2 명령 쉘을 사용하여 컬럼을 사용할 수 있도록 합니다. DAD 파일 및 XML 문서는 295 페이지의 『부록B. 샘플』에 있습니다.

```
dxxadm enable_column SALES_DB sales_tab order getstart.dad
            -v sales_order_view -r invoice_num
```

이 예에서, 컬럼 ORDER는 테이블 SALES_DB.SALES_TAB에서 사용할 수 있습니다. DAD 파일은 getstart.dad이고, 기본 뷰는 sales_order_view이며, ROOT ID는 INVOICE_NUM입니다.

이 예를 사용할 경우 SALES_TAB 테이블의 스키마는 다음과 같습니다.

컬럼 이름	**INVOICE_NUM**	**SALES_PERSON**	**ORDER**
데이터 유형	CHAR(6)	VARCHAR(20)	XMLVARCHAR

다음 부가 테이블은 DAD 스펙을 기초로 작성됩니다.

ORDER_SIDE_TAB:

컬럼 이름	**ORDER_KEY**	**CUSTOMER**	**INVOICE_NUM**
데이터 유형	INTEGER	VARCHAR(50)	CHAR(6)
경로 표현식	/Order/@key	/Order/Customer/Name	N/A

PART_SIDE_TAB:

컬럼 이름	**PART_KEY**	**PRICE**	**INVOICE_NUM**
데이터 유형	INTEGER	DOUBLE	CHAR(6)
경로 표현식	/Order/Part/@key	/Order/Part/ExtendedPrice	N/A

SHIP_SIDE_TAB:

컬럼 이름	**DATE**	**INVOICE_NUM**
데이터 유형	DATE	CHAR(6)

컬럼 이름	DATE	INVOICE_NUM
경로 표현식	/Order/Part/Shipment/ShipDate	N/A

ROOT ID가 응용프로그램 테이블에 있는 기본 키 INVOICE_NUM으로 지정되므로, 모든 부가 테이블에는 같은 유형의 INVOICE_NUM 컬럼이 있습니다. 컬럼을 사용할 수 있게 된 다음에는 INVOICE_NUM 값이 부가 테이블에 삽입됩니다. XML 컬럼 ORDER를 사용할 수 있게 되면 *default_view* 매개변수를 지정하여 기본 뷰 sales_order_view를 작성합니다. 뷰는 다음 명령문을 사용하여 위 테이블을 조인합니다.

```
CREATE VIEW sales_order_view(invoice_num, sales_person, order,
                             order_key, customer, part_key, price, date)
     AS
     SELECT sales_tab.invoice_num, sales_tab.sales_person, sales_tab.order,
            order_tab.order_key, order_tab.customer,
            part_tab.part_key, part_tab.price,
            ship_tab.date
     FROM sales_tab, order_tab, part_tab, ship_tab
     WHERE sales_tab.invoice_num = order_tab.invoice_num
               AND sales_tab.invoice_num = part_tab.invoice_num
               AND sales_tab.invoice_num = ship_tab.invoice_num
```

테이블 공간이 **enable_column** 명령에서 지정되면, 부가 테이블은 지정된 테이블 공간에서 작성됩니다. 테이블 공간이 지정되지 않으면, 기본 테이블 공간에서 부가 테이블이 작성됩니다.

부가 테이블 색인화

XML 컬럼을 사용 가능하게 하고 부가 테이블을 작성한 후에는 부가 테이블을 색인화할 수 있습니다. 부가 테이블에는 DAD 파일을 작성할 때 지정한 컬럼에 XML 데이터가 들어 있습니다. 이 테이블을 색인화하면, XML 문서에 대한 조회의 성능을 향상시킬 수 있습니다.

시작하기 전에

- XML 문서 구조에 대한 부가 테이블을 지정하는 DAD 파일을 작성하십시오.
- 부가 테이블을 작성하는 DAD 파일을 사용하여 XML 컬럼을 사용 가능하게 하십시오.

DB2 CREATE INDEX 명령

DB2 CREATE INDEX 명령을 사용하십시오.

예:

다음 예는 4개의 부가 테이블에서 색인을 작성합니다.

```
DB2 CREATE INDEX KEY_IDX
      ON ORDER_SIDE_TAB(ORDER_KEY)
DB2 CREATE INDEX CUSTOMER_IDX
      ON ORDER_SIDE_TAB(CUSTOMER)
DB2 CREATE INDEX PRICE_IDX
      ON PART_SIDE_TAB(PRICE)
DB2 CREATE INDEX DATE_IDX
      ON SHIP_SIDE_TAB(DATE)
```

XML 컬럼 사용 안함

XML 컬럼에 대해 DAD 파일을 갱신해야 하거나, XML 컬럼을 삭제하거나 이 컬럼이 있는 테이블을 삭제하려면, 컬럼을 사용 불가능하게 하십시오. 컬럼을 사용할 수 없게 된 후에는 갱신된 DAD 파일을 사용하여 컬럼을 다시 사용 가능하게 하거나 컬럼 또는 다른 타스크를 삭제할 수 있습니다. XML Extender 관리 마법사를 사용하거나 DB2 명령 쉘을 사용하여 컬럼을 사용 불가능하게 할 수 있습니다.

XML Extender가 XML 컬럼을 사용할 수 있으면, 다음을 수행합니다.

- XML_USAGE 테이블에서 컬럼 항목을 삭제합니다.
- 이 컬럼과 연결된 부가 테이블을 제거합니다.

중요사항: 먼저 컬럼을 사용 불가능하게 하지 않고 XML 컬럼이 있는 테이블을 제거하면, XML Extender가 XML 컬럼과 연결된 부가 테이블을 제거할 수 없으며, 예상치 못한 결과를 얻을 수 있습니다.

시작하기 전에

사용 불가능한 XML 컬럼이 현재 DB2 데이터베이스에 있는지 확인하십시오.

관리 마법사 사용

XML 컬럼을 사용할 수 없게 하려면, 다음 단계를 사용하십시오.

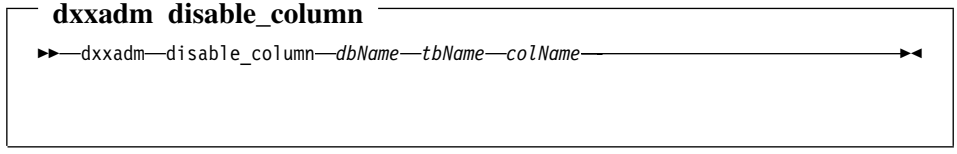
1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. XML Extender 컬럼 관련 타스크를 보려면 LaunchPad 창에서 **XML 컬럼에 대한 작업**을 누르십시오. 타스크 선택 창이 열립니다.
3. **컬럼 사용 안함**을 누른 후, **다음**을 눌러 데이터베이스에 있는 기존 테이블 컬럼을 사용 불가능하게 하십시오.
4. **테이블 이름** 필드에서 XML 컬럼이 포함된 테이블을 선택하십시오.
5. **컬럼 이름** 필드에서 사용 불가능하게 할 컬럼을 선택하십시오.
6. **완료**를 누르십시오.
   - 컬럼이 사용 불가능하게 되면, 컬럼이 사용 불가능하게 됨 메시지가 표시됩니다.
   - 컬럼이 사용 불가능하게 되지 않으면, 오류 상자가 표시됩니다. 컬럼이 완전히 사용 불가능하게 될 때까지 전체 필드의 값을 수정하십시오.

DB2 명령 쉘에서

XML 컬럼을 사용 불가능하게 하려면 다음 명령을 입력하십시오.

구문:

dxxadm disable_column



매개변수:

dbName
데이터베이스 이름.

tbName
사용 불가능하게 될 컬럼이 있는 테이블의 이름.

colName
사용 불가능하게 되는 XML 컬럼의 이름.

예: 다음 예에서는 DB2 명령 쉘을 사용하여 컬럼을 사용 불가능하게 합니다. DAD 파일 및 XML 문서는 295 페이지의 『부록B. 샘플』에 있습니다.

```
dxxadm disable_column SALES_DB sales_tab order
```

이 예에서, ORDER 컬럼은 SALES_DB.SALES_TAB 테이블에서 사용 불가능합니다.

컬럼이 사용 불가능하면, 부가 테이블이 제거됩니다.

XML 콜렉션에 대한 작업

XML 콜렉션을 설정하려면 맵핑 스킴을 작성하고, 선택적으로 DB2 테이블과 DAD 파일을 연결하는 가상 이름으로 콜렉션을 사용할 수 있어야 합니다.

XML 콜렉션을 사용 가능하게 할 필요가 없는 경우라도 이렇게 하면 성능이 향상됩니다.

맵핑 스킴에 대한 DAD 파일 작성 또는 편집

XML 콜렉션을 사용할 경우 DAD 파일을 작성해야 합니다. DAD 파일에서는 XML 데이터와 여러 관계형 테이블간의 관계를 정의합니다. XML Extender는 DAD 파일을 사용하여 다음을 수행합니다.

- 관계형 데이터에서 XML 문서 작성
- XML 문서를 관계형 데이터로 분해

두 개의 메소드, 즉, SQL 맵핑 및 RDB_node 맵핑 중 하나를 사용하여 XML 테이블과 DB2 테이블 간의 데이터를 맵핑할 수 있습니다.

SQL 맵핑

SQL문 요소를 사용하여 XML 데이터를 포함시키는 데 사용되는 테이블 및 컬럼에 대해 SQL 조회를 지정합니다. SQL 맵핑은 XML 문서를 작성하는 데만 사용할 수 있습니다.

RDB_node 맵핑

XML Extender 고유의 요소인 관계형 데이터베이스 노드 또는 RDB_node를 사용하며, 이는 XML 데이터에 대한 테이블, 컬럼, 조건 및

순서를 지정합니다. RDB_node 맵핑에서는 SQL문이 제공할 수 있는 것보다 복잡한 맵핑을 지원합니다. RDB_node 맵핑은 XML 문서를 작성하고 분해하는 데 사용할 수 있습니다.

두 맵핑 메소드 모두 63 페이지의 『DAD 파일』에 설명된 *XPath 데이터 모델*을 사용합니다.

시작하기 전에

- DB2 테이블과 XML 문서 사이의 관계를 맵핑합니다. 이 단계에는 XML 문서 계층 맵핑과, 문서에 있는 데이터를 DB2 테이블에 맵핑하는 방식 정의가 포함됩니다.
- XML 문서의 유효성을 검증하려면, 작성 또는 분해 중인 XML 문서에 대한 DTD를 DTD 참조 테이블 db2xml.DTD_REF에 삽입하십시오.

SQL 맵핑을 사용하여 XML 문서 작성

XML 문서를 작성할 때 SQL을 사용하려면 SQL 맵핑을 사용하십시오.

관리 마법사 사용: 다음 단계를 사용하여 XML 콜렉션 SQL 맵핑을 사용하여 DAD 파일을 작성하십시오.

SQL 맵핑을 사용하여 작성을 위한 DAD 파일을 작성하려면, 다음과 같이 하십시오.

XML 문서를 작성할 때 XML 문서에 있는 데이터를 가져올 테이블과 컬럼을 정의하기 위해 SQL문을 사용하려면, SQL 맵핑을 사용하십시오.

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. LaunchPad 창에서 **DAD 파일에 대한 작업**을 누르십시오. DAD 지정 창이 표시됩니다.
3. 기존 DAD 파일을 편집할 것인지 또는 새 DAD 파일을 작성할 것인지 선택하십시오.

   새 DAD 파일 작성하기:

   a. **파일 이름** 필드를 공백으로 두십시오.

   b. **유형** 메뉴에서, **XML 콜렉션 SQL 맵핑**을 선택하십시오.

c. **다음**을 눌러 유효성 검증 선택 창을 여십시오.

기존 DAD 파일 편집하기:

a. DAD 파일 이름을 **파일 이름** 필드에 입력하거나, **...**을 눌러 기존 DAD 파일을 찾아보십시오.

b. 마법사가 지정된 DAD 파일을 인식하는지 검증하십시오.

- 마법사가 지정된 DAD 파일을 인식하면, **다음**을 선택할 수 있고, XML 콜렉션 SQL 맵핑이 **유형** 필드에 표시됩니다.
- 마법사가 지정된 DAD 파일을 인식하지 않으면, **다음**을 선택할 수 없습니다. DAD 파일 이름을 다시 입력하거나, **...**을 눌러 기존 DAD 파일을 다시 찾아보십시오. **다음**을 선택할 수 있을 때까지 입력 필드 값을 정정하십시오.

c. **다음**을 눌러 유효성 검증 선택 창을 여십시오.

4. 유효성 검증 선택 창에서 DTD로 XML 문서를 유효성 검증할 것인지 선택하십시오.

- 유효성을 검증하려면, 다음과 같이 하십시오.

  a. **DAD로 XML 문서의 유효성 검증**을 누르십시오.

  b. **DTD ID** 메뉴에서 유효성 검증에 사용할 DTD를 선택하십시오.

  데이터베이스에 대한 DTD 저장소로 DTD를 가져오지 않았으면, XML 문서를 유효성 검증할 수 없습니다.

- XML 문서의 유효성을 검증하지 않고 계속 수행하려면 **DAD로 XML 문서의 유효성을 검증하지 않음**을 누르십시오.

5. **다음**을 눌러 텍스트 지정 창을 여십시오.

6. 작성할 XML 문서의 프롤로그를 지정하려면 **프롤로그** 필드에 프롤로그 이름을 입력하십시오.

```
<?xml version="1.0"?>
```

기존의 DAD를 편집중인 경우는, **프롤로그** 필드에 프롤로그가 자동으로 표시됩니다.

7. 텍스트 지정 창의 **Doctype** 필드에 XML 문서의 문서 유형을 입력하여 XML 문서에 대한 DTD를 지정하십시오. 예를 들면 다음과 같습니다.

```
!DOCTYPE DAD SYSTEM "c:\dxx\samples\dtd\getstart.dtd"
```

기존의 DAD를 편집중인 경우는, **Doctype** 필드에 문서 유형이 자동으로 표시됩니다.

8. **다음**을 눌러 SQL문 지정 창을 여십시오.
9. **SQL문** 필드에 올바른 SQL SELECT 문을 입력하십시오. 예를 들면 다음과 같습니다.

```
SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
price, tax, ship_id, date, mode from order_tab o, part_tab p,
table (select substr(char(timestamp(generate_unique())),16)
  as ship_id, date, mode, part_key from ship_tab) s
     WHERE o.order_key = 1 and
              p.price > 20000 and
           p.order_key = o.order_key and
           s.part_key = p.part_key
     ORDER BY order_key, part_key, ship_id
```

기존의 DAD를 편집중인 경우는, **SQL문** 필드에 SQL문이 자동으로 표시됩니다.

10. **SQL 테스트**를 눌러 SQL문의 유효성을 테스트하십시오.
    - SQL문이 유효하면, 샘플 결과가 **샘플 결과** 필드에 표시됩니다.
    - SQL문이 유효하지 않으면, 오류 메시지가 **샘플 결과** 필드에 표시됩니다. 오류 메시지는 SQL SELECT문을 정정하고 다시 시도하도록 알려줍니다.
11. **다음**을 눌러 SQL 맵핑 창을 여십시오.
12. SQL 맵핑 창의 왼쪽에 있는 필드에서, 맵핑할 요소 또는 속성 노드를 눌러 선택하십시오.

    XML 문서에 있는 요소 및 속성들을 DB2 데이터에 해당하는 요소 및 속성 노드에 맵핑합니다. 이러한 노드에서는 XML 데이터에서 DB2 데이터로의 경로를 제공합니다.

    - **루트 노드를 추가하려면, 다음과 같이 하십시오.**
      a. **루트** 아이콘을 선택하십시오.
      b. **새 요소**를 눌러 새 노드를 정의하십시오.
      c. **세부사항** 상자에서 **노드 유형**을 **요소**로 지정하십시오.
      d. **노드 이름** 필드에 최상위 레벨 노드의 이름을 입력하십시오.
      e. **추가**를 눌러 새 노드를 작성하십시오.

맵에 있는 다른 모든 요소 및 속성 노드의 상위가 되는 루트 노드 또는 요소를 작성했습니다. 이제 하위 요소 및 속성을 이 노드에 추가할 수 있습니다.

- **하위 요소 또는 속성 노드를 추가하려면, 다음과 같이 하십시오.**
  a. 하위 요소 또는 속성을 추가하려면 왼쪽 필드에 있는 상위 노드를 누르십시오.

     상위 노드를 선택하지 않으면 **새 노드**를 선택할 수 없습니다.

  b. **새 요소**를 누르십시오.

  c. **세부사항** 상자에 있는 **노드 유형** 메뉴에서 노드 유형을 선택하십시오.

     노드 유형 메뉴에는 맵의 해당 지점에서 유효한 노드 유형만 표시됩니다.

     요소 XML 문서와 연결된 DTD에 정의되어 있는 XML 요소를 나타냅니다. XML 요소를 DB2 테이블의 컬럼과 연결할 때 사용됩니다. 요소 노드에는 속성 노드, 하위 요소 노드 또는 텍스트 노드가 있을 수 있습니다. 최하위 레벨의 노드는 텍스트 노드 및 그와 연결된 컬럼 이름을 트리 보기에 갖습니다.

     속성 XML 문서와 연결된 DTD에 정의되어 있는 XML 속성을 나타냅니다. XML 속성을 DB2 테이블의 컬럼과 연결할 때 사용됩니다. 속성 노드에는 트리 뷰의 텍스트 노드 및 이와 연결된 컬럼 이름이 있습니다.

     텍스트 관계형 테이블에 맵핑될 내용을 가진 요소 또는 속성 노드에 대한 텍스트 내용을 지정합니다. 텍스트 노드에는 트리 뷰의 이와 연결된 컬럼 이름이 있습니다.

     테이블 관계형 테이블에 맵핑될 요소 또는 속성 값에 대한 테이블 이름을 지정합니다.

     컬럼 관계형 테이블에 맵핑될 요소 또는 속성 값에 대한 컬럼 이름을 지정합니다.

     조건 컬럼에 대한 조건을 지정합니다.

  d. **세부사항** 상자의 **노드 이름** 필드에 노드 이름을 입력하십시오. 예를 들면 다음과 같습니다.

주문

e. 최하위 레벨 요소에 대한 **속성**, **요소** 또는 **텍스트**를 노드 유형으로 지정했으면, **세부사항** 상자에 있는 **컬럼** 필드에서 컬럼을 선택하십시오. 예를 들면 다음과 같습니다.

```
Customer_Name
```

제한사항: 관리 마법사를 사용하면 새 컬럼을 작성할 수 없습니다. **컬럼**을 노드 유형으로 지정하면, DB2 데이터베이스에 이미 있는 컬럼만 선택할 수 있습니다.

f. **추가**를 눌러 새 노드를 추가하십시오.

왼쪽에 있는 필드에서 노드를 누르고, **세부사항** 상자에서 필요한 만큼 수정하여 나중에 노드를 수정할 수 있습니다. 요소를 갱신하려면 **변경**을 누르십시오.

추가 프로세스를 반복하고 있는 노드를 강조표시하여 하위 요소 또는 속성을 노드에 추가할 수도 있습니다.

g. SQL 맵 편집을 계속하거나, **다음**을 눌러 DAD 지정 창을 여십시오.

- **노드를 제거하려면, 다음과 같이 하십시오.**

a. 왼쪽에 있는 필드에서 노드를 누르십시오.

b. **제거**를 누르십시오.

c. SQL 맵 편집을 계속하거나, **다음**을 눌러 DAD 지정 창을 여십시오.

최하위 레벨 노드를 제거할 때, 다른 요소가 최하위 레벨이 되고 이에 대한 컬럼 이름이 필요할 수 있다는 점을 주지하십시오.

13. DAD 지정 창의 **파일 이름** 필드에 수정된 DAD 파일의 출력 파일 이름을 입력하십시오.
14. **완료**를 눌러 LaunchPad 창으로 되돌아가십시오.

DB2 명령 쉘에서: XML 문서를 작성할 때 SQL을 사용하려면, SQL 맵핑 표기법을 사용하십시오.

DAD 파일은 모든 텍스트 편집기에서 작성될 수 있는 XML 파일입니다. 다음 단계에서는 샘플 부록 297 페이지의 『문서 액세스 정의 파일』의 조각을 보여줍니다. 광범위한 정보 및 문맥에 대해서는 이들 예를 참조하십시오.

1. 텍스트 편집기를 여십시오.
2. DAD 머리글을 작성하십시오.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> DAD에 대한 DTD의
                                        DAD
```

3. <DAD></DAD> 태그를 삽입하십시오.
4. <DAD> 태그 다음에, DAD 파일을 XML 문서 DTD와 연결시키는 DTD ID를 지정하십시오.

```
<dtdid>path\dtd_name.dtd> --> 응용프로그램에 대한
                                DTD의 경로 및 파일 이름
```

5. 유효성 검증 여부(즉, DTD를 사용하여 XML 문서가 유효한 XML 문서인지 확인함)를 지정합니다. 예를 들면 다음과 같습니다.

```
<validation>NO</validation>   --> YES 또는 NO를 지정하십시오.
```

6. <Xcollection> 요소를 사용하여 액세스 및 저장영역 메소드를 XML 콜렉션으로서 정의하십시오. 액세스 및 저장영역 메소드는 XML 문서에 DB2 테이블에 저장된 데이터의 내용이 있다고 정의합니다.

```
<Xcollection>
  </Xcollection>
```

7. DB2 테이블에서 데이터를 조회하거나 데이터를 삽입하려면 하나 이상의 SQL 문을 지정하십시오. 이에 대한 지침은 69 페이지의 『맵핑 스킴 요구사항』을 참조하십시오. 예를 들어, 다음 예에서처럼 단일 SQL 조회를 지정합니다.

```
    <SQL_stmt>
SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
price, tax, ship_id, date, mode from order_tab o, part_tab p,
table (select substr(char(timestamp(generate_unique())),16)
  as ship_id, date, mode, part_key from ship_tab) s
    WHERE o.order_key = 1 and
             p.price > 20000 and
          p.order_key = o.order_key and
          s.part_key = p.part_key
    ORDER BY order_key, part_key, ship_id
  </SQL_stmt>
```

8. 다음 프롤로그 정보를 추가하십시오.

```
     <prolog>?xml version="1.0"?</prolog>
```

   이 텍스트 그대로가 필요합니다.

9. <doctype></doctype> 태그를 추가하십시오. 예를 들면 다음과 같습니다.

```
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

10. <root_node></root_node> 태그를 사용하여 루트 노드를 정의하십시오. root_node 내부에서는 XML 문서를 구성하는 요소와 속성을 지정합니다.
11. XML 문서에 있는 요소 및 속성들을 DB2 데이터에 해당하는 요소 및 속성 노드에 맵핑합니다. 이러한 노드에서는 XML 데이터에서 DB2 데이터로의 경로를 제공합니다.
    a. DB2 테이블의 컬럼에 맵핑하는 XML 문서의 각 요소마다 <element_node>를 정의하십시오.

    ```
    <element_node name="name"></element_node>
    ```

    element_node는 다음 노드를 가질 수 있습니다.

    - attribute_node
    - 하위 element_node
    - text_node

    b. XML 문서에서 DB2 테이블에 있는 컬럼에 맵핑되는 각 속성에 대해 <attribute_node>를 정의하십시오. DAD 파일의 전체 구문을 제공하는 287 페이지의 『부록A. DAD 파일의 DTD』에 있는 DTD 파일의 DTD와 SQL 맵핑에 대한 이 절의 시작 부분에 있는 DTD 예를 참조하십시오.

    예를 들어, 요소 <Order>의 속성 키가 필요한 경우, 키 값은 PART_KEY에 저장되어 있습니다.

    DAD 파일: DAD 파일에서, 키에 속성 노드를 작성하고, 1 값이 저장될 테이블을 나타냅니다.

    ```
    <attribute_node name="key">
                        <column name="part_key"/>
                </attribute_node>
    ```

    작성된 XML 문서: 키 값은 PART_KEY 컬럼에 있습니다.

    ```
    <Order key="1">
    ```

12. DB2 테이블에서 가져올 내용이 있는 모든 요소나 속성마다 <text_node>를 작성하십시오. 텍스트 노드에는 내용을 제공할 컬럼을 지정하는 <column> 요소가 있습니다.

예를 들면, TAX 컬럼에서 가져올 값이 있는 XML 요소 <Tax>가 있을 수 있습니다.

DAD 요소:

```
<element_node name="Tax">
      <text_node>
         <column name="tax"/>
      </text_node>
   </element_node>
```

컬럼 이름은 DAD 파일의 시작 부분에 있는 SQL문에 있어야 합니다.

작성된 XML 문서

```
<Tax>0.02</Tax>
```

0.02 값은 TAX 컬럼에서 가져옵니다.

13. 마지막 </element_node> 태그 다음에 종료 </root_node> 태그가 있는지 확인하십시오.
14. </root_node> 태그 다음에 종료 </Xcollection> 태그가 있는지 확인하십시오.
15. </Xcollection> 태그 다음에 종료 </DAD> 태그가 있는지 확인하십시오.

RDB_node 맵핑을 사용하여 XML 문서 작성

XML과 같은 구조를 사용하여 XML 문서를 작성하려면, RDB_node 맵핑을 사용하십시오.

이 메소드는 <RDB_node>를 사용하여 요소 또는 속성 노드에 대한 DB2 테이블, 컬럼 및 조건을 지정합니다. <RDB_node>는 다음 요소를 사용합니다.

- <table>: 요소에 해당하는 테이블을 정의합니다.
- <column>: 해당하는 요소를 포함하는 컬럼을 정의합니다.
- <condition>: 컬럼에 대한 조건을 선택적으로 지정합니다.

<RDB_node>에서 사용되는 하위 요소는 노드의 문맥에 따라 다르며, 다음 규칙을 사용합니다.

노드 유형:	RDB 하위 요소 사용		
	테이블	컬럼	조건[1]
루트 요소	Y	N	Y
속성	Y	Y	선택적
텍스트	Y	Y	선택적

(1) 다중 테이블인 경우는 필수

관리 마법사 사용: RDB_node 맵핑을 사용하여 작성할 DAD를 작성하려면, 다음과 같이 하십시오.

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. LaunchPad 창에서 **DAD 파일에 대한 작업**을 누르십시오. DAD 지정 창이 표시됩니다.
3. 기존 DAD 파일을 편집할 것인지 또는 새 DAD를 작성할 것인지 선택하십시오.

   기존 DAD를 편집하려면, 다음과 같이 하십시오.

   a. **파일 이름** 필드에 DAD 파일 이름을 입력하거나, **...**을 눌러 기존 DAD를 찾아보십시오.

   b. 마법사가 지정된 DAD 파일을 인식하는지 검증하십시오.

      - 마법사가 지정된 DAD 파일을 인식하는 경우, **다음**을 선택할 수 있으며 XML 콜렉션 RDB_node 맵핑이 **유형** 필드에 표시됩니다.
      - 마법사가 지정된 DAD 파일을 인식하지 않으면, **다음**을 선택할 수 없습니다. DAD 파일 이름을 **파일 이름** 필드에 다시 입력하거나, **...**을 눌러 기존 DAD 파일을 다시 찾아보십시오. **다음**을 선택할 수 있을 때까지 이 단계를 계속하십시오.

   c. **다음**을 눌러 유효성 검증 선택 창을 여십시오.

   새 DAD를 작성하려면, 다음과 같이 하십시오.

   a. **파일 이름** 필드를 공백으로 두십시오.

   b. **유형** 메뉴에서 XML 콜렉션 RDB_node 맵핑을 선택하십시오.

   c. **다음**을 눌러 유효성 검증 선택 창을 여십시오.

4. 유효성 검증 선택 창에서 DTD로 XML 문서를 유효성 검증할 것인지 선택하십시오.
   - 유효성을 검증하려면, 다음과 같이 하십시오.
     a. **DAD로 XML 문서의 유효성 검증**을 누르십시오.
     b. **DTD ID** 메뉴에서 유효성 검증에 사용할 DTD를 선택하십시오.

     데이터베이스에 대한 DTD 저장소로 DTD를 가져오지 않았으면, XML 문서를 유효성 검증할 수 없습니다.
   - XML 문서의 유효성을 검증하지 않고 계속 수행하려면 **DAD로 XML 문서의 유효성을 검증하지 않음**을 누르십시오.
5. **다음**을 눌러 텍스트 지정 창을 여십시오.
6. 텍스트 지정 창의 **프롤로그** 필드에 프롤로그 이름을 입력하십시오.

   ```
   <?xml version="1.0"?>
   ```

   기존의 DAD를 편집중인 경우는, **프롤로그** 필드에 프롤로그가 자동으로 표시됩니다.
7. 텍스트 지정 창의 **Doctype** 필드에 XML 문서의 문서 유형을 입력하십시오.

   기존의 DAD를 편집중인 경우는, **Doctype** 필드에 문서 유형이 자동으로 표시됩니다.
8. **다음**을 눌러 RDB 맵핑 창을 여십시오.
9. RDB 맵핑 창의 왼쪽에 있는 필드에서, 맵핑할 요소 또는 속성 노드를 눌러 선택하십시오.

   XML 문서에 있는 요소 및 속성들을 DB2 데이터에 해당하는 요소 및 속성 노드에 맵핑합니다. 이러한 노드에서는 XML 데이터에서 DB2 데이터로의 경로를 제공합니다.
10. **루트 노드를 추가하려면, 다음과 같이 하십시오.**
    a. **루트** 아이콘을 선택하십시오.
    b. **새 요소**를 눌러 새 노드를 정의하십시오.
    c. **세부사항** 상자에서 **노드 유형**을 **요소**로 지정하십시오.
    d. **노드 이름** 필드에 최상위 레벨 노드의 이름을 입력하십시오.
    e. **추가**를 눌러 새 노드를 작성하십시오.

맵에 있는 다른 모든 요소 및 속성 노드의 상위가 되는 루트 노드 또는 요소를 작성했습니다. 루트 노드에는 테이블 하위 요소와 조인 조건이 있습니다.

f. 콜렉션의 일부인 각 테이블마다 테이블 노드를 추가하십시오.

1) 루트 노드 이름을 강조표시하고 **새 요소**를 선택하십시오.

2) **세부사항** 상자에서 **노드 유형**을 **테이블**로 지정하십시오.

3) **테이블 이름**에서 테이블의 이름을 선택하십시오. 테이블이 이미 있어야 합니다.

4) 테이블 노드를 추가하려면 **추가**를 누르십시오.

5) 각 테이블마다 이 단계를 반복하십시오.

g. 테이블 노드의 조인 조건을 추가하십시오.

1) 루트 노드 이름을 강조표시하고 **새 요소**를 선택하십시오.

2) **세부사항** 상자에서 **노드 유형**을 **조건**으로 지정하십시오.

3) **조건** 필드에 다음 구문을 사용하여 조인 조건을 입력하십시오.

```
table_name.table_column = table_name.table_column AND
table_name.table_column = table_name.table_column ...
```

4) 조건을 추가하려면 **추가**를 누르십시오.

11. **요소 또는 속성 노드를 추가하려면, 다음과 같이 하십시오.**

a. 하위 요소 또는 속성을 추가하려면 왼쪽 필드에 있는 상위 노드를 누르십시오.

b. **새 요소**를 누르십시오. 상위 노드를 선택하지 않으면 **새 노드**를 선택할 수 없습니다.

c. **세부사항** 상자에 있는 **노드 유형** 메뉴에서 노드 유형을 선택하십시오.

노드 유형 메뉴에서는 맵의 해당 지점에서 유효한 노드 유형만을 표시합니다. **요소** 또는 **속성**.

d. **노드 이름** 필드에 노드 이름을 입력하십시오.

e. **추가**를 눌러 새 노드를 추가하십시오.

f. **요소 또는 속성 노드의 내용을 관계형 테이블에 맵핑하려면,**

1) 텍스트 노드를 지정하십시오.

a) 상위 노드를 누르십시오.

b) **새 요소**를 누르십시오.

c) **노드 유형** 필드에서 **텍스트**를 선택하십시오.

d) **추가**를 눌러 노드를 추가하십시오.

2) 테이블 노드를 추가하십시오.

a) 방금 작성한 텍스트 노드를 선택하고 **새 요소**를 누르십시오.

b) **노드 유형** 필드에서, **테이블**을 선택하고 요소에 테이블 이름을 지정하십시오.

c) **추가**를 눌러 노드를 추가하십시오.

3) 컬럼 노드를 추가하십시오.

a) 텍스트 노드를 다시 선택하고 **새 요소**를 누르십시오.

b) **노드 유형** 필드에서 **컬럼**을 선택하고 요소에 컬럼 이름을 지정하십시오.

c) **추가**를 눌러 노드를 추가하십시오.

제한사항: 관리 마법사를 사용하면 새 컬럼을 작성할 수 없습니다. 노드 유형으로서 컬럼을 지정하면, 이미 DB2 데이터베이스에 있는 컬럼만을 선택할 수 있습니다.

4) 컬럼에 대한 조건을 선택적으로 추가하십시오.

a) 텍스트 노드를 다시 선택하고 **새 요소**를 누르십시오.

b) **노드 유형** 필드에서, **조건**을 선택하고 다음 구문이 있는 조건을 선택하십시오.

operator LIKE|<|>|= *value*

c) **추가**를 눌러 노드를 추가하십시오.

g. RDB 맵 편집을 계속하거나, **다음**을 눌러 DAD 지정 창을 여십시오.

12. **노드를 제거하려면, 다음과 같이 하십시오.**

a. 왼쪽에 있는 필드에서 노드를 누르십시오.

b. **제거**를 누르십시오.

c. RDB_node 맵 편집을 계속하거나, **다음**을 눌러 DAD 지정 창을 여십시오.

13. DAD 지정 창의 **파일 이름** 필드에 수정된 DAD의 출력 파일 이름을 입력하십시오.
14. **완료**를 눌러 노드를 제거하고 LaunchPad 창으로 되돌아가십시오.

DB2 명령 쉘에서: DAD 파일은 모든 텍스트 편집기에서 작성될 수 있는 XML 파일입니다. 다음 단계에서는 샘플 부록 297 페이지의 『문서 액세스 정의 파일』의 조각을 보여줍니다. 광범위한 정보 및 문맥에 대해서는 이들 예를 참조하십시오.

1. 텍스트 편집기를 여십시오.
2. DAD 머리글을 작성하십시오.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> DAD에 대한 DTD의
                                        DAD
```

3. <DAD></DAD> 태그를 삽입하십시오.
4. <DAD> 태그 다음에, DAD 파일을 XML 문서 DTD와 연결시키는 DTD ID를 지정하십시오.

```
<dtdid>path\dtd_name.dtd> --> 사용자 응용프로그램에 대한
                              DTD의 경로 및 파일 이름
```

5. 유효성 검증 여부(즉, DTD를 사용하여 XML 문서가 유효한 XML 문서인지 확인함)를 지정합니다. 예를 들면 다음과 같습니다.

```
<validation>NO</validation>   --> YES 또는 NO를 지정하십시오.
```

6. <Xcollection> 요소를 사용하여 액세스 및 저장영역 메소드를 XML 콜렉션으로서 정의하십시오. 액세스 및 저장영역 메소드는 XML 데이터가 DB2 테이블 콜렉션에 저장되도록 정의합니다.

```
<Xcollection>
  </Xcollection>
```

7. 다음 프롤로그 정보를 추가하십시오.

```
<prolog>?xml version="1.0"?</prolog>
```

이 텍스트 그대로가 필요합니다.

8. <doctype></doctype> 태그를 추가하십시오. 예를 들면 다음과 같습니다.

```
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

9. <root_node>를 사용하여 루트 노드를 정의하십시오. root_node 내부에서는 XML 문서를 구성하는 요소와 속성을 지정합니다.
10. XML 문서에 있는 요소 및 속성들을 DB2 데이터에 해당하는 요소 및 속성 노드에 맵핑합니다. 이러한 노드에서는 XML 데이터에서 DB2 데이터로의 경로를 제공합니다.
    a. 루트 노드 element_node를 정의합니다. 이 element_node는 다음을 포함합니다.
       - 콜렉션을 지정하기 위한 조인 조건이 있는 table_node를 지정하는 RDB_node
       - 하위 요소
       - 속성

       테이블 노드와 조건을 지정하려면, 다음과 같이 하십시오.

       1) RDB_node 요소를 작성하십시오. 예를 들면 다음과 같습니다.

```
<RDB_node>
</RDB_node>
```

       2) XML 문서에 포함될 데이터가 들어 있는 각 테이블마다 <table_node>를 정의하십시오. 예를 들어, 문서에 포함될 컬럼 데이터가 있는 세 개의 테이블, ORDER_TAB, PART_TAB, SHIP_TAB이 있는 경우, 각 테이블마다 테이블 노드를 작성하십시오. 예를 들면 다음과 같습니다.

```
<RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB">
</RDB_node>
```

       3) 이 콜렉션을 사용 가능화할 때, 각 테이블에 대해 선택적으로 키 컬럼을 지정할 수 있습니다. 키 속성은 보통 작성에는 요구되지 않지만, 임의의 콜렉션을 사용 가능화할 때 쓰이는 DAD 파일은 작성 및 분해 둘 다를 지원해야 합니다. 예를 들면 다음과 같습니다.

```
                    <RDB_node>
<table name="ORDER_TAB" key="order_key">
<table name="PART_TAB" key="part_key">
<table name="SHIP_TAB" key="date mode">
                    </RDB_node>
```

4) 콜렉션에 있는 테이블의 조인 조건을 정의하십시오. 구문은 다음과 같습니다.

```
expression = expression AND
expression = expression
```

예를 들면 다음과 같습니다.

```
                    <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB">
<condition>
order_tab.order_key = part_tab.order_key AND
                    part_tab.part_key = ship_tab.part_key
</condition>
                    </RDB_node>
```

b. DB2 테이블의 컬럼에 맵핑하는 XML 문서의 각 요소마다 <element_node> 태그를 정의하십시오. 예를 들면 다음과 같습니다.

```
<element_node name="name">
              </element_node>
```

요소 노드는 다음 요소 유형 중 하나일 수 있습니다.

- <text_node>: 요소에 DB2 테이블에 대한 내용이 있도록 지정할 경우. 이 요소에는 하위 요소가 없습니다.
- <attribute_node>: 속성을 지정할 경우. 속성 노드는 다음 단계에서 정의됩니다.

text_node는 DB2 테이블과 컬럼 이름에 내용을 맵핑할 <RDB_node>를 포함합니다.

RDB_node는 DB2 테이블에 맵핑할 내용이 들어 있는 최하위 레벨 요소에 사용됩니다. RDB_node에는 다음과 같은 하위 요소가 있습니다.

- <table>: 요소에 해당하는 테이블을 정의합니다.

- <column>: 해당하는 요소를 포함하는 컬럼을 정의하며, 유형 속성에 맞는 컬럼 유형을 지정합니다.
- <condition>: 컬럼에 대한 조건을 선택적으로 지정합니다.

예를 들어 TAX 컬럼으로 맵핑되는 XML 요소 <Tax>가 있을 수 있습니다.

XML 문서:

```
<Tax>0.02</Tax>
```

이 경우, 컬럼 TAX 값이 0.02가 되도록 합니다.

```
<element_node name="Tax">
      <text_node>
         <RDB_node>
         <table name="part_tab"/>
         <column name="tax"/>
         </RDB_node>
      </text_node>
   </element_node>
```

이 예에서 <RDB_node>는 <Tax> 요소값이 텍스트 값이고, 데이터가 TAX 컬럼에 있는 PART_TAB 테이블에 저장되도록 지정합니다. DAD 파일의 전체 구문을 제공하는 287 페이지의 『부록A. DAD 파일의 DTD』에 있는 DAD 파일에 대한 DTD와 RDB_node 맵핑에 대한 297 페이지의 『문서 액세스 정의 파일』에 있는 DAD 파일 예도 참조하십시오.

c. 이 콜렉션을 사용 가능화할 때, 각 <column> 요소에 대해 유형 속성을 추가하십시오. 유형 속성은 보통 작성에는 요구되지 않지만, 임의의 콜렉션을 사용 가능화할 때 쓰이는 DAD 파일은 작성 및 분해 둘 다를 지원해야 합니다. 예를 들면 다음과 같습니다.

```
<column name="tax" type="real"/>
```

d. XML 문서에서 DB2 테이블에 있는 컬럼에 맵핑되는 각 속성에 대해 <attribute_node>를 정의하십시오. 예를 들면 다음과 같습니다.

```
<attribute_node name="key">
              </attribute_node>
```

attribute_node에는 속성 값을 DB2 테이블과 컬럼에 맵핑할 <RDB_node>가 있습니다. <RDB_node>에는 다음과 같은 하위 요소가 있습니다.

- <table>: 요소에 해당하는 테이블을 정의합니다.
- <column>: 해당하는 요소를 포함하는 컬럼을 정의합니다.
- <condition>: 컬럼에 대한 조건을 선택적으로 지정합니다.

예를 들어, <Order> 요소에 대한 키 속성이 필요한 경우가 있습니다. 키 값은 컬럼 PART_KEY에 저장되어야 합니다. DAD 파일에서, 키에 대한 <attribute_node>를 작성하고 이 값이 저장될 테이블을 나타내십시오.

DAD 파일

```
<attribute_node name="key">
                    <RDB_node>
  <table name="part_tab">
                    <column name="part_key"/>
                    <RDB_node>
              </attribute_node>
```

작성된 XML 문서

```
<Order key="1">
```

11. 마지막 </element_node> 태그 다음에 종료 </root_node> 태그가 있는지 확인하십시오.
12. </root_node> 태그 다음에 종료 </Xcollection> 태그가 있는지 확인하십시오.
13. </Xcollection> 태그 다음에 종료 </DAD> 태그가 있는지 확인하십시오.

XML 문서에 대한 스타일 시트 지정

문서를 작성하는 경우, XML Extender에서는 <stylesheet> 요소를 사용하여 스타일 시트에 대한 지시사항 처리도 지원합니다. 지시사항 처리는 <Xcollection> 루트 요소 내에 있어야 하며, XML 문서 구조에 대해 정의된 <doctype> 및 <prolog>와 함께 위치해야 합니다. 예를 들면 다음과 같습니다.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dtd\dad.dtd">
<DAD>
    <SQL_stmt>
        ...
    </SQL_stmt>
<Xcollection>
...
<prolog>...</prolog>
<doctype>...</doctype>
<stylesheet>?xml-stylesheet type="text/css" href="order.css"?</stylesheet>
<root_node>...</root_node>
```

```
...
  </Xcollection>
...
</DAD>
```

RDB_node 맵핑을 사용하여 XML 문서 분해

XML 문서를 분해하려면 RDB_node 맵핑을 사용하십시오. 이 메소드는 <RDB_node>를 사용하여 요소 또는 속성 노드에 대한 DB2 테이블, 컬럼 및 조건을 지정합니다. <RDB_node>는 다음 요소를 사용합니다.

- <table>: 요소에 해당하는 테이블을 정의합니다.
- <column>: 해당하는 요소를 포함하는 컬럼을 정의합니다.
- <condition>: 컬럼에 대한 조건을 선택적으로 지정합니다.

<RDB_node>에서 사용되는 하위 요소는 노드의 문맥에 따라 다르며, 다음 규칙을 사용합니다.

노드 유형:	RDB 하위 요소 사용		
	테이블	컬럼	조건[1]
루트 요소	Y	N	Y
속성	Y	Y	선택적
텍스트	Y	Y	선택적

(1) 다중 테이블인 경우는 필수

관리 마법사 사용: 분해를 위한 DAD를 작성하려면, 다음과 같이 하십시오.

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. LaunchPad 창에서 **DAD 파일에 대한 작업**을 누르십시오. DAD 지정 창이 표시됩니다.
3. 기존 DAD 파일을 편집할 것인지 또는 새 DAD를 작성할 것인지 선택하십시오.

   기존 DAD를 편집하려면, 다음과 같이 하십시오.

   a. **파일 이름** 필드에 DAD 파일 이름을 입력하거나, **...**을 눌러 기존 DAD를 찾아보십시오.

   b. 마법사가 지정된 DAD 파일을 인식하는지 검증하십시오.

- 마법사가 지정된 DAD 파일을 인식하면, **다음**을 선택할 수 있고, XML 콜렉션 RDB 노드 맵핑이 **유형** 필드에 표시됩니다.
- 마법사가 지정된 DAD 파일을 인식하지 않으면, **다음**을 선택할 수 없습니다. DAD 파일 이름을 **파일 이름** 필드에 다시 입력하거나, **...**을 눌러 기존 DAD 파일을 다시 찾아보십시오. **다음**을 선택할 수 있을 때까지 이 단계를 계속하십시오.

c. **다음**을 눌러 유효성 검증 선택 창을 여십시오.

새 DAD를 작성하려면, 다음과 같이 하십시오.

a. **파일 이름** 필드를 공백으로 두십시오.

b. **유형** 메뉴에서 XML 콜렉션 RDB_node 맵핑을 선택하십시오.

c. **다음**을 눌러 유효성 검증 선택 창을 여십시오.

4. 유효성 검증 선택 창에서 DTD로 XML 문서를 유효성 검증할 것인지 선택하십시오.
   - 유효성을 검증하려면, 다음과 같이 하십시오.
     a. **DAD로 XML 문서의 유효성 검증**을 누르십시오.
     b. **DTD ID** 메뉴에서 유효성 검증에 사용할 DTD를 선택하십시오.

     데이터베이스에 대한 DTD 저장소로 DTD를 가져오지 않았으면, XML 문서를 유효성 검증할 수 없습니다.
   - XML 문서의 유효성을 검증하지 않고 계속 수행하려면 **DAD로 XML 문서의 유효성을 검증하지 않음**을 누르십시오.
5. **다음**을 눌러 텍스트 지정 창을 여십시오.
6. XML 문서만 분해할 경우, **프롤로그** 필드는 무시하십시오. 작성 및 분해 둘 모두에 DAD 파일을 사용할 경우, 텍스트 지정 창의 **프롤로그** 필드에 프롤로그 이름을 입력하십시오. XML 문서를 DB2 데이터로 분해할 경우에는 프롤로그가 필요없습니다.

   ```
   <?xml version="1.0"?>
   ```

   기존의 DAD를 편집중인 경우는, **프롤로그** 필드에 프롤로그가 자동으로 표시됩니다.

7. XML 문서만 분해할 경우, **Doctype** 필드는 무시하십시오. 작성 및 분해 둘 모두에 DAD 파일을 사용할 경우, **Doctype** 필드에 XML 문서의 문서 유형을 입력하십시오.

   기존의 DAD를 편집중인 경우는, **Doctype** 필드에 문서 유형이 자동으로 표시됩니다.

8. **다음**을 눌러 RDB 맵핑 창을 여십시오.
9. RDB 맵핑 창의 왼쪽에 있는 필드에서, 맵핑할 요소 또는 속성 노드를 눌러 선택하십시오.

   XML 문서에 있는 요소 및 속성들을 DB2 데이터에 해당하는 요소 및 속성 노드에 맵핑합니다. 이러한 노드에서는 XML 데이터에서 DB2 데이터로의 경로를 제공합니다.

10. **루트 노드를 추가하려면, 다음과 같이 하십시오.**
    a. **루트** 아이콘을 선택하십시오.
    b. **새 요소**를 눌러 새 노드를 정의하십시오.
    c. **세부사항** 상자에서 **노드 유형**을 **요소**로 지정하십시오.
    d. **노드 이름** 필드에 최상위 레벨 노드의 이름을 입력하십시오.
    e. **추가**를 눌러 새 노드를 작성하십시오.

       맵에 있는 다른 모든 요소 및 속성 노드의 상위가 되는 루트 노드 또는 요소를 작성했습니다. 루트 노드에는 테이블 하위 요소와 조인 조건이 있습니다.

    f. 콜렉션의 일부인 각 테이블마다 테이블 노드를 추가하십시오.
       1) 루트 노드 이름을 강조표시하고 **새 요소**를 선택하십시오.
       2) **세부사항** 상자에서 **노드 유형**을 **테이블**로 지정하십시오.
       3) **테이블 이름**에서 테이블의 이름을 선택하십시오. 테이블이 이미 있어야 합니다.
       4) **테이블 키** 필드에 테이블에 대한 키 컬럼을 지정하십시오.
       5) 테이블 노드를 추가하려면 **추가**를 누르십시오.
       6) 각 테이블마다 이 단계를 반복하십시오.
    g. 테이블 노드의 조인 조건을 추가하십시오.

1) 루트 노드 이름을 강조표시하고 **새 요소**를 선택하십시오.

2) **세부사항** 상자에서 **노드 유형**을 **조건**으로 지정하십시오.

3) **조건** 필드에서, 다음 구문을 사용하여 조인 조건을 입력하십시오.

```
table_name.table_column = table_name.table_column AND
table_name.table_column = table_name.table_column ...
```

4) 조건을 추가하려면 **추가**를 누르십시오.

이제 하위 요소 및 속성을 이 노드에 추가할 수 있습니다.

11. **요소 또는 속성 노드를 추가하려면, 다음과 같이 하십시오.**

a. 하위 요소 또는 속성을 추가하려면 왼쪽 필드에 있는 상위 노드를 누르십시오.

상위 노드를 선택할 수 없으면, **새로 작성**을 선택할 수 없습니다.

b. **새 요소**를 누르십시오.

c. **세부사항** 상자에 있는 **노드 유형** 메뉴에서 노드 유형을 선택하십시오.

노드 유형 메뉴에서는 맵의 해당 지점에서 유효한 노드 유형만을 표시합니다. **요소** 또는 **속성**.

d. **노드 이름** 필드에 노드 이름을 입력하십시오.

e. **추가**를 눌러 새 노드를 추가하십시오.

f. **요소 또는 속성 노드의 내용을 관계형 테이블에 맵핑하려면,**

1) 텍스트 노드를 지정하십시오.

a) 상위 노드를 누르십시오.

b) **새 요소**를 누르십시오.

c) **노드 유형** 필드에서 **텍스트**를 선택하십시오.

d) **추가**를 눌러 노드를 추가하십시오.

2) 테이블 노드를 추가하십시오.

a) 방금 작성한 텍스트 노드를 선택하고 **새 요소**를 누르십시오.

b) **노드 유형** 필드에서, **테이블**을 선택하고 요소에 테이블 이름을 지정하십시오.

c) **추가**를 눌러 노드를 추가하십시오.

3) 컬럼 노드를 추가하십시오.

a) 텍스트 노드를 다시 선택하고 **새 요소**를 누르십시오.

b) **노드 유형** 필드에서 **컬럼**을 선택하고 요소에 컬럼 이름을 지정하십시오.

c) 태그가 없는 데이터를 저장하기 위해 컬럼이 가져야 할 유형을 지정하려면 **유형** 필드에 컬럼의 기본 데이터 유형을 지정하십시오.

d) **추가**를 눌러 노드를 추가하십시오.

제한사항: 관리 마법사를 사용하면 새 컬럼을 작성할 수 없습니다. 노드 유형으로서 컬럼을 지정하면, 이미 DB2 데이터베이스에 있는 컬럼만을 선택할 수 있습니다.

4) 컬럼에 대한 조건을 선택적으로 추가하십시오.

a) 텍스트 노드를 다시 선택하고 **새 요소**를 누르십시오.

b) **노드 유형** 필드에서, **조건**을 선택하고 다음 구문이 있는 조건을 선택하십시오.

operator LIKE|<|>|= *value*

c) **추가**를 눌러 노드를 추가하십시오.

노드를 선택하고, **세부사항** 상자에서 필드를 변경하고, **변경**을 눌러 이들 노드를 수정할 수 있습니다.

g. RDB 맵 편집을 계속하거나, **다음**을 눌러 DAD 지정 창을 여십시오.

12. **노드를 제거하려면, 다음과 같이 하십시오.**

a. 왼쪽에 있는 필드에서 노드를 누르십시오.

b. **제거**를 누르십시오.

c. RDB_node 맵 편집을 계속하거나, **다음**을 눌러 DAD 지정 창을 여십시오.

13. DAD 지정 창의 **파일 이름** 필드에 수정된 DAD의 출력 파일 이름을 입력하십시오.

14. **완료**를 눌러 노드를 제거하고 LaunchPad 창으로 되돌아가십시오.

DB2 명령 쉘에서: DAD 파일은 모든 텍스트 편집기에서 작성될 수 있는 XML 파일입니다. 다음 단계에서는 샘플 부록 297 페이지의 『문서 액세스 정의 파일』의 조각을 보여줍니다. 광범위한 정보 및 문맥에 대해서는 이들 예를 참조하십시오.

1. 텍스트 편집기를 여십시오.
2. DAD 머리글을 작성하십시오.

   ```
   <?xml version="1.0"?>
   <!DOCTYPE DAD SYSTEM "path\dad.dtd"> --> DAD에 대한 DTD의
                                            DAD
   ```

3. <DAD></DAD> 태그를 삽입하십시오.
4. <DAD> 태그 다음에, DAD 파일을 XML 문서 DTD와 연결시키는 DTD ID를 지정하십시오.

   ```
   <dtdid>path\dtd_name.dtd> --> 사용자 응용프로그램에 대한
                                 DTD의 경로 및 파일 이름
   ```

5. 유효성 검증 여부(즉, DTD를 사용하여 XML 문서가 유효한 XML 문서인지 확인함)를 지정합니다. 예를 들면 다음과 같습니다.

   ```
   <validation>NO</validation>   --> YES 또는 NO를 지정하십시오.
   ```

6. <Xcollection> 요소를 사용하여 액세스 및 저장영역 메소드를 XML 콜렉션으로서 정의하십시오. 액세스 및 저장영역 메소드는 XML 데이터가 DB2 테이블 콜렉션에 저장되도록 정의합니다.

   ```
   <Xcollection>
     </Xcollection>
   ```

7. 다음 프롤로그 정보를 추가하십시오.

   ```
   <prolog>?xml version="1.0"?</prolog>
   ```

   이 텍스트 그대로가 필요합니다.

8. <doctype></doctype> 태그를 추가하십시오. 예를 들면 다음과 같습니다.

   ```
   <doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
   ```

9. <root_node></root_node> 태그를 사용하여 root_node를 정의하십시오. root_node 내부에서는 XML 문서를 구성하는 요소와 속성을 지정합니다.

10. <root_node> 태그 다음에, XML 문서에 있는 요소 및 속성을 DB2 데이터에 해당하는 요소 및 속성 노드로 맵핑하십시오. 이러한 노드에서는 XML 데이터에서 DB2 데이터로의 경로를 제공합니다.

    a. 최상위 레벨의 루트 element_node를 정의하십시오. 이 element_node는 다음을 포함합니다.

       - 콜렉션을 지정할 조인 조건이 있는 테이블 노드
       - 하위 요소
       - 속성

       테이블 노드와 조건을 지정하려면, 다음과 같이 하십시오.

       1) RDB_node 요소를 작성하십시오. 예를 들면 다음과 같습니다.

```
<RDB_node>
</RDB_node>
```

       2) XML 문서에 포함될 데이터가 들어 있는 각 테이블마다 <table_node>를 정의하십시오. 예를 들어, 문서에 포함될 컬럼 데이터가 있는 세 개의 테이블, ORDER_TAB, PART_TAB, SHIP_TAB이 있는 경우, 각 테이블마다 테이블 노드를 작성하십시오. 예를 들면 다음과 같습니다.

```
                    <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB">
                    </RDB_node>
```

       3) 콜렉션에 있는 테이블의 조인 조건을 정의하십시오. 구문은 다음과 같습니다.

```
expression = expression AND
expression = expression ...
```

          예를 들면 다음과 같습니다.

```
                    <RDB_node>
<table name="ORDER_TAB">
<table name="PART_TAB">
<table name="SHIP_TAB">
<condition>
order_tab.order_key = part_tab.order_key AND
```

```
                    part_tab.part_key = ship_tab.part_key
</condition>
                    </RDB_node>
```

4) 각 테이블에 대한 기본 키를 지정하십시오. 기본 키는 복합 키라고 하며, 단일 컬럼이나 다중 컬럼으로 구성됩니다. 기본 키를 지정하려면, 속성 키를 RDB_node의 테이블 요소에 추가하십시오. 다음 예에서는 루트 element_node Order의 RDB_node에서 각 테이블에 대한 기본 키를 정의합니다.

```
        <element_node name="Order">
                    <RDB_node>
        <table name="order_tab" key="order_key"/>
        <table name="part_tab" key="part_key price"/>
        <table name="ship_tab" key="date mode"/>
                       <condition>
order_tab.order_key = part_tab.order_key AND
                    part_tab.part_key = ship_tab.part_key
                       </condition>
                    <RDB_node>
```

분해를 위해 지정된 정보는 XML 문서를 작성할 때는 무시됩니다.

DAD 파일이 작성과 분해 양쪽을 지원하므로 키 속성은 분해 및 임의의 콜렉션을 사용 가능화할 때 요구됩니다.

b. DB2 테이블의 컬럼에 맵핑하는 XML 문서의 각 요소마다 <element_node> 태그를 정의하십시오. 예를 들면 다음과 같습니다.

```
<element_node name="name">
             </element_node>
```

요소 노드는 다음 요소 유형 중 하나일 수 있습니다.

- <text_node>: 요소가 DB2 테이블에 대해 내용을 갖도록 지정할 경우. 이 경우 요소에는 하위 요소가 없습니다.
- <attribute_node>: 속성을 지정할 경우. 속성 노드는 다음 단계에서 정의됩니다.
- 하위 요소

text_node는 DB2 테이블과 컬럼 이름에 내용을 맵핑하기 위한 RDB_node를 포함합니다.

RDB_node는 DB2 테이블에 맵핑할 내용이 들어 있는 최하위 레벨 요소에 사용됩니다. RDB_node에는 다음과 같은 하위 요소가 있습니다.

- <table>: 요소에 해당하는 테이블을 정의합니다.
- <column>: 해당하는 요소를 포함하는 컬럼을 정의합니다.
- <condition>: 컬럼에 대한 조건을 선택적으로 지정합니다.

예를 들면, TAX 컬럼에 태그가 없는 내용을 저장할 XML 요소 <Tax>를 가질 수 있습니다.

XML 문서:

```
<Tax>0.02</Tax>
```

이 경우, 0.02 값을 TAX 컬럼에 저장하게 됩니다.

DAD 파일에서는 XML 요소를 DB2 테이블과 컬럼에 맵핑하기 위해 <RDB_node>를 지정합니다.

DAD 파일

```
<element_node name="Tax">
      <text_node>
         <RDB_node>
         <table name="part_tab"/>
         <column name="tax"/>
         </RDB_node>
      </text_node>
   </element_node>
```

<RDB_node>는 <Tax> 요소의 값이 텍스트 값이며, 이 값이 TAX 컬럼의 PART_TAB 테이블에 저장되도록 지정합니다.

c. XML 문서에서 DB2 테이블에 있는 컬럼에 맵핑되는 각 속성에 대해 <attribute_node>를 정의하십시오. 예를 들면 다음과 같습니다.

```
<attribute_node name="key">
              </attribute_node>
```

attribute_node에는 속성 값을 DB2 테이블 및 컬럼에 맵핑할 RDB_node가 있습니다. RDB_node에는 다음과 같은 하위 요소가 있습니다.

- <table>: 요소에 해당하는 테이블을 정의합니다.

- <column>: 해당하는 요소를 포함하는 컬럼을 정의합니다.
- <condition>: 컬럼에 대한 조건을 선택적으로 지정합니다.

예를 들어 요소 <Order>에 대한 속성 키가 있을 수 있습니다. 키 값은 컬럼 PART_KEY에 저장되어야 합니다.

XML 문서:

```
<Order key="1">
```

DAD 파일에서 키에 대한 attribute_node를 작성하고, 1 값이 저장될 테이블을 나타내십시오.

DAD 파일

```
<attribute_node name="key">
                    <RDB_node>
  <table name="part_tab">
                    <column name="part_key"/>
                    <RDB_node>
              </attribute_node>
```

11. attribute_node 및 text_node 각각에 대한 RDB_node의 컬럼 유형을 지정하십시오. 이렇게 하면, 태그가 없는 데이터가 저장될 각 컬럼마다 올바른 데이터 유형이 확인됩니다. 컬럼 유형을 지정하려면 속성 유형을 컬럼 요소에 추가하십시오. 다음 예에서는 컬럼 유형을 INTEGER로 정의합니다.

```
<attribute_node name="key">
                  <RDB_node>
                  <table name="order_tab"/>
      <column name="order_key" type="integer"/>
                  </RDB_node>
            </attribute_node>
```

12. 마지막 </element_node> 태그 다음에 종료 </root_node> 태그가 있는지 확인하십시오.
13. </root_node> 태그 다음에 종료 </Xcollection> 태그가 있는지 확인하십시오.
14. </Xcollection> 태그 다음에 종료 </DAD> 태그가 있는지 확인하십시오.

XML 콜렉션 사용

XML 콜렉션을 사용하게 되면 DAD 파일을 구문 분석하여 XML 문서와 관련된 테이블 및 컬럼을 식별하고, XML_USAGE 테이블에 제어 정보를 기록합니다. XML 콜렉션을 사용하게 되는 것은 다음의 경우 선택적입니다.

- XML 문서 분해 및 새 DB2 테이블에 데이터 저장
- 여러 DB2 테이블에 있는 기존 데이터로 XML 문서 작성

같은 DAD 파일이 작성과 분해에 사용되는 경우, 작성과 분해 모두에 대해 콜렉션을 사용할 수 있게 됩니다.

XML Extender 관리 마법사를 통해, enable_collection 옵션과 함께 **dxxadm** 명령을 사용하여 XML 콜렉션을 사용할 수 있게 되거나, XML Extender 저장 프로시듀어 dxxEnableCollection()을 사용할 수도 있습니다.

관리 마법사 사용

다음 단계에서 XML 콜렉션을 사용 가능하게 하십시오.

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. LaunchPad 창에서 **XML 콜렉션에 대한 작업**을 누르십시오. 타스크 선택 창이 열립니다.
3. **콜렉션 사용**을 누른 후, **다음**을 누르십시오. 콜렉션 사용 창이 표시됩니다.
4. 풀다운 메뉴에서 **콜렉션 이름** 필드에 사용하려는 콜렉션의 이름을 선택하십시오.
5. **DAD 파일 이름** 필드에 DAD 파일 이름을 입력하거나, **...**을 눌러 기존 DAD 파일을 찾아보십시오.
6. 선택적으로, **테이블 공간** 필드에 이전에 작성했던 테이블 공간의 이름을 입력하십시오.

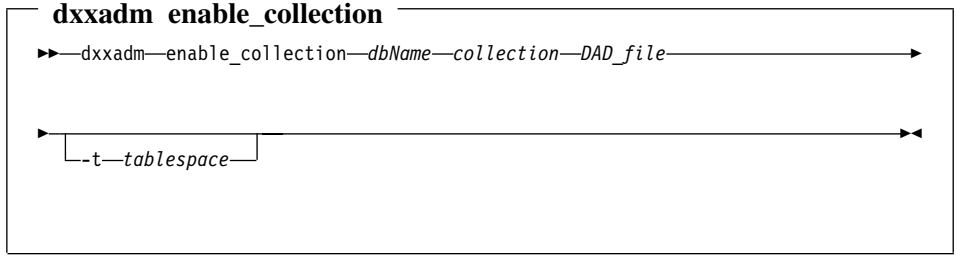
   테이블 공간에는 분해를 위해 생성된 새 DB2 테이블이 포함됩니다.
7. **완료**를 눌러 콜렉션을 사용 가능하게 하고 LaunchPad 창으로 되돌아가십시오.
   - 콜렉션을 사용할 수 있게 되면, 콜렉션 사용 성공 메시지가 표시됩니다.

- 콜렉션을 사용할 수 없으면 오류 메시지가 표시됩니다. 콜렉션을 사용할 수 있게 될 때까지 위 단계를 계속하십시오.

DB2 명령 쉘에서

XML 콜렉션을 사용할 수 있도록 하려면 **dxxadm** 명령을 입력하십시오.

구문:



매개변수:

dbName

데이터베이스 이름.

collection

XML 콜렉션 이름. 이 값은 XML 콜렉션 저장 프로시듀어에 대한 매개변수로서 사용됩니다.

DAD_file

문서 액세스 정의(DAD)가 포함되는 파일 이름.

tablespace

분해를 위해 생성되었던 새 DB2 테이블이 포함되는 기존의 테이블 공간. 지정되지 않았으면 기본 테이블 공간이 사용됩니다.

예: 다음 예에서는 DB2 명령 쉘을 사용하여 데이터베이스 SALES_DB에서 sales_ord 콜렉션을 사용 가능하게 합니다. DAD 파일에서는 SQL 맵핑을 사용하고 이것은 299 페이지의 『DAD 파일: XML 콜렉션 - SQL 맵핑』에 있습니다.

```
dxxadm enable_collection SALES_DB sales_ord getstart.dad
```

XML 콜렉션을 사용할 수 있게 되면, XML Extender 저장 프로시듀어를 사용하여 XML 문서를 작성하거나 분해할 수 있습니다.

XML 콜렉션 사용 안함

XML 콜렉션을 사용할 수 없게 되면, XML_USAGE 테이블에서 테이블과 컬럼을 콜렉션의 일부로 식별하는 레코드가 삭제됩니다. 데이터 테이블은 제거되지 않습니다. DAD를 갱신하거나 콜렉션을 다시 사용해야 하는 경우, 또는 콜렉션을 제거하는 경우에는 콜렉션을 사용할 수 없습니다.

XML Extender 관리 마법사를 통해, disable_collection 옵션과 함께 **dxxadm** 명령을 사용하여 또는 XML Extender 저장 프로시듀어 dxxDisableCollection()을 사용하여 XML 콜렉션을 사용할 수 없도록 할 수 있습니다.

관리 마법사 사용

다음 단계에서 XML 콜렉션을 사용할 수 없도록 하십시오.

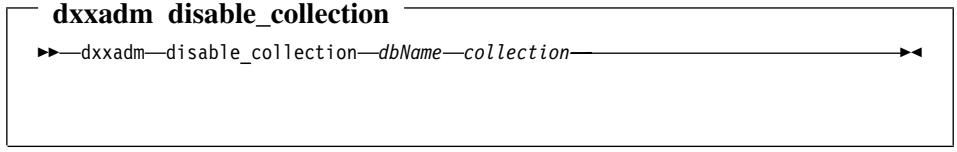
1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. XML Extender 콜렉션 관련 타스크를 보려면, LaunchPad 창에서 **XML 콜렉션에 대한 작업**을 누르십시오. 타스크 선택 창이 열립니다.
3. XML 콜렉션을 사용할 수 없게 하려면, **XML 콜렉션 사용 안함**을 누른 후, **다음**을 누르십시오. 콜렉션 사용 안함 창이 표시됩니다.
4. **콜렉션 이름** 필드에 사용할 수 없게 하려는 콜렉션 이름을 입력하십시오.
5. **완료**를 눌러 콜렉션을 사용할 수 없게 하고 LaunchPad 창으로 되돌아가십시오.
   - 콜렉션을 사용할 수 없게 되면, 콜렉션 사용 안함 메시지가 표시됩니다.
   - 그래도 계속 콜렉션을 사용할 수 있으면, 오류 상자가 표시됩니다. 콜렉션을 사용할 수 없게 될 때까지 위 단계를 계속하십시오.

DB2 명령 쉘에서

XML 콜렉션을 사용할 수 없도록 하려면 **dxxadm** 명령을 입력하십시오.

구문:

dxxadm disable_collection



매개변수:

dbName

데이터베이스 이름.

collection

XML 콜렉션 이름. 이 값은 XML 콜렉션 저장 프로시듀어에 대한 매개변수로서 사용됩니다.

예:

```
dxxadm disable_collection SALES_DB sales_ord
```

XML의 데이터베이스 사용 안함

XML Extender 환경을 정리하고, XML Extender UDT, UDF, 저장 프로시듀어 및 관리 지원 테이블을 제거할 경우 데이터베이스를 사용할 수 없게 됩니다. XML Extender는 현재 인스턴스를 사용하여 사용자가 연결되어 있는 데이터베이스를 사용할 수 없게 됩니다.

XML의 데이터베이스를 사용할 수 없게 되면, XML Extender는 데이터베이스에 대해 다음 조치를 수행합니다.

- 모든 사용자 정의 유형(UDT) 및 사용자 정의 함수(UDF)를 삭제합니다.
- XML Extender에 대한 메타데이터가 있는 제어 테이블을 삭제합니다.
- db2xml 스키마를 삭제합니다.

시작하기 전에

사용할 수 없게 될 데이터베이스에 있는 모든 XML 컬럼이나 콜렉션을 사용할 수 없게 하십시오.

관리 마법사 사용

XML 데이터의 데이터베이스를 사용할 수 없게 하려면 다음 단계를 사용하십시오.

1. 관리 마법사를 설정하고 시작합니다. 세부사항은 75 페이지의 『관리 마법사 시작』을 참조하십시오.
2. 현재 데이터베이스를 사용할 수 없게 하려면 LaunchPad 창에서 **데이터베이스 사용 안함**을 누르십시오.

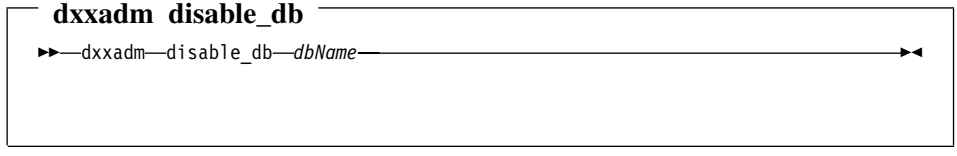
   데이터베이스가 현재 사용할 수 없으면, **데이터베이스 사용**만 선택할 수 있습니다.

   데이터베이스가 사용할 수 없게 되면, LaunchPad 창으로 되돌아갑니다.

DB2 명령 쉘에서

명령 행에서 **dxxadm**을 입력하여 사용 불가능하게 할 데이터베이스를 지정하십시오.

구문:



매개변수:

dbName

사용 불가능하게 될 데이터베이스의 이름.

예: 기존 데이터베이스 SALES_DB를 사용 불가능하게 합니다.

```
dxxadm disable_db SALES_DB
```

제3부 프로그래밍

이 부분에서는 XML 데이터 관리를 위한 프로그래밍 기술에 대해 설명합니다.

제5장 XML 컬럼 데이터 관리

XML 컬럼을 사용할 경우는 전체 XML 문서를 컬럼 데이터로서 저장합니다. 이 액세스 및 저장영역 메소드를 사용하면 XML 문서를 원래 그대로 보존할 수 있는 한편, 문서에 색인 지정 및 검색을 수행할 수 있으며, 문서에서 데이터를 검색하고 문서를 갱신할 수 있습니다. XML 컬럼에는 DB2에서 컬럼 데이터로서 고유 형식으로 되어 있는 XML 문서가 들어 있습니다. XML에 데이터베이스를 사용할 수 있도록 한 다음에는 다음 사용자 정의 유형을 사용할 수 있습니다.

XMLCLOB

DB2에서 문자 대형 오브젝트(CLOB)로서 저장되는 XML 문서 내용.

XMLVARCHAR

DB2에서 VARCHAR로서 저장되는 XML 문서 내용.

XMLFile

지역 파일 시스템상의 파일에 저장되는 XML 문서.

컬럼 데이터 유형으로서 XML UDT를 사용하여 응용프로그램 테이블을 작성하거나 변경할 수 있습니다. 이러한 테이블을 XML 테이블이라고 합니다. XML용 테이블을 작성하거나 변경하는 방법은 88 페이지의 『XML 테이블 작성 또는 변경』을 참조하십시오.

XML용 컬럼을 사용할 수 있게 되면, XML 컬럼의 내용 관리를 시작할 수 있습니다. XML 컬럼이 작성된 다음에는 다음 관리 타스크를 수행할 수 있습니다.

- DB2에 XML 문서 저장
- DB2에서 XML 데이터 또는 문서 검색
- XML 문서 갱신
- XML 데이터 또는 문서 삭제

이러한 타스크를 수행하기 위해 다음 두 메소드를 사용할 수 있습니다.

- *기본 유형 변환 함수*, SQL 기본 유형을 XML UDT로 변환합니다.
- XML Extender 제공 사용자 정의 기능(UDF)

이 책에서는 각 타스크에서 두 메소드 모두에 대해 설명합니다.

UDT 및 UDF 이름

DB2 함수의 전체 이름은 *schema-name.function-name*이며, 여기서 *schema-name*은 SQL 오브젝트에 대해 논리적 그룹화를 제공하는 식별자입니다. XML Extender UDF에 대한 스키마 이름은 db2xml 입니다. db2xml 스키마 이름은 또한 XML Extender UDT에 대한 규정자입니다. 이 책에서는 함수 이름만을 참조합니다.

함수 경로는 순서가 지정된 스키마 이름의 목록입니다. DB2는 목록에서의 스키마 이름 순서를 사용하여 기능 및 UDT의 참조를 설명합니다. SQL문 SET CURRENT FUNCTION PATH를 지정하여 함수 경로를 지정할 수 있습니다. 이 명령은 CURRENT FUNCTION PATH 특수 레지스터에서 함수 경로를 설정합니다.

XML Extender의 경우, db2xml 스키마를 함수 경로에 추가하는 것이 좋습니다. 이렇게 하면 앞에 db2xml을 삽입하지 않고도 XML Extender UDF 및 UDT 이름을 입력할 수 있습니다. 다음 예에서는 db2xml 스키마를 함수 경로에 추가하는 방법을 보여줍니다.

```
SET CURRENT FUNCTION PATH = db2xml, CURRENT FUNCTION PATH
```

중요사항: db2xml로서 로그온했으면 함수 경로에서 첫번째 스키마로서 db2xml을 추가하지 마십시오. db2xml은 db2xml로서 로그온할 때 첫번째 스키마로서 자동으로 설정됩니다. 이렇게 하면 함수 경로가 두 개의 db2xml 스키마로 시작되므로 오류 조건이 생성됩니다.

데이터 저장

XML Extender를 사용하면 원래 그대로의 XML 문서를 XML 컬럼에 삽입할 수 있습니다. 부가 테이블을 정의할 경우, XML Extender는 자동으로 이러한 테이블을 갱신합니다. XML 문서를 직접 저장할 때, XML Extender는 기본 유형을 XML 유형으로 저장합니다.

타스크 개요:

1. DAD 파일을 작성했거나 갱신했는지 확인합니다.

2. 문서를 저장할 때 사용할 데이터 유형을 판별합니다.
3. DB2 테이블에 데이터를 저장하기 위한 메소드를 선택합니다(유형 변환 함수 또는 UDF).
4. XML 문서가 들어갈 XML 테이블 및 컬럼을 지정하는 SQL INSERT문을 지정합니다.

XML Extender는 XML 문서를 저장하는 두 가지 메소드 즉, 기본 유형 변환 함수와 저장영역 UDF를 제공합니다. 표8에서는 각 메소드의 사용 경우를 보여줍니다.

표 8. XML Extender 저장영역 기능

기본 유형	**DB2에 저장되는 내용...**		
	XMLVARCHAR	**XMLCLOB**	**XMLFILE**
VARCHAR	XMLVARCHAR()	N/A	XMLFileFromVarchar()
CLOB	N/A	XMLCLOB()	XMLFileFromCLOB()
FILE	XMLVarcharFromFile()	XMLCLOBFromFile()	XMLFILE

기본 유형 변환 함수 사용

각 UDT의 경우, SQL 기본 유형을 UDT로 유형 변환하기 위해 *기본 유형 변환 함수*가 제공됩니다. VALUES 절에서 XML Extender가 제공한 유형 변환 함수를 사용하여 데이터를 삽입할 수 있습니다. 표9에서는 제공된 유형 변환 함수를 보여줍니다.

표 9. XML Extender 기본 유형 변환(cast) 함수

SELECT 절에 사용된 유형 변환	**리턴 유형**	**설명**
XMLVARCHAR(VARCHAR)	XMLVARCHAR	VARCHAR의 메모리 버퍼에서 입력
XMLCLOB(CLOB)	XMLCLOB	CLOB 또는 CLOB 위치 지정자의 메모리 버퍼에서 입력
XMLFILE(VARCHAR)	XMLFILE	파일 이름만을 저장

예: 다음 명령문은 유형 변환된 VARCHAR 유형을 XMLVARCHAR 유형으로 삽입합니다.

```
INSERT INTO sales_tab
VALUES('123456', 'Sriram Srinivasan', db2xml.XMLVarchar(:xml_buff))
```

저장영역 UDF를 사용하려면, 다음과 같이 하십시오.

각 XML Extender UDT의 경우, 기본 유형 이외의 자원에서 DB2로 데이터를 가져오기하기 위해 저장영역 UDF가 제공됩니다. 예를 들어, XML 파일 문서를 XMLCLOB로서 DB2에 가져오기하려는 경우, 함수 XMLCLOBFromFile()을 사용할 수 있습니다.

표10에서는 XML Extender가 제공하는 저장영역 기능을 보여줍니다.

표 10. XML Extender 저장영역 UDF

저장영역 사용자 정의 함수	리턴 유형	설명
XMLVarcharFromFile()	XMLVARCHAR	서버에 있는 파일에서 XML 문서를 읽고 XMLVARCHAR 유형 값을 리턴합니다.
XMLCLOBFromFile()	XMLCLOB	서버에 있는 파일에서 XML 문서를 읽고 XMLCLOB 유형 값을 리턴합니다.
XMLFileFromVarchar()	XMLFILE	메모리에서 VARCHAR로서 XML 문서를 읽고, 이를 외부 파일에 기록한 다음, XMLFILE 유형 값을 리턴합니다. 이것이 파일 이름입니다.
XMLFileFromCLOB()	XMLFILE	메모리에서 CLOB 또는 CLOB 위치 지정자로서 XML 문서를 읽고, 이를 외부 파일에 기록한 다음, XMLFILE 유형 값을 리턴합니다. 이것이 파일 이름입니다.

예: 다음 명령문은 XMLCLOBFromFile() 함수를 사용하여 XMLCLOB로서 XML 테이블에 레코드를 기록합니다.

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
    VALUES( '1234', 'Sriram Srinivasan,
      XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

앞의 예에서는 `c:\dxx\samples\cmd\getstart.xml` 파일에서 SALES_TAB 테이블에 있는 ORDER 컬럼으로 XML 오브젝트를 가져옵니다.

데이터 검색

XML Extender를 사용하면 문서 전체나 요소 및 속성의 내용을 검색할 수 있습니다. XML 컬럼을 직접 검색할 때, XML Extender는 컬럼 유형으로서 UDT를 리턴합니다. 데이터 검색에 대한 세부사항은 다음 절을 참조하십시오.

- 『전체 문서 검색』
- 140 페이지의 『요소 내용 및 속성 값 검색』

XML Extender는 데이터를 검색하는 두 가지 메소드 즉, 기본 유형 변환 함수 및 Content() 오버로드된 UDF를 제공합니다. 표11에서는 각 메소드 사용 경우를 보여줍니다.

표 11. XML Extender 검색 함수

XML 유형	**DB2 검색 유형**		
	VARCHAR	**CLOB**	**FILE**
XMLVARCHAR	VARCHAR	N/A	Content()
XMLCLOB	N/A	XMLCLOB	Content()
XMLFILE	N/A	Content()	FILE

전체 문서 검색

타스크 개요:

1. XML 테이블에 XML 문서를 저장했는지 확인하고, 검색하려는 데이터를 판별합니다.
2. DB2 테이블에서 데이터를 검색하기 위한 메소드를 선택합니다(유형 변환 함수 또는 UDF).
3. 오버로드된 Content() UDF를 사용하여 검색하는 데이터와 연결할 데이터 유형 및 가져가기할 데이터 유형을 결정합니다.
4. XML 문서를 검색할 XML 테이블 및 컬럼을 지정하는 SQL 조회를 지정합니다.

XML Extender는 데이터 검색을 위한 두 메소드를 제공합니다.

기본 유형 변환 함수 사용

UDT에 대해 DB2가 제공하는 기본 유형 변환 함수를 사용하여, XML

UDT를 SQL 기본 유형으로 변환한 다음 이에 대해 동작하십시오. SELECT문에서 XML Extender가 제공하는 유형 변환 함수를 사용하여 데이터를 검색할 수 있습니다. 표12에서는 제공된 유형 변환 함수를 보여 줍니다.

표 12. XML Extender 기본 유형 변환(cast) 함수

select 절에 사용된 유형 변환	**리턴 유형**	**설명**
varchar(XMLVARCHAR)	VARCHAR	VARCHAR로 된 XML 문서
clob(XMLCLOB)	CLOB	CLOB로 된 XML 문서
varchar(XMLFile)	VARCHAR	VARCHAR로 된 XML 파일 이름

예: 다음 예에서는 XMLVARCHAR을 검색하고, 이를 메모리에서 VARCHAR 데이터 유형으로 저장합니다.

```
EXEC SQL SELECT db2xml.varchar(order) from sales_tab
```

Content() 오버로드된 UDF 사용

Content() UDF를 사용하여 외부 저장영역에서 메모리로 문서 내용을 검색하거나, 내부 저장영역에서 DB2 서버에 있는 *외부 파일*로 문서를 가져가기합니다.

예를 들어 XML 문서가 XMLFILE로서 저장됩니다. 그리고 메모리에서 이에 대해 동작하고 Content() UDF를 사용할 수 있습니다. XMLFILE 데이터 유형을 입력으로 사용하고 CLOB를 리턴합니다.

Content() UDF는 지정된 데이터 유형에 따라 두개의 서로 다른 검색 함수를 수행합니다. 다음도 수행합니다.

외부 저장영역에서 문서를 검색하고 이를 메모리에 삽입

문서가 외부 파일로서 저장될 때 Content()를 사용하여 메모리 버퍼나 CLOB 위치 지정자로 XML 문서를 검색할 수 있습니다. 다음 함수 구문을 사용하며, 여기서 *xmlobj*는 조회하고 있는 XML 컬럼입니다.

XMLFILE to CLOB: 파일에서 데이터를 검색하고 CLOB 위치 지정자로 가져가기합니다.

```
Content(xmlobj XMLFile)
```

내부 저장영역에서 문서를 검색하고 이를 외부 파일에 가져가기

또한 Content()를 사용하여 DB2에서 XMLCLOB 데이터 유형으로 저장되는 XML 문서를 검색하고, 이를 데이터베이스 서버 파일 시스템에 있는 파일로 가져가기할 수 있습니다. VARCHAR 유형의 파일 이름을 리턴합니다. 다음 함수 구문을 사용합니다. 여기서 *xmlobj*는 조회되는 XML 컬럼이고, *filename*은 외부 파일입니다. *XML 유형*으로 XMLVARCHAR 또는 XMLCLOB 데이터 유형이 될 수 있습니다.

외부 파일에 대한 *XML 유형*: XML 데이터 유형으로 저장된 XML 내용을 검색하고, 이를 외부 파일로 가져가기합니다.

```
Content(xmlobj XML type, filename varchar(512))
```

여기서,

xmlobj XML 내용을 검색할 XML 컬럼 이름. *xmlobj*는 유형 XMLVARCHAR 또는 XMLCLOB이 될 수 있습니다.

filename
XML 데이터가 저장될 파일 이름.

아래 예에서, *Embedded SQL*이 있는 작은 C 프로그램 세그먼트에서는 XML 문서가 파일에서 메모리로 검색되는 방식을 설명합니다. 이 예에서는 컬럼 BOOK의 유형이 XMLFILE인 것으로 가정합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
      SQL TYPE IS CLOB_LOCATOR xml_buff;
  EXEC SQL END DECLARE SECTION;
  EXEC SQL CONNECT TO SALES_DB
  EXEC SQL DECLARE c1 CURSOR FOR
      SELECT Content(order) from sales_tab
                EXEC SQL OPEN c1;
  do {
     EXEC SQL FETCH c1 INTO :xml_buff;
     if (SQLCODE != 0) {
        break;
        }
     else {
       /* 버퍼 내의 XML 문서에 대해, 필요한 모든 조치를 하십시오.*/
       }
     }
  EXEC SQL CLOSE c1;
  EXEC SQL CONNECT RESET;
```

요소 내용 및 속성 값 검색

하나 이상의 XML 문서(단일 문서 또는 콜렉션 문서 검색)에서 요소 또는 속성 값의 내용을 검색(추출)할 수 있습니다. XML Extender는 SQL 데이터 유형 각각에 대한 SQL SELECT 절에서 지정할 수 있는 사용자 정의 추출 함수를 제공합니다.

요소 및 속성의 내용과 값을 검색하는 작업은 응용프로그램 개발시 유용하며, 이는 사용자가 XML 데이터를 관계형 데이터로서 액세스할 수 있기 때문입니다. 예를 들어, 테이블 SALES_TAB에 있는 컬럼 ORDER에 1000개의 XML 문서가 저장되어 있다고 가정합니다. 이 정보를 검색하기 위해 SELECT 절에서 UDF를 추출하며, 다음 SQL 명령문을 사용하여 항목을 주문했던 모든 고객 이름을 검색할 수 있습니다.

```
SELECT extractVarchar(Order, '/Order/Customer/Name') from sales_order_view
   WHERE price > 2500.00
```

이 예에서, 추출 UDF는 VARCHAR 데이터 유형으로서 컬럼 ORDER에서 요소 <customer>를 검색합니다. 위치 경로는 /Order/Customer/Name입니다(위치 경로 구문에 대해서는 58 페이지의 『위치 경로』를 참조하십시오). 또한, WHERE 절을 사용하면 리턴되는 값의 수가 감소되며, 이는 하위 요소가 <ExtendedPrice>인 <customer> 요소의 내용만 2500.00 이상의 값을 가짐을 지정합니다.

요소 내용 또는 속성값을 추출하려면, 다음과 같이 하십시오. 테이블이나 *스칼라 함수*로서 다음 구문을 사용하여 141 페이지의 표13에 나열된 추출 UDF를 사용하십시오.

```
extract<retrieved_datatype>(xmlobj, path)
```

여기서,

retrieved_datatype

추출 함수에서 리터되는 데이터 유형. 이것은 다음 유형 중 하나가 될 수 있습니다.

- INTEGER
- SMALLINT
- DOUBLE

- REAL
- CHAR
- VARCHAR
- CLOB
- DATE
- TIME
- TIMESTAMP
- FILE

xmlobj 요소 또는 속성이 추출되는 XML 컬럼 이름. 이 컬럼은 다음 XML 사용자 정의 유형중 하나로서 정의되어야 합니다.

- XMLVARCHAR
- LOCATOR로서의 XMLCLOB
- XMLFILE

경로 XML 문서에서 요소 또는 속성의 위치 경로(예: `/Order/Customer/Name`) 입니다. 위치 경로 구문에 대해서는 58 페이지의 『위치 경로』를 참조하십시오.

중요사항: 추출 UDF는 요소가 아닌 속성을 지닌 술어를 보유한 위치 경로를 지원합니다. 예를 들어, 다음과 같은 술어가 지원됩니다.

```
'/Order/Part[@color="black "]/ExtendedPrice'
```

다음 술어는 지원되지 않습니다.

```
'/Order/Part/Shipment/[Shipdate < "11/25/00"]'
```

표13에서는 스칼라 및 테이블 형식 모두로 추출 함수를 보여줍니다.

표 13. XML Extender 추출 함수

스칼라 함수	테이블 함수	리턴된 컬럼 이름 (테이블 함수)	리턴 유형
extractInteger()	extractIntegers()	returnedInteger	INTEGER
extractSmallint()	extractSmallints()	returnedSmallint	SMALLINT
extractDouble()	extractDoubles()	returnedDouble	DOUBLE

표 13. XML Extender 추출 함수 (계속)

스칼라 함수	테이블 함수	리턴된 컬럼 이름 (테이블 함수)	리턴 유형
extractReal()	extractReals()	returnedReal	REAL
extractChar()	extractChars()	returnedChar	CHAR
extractVarchar()	extractVarchars()	returnedVarchar	VARCHAR
extractCLOB()	extractCLOBs()	returnedCLOB	CLOB
extractDate()	extractDates()	returnedDate	DATE
extractTime()	extractTimes()	returnedTime	TIME
extractTimestamp()	extractTimestamps()	returnedTimestamp	TIMESTAMP

스칼라 함수 예:

다음 예에서는 키의 속성 값이 ″1″일 때 하나의 값이 리턴됩니다. 이 값은 자동으로 DECIMAL 유형으로 변환되는 정수로 추출됩니다.

```
CREATE TABLE t1(key decimal(3,2));
INSERT into t1 values
SELECT * from table(db2xml.extractInteger(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/[@key="1"]'));
SELECT * from t1;
```

테이블 함수 예:

다음 예에서, 판매 주문에 대한 각각의 키 값은 INTEGER로 추출됩니다.

```
SELECT * from table(db2xml.extractIntegers(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/@key')) as x;
```

XML 데이터 갱신

XML Extender를 사용하면, XML 컬럼 데이터를 바꾸어 XML 문서 전체를 갱신하거나, 지정된 요소나 속성값을 갱신할 수 있습니다.

타스크 개요:

1. XML 테이블에 XML 문서를 저장했는지 확인하고, 검색하려는 데이터를 판별합니다.
2. DB2 테이블에서 데이터를 검색하기 위한 메소드를 선택합니다(유형 변환 함수 또는 UDF).

3. 갱신할 XML 테이블 및 컬럼을 지정하는 SQL 조회를 지정합니다.

중요사항: XML에 사용 가능한 컬럼을 갱신할 때, XML Extender는 변경사항을 반영하도록 자동으로 부가 테이블을 갱신합니다. 그러나, 해당 XML 요소나 속성 값을 변경함으로써 XML 컬럼에 저장된 원래 XML 문서를 갱신하지 않고 이들 테이블을 직접 갱신하지 마십시오. 이러한 갱신으로 인해 데이터 불일치 문제점이 발생할 수 있습니다.

XML 문서를 갱신하려면, 다음과 같이 하십시오.

다음 메소드 중 하나를 사용하십시오.

기본 유형 변환 함수 사용

사용자 정의 유형(UDT) 각각에 대해, SQL 기본 유형을 UDT로 변환하기 위한 기본 유형 변환 함수가 제공됩니다. XML Extender가 제공하는 유형 변환 함수를 사용하여 XML 문서를 갱신할 수 있습니다. 135 페이지의 표9에서는 제공되는 유형 변환 함수를 보여주며, XML Extender가 제공한 다른 UDT에 대한 컬럼 UDT가 작성된 것으로 가정합니다.

예: 유형 변환된 VARCHAR 유형에서 XMLVARCHAR 유형을 갱신합니다. 이때, `xml_buf`는 VARCHAR 유형으로서 정의된 호스트 변수인 것으로 간주합니다.

```
UPDATE sales_tab VALUES('123456', 'Sriram Srinivasan',
  db2xml.XMLVarchar(:xml_buff))
```

저장영역 UDF 사용

각 XML Extender UDT에 대해, 기본 유형 이외의 자원에서 DB2로 데이터를 가져오기하기 위해 저장영역 UDF가 제공됩니다. 저장영역 UDF를 사용하여 전체 XML 문서를 바꿈으로써 이를 갱신할 수 있습니다.

예: 다음 예에서는 XMLVarcharFromFile() 함수를 사용하여 XML 문서를 갱신합니다.

```
UPDATE sales_tab
       set order = XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml')
   WHERE sales_person = 'Sriram Srinivasan'
```

앞의 예에서는 XML 오브젝트를 `c:\dxx\samples\cmd\getstart.xml` 파일에서 SALES_TAB 테이블에 있는 ORDER 컬럼으로 갱신합니다.

XML Extender가 제공하는 저장영역 함수 목록에 대해서는 136 페이지의 표10을 참조하십시오.

XML 문서의 특정 요소 및 속성을 갱신하려면, 다음과 같이 하십시오.

Update() UDF를 사용하면 전체 문서를 갱신하지 않고 문서에서 한 번에 하나의 값을 갱신할 수 있습니다. UDF를 사용하여, 대체될 위치 경로로 표현되는 요소 또는 속성값과 위치 경로를 지정합니다(위치 경로 구문에 대해서는 58 페이지의 『위치 경로』를 참조하십시오). XML 문서를 편집할 필요는 없습니다. XML Extender가 변경합니다.

갱신 UDF는 전체 XML 파일을 갱신하며, XML 구문분석기의 정보를 기반으로 파일을 재구성합니다. 갱신 UDF가 문서를 처리하는 방법 및 갱신 전후의 예제 문서에 대한 정보는 226 페이지의 『갱신 함수가 XML 문서를 처리하는 방법』의 내용을 참조하십시오.

구문:

```
Update(xmlobj, path, value)
```

여기서,

xmlobj 갱신될 요소나 속성값에 대한 XML 컬럼 이름.

경로 갱신될 요소나 속성의 위치 경로. 위치 경로 구문에 대해서는 58 페이지의 『위치 경로』를 참조하십시오. 다중 발생에 대한 고려사항은 228 페이지의 『다중 발생』의 내용을 참조하십시오.

value 갱신될 값.

예: 다음 명령문에서는 Update() UDF를 사용하여 <Customer> 요소값을 문자열 IBM으로 갱신합니다.

```
UPDATE sales_tab
       set order = Update(order, '/Order/Customer/Name', 'IBM')
   WHERE sales_person = 'Sriram Srinivasan'
```

다중 발생:

위치 경로가 Update() UDF에서 제공되는 경우, 일치 경로를 지닌 모든 요소나 속성의 내용은 제공되는 값으로 갱신됩니다. 이는 문서가 다중 발생 위치 경로를 보유하는 경우에 갱신 함수가 기존 값을 값 매개변수에서 제공되는 값으로 대체함을 의미합니다.

XML 문서 검색

XML 데이터 탐색(search)은 XML 데이터 탐색(retrieve)과 비슷합니다. 두 기술 모두 작업을 위해 데이터를 검색하지만 Where 절을 사용하여 검색 기준으로서 술어를 정의하여 검색합니다.

XML Extender는 응용프로그램 필요에 따라 XML 컬럼에서 XML 문서를 검색하는 여러 메소드를 제공합니다. 문서 구조를 검색하고 요소 내용 및 속성값에 따라 결과를 리턴하는 기능도 제공합니다. XML 컬럼 및 부가 테이블의 뷰를 검색할 수 있고, 보다 나은 성능을 위해 부가 테이블을 직접 검색하거나 Where 절과 함께 추출 UDF를 사용할 수도 있습니다. 뿐만 아니라 DB2 Text Extender를 사용하고, 텍스트 문자열에 대한 내용에서 컬럼 데이터를 검색할 수 있습니다.

XML Extender를 사용하여 부가 테이블 컬럼에 대해 색인을 사용할 수 있으며, 이 컬럼에는 고속 검색을 위해 XML 문서에서 추출된 XML 요소 내용 또는 속성값이 들어 있습니다. 요소나 속성의 데이터 유형을 지정함으로써, SQL 일반 데이터를 검색하거나 범위 검색을 수행할 수 있습니다. 예를 들어 구매 주문 예에서, 가격이 2500.00을 초과하는 모든 주문을 검색할 수 있습니다.

뿐만 아니라, DB2 UDB Text Extender를 사용하여 구조적인 텍스트 검색 또는 전체 텍스트 검색을 수행할 수 있습니다. 예를 들어, XML 형식으로 된 이력서가 들어있는 RESUME 컬럼이 있습니다. Java에 능숙한 지원자 모두의 이름을 원합니다. DB2 Text Extender를 사용하여 XML 문서에서 <skill> 요소에 JAVA 문자열이 들어있는 이력서 모두를 검색할 수 있습니다.

다음 절에서는 검색 메소드에 대해 설명합니다.

- 146 페이지의『구조별 XML 문서 검색』
- 148 페이지의『구조적인 텍스트 검색을 위한 Text Extender 사용』

구조별 XML 문서 검색

XML Extender 검색 기능을 사용하여, 문서 구조 즉 요소 및 속성을 기초로 컬럼에서 XML 데이터를 검색할 수 있습니다. 컬럼 데이터를 검색하기 위해, 여러 방식으로 SELECT문을 사용하며, 문서 요소 및 속성에 대한 일치 여부를 근거로 *결과 세트*를 리턴합니다. 다음 메소드를 사용하여 컬럼 데이터를 검색할 수 있습니다.

- 부가 테이블에 대한 직접 조회로 검색
- *조인된 뷰*에서 검색
- 추출 UDF로 검색
- 여러 번 발생할 요소 또는 속성 검색

이러한 메소드에 대해서는 다음 절에서 설명되며, 다음 시나리오로 예를 사용합니다. 응용프로그램 테이블 SALES_TAB에는 ORDER라는 XML 컬럼이 있습니다. 이 컬럼에는 세 개의 부가 테이블 즉, ORDER_SIDE_TAB, PART_SIDE_TAB 및 SHIP_SIDE_TAB이 있습니다. ORDER 컬럼을 사용할 수 있고 다음 CREATE VIEW 명령문을 사용하여 이들 테이블을 조인할 때 기본 뷰 sales_order_view가 지정되었습니다.

```
CREATE VIEW sales_order_view(invoice_num, sales_person, order,
                             order_key, customer, part_key, price, date)
     AS
     SELECT sales_tab.invoice_num, sales_tab.sales_person, sales_tab.order,
            order_side_tab.order_key, order_side_tab.customer,
            part_side_tab.part_key, ship_side_tab.date
     FROM sales_tab, order_side_tab, part_side_tab, ship_side_tab
     WHERE sales_tab.invoice_num = order_side_tab.invoice_num
                AND sales_tab.invoice_num = part_side_tab.invoice_num
                AND sales_tab.invoice_num = ship_side_tab.invoice_num
```

부가 테이블에 대한 직접 조회로 검색

부속 조회 검색 기능이 있는 직접 조회에서는 부가 테이블에 색인을 지정할 때 최고 성능의 구조적 검색을 제공합니다. 조회나 부속 조회를 사용하여 부가 테이블을 정확하게 검색할 수 있습니다.

예: 다음 명령문에서는 조회 및 부속 조회를 사용하여 부가 테이블을 직접 검색합니다.

```
SELECT sales_person from sales_tab
       WHERE invoice_num in
          (SELECT invoice_num from part_side_tab
           WHERE price > 2500.00)
```

이 예에서, invoice_num는 SALES_TAB 테이블에 있는 기본 키입니다.

조인된 뷰에서 검색

XML Extender가 고유 ID를 사용하여 응용프로그램 테이블과 부가 테이블을 조인하는 기본 뷰를 작성하도록 할 수 있습니다. 이 기본 뷰나, 응용프로그램 테이블과 부가 테이블을 조인하는 임의 뷰를 사용하여 컬럼 데이터를 검색하고 부가 테이블을 조회할 수 있습니다. 이 메소드에서는 응용프로그램 테이블 및 부가 테이블의 단일 가상 뷰를 제공합니다. 그러나, 부가 테이블을 작성하면 할수록 조회 시 비용이 추가됩니다.

추가 정보: *root_id*, 또는 DXXROOT_ID(XML Extender가 작성함)를 사용하여 자신의 뷰를 작성할 때 테이블을 조인할 수 있습니다.

예: 다음 명령문은 뷰를 검색합니다.

```
SELECT sales_person from sales_order_view
   WHERE price > 2500.00
```

SQL문은 가격이 2500.00 이상인 항목 주문이 있는 조인된 뷰 sales_order_view 테이블에서 sales_person 값을 리턴합니다.

추출 UDF로 검색

또한 응용프로그램 테이블에 대해 색인이나 부가 테이블을 작성하지 않은 경우 XML Extender의 추출 UDF를 사용하여 요소 및 속성을 검색할 수도 있습니다. 추출 UDF를 사용하여 XML 데이터를 스캔하는 것은 비용이 많이 소요되며, 검색시 포함될 XML 문서 수를 제한하는 WHERE 절과 함께만 사용되어야 합니다.

예: 다음 명령문은 추출 XML Extender UDF로 검색합니다.

```
SELECT sales_person from sales_tab
     WHERE extractVarchar(order, '/Order/Customer/Name')
                          like '%IBM%'
                                  AND invoice_num > 100
```

이 예에서, 추출 UDF는 IBM 값으로 </Order/Customer/Name> 요소를 추출합니다.

여러 번 발생한 요소 및 속성 검색

여러 번 발생한 요소나 속성을 검색할 경우, DISTINCT 절을 사용하여 값이 중복되지 않도록 하십시오.

예: 다음 명령문에서는 DISTINCT 절로 검색합니다.

```
SELECT sales_person from sales_tab
       WHERE invoice_num in
              (SELECT DISTINCT invoice_num from part_side_tab
          WHERE price > 2500.00)
```

이 예에서, DAD 파일은 /Order/Part/Price가 여러 번 발생하도록 지정하고, 이에 대해 PART_SIDE_TAB 부가 테이블을 작성합니다. PART_SIDE_TAB 테이블에는 invoice_num가 같은 행이 여러 개 있습니다. DISTINCT를 사용하면 고유한 값만이 리턴됩니다.

구조적인 텍스트 검색을 위한 Text Extender 사용

XML 문서를 검색할 때, XML Extender는 일반 데이터 유형으로 변환된 요소 및 속성값을 검색하지만, 텍스트는 검색하지 않습니다. XML에 사용 가능한 컬럼에 대해 구조적 또는 전체 텍스트 검색을 수행하기 위해 DB2 UDB Text Extender를 사용할 수 있습니다. Text Extender는 DB2 UDB 버전 6.1 이상에서 XML 문서 검색을 지원합니다. Text Extender는 Windows 운영 체제, AIX 및 Sun Solaris에서 사용할 수 있습니다.

구조적인 텍스트 검색

XML 문서의 트리 구조를 기초로 텍스트 문자열을 검색합니다. 예를 들어 문서 구조가 /Order/Customer/Name이고 <Customer> 부속요소내에서 문자열『IBM』을 찾으려고 할 경우, 구조적인 텍스트 검색을 사용할 수 있습니다. 문서에는 <Comment> 부속요소에 또는 제품 파트명으로서 문자열 IBM이 있을 수 있습니다. 구조적인 텍스트 검색에서는 지정된 요소에서 문자열만을 검색합니다. 이 예에서는, </Order/Customer/Name> 부속요소에 IBM이 있는 문서만 발견되며, 다른 요소에는 IBM이 있지만 </Order/Customer/Name> 부속요소에는 없는 문서는 리턴되지 않습니다.

전체 텍스트 검색

요소나 속성에 관계없이, 문서 구조 어느 위치에서나 텍스트 문자열을 검색합니다. 이전 예에서는, 문자열 IBM이 있는지 여부에 관계없이 문자열 IBM이 있는 모든 문서가 리턴됩니다.

Text Extender 검색을 사용하려면, Text Extender를 설치하고 아래에서 설명한 것처럼 데이터베이스와 테이블을 사용할 수 있어야 합니다. Text Extender 검색의 사용 방법을 알려면, *DB2 Universal Database Text Extender 관리 및 프로그래밍*에 있는 Text Extender UDF 검색 장을 참조하십시오.

Text Extender에 XML 컬럼 사용 가능

XML에 사용 가능한 데이터베이스가 있는 것으로 가정하고, 다음 단계에서 XML 가능 컬럼의 내용을 검색하기 위해 Text Extender를 사용 가능으로 설정합니다. 예를 들어, 데이터베이스의 이름은 SALES_DB, 테이블 이름은 ORDER, 그리고 XML 컬럼 이름은 XVARCHAR 및 XCLOB입니다.

1. Text Extender 설치 방법에 대해서는 Extender CD에 있는 `install.txt` 파일을 참조하십시오.
2. 다음 위치 중 하나에서 **txstart** 명령을 입력하십시오.
   - UNIX 운영 체제에서는, 인스턴스 소유자의 명령 프롬프트에서 명령을 입력하십시오.
   - Windows NT에서는, DB2INSTANCE가 지정된 명령 창에서 명령을 입력하십시오.
3. Text Extender 명령행 창을 여십시오. 이 단계에서는 데이터베이스의 이름이 SALES_DB이고, 테이블 이름이 ORDER인 것으로 가정합니다. `dxx\samples\c`에서 샘플 프로그램을 실행해야 합니다.
4. 데이터베이스에 연결하십시오. **db2tx** 명령 프롬프트에서 다음을 입력하십시오.

   ```
   'connect to SALES_DB'
   ```

5. Text Extender용으로 데이터베이스를 사용할 수 있도록 합니다.

   db2tx 명령 프롬프트에서 다음을 입력하십시오.

   ```
   'enable database'
   ```

6. XML 테이블에서 Text Extender용으로 컬럼을 사용하게 합니다. 이 때 XML 문서의 데이터 유형, 언어, 코드 페이지 및 컬럼에 대한 기타 정보를 정의합니다.

   - VARCHAR 컬럼 XVARCHAR에서 다음을 입력하십시오.

     ```
     'enable text column order xvarchar function db2xml.varchartovarchar handle
     varcharhandle ccsid 850 language us_english format xml indextype precise
         indexproperty sections_enabled
     documentmodel (Order) updateindex update'
     ```

   - CLOB 컬럼 XCLOB에서, 다음을 입력하십시오.

     ```
     'enable text column order xclob function db2xml.clob handle clobhandle
      ccsid 850 language us_english indextype precise updateindex update'
     ```

7. 색인 상태를 확인하십시오.

   - 컬럼 XVARCHAR의 경우, 다음을 입력하십시오: `get index status order handle varcharhandle`
   - 컬럼 XCLOB의 경우, 다음을 입력하십시오: `get index status order handle clobhandle`

8. XML 문서 모델을 이름이 `desmodel.ini`인 문서 모델 INI 파일에 정의합니다. 이 파일은 UNIX의 경우 `/db2tx/txins000`에, Windows NT의 경우 `\instance\\db2tx\txins000`에, 초기 설정 파일에 있는 섹션에 있습니다. 예를 들어 `textmodel.ini`의 경우,

   ```
   ;list of document models
   [MODELS]
   modelname=Order
   ; an 'Order' document model definition
   ; left side = section name identifier
   ; right side = section name tag
   [Order]
   Order = /Order
   Order/Customer/Name = /Order/Customer/Name
   Order/Customer/Email = /Order/Customer/Email
   Order/Part/@color = /Order/Part/@color
   Order/Part/Shipment/ShipMode = /Order/Part/Shipment/ShipMode
   ```

Text Extender를 사용한 텍스트 검색

Text Extender의 검색 기능은 XML Extender 문서 구조 검색에 잘 동작합니다. 권장되는 방법은 문서 요소나 속성을 검색하는 조회를 작성하고, Text Extender를 사용하여 요소 내용 또는 속성 값을 검색하는 것입니다.

예: 다음 명령문에서는 Text Extender로 XML 문서 텍스트를 검색합니다. DB2 명령 창에서 다음을 입력하십시오.

```
'connect to SALES_DB'
'select xvarchar from order where db2tx.contains(varcharhandle,
        'model Order section(Order/Customer/Name) "Motors"')=1'
'select xclob from order where db2tx.contains(clobhandle,
        'model Order section(Order/Customer/Name) "Motors"')=1'
```

Text Extender Contains() UDF가 검색합니다.

이 예에 컬럼 데이터를 검색하기 위해 Text Extender를 사용해야 하는 모든 단계가 포함되는 것은 아닙니다. Text Extender 검색 개념 및 기능을 알려면, *DB2 Universal Database Text Extender 관리 및 프로그래밍*에 있는 Text Extender UDF로 검색 장을 참조하십시오.

XML 문서 삭제

SQL DELETE문을 사용하여 XML 컬럼에서 XML 문서 행을 삭제하십시오. Where 절을 지정하여 삭제할 문서를 정의할 수 있습니다.

예: 다음 명령문은 <ExtendedPrice>가 2500.00 이상인 모든 문서를 삭제합니다.

```
DELETE from sales_tab
       WHERE invoice_num in
          (SELECT invoice_num from part_side_tab
           WHERE price > 2500.00)
```

JDBC에서 함수 호출시 제한사항

함수에서 매개변수 표시문자를 사용하는 경우, JDBC 제한사항은 함수에 대한 매개변수 표시문자가 리턴되는 데이터가 삽입되는 컬럼의 데이터 유형으로 변환되어야 함을 요구합니다. 함수 선택 로직은 인수의 최종 데이터 유형을 알 수 없으며, 이는 참조를 분석할 수 없습니다.

결과적으로, JDBC는 다음 코드를 분석할 수 없습니다.

```
db2xml.XML_default_casting_function(length)
```

CAST 스펙을 사용하면 VARCHAR와 같은 매개변수 표시문자에 대한 유형을 제공할 수 있으며, 그 후에 함수 선택 로직이 처리될 수 있습니다.

```
db2xml.XML default_casting_function(CAST(? AS cast_type(length))
```

예 1: 다음 예에서, 매개변수 표시문자는 VARCHAR로 유형 변환됩니다. 전달되는 매개변수는 XML 문서이며, 이는 VARCHAR(1000)로 유형 변환되어 ORDER 컬럼에 삽입됩니다.

```
String query = "insert into sales_tab(invoice_num, sales_person, order) values
    (?,?,db2xml.XMLVarchar(cast (? as varchar(1000))))";
```

예 2: 다음 예에서, 매개변수 표시문자는 VARCHAR로 유형 변환됩니다. 전달되는 매개변수는 파일 이름이며, 이 내용은 VARCHAR로 유형 변환되어 ORDER 컬럼에 삽입됩니다.

```
String query = "insert into sales_tab(invoice_num, sales_person, order) values
    (?,?,db2xml.XMLVarcharfromFILE(cast (? as varchar(1000))))";
```

제6장 XML 콜렉션 데이터 관리

XML 콜렉션은 XML 문서로 맵핑되는 데이터가 포함된 관계형 테이블 세트입니다. 이 액세스 및 저장영역 메소드를 사용하면 기존 데이터로 XML 문서를 작성하고, XML 문서를 분해하며, XML을 교환 메소드로서 사용할 수 있습니다.

관계형 테이블은 XML 문서를 분해할 때 XML Extender가 생성하는 새 테이블이거나, XML Extender가 응용프로그램에 대한 XML문서를 생성하기 위해 사용하는 데이터가 들어있는 기존 테이블이 될 수 있습니다. 이러한 테이블에 있는 컬럼 데이터에는 XML 태그가 들어있지 않고, 요소 및 속성과 연관된 내용이나 값들이 각각 들어 있습니다. 저장 프로시듀어는 XML 콜렉션 데이터의 저장, 탐색(retrieve), 갱신, 검색(search) 및 삭제를 위한 액세스 및 저장영역 메소드로서 동작합니다.

XML 콜렉션 저장 프로시듀어에 의해 사용되는 매개변수 한계는 317 페이지의 『부록D. XML Extender 한계』에 문서화되어 있습니다.

233 페이지의 『CLOB 한계 증가』에 문서화된 바와 같이 저장 프로시듀어 결과에 대한 CLOB *크기*를 증시킬 수 있습니다.

XML 콜렉션 관리에 대해서는 다음 절을 참조하십시오.

- 『DB2 데이터에서 XML 문서 작성』
- 162 페이지의 『DB2 데이터로 XML 문서 분해』

DB2 데이터에서 XML 문서 작성

작성은 XML 콜렉션에 있는 관계형 데이터에서 XML 문서 세트를 생성하는 것입니다. 저장 프로시듀어를 사용하여 XML 문서를 작성할 수 있습니다. 이러한 저장 프로시듀어를 사용하려면, DAD 파일을 작성해야 하며, 이 파일은 XML 문서와 DB2 테이블 구조간의 맵핑을 지정합니다. 저장 프로시듀어는 DAD 파일을 사용하여 XML 문서를 작성합니다. DAD 파일 작성법에 대해서는 62 페이지의 『XML 콜렉션 플랜』을 참조하십시오.

시작하기 전에

- 요소 내용과 속성값이 들어있는 관계형 테이블로 XML 문서의 구조를 맵핑하십시오.
- 맵핑 메소드 즉, SQL 또는 RDB_node 맵핑을 선택하십시오.
- DAD 파일을 준비하십시오. 완전한 세부사항에 대해서는 62 페이지의 『XML 콜렉션 플랜』을 참조하십시오.
- 선택적으로, XML 콜렉션을 사용할 수 있도록 설정하십시오.

XML 문서 작성

XML Extender는 두개의 저장 프로시듀어 즉, dxxGenXML() 및 dxxRetrieveXML()을 제공하여 XML 문서를 작성합니다.

dxxGenXML()

이 저장 프로시듀어는 때때로 갱신하는 응용프로그램 또는 XML 데이터 관리 오버헤드를 원치않는 응용프로그램에 사용됩니다. 저장 프로시듀어 dxxGenXML()에서는 사용 가능한 콜렉션이 필요없습니다. 대신 DAD 파일을 사용합니다.

저장 프로시듀어 dxxGenXML()은 XML 콜렉션 테이블에 저장된 데이터를 사용하여 XML 문서를 생성하며, DAD 파일에서 <Xcollection> 요소가 지정합니다. 이 저장 프로시듀어는 하나의 행으로 각각의 XML 문서를 *결과 테이블*에 삽입합니다. 또한 커서로 결과 테이블을 열고 결과 세트를 사전 추출할 수 있습니다. 결과 테이블은 응용프로그램에 의해 작성되어야 하며, 항상 VARCHAR, CLOB, XMLVARCHAR 또는 XMLCLOB 유형의 컬럼이 있어야 합니다.

또한, DAD 파일의 유효성 검증 요소를 YES로 지정하는 경우에 XML Extender는 유형이 INTEGER인 DXX_VALID 컬럼을 결과 테이블에 추가하고, 유효한 XML 문서인 경우에는 값 1을, 유효하지 않은 문서인 경우에는 0을 삽입합니다.

저장 프로시듀어 dxxGenXML()을 사용하면 결과 테이블에서 생성될 최대 행 수를 지정할 수도 있습니다. 이것은 처리 시간을 줄여줍니다. 저장 프로시듀어는 리턴 코드 및 메시지와 함께, 테이블에 있는 실제 행 수를 리턴합니다.

분해에 해당하는 저장 프로시듀어는 dxxShredXML()이며, 입력 매개변수로서 DAD를 사용하지만 XML 콜렉션을 사용 가능하도록 설정할 필요는 없습니다.

XML 콜렉션 작성하기: dxxGenXML()

다음 저장 프로시듀어 선언을 사용하여 응용프로그램에 저장 프로시듀어 호출을 삽입하십시오.

```
dxxGenXML(CLOB(100K)     DAD,              /* input */
          char(resultTabName) resultTabName,   /* input */
               integer        overrideType,    /* input */
               varchar(1024)  override,        /* input */
               integer        maxRows,         /* input */
               integer        numRows,         /* output */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

전체 구문과 예에 대해서는 243 페이지의 『dxxGenXML()』을 참조하십시오.

예: 다음 예에서는 XML 문서를 작성합니다.

```
#include "dxx.h"
#include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE is CLOB(100K) dad;             /* DAD */
     SQL TYPE is CLOB_FILE  dadFile;         /* dad file */
         char    result_tab[32];  /* name of the result table */
              char    override[2];      /* override, will set to NULL*/
              short   overrideType;     /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
         char                  returnMsg[1024];  /* error message text */
         short                 dad_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                 returnCode_ind;
         short                 returnMsg_ind;
          EXEC SQL END DECLARE SECTION;
         /* create table */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);
     /* read data from a file to a CLOB */
     strcpy(dadfile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
     dadfile.name_length = strlen("c:\dxx\samples\dad\getstart_xcollection.dad");
     dadfile.file_options = SQL_FILE_READ;
     EXEC SQL VALUES (:dadfile) INTO :dad;
         strcpy(result_tab,"xml_order_tab");
     override[0] = '\0';
     overrideType = NO_OVERRIDE;
         max_row = 500;
         num_row = 0;
        returnCode = 0;
```

```
    msg_txt[0] = '\0';
    collection_ind = 0;
   dad_ind = 0;
    rtab_ind = 0;
ov_ind = -1;
    ovtype_ind = 0;
    maxrow_ind = 0;
    numrow_ind = -1;
   returnCode_ind = -1;
   returnMsg_ind = -1;
   /* Call the store procedure */
EXEC SQL CALL dxxGenXML(:dad:dad_ind;
          :result_tab:rtab_ind,
     :overrideType:ovtype_ind,:override:ov_ind,
          :max_row:maxrow_ind,:num_row:numrow_ind,
               :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

저장 프로시듀어가 호출된 다음에 결과 테이블에는 250 행이 들어있게 되며, 이는 DAD 파일에서 지정된 SQL 조회에서 250 XML 문서를 생성하기 때문입니다.

dxxRetrieveXML()

이 저장 프로시듀어는 정규 갱신을 수행하는 응용프로그램에 사용됩니다. 같은 타스크가 반복되므로, 성능 향상이 중요합니다. XML 콜렉션을 사용 가능으로 설정하고 저장 프로시듀어에 콜렉션 이름을 사용하면 성능이 향상됩니다.

저장 프로시듀어 dxxRetrieveXML()은 저장 프로시듀어 dxxGenXML()과 동일하게 동작하지만, DAD 파일 대신 사용 가능한 XML 콜렉션 이름을 사용합니다. XML 콜렉션을 사용할 수 있을 때, DAD 파일은 XML_USAGE 테이블에 저장됩니다. 그러므로, XML Extender는 DAD 파일을 검색하고, 이 때부터 계속 dxxRetrieveXML()은 dxxGenXML()과 같게 됩니다.

dxxRetrieveXML()을 사용하면 작성과 분해 모두에 같은 DAD 파일을 사용할 수 있습니다. 이 저장 프로시듀어는 또한 분해된 XML 문서를 검색할 때에도 사용될 수 있습니다.

분해에 해당하는 저장 프로시듀어는 dxxInsertXML()이며, 사용 가능한 XML 콜렉션의 이름을 사용합니다.

XML 콜렉션 작성하기: dxxRetrieveXML()

다음 저장 프로시듀어 선언을 사용하여 응용프로그램에 저장 프로시듀어 호출을 삽입하십시오.

```
dxxRetrieveXML(char(collectionName) collectionName,  /* input */
               char(resultTabName) resultTabName,   /* input */
               integer        overrideType,     /* input */
               varchar(1024)  override,        /* input */
               integer        maxRows,         /* input */
               integer        numRows,         /* output */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

전체 구문과 예에 대해서는 246 페이지의 『dxxRetrieveXML()』을 참조하십시오.

예: 다음 예는 dxxRetrieveXML()을 호출하는 예입니다. 결과 테이블이 XML_ORDER_TAB 이름으로 작성되고 XMLVARCHAR 유형의 한 컬럼이 있는 것으로 가정합니다.

```
#include "dxx.h"
#include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
              char    override[2];      /* override, will set to NULL*/
              short   overrideType;     /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
         char                    returnMsg[1024];  /* error message text */
         short   dadbuf_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                   returnCode_ind;
         short                   returnMsg_ind;
          EXEC SQL END DECLARE SECTION;
         /* create table */
         EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);
        /* initialize host variable and indicators */
         strcpy(collection,"sales_ord");
         strcpy(result_tab,"xml_order_tab");
     override[0] = '\0';
     overrideType = NO_OVERRIDE;
         max_row = 500;
         num_row = 0;
        returnCode = 0;
         msg_txt[0] = '\0';
         collection_ind = 0;
         rtab_ind = 0;
     ov_ind = -1;
         ovtype_ind = 0;
         maxrow_ind = 0;
         numrow_ind = -1;
        returnCode_ind = -1;
        returnMsg_ind = -1;
        /* Call the store procedure */
              EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
```

```
        :result_tab:rtab_ind,
    :overrideType:ovtype_ind,:override:ov_ind,
        :max_row:maxrow_ind,:num_row:numrow_ind,
            :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

DAD 파일에서 동적으로 값 겹쳐쓰기

동적 조회의 경우, 두 개의 선택적인 매개변수, 즉 *override* 및 *overrideType*을 사용하여 DAD 파일에서 조건을 겹쳐쓰기할 수 있습니다. *overrideType*으로부터의 입력에 따라, 응용프로그램은 DAD에서 SQL 맵핑의 경우 <SQL_stmt> 태그값을, RDB_node 맵핑의 경우 RDB_node 조건을 겹쳐쓰기할 수 있습니다.

이러한 매개변수에는 다음 값과 규칙이 있습니다.

overrideType

이 매개변수는 *override* 매개변수의 유형을 표시하는 필수 입력 매개변수(IN)입니다. *overrideType*에는 다음 값이 있을 수 있습니다.

NO_OVERRIDE

DAD 파일에서 조건을 겹쳐쓰기하지 않도록 지정합니다.

SQL_OVERRIDE

DAD 파일에 있는 조건을 SQL문으로 겹쳐쓰기하도록 지정합니다.

XML_OVERRIDE

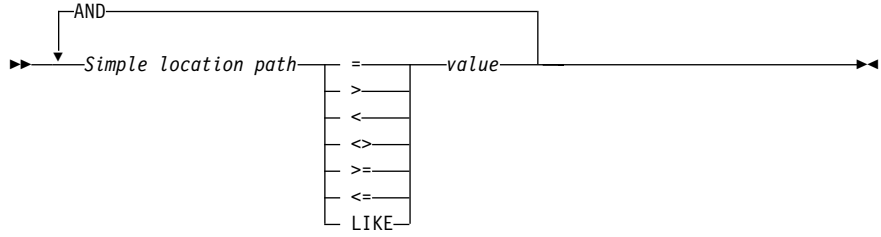
DAD 파일에 있는 조건을 XPath 기본의 조건으로 겹쳐쓰기하도록 지정합니다.

override

이 매개변수는 DAD 파일에 대한 겹쳐쓰기 조건을 지정하는 선택적인 입력 매개변수(IN)입니다. 입력 값 구문은 *overrideType*에서 지정된 값과 대응됩니다.

- NO_OVERRIDE를 지정하면, 입력값은 널 문자열입니다.
- SQL_OVERRIDE를 지정하면, 입력값은 유효한 SQL문입니다. ***필수:*** SQL_OVERRIDE 및 SQL문을 사용하는 경우, DAD 파일에 SQL 맵핑 스킴을 사용해야 합니다. 입력 SQL문은 DAD 파일에서 <SQL_stmt> 요소가 지정한 SQL문을 겹쳐쓰게 됩니다.
- XML_OVERRIDE를 사용하는 경우, 입력값은 하나 이상의 표현식이 포함된 문자열입니다. ***필수:*** XML_OVERRIDE 및 표현식을 사용하는 경우에

는 DAD 파일에서 RDB_node 맵핑 스킴을 사용해야 합니다. 입력 XML 표현식은 DAD파일에서 지정된 RDB_node 조건을 겹쳐쓰기합니다. 표현식에서는 다음 구문을 사용합니다.



여기서,

Simple location path

XPATH에서 정의된 구문을 사용한 단순 위치 경로. 구문에 대해서는 60 페이지의 표5 참조.

연산자

연산자를 표현식의 다른 부분과 구분하기 위해 공백을 사용할 수 있습니다.

value

숫자 값 또는 작은 따옴표로 묶인 문자열.

연산 주위에 선택적 공간을 가질 수 있습니다. LIKE 연산자 주위에는 공간을 필수입니다.

XML_OVERRIDE 값이 지정되면, 단순 위치 경로와 일치하는 text_node 또는 attribute_node에서 RDB_node의 조건이 지정된 표현식으로 겹쳐쓰기 됩니다.

XML_OVERRIDE는 XPath를 완전히 준수하지 않습니다. 단순 위치 경로는 컬럼에 맵핑된 요소나 속성을 식별하는 데만 사용됩니다.

예:

다음 예에서는 SQL_OVERRIDE 및 XML_OVERRIDE를 사용한 동적 겹쳐쓰기를 보여줍니다. 이 책에 있는 대부분의 저장 프로시듀어에서는 NO_OVERRIDE를 사용합니다.

예: SQL_OVERRIDE를 사용하는 저장 프로시듀어.

```
include "dxx.h"
include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
         char    override[256];   /* override, SQL_stmt */
         short   overrideType;      /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
         char                   returnMsg[1024];  /* error message text */
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                  returnCode_ind;
         short                  returnMsg_ind;
          EXEC SQL END DECLARE SECTION;
         /* create table */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);
        /* initialize host variable and indicators */
         strcpy(collection,"sales_ord");
         strcpy(result_tab,"xml_order_tab");
         sprintf(override,"%s %s %s %s %s %s %s",
                   "SELECT o.order_key, customer, p.part_key, quantity, price,",
                   "tax, ship_id, date, mode ",
                   "FROM order_tab o, part_tab p,",
                   "table(select substr(char(timestamp(generate_unique())),16",
                   "as ship_id,date,mode from ship_tab)as s",
                   "WHERE p.price > 50.00 and s.date >'1998-12-01' AND",
                   "p.order_key = o.order_key and s.part_key = p.part_key");
        overrideType = SQL_OVERRIDE;
         max_row = 500;
         num_row = 0;
        returnCode = 0;
         msg_txt[0] = '\0';
         collection_ind = 0;
         rtab_ind = 0;
         ov_ind = 0;
         ovtype_ind = 0;
         maxrow_ind = 0;
         numrow_ind = -1;
        returnCode_ind = -1;
        returnMsg_ind = -1;
        /* Call the store procedure */
              EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
                :result_tab:rtab_ind,
```

```
:overrideType:ovtype_ind,:override:ov_ind,
:max_row:maxrow_ind,:num_row:numrow_ind,
      :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

이 예에서, DAD 파일에 있는 <xcollection> 요소에는 <SQL_stmt> 요소가 있어야 합니다. *override* 매개변수는 값을 50.00 이상이 되도록 변경함으로써 <SQL_stmt> 값을 겹쳐쓰기 하고, 날짜는 1998-12-01보다 크게 변경됩니다.

예: XML_OVERRIDE를 사용한 저장 프로시듀어.

```
include "dxx.h"
include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
         char    override[256];   /* override, SQL_stmt */
         short   overrideType;    /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
         char                   returnMsg[1024];  /* error message text */
         short   dadbuf_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                  returnCode_ind;
         short                  returnMsg_ind;
          EXEC SQL END DECLARE SECTION;
         /* create table */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);
        /* initialize host variable and indicators */
         strcpy(collection,"sales_ord");
         strcpy(result_tab,"xml_order_tab");
         sprintf(override,"%s %s",
                   "/Order/Part/Price > 50.00 AND ",
                   "Order/Part/Shipment/ShipDate > '1998-12-01'");
        overrideType = XML_OVERRIDE;
         max_row = 500;
         num_row = 0;
        returnCode = 0;
         msg_txt[0] = '\0';
         collection_ind = 0;
         rtab_ind = 0;
         ov_ind = 0;
         ovtype_ind = 0;
         maxrow_ind = 0;
         numrow_ind = -1;
        returnCode_ind = -1;
        returnMsg_ind = -1;
        /* Call the store procedure */
              EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
                :result_tab:rtab_ind,
```

```
:overrideType:ovtype_ind,:override:ov_ind,
:max_row:maxrow_ind,:num_row:numrow_ind,
      :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

이 예에서, DAD 파일에 있는 <collection> 요소에는 루트 element_node에 대한 RDB_node가 있어야 합니다. *override* 값은 XML 내용에 따라 달라집니다. XML Extender는 단순 위치 경로를 맵핑된 DB2 컬럼으로 변환시킵니다.

DB2 데이터로 XML 문서 분해

XML 문서를 분해하는 것은 XML 문서 내에서 데이터를 분할하고 이를 관계형 테이블에 저장하는 것입니다. XML Extender는 소스 XML 문서에서 관계형 테이블로 XML 데이터를 분해하기 위해 저장 프로시듀어를 제공합니다. 이 저장 프로시듀어를 사용하려면, XML 문서와 DB2 테이블 구조 간의 맵핑을 지정하는 DAD 파일을 작성해야 합니다. 저장 프로시듀어는 DAD 파일을 사용하여 XML 문서를 분해합니다. DAD 파일 작성법에 대해서는 62 페이지의 『XML 콜렉션 플랜』을 참조하십시오.

분해를 위해 XML 콜렉션을 사용

대부분의 경우, 저장 프로시듀어를 사용하기 전에 XML 콜렉션을 사용할 수 있어야 합니다. 다음 경우, XML 콜렉션을 사용할 수 있게 되어야 합니다.

- XML 문서를 새 테이블로 분해할 때, XML 콜렉션에 있는 모든 테이블은 콜렉션을 사용할 수 있을 때 XML Extender가 작성하므로 XML 콜렉션을 사용할 수 있어야 합니다.
- 여러 번 발생하는 요소와 속성의 순서를 유지하는 것이 중요한 경우. XML Extender는 콜렉션을 사용할 수 있는 동안 작성된 테이블에서 여러 번 발생한 요소나 속성의 순서만을 유지합니다. XML 문서를 기존 관계형 테이블로 분해할 때 순서가 보존되는지는 보장할 수 없습니다.

테이블이 이미 데이터베이스에 있을 때 DAD 파일을 동시에 전달하려 하는 경우, XML 콜렉션을 사용 가능하도록 설정할 필요는 없습니다.

분해 테이블 크기 한계

분해는 요소 및 속성 값을 테이블 행으로 추출함으로써 XML 문서가 DB2 테이블에 분해되는 방법을 지정하기 위해 RDB_node 맵핑을 사용합니다. 각 XML 문서의 값은 하나 이상의 DB2 테이블에 저장됩니다. 각 테이블에는 각 문서에서 분해된 최대 1024행이 있을 수 있습니다.

예를 들어, XML 문서가 5개의 테이블로 분해된 경우, 5개의 테이블 각각에는 특정 문서에 대해 최대 1024행이 있을 수 있습니다. 이 테이블에 여러 문서의 행이 들어 있는 경우, 테이블에는 각 문서마다 최대 1024행이 있을 수 있습니다. 테이블에 20개의 문서가 들어 있는 경우, 테이블에는 각 문서에 대해 1024행씩 모두 20,480행이 있을 수 있습니다.

다중 발생 요소(XML 구조에서 두 번 이상 발생할 수 있는 위치 경로가 있는 요소)는 행 수에 영향을 줍니다. 예를 들어, 20번 발생하는 요소 <Part>가 들어 있는 문서는 한 테이블에서 20행으로 분해될 수 있습니다. 다중 발생 요소를 사용한다면 테이블 *크기* 제한사항을 고려하십시오.

시작하기 전에

- 요소 내용과 속성값이 들어있는 관계형 테이블로 XML 문서의 구조를 맵핑하십시오.
- RDB_node 맵핑을 사용하여 DAD 파일을 준비하십시오. 세부사항은 62 페이지의 『XML 콜렉션 플랜』을 참조하십시오.
- 선택적으로, XML 콜렉션을 사용할 수 있도록 설정하십시오.

XML 문서 분해

XML Extender는 두 개의 저장 프로시듀어 즉, dxxShredXML() 및 dxxInsertXML을 제공하여 XML 문서를 분해합니다.

dxxShredXML()

이 저장 프로시듀어는 때때로 갱신하는 응용프로그램 또는 XML 데이터 관리 오버헤드를 원치않는 응용프로그램에 사용됩니다. 저장 프로시듀어 dxxShredXML()에서는 사용 가능한 콜렉션이 필요없습니다. 대신 DAD 파일을 사용합니다.

저장 프로시듀어 dxxShredXML()은 두 개의 입력 매개변수 즉, DAD 파일과 분해할 XML 문서를 사용하며, 두 개의 출력 매개변수 즉 리턴 코드와 리턴 메시지를 리턴합니다.

저장 프로시듀어 dxxShredXML()은 입력 DAD 파일에 있는 <Xcollection> 스펙에 따라 XML 문서를 XML 콜렉션으로 삽입합니다. DAD 파일의 <Xcollection>에서 사용되는 테이블이 있는 것으로 가정하고, 컬럼은 DAD 맵핑에서 지정된 데이터 유형을 만족하는 것으로 가정합니다. 이 가정이 참이 아니면, 오류 메시지가 리턴됩니다. 저장 프로시듀어 dxxShredXML()은 XML 문서를 분해하고, 태그가 지정되지 않은 XML 데이터를 DAD 파일에서 지정된 테이블로 삽입합니다.

작성에 사용되는 저장 프로시듀어는 dxxGenXML()이며, 여기에서는 입력 매개변수로서 DAD를 사용하며, XML 콜렉션을 사용할 수 있도록 설정할 필요가 없습니다.

XML 콜렉션 분해하기: dxxShredXML()

다음 저장 프로시듀어 선언을 사용하여 응용프로그램에 저장 프로시듀어 호출을 삽입하십시오.

```
dxxShredXML(CLOB(100K)    DAD,            /* input */
            CLOB(1M)        xmlobj,           /* input */
              long            returnCode,      /* output */
              varchar(1024)   returnMsg)       /* output */
```

전체 구문과 예에 대해서는 250 페이지의 『dxxShredXML()』을 참조하십시오.

예: 다음은 dxxShredXML()에 대한 호출 예입니다.

```
#include "dxx.h"
#include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
         SQL TYPE is CLOB      dad;             /* DAD*/
         SQL TYPE is CLOB_FILE dadFile;         /* DAD file*/
         SQL TYPE is CLOB      xmlDoc;          /* input XML document */
         SQL TYPE is CLOB_FILE xmlFile;          /* input XML file */
         long                  returnCode;       /* error code */
         char                  returnMsg[1024];  /* error message text */
         short                 dad_ind;
         short                 xmlDoc_ind;
         short                 returnCode_ind;
         short                 returnMsg_ind;
          EXEC SQL END DECLARE SECTION;
        /* initialize host variable and indicators */
        strcpy(dadFile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
```

```
dadFile.name_length=strlen("c:\dxx\samples\dad\getstart_xcollection.dad");
dadFile.file_option=SQL_FILE_READ;
strcpy(xmlFile.name,"c:\dxx\samples\cmd\getstart_xcollection.xml");
xmlFile.name_length=strlen("c:\dxx\samples\cmd\getstart_xcollection.xml");
xmlFile.file_option=SQL_FILE_READ;
SQL EXEC VALUES (:dadFile) INTO :dad;
SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
returnCode = 0;
 returnMsg[0] = '\0';
dad_ind = 0;
xmlDoc_ind = 0;
returnCode_ind = -1;
returnMsg_ind = -1;
/* Call the store procedure */
EXEC SQL CALL db2xml.dxxShredXML(:dad:dad_ind;
              :xmlDoc:xmlDoc_ind,
              :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxInsertXML()

이 저장 프로시듀어는 정규 갱신을 수행하는 응용프로그램에 사용됩니다. 같은 타스크가 반복되므로, 성능 향상이 중요합니다. XML 콜렉션을 사용 가능으로 설정하고 저장 프로시듀어에 콜렉션 이름을 사용하면 성능이 향상됩니다. 저장 프로시듀어 dxxInsertXML()은 dxxShredXML()과 동일하게 동작하지만, 첫번째 입력 매개변수로서 사용 가능한 XML 콜렉션을 사용하는 것은 예외입니다.

저장 프로시듀어 dxxInsertXML()은 XML 문서를 사용 가능한 XML 콜렉션에 삽입하며, 이것은 DAD 파일과 연관됩니다. DAD 파일에는 콜렉션 테이블 및 맵핑에 대한 스펙이 들어 있습니다. 콜렉션 테이블은 <Xcollection>에 있는 스펙에 따라 체크되거나 작성됩니다. 그런 다음 저장 프로시듀어 dxxInsertXML()은 맵핑에 따라 XML 문서를 분해하고, 태그가 지정되지 않은 XML 데이터를 이름이 지정된 XML 콜렉션 테이블로 삽입합니다.

작성에 사용되는 저장 프로시듀어는 dxxRetrieveXML()이며, 사용 가능한 XML 콜렉션의 이름을 사용합니다.

XML 콜렉션 분해하기: dxxInsertXML()

다음 저장 프로시듀어 선언을 사용하여 응용프로그램에 저장 프로시듀어 호출을 삽입하십시오.

```
dxxInsertXML(char(collectionName) collectionName,   /* input */
             CLOB(1M)       xmlobj,           /* input */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

전체 구문과 예에 대해서는 252 페이지의 『dxxInsertXML()』을 참조하십시오.

예: 다음은 dxxInsertXML()에 대한 호출 예입니다.

```
#include "dxx.h"
#include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
         char                 collection[64];   /* name of an XML collection */
         SQL TYPE is CLOB_FILE xmlFile;         /* input XML file */
         SQL TYPE is CLOB      xmlDoc;          /* input XML doc */
         long                 returnCode;       /* error code */
         char                 returnMsg[1024];  /* error message text */
         short                collection_ind;
         short                xmlDoc_ind;
         short                returnCode_ind;
         short                returnMsg_ind;
          EXEC SQL END DECLARE SECTION;
        /* initialize host variable and indicators */
         strcpy(collection,"sales_ord")
         strcpy(xmlobj.name,"c:\dxx\samples\cmd\getstart_xcollection.xml");
         xmlobj.name_length=strlen("c:\dxx\samples\cmd\getstart_xcollection.xml");
         xmlobj.file_option=SQL_FILE_READ;
         SQL EXEC VALUES (:xmlFile) INTO (:xmlDoc);
        returnCode = 0;
         returnMsg[0] = '\0';
         collection_ind = 0;
         xmlobj_ind = 0;
        returnCode_ind = -1;
        returnMsg_ind = -1;
        /* Call the store procedure */
         EXEC SQL CALL db2xml.dxxInsertXML(:collection:collection_ind;
                     :xmlDoc:xmlDoc_ind,
                     :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

XML 콜렉션 액세스

XML 콜렉션을 갱신, 삭제, 탐색(search) 및 검색(retrieve)할 수 있습니다. 그러나 XML 콜렉션을 사용하는 목적은 데이터베이스 테이블에 태그가 지정되지 않은 순수한 데이터를 저장하거나 검색하기 위한 것임을 기억하십시오. 기존 데이터베이스 테이블에 있는 데이터는 가져오는 XML 문서에 대해 아무것도 수행하지 않습니다. 갱신, 삭제 및 검색 작업은 이러한 테이블에 대한 정상적인 SQL 액세스로 구성됩니다. 데이터가 가져오는 XML 문서에서 분해된 경우, 원래 XML 문서는 계속해서 존재하지 않습니다.

XML Extender는 XML 콜렉션 뷰에서 데이터에 대한 동작을 수행하는 기능을 제공합니다. UPDATE 및 DELETE SQL문을 사용하여, XML 문서를 작성하는 데 사용된 데이터를 수정할 수 있으므로 XML 콜렉션을 갱신할 수 있습니다.

고려사항:

- 문서를 갱신하려면, 테이블의 기본 키가 들어있는 행을 삭제하지 마십시오. 이것은 다른 콜렉션 테이블의 외부 키 행입니다. 기본 키와 외부 키 행이 삭제되면, 문서가 삭제됩니다.
- 요소와 속성값을 바꾸거나 삭제하기 위해, 문서를 삭제하지 않고 하위 테이블에서 행을 삭제한 다음 삽입할 수 있습니다.
- 문서를 삭제하려면, DAD에서 지정된 맨 위 element_node를 작성한 행을 삭제하십시오.

XML 콜렉션에서 데이터 갱신

XML Extender를 사용하면 XML 콜렉션 테이블에 저장된 태그가 지정되지 않은 데이터를 갱신할 수 있습니다. XML 콜렉션 테이블 값을 갱신함으로써, XML 요소의 텍스트나 XML 속성값을 갱신하는 것입니다. 뿐만 아니라, 갱신에서는 여러 번 발생하는 요소나 속성에서 데이터 인스턴스를 삭제할 수 있습니다.

SQL 측면에서 보면, 요소나 속성값을 변경하는 것은 갱신 동작이고, 요소나 속성의 인스턴스를 삭제하는 것은 삭제 동작입니다. XML 측면에서 보면 루트 element_node의 요소 텍스트나 속성값이 존재하는 동안 XML 문서도 계속 존재하므로, 이것이 갱신 동작입니다.

요구사항: XML 콜렉션에서 데이터를 갱신하려면 다음 규칙을 준수하십시오.

- 기존 테이블에 이 관계가 있으면 콜렉션 테이블 사이에서 기본-외부 키 관계를 지정하십시오. 지정하지 않으면, 조인될 수 있는 컬럼이 있는지 확인하십시오.
- DAD 파일에서 지정된 조인 조건을 삽입하십시오.
  - SQL 맵핑의 경우 <SQL_stmt> 요소에서
  - RDB_node 맵핑의 경우, 루트 요소 노드의 RDB_node에서

요소 및 속성값 갱신

XML 콜렉션에서, 요소 텍스트와 속성값은 모두 데이터베이스 테이블에 있는 컬럼으로 맵핑됩니다. 이전에 컬럼 데이터가 있었는지 또는 가져오는 XML 문서에서 분해된 것인지에 관계없이, 일반 SQL 갱신 기법을 사용하여 데이터를 바꿉니다.

요소나 속성값을 갱신하려면, DAD 파일에서 지정된 조인 조건이 들어있는 SQL UPDATE문에서 WHERE 절을 지정하십시오.

예를 들면 다음과 같습니다.

```
UPDATE SHIP_TAB
    set MODE = 'BOAT'
  WHERE MODE='AIR' AND PART_KEY in
    (SELECT PART_KEY from PART_TAB WHERE ORDER_KEY=68)
```

<ShipMode> 요소값은 SHIP_TAB 테이블에서 AIR에서 BOAT로 갱신되며, 여기서 키는 68입니다.

요소 및 속성 인스턴스 삭제

여러 번 발생하는 요소나 속성들을 제거하여 작성된 XML 문서를 갱신하기 위해, Where 절을 사용하여 요소나 속성값에 해당하는 필드값이 들어있는 행을 삭제하십시오. 맨 위 element_node 값이 들어있는 행을 삭제하지 않는 동안, 요소값을 삭제하는 것은 XML 문서를 갱신하는 것으로 간주됩니다.

예를 들어 다음 DELETE문에서, 부속요소중 하나의 고유한 값을 지정하여 <shipment> 요소를 삭제합니다.

```
DELETE from SHIP_TAB
   WHERE DATE='1999-04-12'
```

DATE 값을 지정하면 이 값과 일치하는 행이 삭제됩니다. 작성된 문서에는 원래 두 개의 <shipment> 요소가 들어있었지만, 이제는 하나만 들어 있게 됩니다.

XML 콜렉션에서 XML 문서 삭제

콜렉션에서 작성된 XML 문서를 삭제할 수 있습니다. 이것은 여러 XML 문서를 작성한 XML 콜렉션이 있는 경우 이들 작성된 문서 중 하나를 삭제할 수 있음을 의미합니다.

문서를 삭제하기 위해, DAD 파일에서 지정된 맨 위 element_node를 작성한 행을 테이블에서 삭제합니다. 이 테이블에는 맨 위 콜렉션 테이블의 기본 키와, 하위 테이블의 외부 키가 포함됩니다.

예를 들어, 다음 DELETE문에서는 기본 키 컬럼 값을 지정합니다.

```
DELETE from order_tab
   WHERE order_key=1
```

ORDER_KEY는 테이블 ORDER_TAB에 있는 기본 키이고, XML 문서가 작성될 때는 맨 위 element_node입니다. 이 행을 삭제하는 것은 작성하는 동안 생성된 하나의 XML 문서를 삭제하는 것입니다. 그러므로 XML 측면에서 보면, 하나의 XML 문서가 XML 콜렉션에서 삭제됩니다.

XML 콜렉션에서 XML 문서 검색

XML 콜렉션에서 XML 문서를 검색하는 것은 콜렉션에서 문서를 작성하는 것과 비슷합니다.

XML 문서를 검색하려면 저장 프로시듀어 dxxRetrieveXML()을 사용하십시오. 구문 및 예에 대해서는 246 페이지의 『dxxRetrieveXML()』을 참조하십시오.

DAD 파일 고려사항: XML 문서를 XML 콜렉션으로 분해할 때, DAD 파일에서 순서를 지정하지 않으면 여러 번 발생하는 요소와 속성값의 순서를 잃게 될 수도 있습니다. 이 순서를 유지하기 위해서는 RDB_node 맵핑 스킴을 사용해야 합니다. 이 맵핑 스킴을 사용하면 RDB_node에서 루트 요소가 들어있는 테이블에 대해 orderBy 속성을 지정할 수 있습니다.

XML 콜렉션 검색

이 절에서는 다음 목적으로 XML 콜렉션을 검색하는 작업에 대해 설명합니다.

- **검색 기준을 사용한 XML 문서 생성**

  이 타스크는 실제로 조건을 사용한 작성입니다. 다음 검색 기준을 사용하여 검색 기준을 지정할 수 있습니다.

  – DAD 파일의 text_node 및 attribute_node에서 조건 지정

  – dxxGenXML() 및 dxxRetrieveXML() 저장 프로시듀어를 사용할 경우 *overwrite* 매개변수 지정

  예를 들어 DAD 파일 `order.dad`를 사용하여 XML 콜렉션 sales_ord를 사용할 수 있게 되었지만 이제는 웹에서 추출한 양식 데이터를 사용하여 가격을 겹쳐쓰기하려는 경우, 다음과 같이 <SQL_stmt> DAD 요소값을 겹쳐쓰기할 수 있습니다.

```
EXEC SQL INCLUDE SQLCA;
 EXEC SQL BEGIN DECLARE SECTION;
...

   EXEC SQL END DECLARE SECTION;
  float    price_value;
  /* create table */
  EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);
 /* initialize host variable and indicators */
  strcpy(collection,"sales_ord");
  strcpy(result_tab,"xml_order_tab");
  overrideType = SQL_OVERRIDE;
  max_row = 20;
  num_row = 0;
 returnCode = 0;
  msg_txt[0] = '\0';
  override_ind = 0;
  overrideType_ind = 0;
  rtab_ind = 0;
  maxrow_ind = 0;
  numrow_ind = -1;
 returnCode_ind = -1;
 returnMsg_ind = -1;
  /* get the price_value from some place, such as form data */
  price_value = 1000.00         /* for example*/
  /* specify the overwrite */
  sprintf(overwrite,
         "SELECT o.order_key, customer, p.part_key, quantity, price,
                tax, ship_id, date, mode
         FROM order_tab o, part_tab p,
table (select substr(char(timestamp(generate_unique())),16)
             as ship_id, date, mode from ship_tab)as s
         WHERE p.price > %d and s.date >'1996-06-01' AND
             p.order_key = o.order_key and s.part_key = p.part_key",
             price_value);
  /* Call the store procedure */
  EXEC SQL CALL db2xml.dxxRetrieve(:collection:collection_ind,
         :result_tab:rtab_ind,
         :overrideType:overrideType_ind,:overwrite:overwrite_ind,
         :max_row:maxrow_ind,:num_row:numrow_ind,
               :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

order.dad에 있는 price > 2500.00 조건은 > ?로 겹쳐쓰기됩니다. 여기서 ?는 입력 변수 *price_value*를 기초로 합니다.

- **분해된 XML 데이터 검색:**

  일반 SQL 조회 조작을 수행하여 콜렉션 테이블을 검색할 수 있습니다. 콜렉션 테이블을 조인하거나, 부속조회를 사용한 다음, 텍스트 컬럼에 대해 구조적인 텍스트 검색을 수행할 수 있습니다. 구조 검색 결과를 사용하면, 그 결과를 적용하여 지정된 XML 문서를 검색하거나 생성할 수 있습니다.

제4부 참조

이 부분에서는 XML Extender 관리 명령, 사용자 정의 데이터 유형(UDT), 사용자 정의 기능(UDF) 및 저장 프로시듀어에 대한 구문 정보를 제공합니다. 문제점 판별 활동에 대한 메시지 텍스트도 제공됩니다.

제7장 XML Extender 관리 명령: *dxxadm*

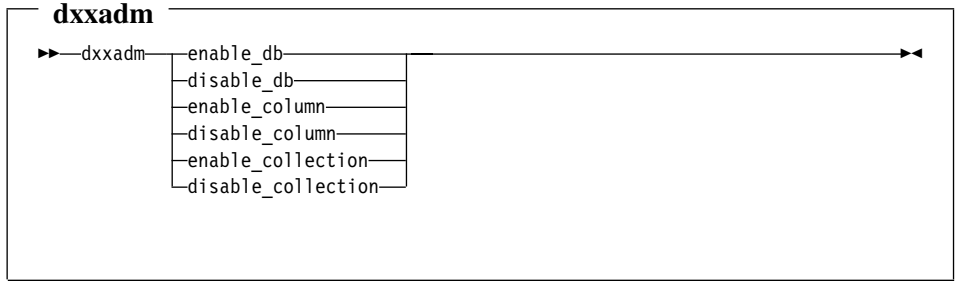
XML Extender는 다음과 같은 관리 타스크를 완료하기 위해 관리 명령 **dxxadm**을 제공합니다. 다음의 XML Extender 관리 타스크는 다양한 명령 옵션을 사용하여 **dxxadm**을 호출함으로써 수행할 수 있습니다.

- XML Extender에 대해 데이터베이스 사용 또는 사용 안함
- XML 컬럼 사용 또는 사용 안함
- XML 콜렉션 사용 또는 사용 안함

XML Extender 관리 마법사나 저장 프로시듀어를 사용하여 각 관리 타스크를 수행할 수도 있습니다.

고급 구문

다음 구문 도표에서는 **dxxadm** 명령의 고급 구문을 제공합니다. 각 옵션에 대한 설명은 다음 절에서 제공됩니다.



관리 명령 옵션

다음 절에서는 시스템 프로그래머가 사용할 수 있는 각 **dxxadm** 명령 옵션에 대해 설명합니다.

- 175 페이지의『enable_db』
- 177 페이지의『disable_db』
- 179 페이지의『enable_column』
- 181 페이지의『disable_column』
- 183 페이지의『enable_collection』
- 185 페이지의『disable_collection』

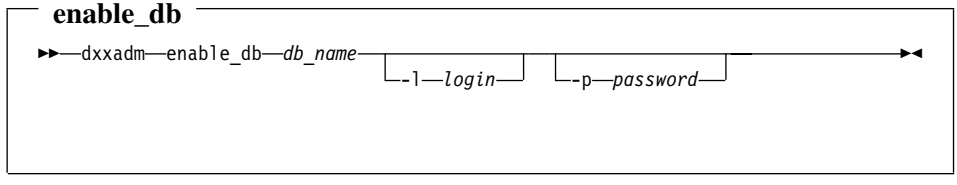
enable_db

목적

XML Extender가 데이터베이스를 사용할 수 있도록 데이터베이스에 연결하고 이를 사용 가능한 것으로 설정합니다. 데이터베이스를 사용할 수 있게 되면, XML Extender는 다음 오브젝트를 작성합니다.

- XML Extender 사용자 정의 유형(UDT).
- XML Extender 사용자 정의 기능(UDF).
- XML Extender DTD 참조 테이블 DTD_REF, 여기에는 DTD 및 각 DTD 정보를 저장합니다. DTD_REF 테이블에 대한 완전한 설명은 255 페이지의 『DTD 참조 테이블』를 참조하십시오.
- XML Extender 이용 테이블 XML_USAGE, 여기에는 XML 및 각 콜렉션에 대해 사용할 수 있는 각 컬럼의 일반 정보가 저장됩니다. XML_USAGE 테이블에 대한 완전한 설명은 256 페이지의 『XML 사용 테이블』을 참조하십시오.

형식



매개변수

표 14. enable_db 매개변수

매개변수	설명
db_name	XML 데이터가 있는 데이터베이스 이름.
-l *login*	사용자 ID, 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 사용자 ID가 사용됩니다.
-p *password*	암호, 이것은 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 암호가 사용됩니다.

예

다음 예에서는 데이터베이스 SALES_DB를 사용할 수 있도록 설정합니다.

```
dxxadm enable_db SALES_DB
```

disable_db

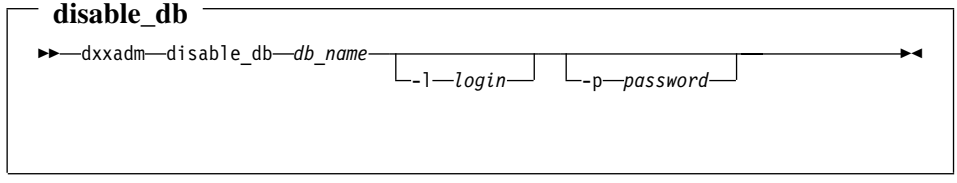
목적

XML 가능 데이터베이스에 연결하여 사용 안함으로 설정합니다. 데이터베이스를 사용할 수 없게 되면, 더 이상 XML Extender가 사용할 수 없게 됩니다. XML Extender가 데이터베이스를 사용할 수 없게 되면, 다음 오브젝트를 제거합니다.

- XML Extender 사용자 정의 유형(UDT).
- XML Extender 사용자 정의 기능(UDF).
- XML Extender DTD 참조 테이블 DTD_REF, 여기에는 DTD 및 각 DTD 정보를 저장합니다. DTD_REF 테이블에 대한 완전한 설명은 255 페이지의 『DTD 참조 테이블』를 참조하십시오.
- XML Extender 이용 테이블 XML_USAGE, 여기에는 XML 및 각 콜렉션에 대해 사용할 수 있는 각 컬럼의 일반 정보가 저장됩니다. XML_USAGE 테이블에 대한 완전한 설명은 256 페이지의 『XML 사용 테이블』을 참조하십시오.

중요사항: 데이터베이스를 사용 불능으로 설정하기 전에 모든 XML 컬럼들을 사용 불능으로 설정해야 합니다. XML Extender는 XML에 사용 가능한 컬럼이나 콜렉션이 있는 데이터베이스는 사용 불가능하게 할 수 없습니다.

형식



매개변수

표 15. disable_db 매개변수

매개변수	설명
db_name	XML 데이터가 있는 데이터베이스 이름.
-l *login*	사용자 ID, 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 사용자 ID가 사용됩니다.

표 15. disable_db 매개변수 (계속)

매개변수	설명
-p *password*	암호, 이것은 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 암호가 사용됩니다.

예

다음 예에서는 데이터베이스 SALES_DB를 사용할 수 없도록 설정합니다.

```
dxxadm disable_db SALES_DB
```

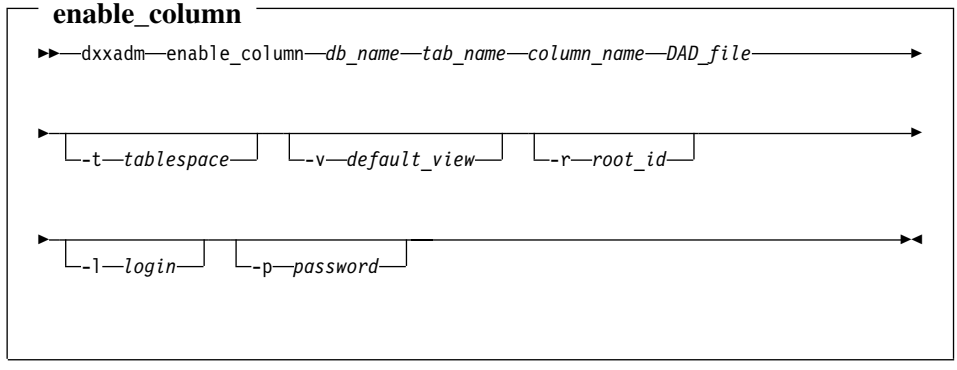
enable_column

목적

데이터베이스에 연결하고, XML Extender UDT가 들어갈 수 있도록 XML을 사용 가능으로 설정하십시오. 컬럼을 사용할 수 있게 되면, XML Extender는 다음 타스크를 완료합니다.

- XML 테이블에 기본 키가 있는지 판별합니다. 기본 키가 없으면, XML Extender는 XML 테이블을 변경하고 DXXROOT_ID 컬럼을 추가합니다.
- XML 테이블에서 각 행에 대한 고유한 식별자가 들어있는 컬럼으로 DAD 파일에서 지정된 부가 테이블을 작성합니다. 이 컬럼은 사용자가 지정한 *root_id* 이거나, XML Extender가 명명한 DCCROOT_ID입니다.
- 선택적으로 사용자가 지정한 이름을 사용하여 XML 테이블 및 부가 테이블에 대한 기본 뷰를 작성합니다.

형식



매개변수

표 16. enable_column 매개변수

매개변수	설명
db_name	XML 데이터가 있는 데이터베이스 이름.
tab_name	XML 컬럼이 있는 테이블 이름.
column_name	XML 컬럼 이름.

표 16. enable_column 매개변수 (계속)

매개변수	설명
DAD_file	XML 문서를 XML 컬럼과 부가 테이블로 맵핑하는 DAD 파일 이름.
-t *tablespace*	테이블 공간, 선택적이며 XML 컬럼과 연관된 부가 테이블이 들어 있습니다. 지정되지 않았으면 기본 테이블 공간이 사용됩니다.
-v *default_view*	기본 보기 이름, 선택적이며 XML 컬럼 및 부가 테이블을 조인합니다.
-r *root_id*	부가 테이블에 대한 root_id로서 사용될, XML 컬럼에 있는 기본 키 이름. root_id는 선택적입니다.
-l *login*	사용자 ID, 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 사용자 ID가 사용됩니다.
-p *password*	암호, 이것은 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 암호가 사용됩니다.

예

다음 예에서는 XML 컬럼을 사용할 수 있도록 설정합니다.

```
dxxadm enable_column SALES_DB SALES_TAB ORDER -v sales_order_view -r INVOICE_NUMBER
```

disable_column

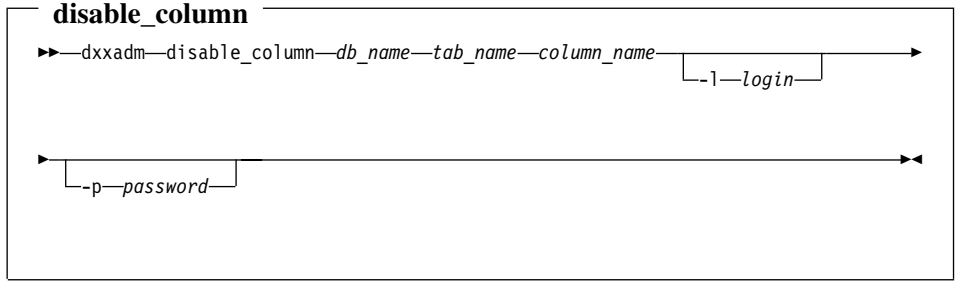
목적

데이터베이스에 연결하여 XML 가능 컬럼을 사용할 수 없도록 설정합니다. 컬럼을 사용할 수 없게 되면, 더 이상 XML 데이터 유형이 포함될 수 없습니다. XML 가능 컬럼을 사용할 수 없게 되면, 다음 조치가 수행됩니다.

- XML 컬럼 이용 항목이 XML_USAGE 테이블에서 삭제됩니다.
- USAGE_COUNT가 DTD_REF 테이블에서 줄어듭니다.
- 이 컬럼과 연관된 모든 트리거가 삭제됩니다.
- 이 컬럼과 연관된 모든 부가 테이블이 삭제됩니다.

중요사항: XML 테이블을 삭제하기 전에 XML 컬럼을 사용 불능으로 설정해야 합니다. XML 테이블이 삭제되었지만 이것의 XML 컬럼을 계속해서 사용할 수 있으면, XML Extender는 작성했던 부가 테이블과 XML 컬럼 항목 모두를 XML_USAGE 테이블에서 보존합니다.

형식



매개변수

표 17. disable_column 매개변수

매개변수	설명
db_name	데이터가 있는 데이터베이스 이름.
tab_name	XML 컬럼이 있는 테이블 이름.
column_name	XML 컬럼 이름.

표 17. disable_column 매개변수 (계속)

매개변수	설명
-l *login*	사용자 ID, 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 사용자 ID가 사용됩니다.
-p *password*	암호, 이것은 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 암호가 사용됩니다.

예

다음 예에서는 XML 가능 컬럼을 사용 불능으로 설정합니다.

```
dxxadm disable_column SALES_DB SALES_TAB ORDER
```

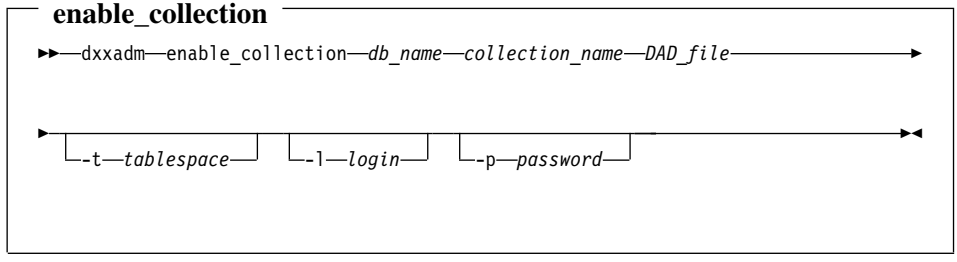
enable_collection

목적

데이터베이스에 연결하고, 지정된 DAD에 따라 XML 콜렉션을 사용 가능하게 설정합니다. 콜렉션을 사용할 수 있게 되면, XML Extender는 다음 타스크를 수행합니다.

- XML_USAGE 테이블에서 XML 콜렉션 이용 항목을 작성합니다.
- RDB_node 맵핑의 경우, 테이블이 데이터베이스에 없는 경우 DAD에서 지정된 콜렉션 테이블을 작성합니다.

형식



매개변수

표 18. enable_collection 매개변수

매개변수	설명
db_name	데이터가 있는 데이터베이스 이름.
collection_name	XML 콜렉션 이름.
DAD_file	콜렉션에 관계형 테이블로 XML 문서를 맵핑하는 DAD 파일 이름.
-t *tablespace*	테이블 공간 이름, 선택적이며 콜렉션과 연관됩니다. 지정되지 않았으면 기본 테이블 공간이 사용됩니다.
-l *login*	사용자 ID, 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 사용자 ID가 사용됩니다.

표 18. enable_collection 매개변수 (계속)

매개변수	설명
-p *password*	암호, 이것은 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 암호가 사용됩니다.

예

다음 예에서는 XML 콜렉션을 사용할 수 있도록 설정합니다.

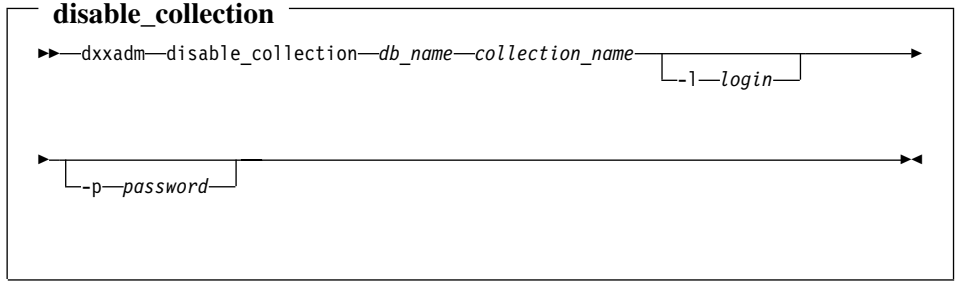
```
dxxadm enable_collection SALES_DB sales_ord getstart_xcollection.dad -t orderspace
```

disable_collection

목적

데이터베이스에 연결하여 XML 가능 콜렉션을 사용할 수 없도록 설정합니다. 콜렉션 이름이 작성(dxxRetrieveXML) 및 분해(dxxInsertXML) 저장 프로시듀어에 사용될 수 없습니다. XML 콜렉션을 사용할 수 없게 되면, 연관된 콜렉션 항목이 XML_USAGE 테이블에 삭제됩니다. 콜렉션을 사용 불능으로 설정하는 과정에서는 enable_collection 단계에서 작성했던 콜렉션 제거를 삭제하지 않는다는 점에 유의하십시오.

형식



매개변수

표 19. disable_collection 매개변수

매개변수	설명
db_name	데이터가 있는 데이터베이스 이름.
collection_name	XML 콜렉션 이름.
-l *login*	사용자 ID, 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 사용자 ID가 사용됩니다.
-p *password*	암호, 이것은 선택적이며, 지정된 경우는 데이터베이스로 연결될 때 사용됩니다. 지정되지 않으면 현재 암호가 사용됩니다.

예

다음 예에서는 XML 콜렉션을 사용할 수 없도록 설정합니다.

```
dxxadm disable_collection SALES_DB sales_ord
```

제8장 XML Extender 사용자 정의 유형

XML Extender 사용자 정의 유형(UDT)은 XML 컬럼 및 XML 콜렉션에 사용되는 데이터 유형입니다. 모든 UDT의 스키마 이름은 db2xml입니다. XML Extender는 XML 문서를 저장하고 검색하기 위해 UDT를 작성합니다. 표20에는 UDT 개요가 들어 있습니다.

표 20. XML Extender UDT

사용자 정의 유형 컬럼	소스 데이터 유형	사용법 설명
XMLVARCHAR	VARCHAR(*varchar_len*)	DB2에서 전체 XML 문서를 VARCHAR로서 저장합니다.
XMLCLOB	CLOB(*clob_len*)	DB2에서 전체 XML 문서를 문자 대형 오브젝트(CLOB)로서 저장합니다.
XMLFILE	VARCHAR(512)	지역 파일 서버의 파일 이름을 지정합니다. XML 컬럼에 대해 XMLFILE이 지정되면, XML Extender는 외부 서버 파일에서 XML 문서를 저장합니다. Text Extender는 XMLFILE과 함께 사용될 수 없습니다. 색인화를 위해 작성된 부가 테이블 뿐만 아니라, 파일 내용과 DB2 간의 무결성을 유지하는 것은 사용자 책임입니다.

여기서 *varchar_len* 및 *clob_len*은 운영 체제마다 다릅니다.

DB2 UDB의 경우, *varchar_len* = 3K 및 *clob_len* = 2G.

이러한 UDT는 응용프로그램 컬럼 유형을 지정하는 데에만 사용되며, XML Extender가 작성한 부가 테이블에는 적용되지 않습니다.

제9장 XML Extender 사용자 정의 기능

XML Extender는 XML 문서를 저장, 검색(retrieve), 탐색(search) 및 갱신하고, XML 요소나 속성을 추출하는 기능을 제공합니다. XML 컬럼에 대해서는 XML 사용자 정의 기능(UDF)을 사용하지만, XML 콜렉션에 대해서는 사용하지 마십시오. 모든 UDF의 스키마 이름은 db2xml이고, 이것은 UDF 앞에서 생략될 수 있습니다.

XML Extender 기능의 4 가지 유형은 저장영역 기능, 검색 기능, 추출 기능 및 갱신 기능입니다.

저장영역 기능

저장영역 기능은 XML 문서를 DB2 데이터베이스에 삽입합니다. 구문 및 예에 대해서는 190 페이지의 『저장영역 기능』을 참조하십시오.

검색 기능

검색 기능은 DB2 데이터베이스에 있는 XML 컬럼에서 XML 문서를 검색합니다. 구문 및 예에 대해서는 197 페이지의 『검색 기능』을 참조하십시오.

추출 기능

추출 기능은 XML 문서에서 요소 내용이나 속성값을 추출하고, 기능 이름에서 지정된 데이터 유형으로 변환시킵니다. XML Extender는 여러 SQL 데이터 유형에 대해 추출 기능 세트를 제공합니다. 구문 및 예에 대해서는 204 페이지의 『추출 기능』을 참조하십시오.

갱신 함수

Update() 함수는 요소 내용이나 속성값을 수정하고, 위치 경로에서 지정된 갱신된 값과 함께 XML 문서 사본을 리턴합니다. Update() 함수를 사용하면 응용프로그램 프로그래머가 갱신될 요소나 속성을 지정할 수 있습니다. 구문 및 예에 대해서는 225 페이지의 『갱신 함수』를 참조하십시오.

표21에서는 XML Extender 기능 요약을 제공합니다.

표 21. XML Extender 사용자 정의 기능

유형	함수
저장영역 기능	XMLVarcharFromFile()
	XMLCLOBFromFile()
	XMLFileFromVarchar()
	XMLFileFromCLOB()
검색 기능	Content(): XMLFile에서 CLOB로 검색
	Content(): XMLVarchar에서 외부 서버 파일로 검색
	Content(): XMLCLOB에서 외부 서버로 검색
추출 기능	extractInteger() 및 extractIntegers()
	extractSmallint() 및 extractSmallints()
	extractDouble() 및 extractDoubles()
	extractReal() 및 extractReals()
	extractChar() 및 extractChars()
	extractVarchar() 및 extractVarchars()
	extractCLOB() 및 extractCLOBs()
	extractDate() 및 extractDates()
	extractTime() 및 extractTimes()
	extractTimestamp() 및 extractTimestamps()
갱신 함수	Update()

UDF에서 매개변수 표시문자를 사용하는 경우, JDBC 제한사항은 UDF에 대한 매개변수 표시문자가 리턴되는 데이터가 삽입될 컬럼의 데이터 유형으로 변환되어야 함을 요구합니다. 매개변수 표시문자를 유형 변환하는 방법은 151 페이지의 『JDBC에서 함수 호출시 제한사항』을 참조하십시오.

저장영역 기능

저장영역 기능을 사용하여 XML 문서를 DB2 데이터베이스에 삽입합니다. 134 페이지의 『데이터 저장』에서 설명한 바와 같이 INSERT 또는 SELECT 문에서 직접 UDT의 기본 유형 변환 기능을 사용할 수 있습니다. 또한, XML Extender는 UDT 기본 데이터 유형 이외의 소스로부터 XML 문서를 액세스하고, 이를 원하는 UDT로 변환하기 위한 UDF를 제공합니다.

XML Extender는 다음 저장영역 기능을 제공합니다.

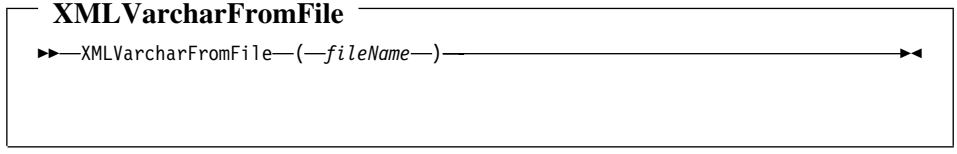
- 192 페이지의『XMLVarcharFromFile()』
- 193 페이지의『XMLCLOBFromFile()』
- 194 페이지의『XMLFileFromVarchar()』
- 196 페이지의『XMLFileFromCLOB()』

XMLVarcharFromFile()

목적

서버 파일에서 XML 문서를 읽고 이 문서를 XMLVARCHAR 유형으로 리턴합니다.

구문



매개변수

표 22. XMLVarcharFromFile 매개변수

매개변수	데이터 유형	설명
fileName	VARCHAR(512)	완전한 서버 파일 이름.

리턴 유형

XMLVARCHAR

예

다음 예는 서버 파일에서 XML 문서를 읽고 이를 XMLVARCHAR 유형으로 XML 컬럼에 삽입합니다.

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLVarcharFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

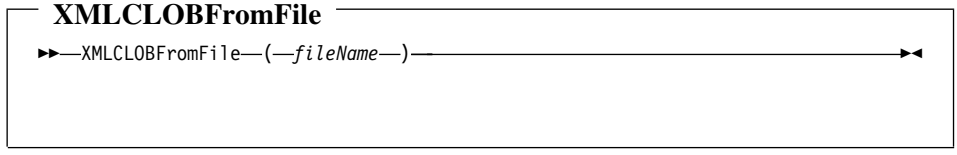
이 예에서, 레코드는 SALES_TAB 테이블에 삽입됩니다. XMLVarcharFromFile() 함수는 파일에서 DB2로 XML 문서를 가져오기하고, 이를 XMLVARCHAR로서 저장합니다.

XMLCLOBFromFile()

목적

서버 파일에서 XML 문서를 읽고 이 문서를 XMLCLOB 유형으로 리턴합니다.

구문



매개변수

표 23. XMLCLOBFromFile 매개변수

매개변수	데이터 유형	설명
fileName	VARCHAR(512)	완전한 서버 파일 이름.

리턴 유형

LOCATOR로서의 XMLCLOB

예

다음 예는 서버 파일에서 XML 문서를 읽고 이를 XMLCLOB 유형으로 XML 컬럼에 삽입합니다.

```
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLCLOBFromFile('c:\dxx\samples\cmd\getstart.xml'))
```

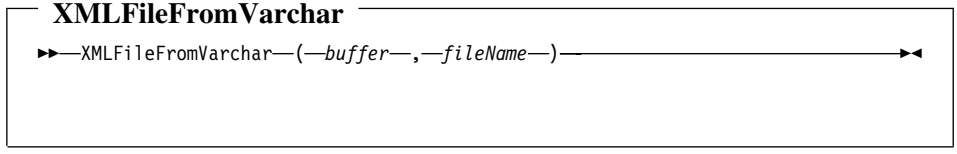
SALES_TAB 테이블에 있는 컬럼 OREDER는 XMLCLOB 유형으로 정의됩니다. 위 예에서는 컬럼 ORDER가 SALES_TAB 테이블에 삽입되는 방법을 보여줍니다.

XMLFileFromVarchar()

목적

메모리에서 XML 문서를 VARCHAR로 읽고 이를 외부 서버 파일에 기록한 후, 그 파일 이름과 경로를 XMLFILE 유형으로 리턴합니다.

구문



매개변수

표 24. XMLFileFromVarchar 매개변수

매개변수	데이터 유형	설명
buffer	VARCHAR(3K)	메모리 버퍼.
fileName	VARCHAR(512)	완전한 서버 파일 이름.

리턴 유형

XMLFILE

예

다음 예에서는 메모리에서 XML 문서를 VARCHAR로 읽고, 이를 외부 서버 파일로 기록하며, 파일 이름과 경로를 XML 컬럼에서 XMLFILE 유형으로 삽입합니다.

```
        EXEC SQL BEGIN DECLARE SECTION;
        struct {        short len; char data[3000]; } xml_buff;
          EXEC SQL END DECLARE SECTION;
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
               XMLFileFromVarchar(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml'))
```

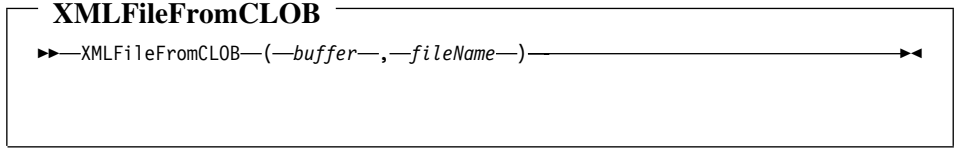
SALES_TAB 테이블에 있는 컬럼 OREDER는 XMLFILE 유형으로 정의됩니다. 위 예에서는 버퍼에 XML 문서가 있는 경우 이를 서버 파일에 저장할 수 있음을 보여줍니다.

XMLFileFromCLOB()

목적

XML 문서를 CLOB 위치 지정자로 읽고, 이를 외부 서버 파일에 기록한 후, 그 파일 이름과 경로를 XMLFILE 유형으로 리턴합니다.

구문



매개변수

표 25. XMLFileFromCLOB() 매개변수

매개변수	데이터 유형	설명
buffer	LOCATOR로서의 CLOB	XML 문서가 들어있는 버퍼.
fileName	VARCHAR(512)	완전한 서버 파일 이름.

리턴 유형

XMLFILE

예

다음 예는 XML 문서를 CLOB 위치 지정자로 읽고, 이를 외부 서버 파일에 기록한 후, 그 파일 이름과 경로를 XMLFILE 유형으로 XML 컬럼에 삽입합니다.

```
        EXEC SQL BEGIN DECLARE SECTION;
              SQL TYPE IS CLOB_LOCATOR xml_buff;
          EXEC SQL END DECLARE SECTION;
EXEC SQL INSERT INTO sales_tab(ID, NAME, ORDER)
   VALUES('1234', 'Sriram Srinivasan',
           XMLFileFromCLOB(:xml_buf, 'c:\dxx\samples\cmd\getstart.xml'))
```

SALES_TAB 테이블에 있는 컬럼 OREDER는 XMLFILE 유형으로 정의됩니다. 버퍼에 XML 문서가 있는 경우, 이를 서버 파일에 저장할 수 있습니다.

검색 기능

137 페이지의『전체 문서 검색』에서 설명한 바와 같이 기본 유형 변환 함수를 사용하면 XML UDT를 기본 데이터 유형으로 변환할 수 있습니다. XML Extender는 또한 *오버로드된 함수* Content()를 제공하며, 이것은 검색에 사용됩니다.

Content() 함수는 다음 유형의 검색을 제공합니다.

- **서버의 외부 저장영역으로부터 클라이언트의 호스트 변수로 검색.**

  XML 문서가 외부 서버 파일로서 저장될 때 Content()를 사용하여 메모리 버퍼로 검색할 수 있습니다. 이를 위해 198 페이지의『Content(): XMLFILE에서 CLOB로 검색』을 사용할 수 있습니다.

- **내부 저장영역에서 외부 서버 파일로 검색**

  또한 Content()를 사용하여 DB2내에 저장된 XML 문서를 검색하고, 이를 DB2 서버의 파일 시스템에 있는 서버 파일로 저장할 수 있습니다. 다음 Content() 함수는 외부 서버 파일에 정보를 저장할 때 사용됩니다.

  - 200 페이지의『Content(): XMLVARCHAR에서 외부 서버 파일로 검색』
  - 202 페이지의『Content(): XMLCLOB에서 외부 서버 파일로 검색』

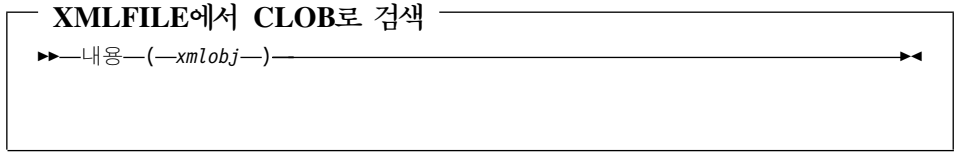
다음 절에 있는 예에서는 사용자가 DB2 명령 쉘을 사용하고 있는 것으로 가정하며, 여기에서는 각 명령 시작에『DB2』를 입력할 필요가 없습니다.

Content(): XMLFILE에서 CLOB로 검색

목적

서버 파일에서 데이터를 검색하고 이를 CLOB LOCATOR에 저장합니다.

구문



매개변수

표 26. XMLFILE에서 CLOB로 검색 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLFILE	XML 문서.

리턴 유형

LACATOR로서의 CLOB (*clob_len*)

DB2 UDB에 대한 *clob_len*은 2G입니다.

예

다음 예는 서버 파일에서 데이터를 검색하고, 이를 CLOB 위치 지정자에 저장합니다.

```
        EXEC SQL BEGIN DECLARE SECTION;
             SQL TYPE IS CLOB_LOCATOR xml_buff;
          EXEC SQL END DECLARE SECTION;
          EXEC SQL CONNECT TO SALES_DB
EXEC SQL DECLARE c1 CURSOR FOR
              SELECT Content(order) from sales_tab
   WHERE sales_person = 'Sriram Srinivasan'
                        EXEC SQL OPEN c1;
          do {
             EXEC SQL FETCH c1 INTO :xml_buff;
             if (SQLCODE != 0) {
                break;
      }
```

```
            else {
      /* do with the XML doc in buffer */
      }
   }
          EXEC SQL CLOSE c1;
          EXEC SQL CONNECT RESET;
```

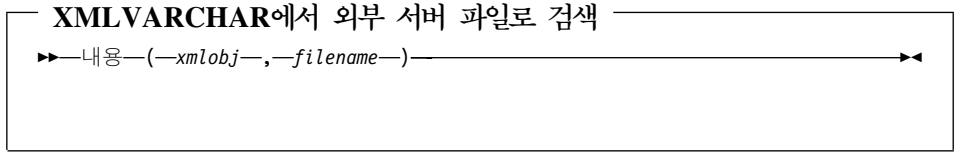
SALES_TAB 테이블에 있는 컬럼 ORDER의 유형은 XMLFILE이므로, Content() UDF는 서버 파일에서 데이터를 검색하고, 이를 CLOB 위치 지정자에 저장합니다.

Content(): XMLVARCHAR에서 외부 서버 파일로 검색

목적

XMLVARCHAR 유형으로서 저장된 XML 내용을 검색하고 외부 서버 파일에 이를 저장합니다.

구문



중요사항: 지정된 이름을 가진 파일이 이미 있으면, Content 함수는 그 내용을 겹쳐쓰기 합니다.

주:

매개변수

표 27. XMLVarchar에서 외부 서버 파일로 검색 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR	XML 문서.
filename	VARCHAR(512)	완전한 서버 파일 이름.

리턴 유형

VARCHAR(512)

예

다음 예는 XMLVARCHAR 유형으로서 저장된 XML 내용을 검색하고 이를 외부 서버 파일에 저장합니다.

```
CREATE table app1 (id int NOT NULL, order db2xml.XMLVarchar);
INSERT into app1 values (1, '<?xml version="1.0"?>
  <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
    <Order key="1">
                                                <Customer> 식별
    <Name>American Motors</Name>
```

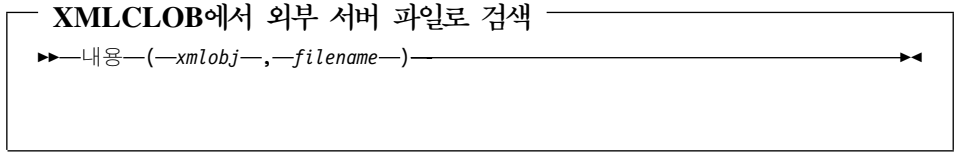
```
    <Email>parts@am.com</Email>
      </Customer>
      <Part color="black">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
    </Order>');
SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart_.dad')
  from app1 where ID=1;
```

Content(): XMLCLOB에서 외부 서버 파일로 검색

목적

XMLCLOB 유형으로서 저장된 XML 내용을 검색하고 외부 서버 파일에 이를 저장합니다.

구문



중요사항: 지정된 이름을 가진 파일이 이미 있으면, Content 함수는 그 내용을 겹쳐쓰기 합니다.

매개변수

표 28. XMLCLOB에서 외부 서버 파일로 검색 매개변수

매개변수	데이터 유형	설명
xmlobj	LOCATOR로서의 XMLCLOB	XML 문서.
filename	VARCHAR(512)	완전한 서버 파일 이름.

리턴 유형

VARCHAR(512)

예

다음 예는 XMLCLOB 유형으로서 저장된 XML 내용을 검색하고 이를 외부 서버 파일에 저장합니다.

```
CREATE table app1 (id int NOT NULL, order db2xml.XMLCLOB not logged);
INSERT into app1 values (1, '<?xml version="1.0"?>
  <!DOCTYPE SYSTEM c:\dxx\samples\dtd\getstart.dtd"->
    <Order key="1">
                                                  <Customer> 식별
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
      </Customer>
```

```
      <Part color="black">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
    </Shipment>
    <Shipment>
       <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
    </Shipment>
    </Order>');
SELECT db2xml.Content(order, 'c:\dxx\samples\cmd\getstart.xml')
        from app1 where ID=1;
```

추출 기능

추출 기능은 XML 문서에서 요소 내용이나 속성값을 추출하고, 요청된 SQL 데이터 유형을 리턴합니다. XML Extender는 여러 SQL 데이터 유형에 대해 추출 기능 세트를 제공합니다. 추출 기능은 두 개의 입력 매개변수를 사용합니다. 첫번째 매개변수는 XML Extender UDT이고, XML UDT 중 하나가 될 수 있습니다. 두번째 매개변수는 XML 요소나 속성을 지정하는 위치 경로입니다. 추출 함수 각각은 위치 경로에서 지정한 값이나 내용을 리턴합니다.

몇몇 요소나 속성값은 여러 번 발생하므로, 추출 기능은 스칼라나 테이블 값을 리턴하며, 후자를 테이블 함수라고 합니다.

XML Extender는 다음 추출 기능을 제공합니다.

- 205 페이지의 『extractInteger() 및 extractIntegers()』
- 207 페이지의 『extractSmallint() 및 extractSmallints()』
- 209 페이지의 『extractDouble() 및 extractDoubles()』
- 211 페이지의 『extractReal() 및 extractReals()』
- 213 페이지의 『extractChar() 및 extractChars()』
- 215 페이지의 『extractVarchar() 및 extractVarchars()』
- 217 페이지의 『extractCLOB() 및 extractCLOBs()』
- 219 페이지의 『extractDate() 및 extractDates()』
- 221 페이지의 『extractTime() 및 extractTimes()』
- 223 페이지의 『extractTimestamp() 및 extractTimestamps()』

다음 절에 있는 예에서는 사용자가 DB2 명령 쉘을 사용하고 있는 것으로 가정하며, 여기에서는 각 명령 시작에 『DB2』를 입력할 필요가 없습니다.

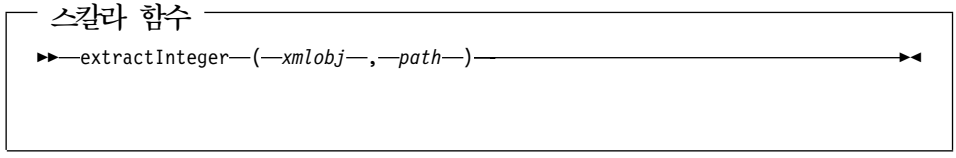
extractInteger() 및 extractIntegers()

목적

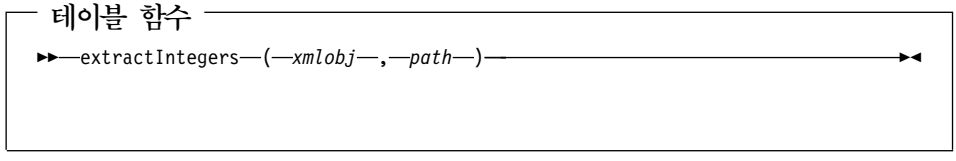
XML 문서에서 내용 요소나 속성값을 추출하고, INTEGER 유형으로서 데이터를 리턴합니다.

구문

스칼라 함수



테이블 함수



매개변수

표 29. extractInteger 및 extractIntegers 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

INTEGER

리턴된 컬럼 이름(테이블 함수)

returnedInteger

예

스칼라 함수 예:

다음 예에서는 키의 속성 값이 ″1″일 때 하나의 값이 리턴됩니다. 이 값은 자동으로 DECIMAL 유형으로 변환되는 정수로 추출됩니다.

```
CREATE TABLE t1(key decimal(3,2));
INSERT into t1 values
SELECT * from table(db2xml.extractInteger(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/[@key="1"]'));
SELECT * from t1;
```

테이블 함수 예:

다음 예에서, 판매 주문에 대한 각각의 키 값은 INTEGER로 추출됩니다.

```
SELECT * from table(db2xml.extractIntegers(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/@key')) as x;
```

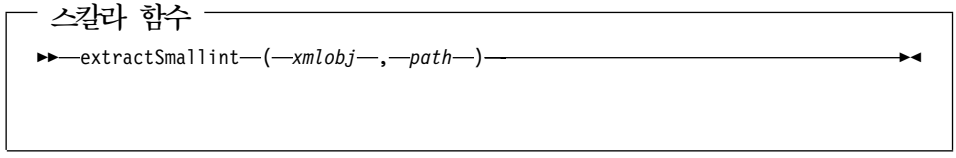
extractSmallint() 및 extractSmallints()

목적

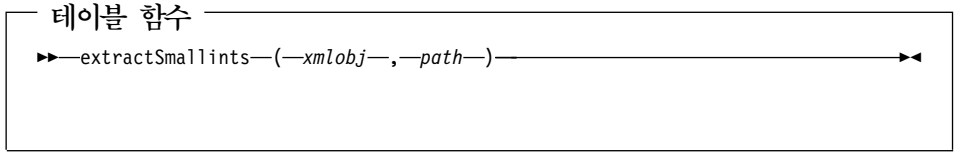
XML 문서에서 내용 요소나 속성값을 추출하고, SMALLINT 유형으로서 데이터를 리턴합니다.

구문

스칼라 함수



테이블 함수



매개변수

표 30. extractSmallint 및 extractSmallints 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

SMALLINT

리턴된 컬럼 이름(테이블 함수)

returnedSmallint

예

다음 예에서, 모든 판매 주문서의 Quantity 값은 SMALLINT로 추출됩니다.

```
SELECT * from table(db2xml.extractSmallints(db2xml.File
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Quantity')) as x;
```

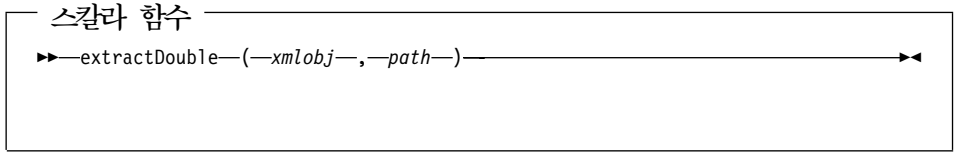
extractDouble() 및 extractDoubles()

목적

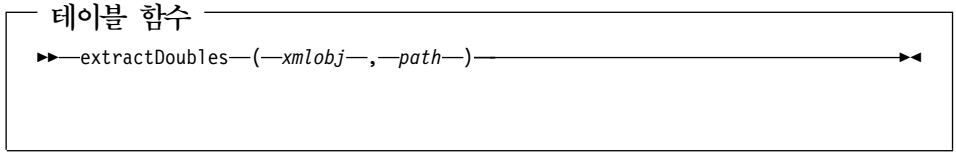
XML 문서에서 내용 요소나 속성값을 추출하고, DOUBLE 유형으로서 데이터를 리턴합니다.

구문

스칼라 함수



테이블 함수



매개변수

표 31. extractDouble 및 extractDoubles 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

DOUBLE

리턴된 컬럼 이름(테이블 함수)

returnedDouble

예

스칼라 함수 예:

다음 예는 주문서의 가격을 자동으로 DOUBLE 유형에서 DECIMAL 유형으로 변환합니다.

```
CREATE TABLE t1(price, DECIMAL(5,2));
INSERT into t1 values (db2xml.extractDouble(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'),
    '/Order/Part[@color="black "]/ExtendedPrice'));
SELECT * from t1;
```

테이블 함수 예:

다음 예에서, 판매 주문서의 각 파트의 ExtendedPrice의 값은 DOUBLE로 추출됩니다.

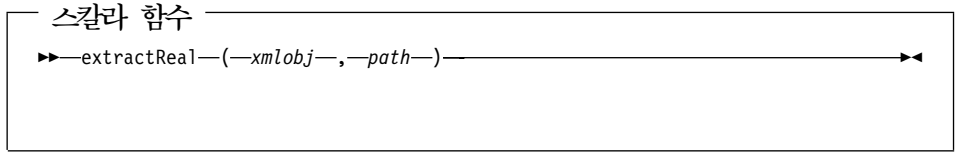
```
SELECT * from table(db2xml.extractDoubles(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/ExtendedPrice')) as x;
```

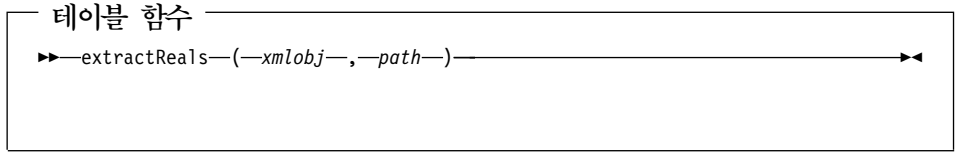
extractReal() 및 extractReals()

목적

XML 문서에서 내용 요소나 속성값을 추출하고, REAL 유형으로서 데이터를 리턴합니다.

구문





매개변수

표 32. extractReal 및 extractReals 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

REAL

리턴된 컬럼 이름(테이블 함수)

returnedReal

예

다음 예에서 Tax의 각 값은 REAL로 추출됩니다.

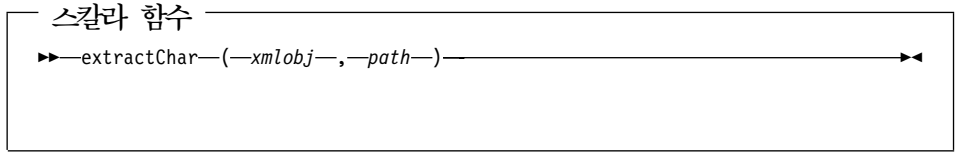
```
SELECT * from table(db2xml.extractReals(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Tax')) as x;
```

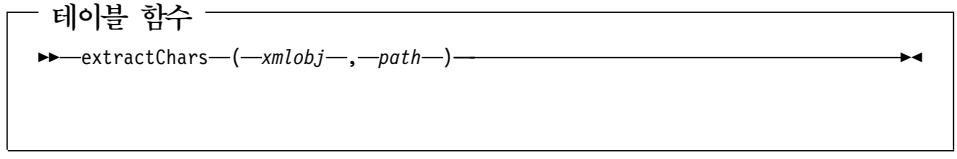
extractChar() 및 extractChars()

목적

XML 문서에서 내용 요소나 속성값을 추출하고, CHAR 유형으로서 데이터를 리턴합니다.

구문





매개변수

표 33. extractChar 및 extractChars 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

CHAR

리턴된 컬럼 이름(테이블 함수)

returnedChar

예

다음 예에서, Color의 값은 CHAR로 추출됩니다.

```
SELECT * from table(db2xml.extractChars(Order,
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/@Color')) as x;
```

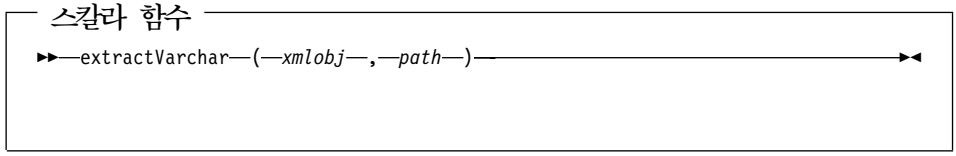
extractVarchar() 및 extractVarchars()

목적

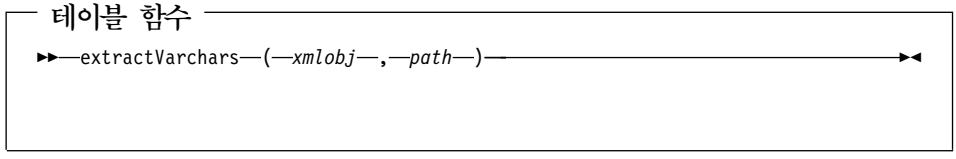
XML 문서에서 내용 요소나 속성값을 추출하고, VARCHAR 유형으로서 데이터를 리턴합니다.

구문

스칼라 함수



테이블 함수



매개변수

표 34. extractVarchar 및 extractVarchars 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

VARCHAR(4K)

리턴된 컬럼 이름(테이블 함수)

returnedVarchar

예

SALES_TAB 테이블에 있는 ORDER 컬럼에 1000개 이상의 XML 문서가 저장되어 있는 데이터베이스에서, ExtendedPrice가 2500.00 이상인 항목을 주문한 모든 고객을 탐색하려 합니다. 다음 SQL문에서는 SELECT 절에서 추출 UDF를 사용합니다.

```
SELECT extractVarchar(Order, '/Order/Customer/Name') from sales_order_view
   WHERE price > 2500.00
```

UDF extractVarchar()는 ORDER 컬럼을 입력으로 사용하고, /Order/Customer/Name 위치 경로를 선택 식별자로서 사용합니다. UDF는 고객 이름을 리턴합니다. Where 절에서, 추출 함수는 ExtendedPrice가 2500.00 이상인 주문만을 평가합니다.

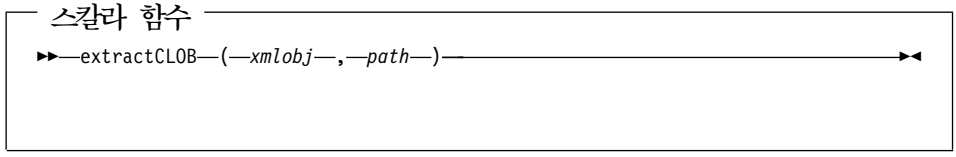
extractCLOB() 및 extractCLOBs()

목적

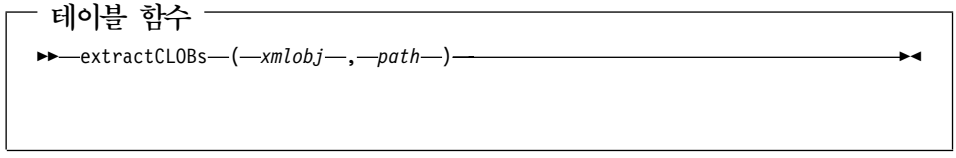
요소와 속성 마크업, 요소와 속성의 내용 및 하부 요소와 함께 XML 문서의 조각을 추출합니다. 이 함수는 다른 추출 함수들과 상이합니다. 이들 함수는 요소와 속성의 내용만 리턴합니다. extractClob(s) 함수는 문서 조각을 추출하는 경우에만 사용해야 하며, extractVarchar(s) 및 extractChar(s)는 단순 값을 추출하는 경우에 사용하십시오.

구문

스칼라 함수



테이블 함수



매개변수

표 35. extractCLOB 및 extractCLOBs 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

CLOB(10K)

리턴된 컬럼 이름(테이블 함수)

returnedCLOB

예

다음 예에서, 모든 파트 요소는 구매 주문서에서 추출됩니다.

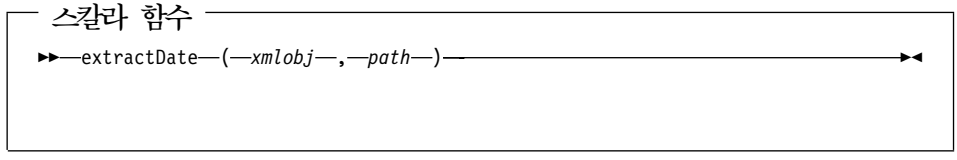
```
SELECT returnedCLOB as part
   from table(db2xml.extractCLOBs(db2xml.XMLFile('c:\dxx\samples\xml\getstart.xml'),
   '/Order/Part')) as x;
```

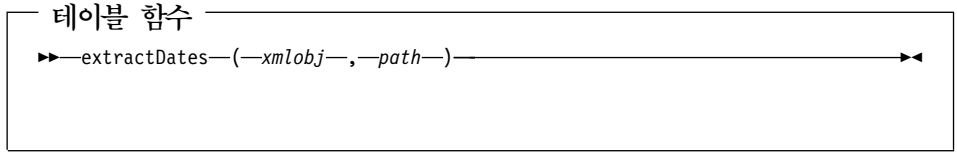
extractDate() 및 extractDates()

목적

XML 문서에서 내용 요소나 속성값을 추출하고, DATE 유형으로서 데이터를 리턴합니다.

구문





매개변수

표 36. extractDate 및 extractDates 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

DATE

리턴된 컬럼 이름(테이블 함수)

returnedDate

예

다음 예에서 ShipDate의 값은 DATE로 추출됩니다.

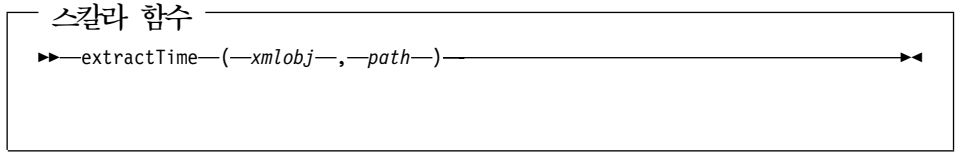
```
SELECT * from table(db2xml.extractDates(db2xml.XMLFile
   ('c:\dxx\samples\xml\getstart.xml'), '/Order/Part/Shipment/ShipDate')) as x;
```

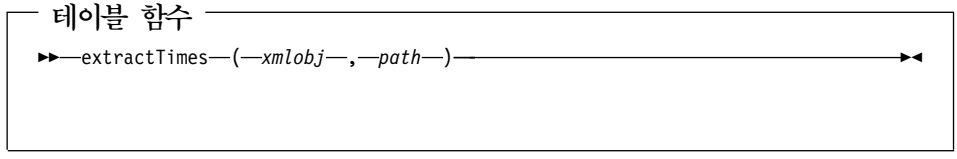
extractTime() 및 extractTimes()

목적

XML 문서에서 내용 요소나 속성값을 추출하고, TIME 유형으로서 데이터를 리턴합니다.

구문





매개변수

표 37. extractTime 및 extractTimes 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

TIME

리턴된 컬럼 이름(테이블 함수)

returnedTime

예

이 예는 책 샘플 파일을 사용합니다. XML 파일 book.xml에서 책의 가격이 매겨진 시간을 검색하여 그 값을 TIME으로 보냅니다.

XML 파일:

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
        <chapter id="1" date="07/01/97">
                <section>This is a section in Chapter One.</section>
        <chapter id="2" date="01/02/1997">
                <section>This is a section in Chapter Two.</section>
        </chapter>
        <price date="12/22/1998" time="11.12.13" timestamp="1998-12-22-11.12.13.888888">
                38.281
        </price>
</book>
```

추출 예:

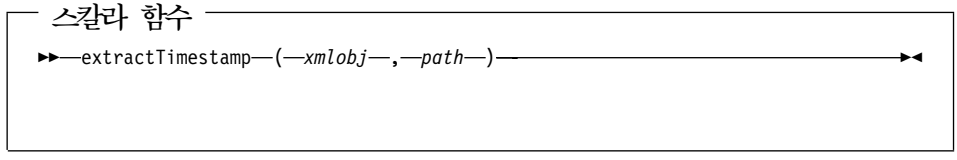
```
CREATE TABLE t1(SaleTime TIME);
INSERT INTO t1 values(db2xml.extractTime(doc, '/book/price/@time'));
SELECT * from t1;
```

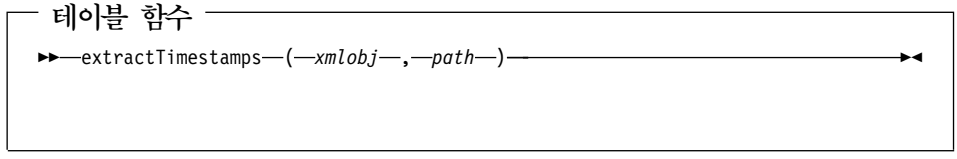
extractTimestamp() 및 extractTimestamps()

목적

XML 문서에서 내용 요소나 속성값을 추출하고, TIMESTAMP 유형으로서 데이터를 리턴합니다.

구문





매개변수

표 38. extractTimestamp 및 extractTimestamps 함수 매개변수

매개변수	데이터 유형	설명
xmlobj	XMLVARCHAR, XMLFILE 또는 XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.

리턴 유형

TIMESTAMP

리턴된 컬럼 이름(테이블 함수)

returnedTimestamp

예

이 예는 책 샘플 파일을 사용합니다. XML 파일 book.xml에서 각 책의 가격이 매겨진 때를 나타내는 시간을 검색하여 그 값을 TIMESTAMP로 추출합니다.

XML 파일:

```
<?xml version="1.0">
<DOCTYPE book SYSTEM "c:\dxx\samples\book.dtd">
<book>
        <chapter id="1" date="07/01/97">
                <section>This is a section in Chapter One.</section>
        <chapter id="2" date="01/02/1997">
                <section>This is a section in Chapter Two.</section>
        </chapter>
        <price date="12/22/1998" time="11.12.13" timestamp="1998-12-22-11.12.13.888888">
                38.281
        </price>
</book>
```

추출 예:

```
CREATE TABLE t1(SaleTime TIMESTAMP);
INSERT INTO t1 values(db2xml.extractTimestamp(doc, '/book/price/@timestamp'));
SELECT * from t1;
```

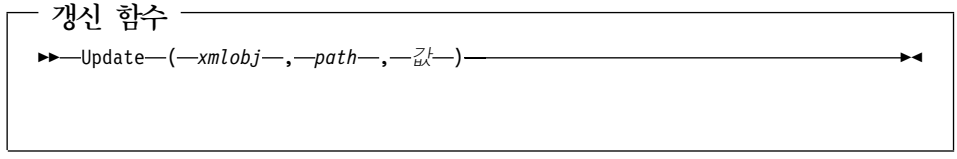
갱신 함수

Update() 함수는 XML 컬럼에 저장된 하나 이상의 XML 문서에서 지정된 요소나 속성 값을 갱신합니다. 142 페이지의 『XML 데이터 갱신』에 설명된 바와 같이, 기본 유형 변환 함수를 사용하여 SQL 기본 유형을 XML UDT로 변환할 수도 있습니다.

목적

XML UDT의 컬럼 이름, 위치 경로 및 갱신 값 문자열을 사용하고, 첫번째 입력 매개변수와 같은 XML UDT를 리턴합니다. Update() 함수를 사용하면 갱신될 요소나 속성을 지정할 수 있습니다.

구문



매개변수

표 39. UDF Update 매개변수

매개변수	데이터 유형	설명
xmlobj	위치 지정자로서 XMLVARCHAR, XMLCLOB	컬럼 이름.
경로	VARCHAR	요소나 속성의 위치 경로.
value	VARCHAR	갱신 문자열.

중요사항: 갱신 UDF는 요소가 아닌, 속성을 지닌 술어를 보유한 위치 경로를 지원합니다. 예를 들어, 다음과 같은 술어가 지원됩니다.

```
'/Order/Part[@color="black "]/ExtendedPrice'
```

다음 술어는 지원되지 않습니다.

```
'/Order/Part/Shipment/[Shipdate < "11/25/00"]'
```

리턴 유형

데이터 유형	리턴 유형
XMLVARCHAR	XMLVARCHAR
LOCATOR로서의 XMLCLOB	XMLCLOB

예

다음 예는 판매원 Sriram Srinivasan에 의해 처리되는 구매 주문서를 갱신합니다.

```
UPDATE sales_tab
       set order = Update(order, '/Order/Customer/Name', 'IBM')
   WHERE sales_person = 'Sriram Srinivasan'
```

이 예에서 /Order/Customer/Name의 내용은 IBM으로 갱신됩니다.

사용법

갱신 함수를 사용하여 하나 이상의 XML 문서에 있는 값을 변경하는 경우, 이는 실제로 XML 컬럼 내의 XML 문서를 대체합니다. XML 구문분석기의 출력에 따라 원래 문서의 일부가 유지되며, 다른 부분은 유실되거나 변경됩니다. 다음 섹션은 문서가 처리되는 방법을 설명하며, 갱신 전후 문서의 형태에 대한 예를 제공합니다.

갱신 함수가 XML 문서를 처리하는 방법

갱신 함수가 XML 문서를 대체하는 경우, 이 함수는 XML 구문분석기 출력에 따라 문서를 재구성해야 합니다. 227 페이지의 표40은 예를 사용하여 문서의 일부를 처리하는 방법을 설명합니다. 갱신 전후 XML 문서를 비교하는 예의 경우에는 228 페이지의 『예』를 참조하십시오.

표 40. 갱신 함수 규칙

항목 또는 노드 유형	XML 문서 코드 예	갱신 이후 상태
XML 선언	<?xml version='1.0' encoding='utf-8' standalone='yes' >	XML 선언이 유지됩니다. • 버전 정보가 유지됩니다. • 코드화 선언이 유지되며, 원래 문서에 지정되는 경우에 나타납니다. • 독립형 선언이 유지되며, 원래 문서에 지정되는 경우에 나타납니다. • 갱신 이후, 작은 따옴표를 사용하여 값을 구분합니다.
DOCTYPE 선언	<!DOCTYPE books SYSTEM "http://dtds.org/books.dtd" > <!DOCTYPE books PUBLIC "local.books.dtd" "http://dtds.org/books.dtd" > <!DOCTYPE books> -Any of <!DOCTYPE books (S ExternalID) ? [internal-dtd-subset] > -Such as <!DOCTYPE books [<!ENTITY mydog "Spot">] >? [internal-dtd-subset] >	문서 유형 선언이 유지됩니다. • 루트 요소 이름이 지원됩니다. • 공용 및 시스템 ExternalID가 유지되며, 원래 문서에 지정될 때 나타납니다. • 내부 DTD 부분 집합이 유지되지 않습니다. 엔터티가 대체됩니다. 속성에 대한 기본값이 유지되며, 출력 문서에 나타납니다. • 갱신 이후, 인용 부호를 사용하여 공용 및 시스템 URI 값을 서술합니다. • 현재의 XML4c 구문분석기는 ExternalID 또는 내부 DTD 부분 집합을 포함하지 않는 XML 선언을 보고하지 않습니다. 갱신 이후, DOCTYPE 선언은 이 케이스에서 누락됩니다.
지시사항 처리	<?xml-stylesheet title="compact" href="datatypes1.xsl" type="text/xsl"?>	지시사항 처리가 유지됩니다.

표 40. 갱신 함수 규칙 (계속)

항목 또는 노드 유형	XML 문서 코드 예	갱신 이후 상태
주석	`<!-- comment -->`	루트 요소 내에서는 주석이 유지됩니다. 루트 요소 외부의 주석은 버려집니다.

요소	`<books>`
	내용
	`</books>`
		요소가 유지됩니다.

속성	`id='1' date='01/02/1997"`	요소의 속성이 유지됩니다. • 갱신 이후, 인용 부호를 사용하여 값을 서술합니다. • 속성 내의 데이터가 확장됩니다. • 엔터티가 대체됩니다.
텍스트 노드	이 장은 `my dog &mydoc;`에 대한 내용입니다. 이 장은 `my dog Spot`에 대한 내용입니다.	텍스트 노드(요소 내용)가 유지됩니다. • 텍스트 노드 내의 데이터가 확장됩니다. • 엔터티가 대체됩니다.

다중 발생

위치 경로가 Update() UDF에서 제공되는 경우, 일치 경로를 지닌 모든 요소나 속성의 내용은 제공되는 값으로 갱신됩니다. 이는 문서가 다중 발생 위치 경로를 보유하는 경우, 갱신 함수가 기존 값을 *값* 매개변수에서 제공되는 값으로 대체함을 의미합니다.

경로 매개변수에서 술어 지정을 사용하면, 의도하지 않은 갱신을 예방하기 위한 독립적인 위치 경로를 제공할 수 있습니다. 갱신 UDF는 요소가 아닌 속성를 지닌 술어를 보유한 위치 경로를 지원합니다. 자세한 정보는 225 페이지의 『매개변수』를 참조하십시오.

예

다음 예는 갱신 전후 XML 문서의 인스턴스를 나타냅니다.

예 1:

이전:

```
<?xml version='1.0'   encoding='utf-8'    standalone="yes"?>
<!DOCTYPE book PUBLIC "public.dtd" "system.dtd">
<?pitarget option1='value1' option2='value2'?>
<!-- comment -->
<book>
  <chapter id="1" date='07/01/1997'>
    <!-- first section -->
    <section>This is a section in Chapter One.</section>
  </chapter>
  <chapter id="2" date="01/02/1997">
    <section>This is a section in Chapter Two.</section>
    <footnote>A footnote in Chapter Two is here.</footnote>
  </chapter>
  <price date="12/22/1998" time="11.12.13"
       timestamp="1998-12-22-11.12.13.888888">38.281</price>
</book>
```

- XML 선언에 공백 포함
- 지시사항 처리를 지정
- 루트 노드 외부에 주석 포함
- PUBLIC ExternalID 지정
- 루트 노드 내부에 주석 포함

이후:

```
<?xml version='1.0' encoding='utf-8' standalone='yes'?>
<!DOCTYPE book PUBLIC "public.dtd" "system.dtd">
<?pitarget option1='value1' option2='value2'?><book>
  <chapter id="1" date="07/01/1997">
        <!-- first section -->
        <section>This is a section in Chapter One.</section>
  </chapter>
  <chapter id="2" date="01/02/1997">
        <section>This is a section in Chapter Two.</section>
        <footnote>A footnote in Chapter Two is here.</footnote>
  </chapter>
  <price date="12/22/1998" time="11.12.13"
       timestamp="1998-12-22-11.12.13.888888">60.02</price>
</book>
```

- 마크업 내부의 공백이 제거됨
- 지시사항 처리가 유지됨
- 루트 노드 외부의 주석이 유지되지 않음
- PUBLIC ExternalID가 유지됨
- 루트 노드 내부의 주석이 유지됨
- 변경 값이 <price> 요소의 값임

예 2:

이전:

```
<?xml version='1.0'     ?>
<!DOCTYPE book>
<!-- comment -->
<book>
  ...
</book>
```

ExternalID 또는 내부 DTD 부분 집합이 없는 DOCTYPE 선언을 포함합니다. 지원되지 않습니다.

이후:

```
<?xml version='1.0'?>
<book>
  ...
</book>
```

DOCTYPE 선언은 XML 구문분석기에 의해 보고되지 않으며, 유지되지 않습니다.

예 3:

이전:

```
<?xml version='1.0'     ?>
<!DOCTYPE book [ <!ENTITY myDog "Spot"> ]>
<!-- comment -->
<book>
  <chapter id="1" date='07/01/1997'>
       <!-- first section -->
       <section>This is a section in Chapter
         One about my dog &;myDog;.</section>
       ...
  </chapter>
  ...
</book>
```

- 마크업의 공백 포함
- 내부 DTD 부분 집합 지정
- 텍스트 노드에 엔터티 지정

이후:

```
<?xml version='1.0'?>
<!DOCTYPE book>
<book>
  <chapter id="1" date="07/01/1997">
       <!-- first section -->
       <section>This is a section in Chapter
         One about my dog Spot.</section>
          ...
  </chapter>
  ...
</book>
```

- 마크업의 공백이 제거됨
- 내부 DTD 부분 집합이 유지되지 않음
- 텍스트 노드의 엔터티가 분석되고 대체됨

제10장 XML Extender 저장 프로시듀어

XML Extender는 XML 컬럼 및 콜렉션의 관리를 위해 저장 프로시듀어를 제공합니다. 이러한 저장 프로시듀어들은 DB2 클라이언트에서 호출될 수 있습니다. 클라이언트 인터페이스는 SQL, ODBC 또는 JDBC에 내장될 수 있습니다. 저장 프로시듀어 호출 방법에 대한 자세한 내용은 *DB2 UDB 관리 안내서*에 있는 저장 프로시듀어에 관한 절을 참조하십시오.

저장 프로시듀어는 db2xml 스키마를 사용하며, 이는 XML Extender의 스키마 이름입니다.

XML Extender에서는 세가지 유형의 저장 프로시듀어를 제공합니다.

- 234 페이지의 『관리 저장 프로시듀어』, 사용자가 관리 타스크를 완료하는 데 도움이 됩니다.
- 242 페이지의 『작성 저장 프로시듀어』, 기존 데이터베이스 테이블에 있는 데이터를 사용하여 XML 문서를 생성합니다.
- 249 페이지의 『분해 저장 프로시듀어』, 들어오는 XML 문서를 분할하여 새 데이터베이스 테이블이나 기존 데이터베이스 테이블에 데이터를 저장합니다.

XML 콜렉션 저장 프로시듀어에 의해 사용되는 매개변수 한계는 317 페이지의 『부록D. XML Extender 한계』에 문서화되어 있습니다.

include 파일 지정

저장 프로시듀어를 호출하는 프로그램에서 XML Extender 외부 머리글 파일을 포함시키도록 하십시오. 머리글 파일은 `DXX_INSTALL/include` 디렉토리에 있습니다. `DXX_INSTALL`은 XML Extender의 설치 디렉토리입니다. 이것은 운영 체제에 따라 다릅니다. 머리글 파일은 다음과 같습니다.

dxx.h XML Extender에서 정의된 상수 및 데이터 유형

dxxrc.h XML Extender 리턴 코드

이러한 머리글 파일을 포함시키는 구문은 다음과 같습니다.

```
#include "dxx.h"
#include "dxxrc.h"
```

include 파일의 경로가 makefile에서 컴파일 옵션과 함께 지정되었는지 확인하십시오.

XML Extender 저장 프로시듀어 호출

일반적으로, 다음 구문을 사용하여 XML Extender를 호출합니다.

```
CALL db2xml.function_entry_point
```

여기서,

db2xml
XML Extender 저장 프로시듀어의 라이브러리를 지정합니다. UNIX 운영 체제에서, 라이브러리는 sqllib/function 디렉토리에 저장됩니다. Windows 운영 체제에서, 라이브러리는 *DXX_INSTALL*/bin 디렉토리에 저장됩니다.

function_entry_point
저장 프로시듀어로 전달될 인수를 지정합니다.

CALL문에서, 저장 프로시듀어에 전달될 인수는 상수나 표현식이 아닌 호스트 변수여야 합니다. 호스트 변수는 널 표시기가 될 수 있습니다. DXX_INSTALL/samples/c 및 DXX_INSTALL/samples/cli 디렉토리에 있는 저장 프로시듀어 호출 샘플, 이 책에서는 44 페이지의 『XML 문서 작성』 및 153 페이지의 『제6장 XML 콜렉션 데이터 관리』 절을 참조하십시오.

DXX_INSTALL/samples/c 디렉토리에서는, 내장 SQL을 사용하여 XML 콜렉션 저장 프로시듀어를 호출하는 SQC 코드 파일이 제공됩니다. DXX_INSTALL/samples/cli 디렉토리에서, 샘플 파일은 CLI(Call Level Interface)를 사용하여 저장 프로시듀어를 호출하는 방법을 보여줍니다.

CLOB 한계 증가

저장 프로시듀어에 전달될 때 CLOB 매개변수의 기본 한계는 1 MB입니다. 다음 단계를 수행하면 이 한계를 증가시킬 수 있습니다.

1. 각각의 저장 프로시듀어를 제거하십시오. 예를 들면 다음과 같습니다.

```
db2   "drop procedure db2xml.dxxShredXML"
```

2. 증가된 CLOB 한계를 지닌 새 프로시듀어를 작성하십시오. 예를 들면 다음과 같습니다.

```
db2   "create procedure db2xml.dxxShredXML(in     dadBuf      clob(100K),
                                           in     XMLObj      clob(10M),
                                           out    returnCode  integer,
                                           out    returnMsg   varchar(1024)
                                           )
       external name 'db2xml.dxxShredXML'
       language C
       parameter style DB2DARI
       not deterministic
       fenced
       null call;
```

시작하기 전에

데이터베이스를 XML Extender 저장 프로시듀어 및 DB2 CLI 바인드 파일과 바인드하십시오. 샘플 명령 파일 getstart_prep.cmd를 사용하여 파일들을 바인드할 수 있습니다. 이 명령 파일은 DXX_INSTALL/samples/cmd 디렉토리에 있습니다.

1. 데이터베이스에 연결하십시오. 예를 들면 다음과 같습니다.

```
db2 "connect to SALES_DB"
```

2. DXX_INSTALL/bnd 디렉토리로 변경한 다음, XML Extender를 데이터베이스로 바인드하십시오.

```
db2 "bind @dxxbind.lst"
```

3. sqllib/bnd 디렉토리로 변경한 뒤 CLI를 데이터베이스와 바인드하십시오.

```
db2 "bind @db2cli.lst"
```

4. 연결을 종료하십시오.

```
db2 "terminate"
```

관리 저장 프로시듀어

이러한 저장 프로시듀어는 XML 컬럼 또는 콜렉션의 사용 가능 또는 사용 불능과 같은 관리 타스크에 사용됩니다. 이들은 XML Extender 관리 마법사 및 관리 명령 **dxxadm**으로 호출됩니다.

dxxEnableDB()

목적

데이터베이스를 사용할 수 있도록 설정합니다. 데이터베이스를 사용할 수 있게 되면, XML Extender는 다음 오브젝트를 작성합니다.

- XML Extender 사용자 정의 유형(UDT).
- XML Extender 사용자 정의 기능(UDF).
- XML Extender DTD 참조 테이블 DTD_REF, 여기에는 DTD 및 각 DTD 정보를 저장합니다. DTD_REF 테이블에 대한 완전한 설명은 255 페이지의 『DTD 참조 테이블』을 참조하십시오.
- XML Extender 이용 테이블 XML_USAGE, 여기에는 XML 및 각 콜렉션에 대해 사용할 수 있는 각 컬럼의 일반 정보가 저장됩니다. XML_USAGE 테이블에 대한 완전한 설명은 256 페이지의 『XML 사용 테이블』을 참조하십시오.

```
dxxEnableDB(char(dbName) dbName,       /* input */
            long         returnCode,   /* output */
            varchar(1024) returnMsg)   /* output */
```

매개변수

표 41. dxxEnableDB() 매개변수

매개변수	설명	IN/OUT 매개변수
dbName	데이터베이스 이름.	IN
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

dxxDisableDB()

목적

데이터베이스를 사용할 수 없도록 설정합니다. XML Extender가 데이터베이스를 사용할 수 없게 되면, 다음 오브젝트를 제거합니다.

- XML Extender 사용자 정의 유형(UDT).
- XML Extender 사용자 정의 기능(UDF).
- XML Extender DTD 참조 테이블 DTD_REF, 여기에는 DTD 및 각 DTD 정보를 저장합니다. DTD_REF 테이블에 대한 완전한 설명은 255 페이지의 『DTD 참조 테이블』을 참조하십시오.
- XML Extender 이용 테이블 XML_USAGE, 여기에는 XML 및 각 콜렉션에 대해 사용할 수 있는 각 컬럼의 일반 정보가 저장됩니다. XML_USAGE 테이블에 대한 완전한 설명은 256 페이지의 『XML 사용 테이블』을 참조하십시오.

중요사항: 데이터베이스를 사용 불능으로 설정하기 전에 모든 XML 컬럼들을 사용 불능으로 설정해야합니다. XML Extender는 XML에 사용할 수 있는 컬럼이나 콜렉션이 포함된 데이터베이스를 사용 불능으로 설정할 수 없습니다.

```
dxxDisableDB(char(dbName)     dbName,       /* input */
             long             returnCode,      /* output */
             varchar(1024)    returnMsg)       /* output */
```

매개변수

표 42. dxxDisableDB() 매개변수

매개변수	설명	IN/OUT 매개변수
dbName	데이터베이스 이름.	IN
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

dxxEnableColumn()

목적

XML 컬럼을 사용할 수 있도록 설정합니다. 컬럼을 사용할 수 있게 되면, XML Extender는 다음 타스크를 완료합니다.

- XML 테이블에 기본 키가 있는지 판별합니다. 기본 키가 없으면, XML Extender는 XML 테이블을 변경하고 DXXROOT_ID 컬럼을 추가합니다.
- XML 테이블에서 각 행에 대한 고유한 식별자가 들어있는 컬럼으로 DAD 파일에서 지정된 부가 테이블을 작성합니다. 이 컬럼은 사용자가 지정한 *root_id* 이거나, XML Extender가 명명한 DXXROOT_ID입니다.
- 선택적으로 사용자가 지정한 이름을 사용하여 XML 테이블 및 부가 테이블에 대한 기본 뷰를 작성합니다.

```
dxxEnableColumn(char(dbName) dbName,      /* input */
                char(tbName) tbName,      /* input */
                char(colName) colName,     /* input */
                    CLOB(100K)     DAD,         /* input */
                char(tablespace) tablespace,  /* input */
                char(defaultView) defaultView, /* input */
                char(rootID) rootID,      /* input */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

매개변수

표 43. dxxEnableColumn() 매개변수

매개변수	설명	IN/OUT 매개변수
dbName	데이터베이스 이름.	IN
tbName	XML 컬럼이 있는 테이블 이름.	IN
colName	XML 컬럼 이름.	IN
DAD	DAD 파일이 들어있는 CLOB	IN
tablespace	기본 테이블 공간 이외의 부가 테이블이 들어있는 테이블 공간. 지정되지 않았으면 기본 테이블 공간이 사용됩니다.	IN
defaultView	응용프로그램 테이블과 부가 테이블을 조인하는 기본 보기의 이름.	IN

표 43. dxxEnableColumn() 매개변수 (계속)

매개변수	설명	IN/OUT 매개변수
rootID	응용프로그램 테이블에서 부가 테이블에 대한 *root_id*로서 사용될 단일 기본 키 이름.	IN
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

dxxDisableColumn()

목적

XML이 사용할 수 있는 컬럼을 사용할 수 없도록 설정합니다. XML 컬럼을 사용할 수 없게 되면, 더 이상 XML 데이터 유형이 포함될 수 없습니다.

```
dxxDisableColumn(char(dbName) dbName,      /* input */
                 char(tbName) tbName,      /* input */
                 char(colName) colName,     /* input */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

매개변수

표 44. dxxDisableColumn() 매개변수

매개변수	설명	IN/OUT 매개변수
dbName	데이터베이스 이름.	IN
tbName	XML 컬럼이 있는 테이블 이름.	IN
colName	XML 컬럼 이름.	IN
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

dxxEnableCollection()

목적

응용프로그램 테이블과 연관된 XML 콜렉션을 사용할 수 있도록 설정합니다.

```
dxxEnableCollection(char(dbName) dbName,      /* input */
                    char(colName) colName,     /* input */
                    CLOB(100K)     DAD,         /* input */
                    char(tablespace) tablespace,  /* input */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

매개변수

표 45. dxxEnableCollection() 매개변수

매개변수	설명	IN/OUT 매개변수
dbName	데이터베이스 이름.	IN
colName	XML 콜렉션 이름.	IN
DAD	DAD 파일이 들어있는 CLOB	IN
tablespace	기본 테이블 공간 이외의 부가 테이블이 들어있는 테이블 공간. 지정되지 않았으면 기본 테이블 공간이 사용됩니다.	IN
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

dxxDisableCollection()

목적

테이블과 컬럼을 콜렉션의 일부로서 표시하는 표시문자를 제거하여, XML이 사용할 수 있는 콜렉션을 사용할 수 없도록 합니다.

```
dxxDisableCollection(char(dbName) dbName,      /* input */
                     char(colName) colName,    /* input */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

매개변수

표 46. dxxDisableCollection() 매개변수

매개변수	설명	IN/OUT 매개변수
dbName	데이터베이스 이름.	IN
colName	XML 콜렉션 이름.	IN
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

작성 저장 프로시듀어

작성 저장 프로시듀어 dxxGenXML() 및 dxxRetrieveXML()은 기존 데이터베이스 테이블에 있는 데이터를 사용하여 XML 문서를 생성하는 데 사용됩니다. dxxGenXML() 저장 프로시듀어는 입력으로서 DAD 파일을 사용하며, 사용 가능한 XML 콜렉션이 필요하지 않습니다. dxxRetrieveXML() 저장 프로시듀어는 입력으로서 사용 가능한 XML 콜렉션을 사용합니다.

dxxGenXML()

목적

DAD 파일에서 <Xcollection>으로 지정된 XML 콜렉션 테이블에 저장되어 있는 데이터를 사용하여 XML 문서를 생성하고, 각각의 XML 문서를 결과 테이블에 행으로서 삽입합니다. 또한 커서로 결과 테이블을 열고 결과 세트를 사전 추출할 수 있습니다.

유연성을 제공하기 위해, dxxGenXML()은 사용자가 결과 테이블에서 생성될 최대 행 수를 지정할 수 있도록 합니다. 이렇게 하면 시도 프로세스 동안 응용프로그램이 결과를 기다려야 하는 시간이 줄어듭니다. 저장 프로시듀어는 테이블에 있는 실제 행 수와, 오류 코드 및 오류 메시지가 포함된 오류 정보를 리턴합니다.

동적 조회를 지원하기 위해, dxxGenXML()은 입력 매개변수 *override*를 사용합니다. 입력 *overrideType*에 따라, 응용프로그램은 DAD 파일에서 SQL 맵핑의 경우 SQL_stmt를, RDB_node 맵핑의 경우 RDB_node에 있는 조건을 겹쳐쓰기할 수 있습니다. 입력 매개변수 *overrideType*은 *override* 유형을 명시하는 데 사용됩니다. *override* 매개변수에 대한 세부사항은 158 페이지의 『DAD 파일에서 동적으로 값 겹쳐쓰기』를 참조하십시오.

```
dxxGenXML(CLOB(100K)     DAD,             /* input */
          char(resultTabName) resultTabName,   /* input */
          integer        overrideType     /* input */
               varchar(1024)  override,        /* input */
               integer        maxRows,         /* input */
               integer        numRows,         /* output */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

매개변수

표 47. dxxGenXML() 매개변수

매개변수	설명	IN/OUT 매개변수
DAD	DAD 파일이 들어있는 CLOB	IN
resultTabName	결과 테이블 이름으로서, 호출하기 전에 있어야 합니다. 테이블에는 XMLVARCHAR 또는 XMLCLOB 유형의 한 컬럼만이 들어갑니다.	IN

표 47. *dxxGenXML() 매개변수 (계속)*

매개변수	설명	IN/OUT 매개변수
overrideType	다음 *override* 매개변수 유형을 나타내는 플래그. • **NO_OVERRIDE**: 겹쳐쓰기 하지 않음. • **SQL_OVERRIDE**: SQL_stmt가 겹쳐쓰기함. • **XML_OVERRIDE**: XPath 기본 조건으로 겹쳐쓰기됨.	IN
override	DAD 파일에 있는 조건을 겹쳐쓰기합니다. 입력값은 *overrideType*을 기초로 합니다. • **NO_OVERRIDE**: 널 문자열. • **SQL_OVERRIDE**: 유효한 SQL문. 이 *overrideType*을 사용하려면 DAD 파일에서 SQL 맵핑이 사용되어야 합니다. 입력 SQL문은 DAD 파일에 있는 SQL_stmt를 겹쳐쓰기합니다. • **XML_OVERRIDE**: "AND"로 구분되고 큰 따옴표로 묶여진 하나 이상의 표현식이 들어있는 문자열. 이 *overrideType*을 사용하려면 DAD 파일에서 RDB_node 맵핑이 사용되어야 합니다.	IN
maxRows	결과 테이블에 있는 최대 행 수.	IN
numRows	결과 테이블에서 생성된 실제 행 수.	OUT
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

예

다음 예에서는 결과 테이블이 XML_ORDER_TAB 이름으로 작성되고 XMLVARCHAR 유형의 한 컬럼이 있는 것으로 가정합니다.

```
#include "dxx.h"
#include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE is CLOB(100K) dad;              /* DAD */
     SQL TYPE is CLOB_FILE  dadFile;          /* dad file */
         char    result_tab[32];  /* name of the result table */
              char    override[2];      /* override, will set to NULL*/
              short   overrideType;     /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
         char                  returnMsg[1024];  /* error message text */
         short                 dad_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                 returnCode_ind;
         short                 returnMsg_ind;
          EXEC SQL END DECLARE SECTION;

         /* create table */
         EXEC CREATE TABLE xml_order_tab (xmlorder XMLVarchar);
     /* read data from a file to a CLOB */
              strcpy(dadfile.name,"e:\dxx\dad\litem3.dad");
              dadfile.name_length = strlen("e:\dxx\dad\litem3.dad");
     dadfile.file_options = SQL_FILE_READ;
     EXEC SQL VALUES (:dadfile) INTO :dad;
         strcpy(result_tab,"xml_order_tab");
     override[0] = '\0';
     overrideType = NO_OVERRIDE;
         max_row = 500;
         num_row = 0;
        returnCode = 0;
         msg_txt[0] = '\0';
         collection_ind = 0;
        dad_ind = 0;
         rtab_ind = 0;
     ov_ind = -1;
         ovtype_ind = 0;
         maxrow_ind = 0;
         numrow_ind = -1;
        returnCode_ind = -1;
        returnMsg_ind = -1;
        /* Call the store procedure */
     EXEC SQL CALL dxxGenXML(:dad:dad_ind;
                :result_tab:rtab_ind,
          :overrideType:ovtype_ind,:override:ov_ind,
                :max_row:maxrow_ind,:num_row:numrow_ind,
                      :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxRetrieveXML()

목적

작성과 분해 모두에 같은 DAD 파일을 사용할 수 있도록 합니다. 저장 프로시듀어 dxxRetrieveXML()은 분해된 XML 문서를 검색하기 위한 방법으로도 제공됩니다. 입력으로서 dxxRetrieveXML()은 DAD 파일이 들어있는 버퍼, 작성된 결과 테이블 이름, 리턴될 최대 행 수를 사용합니다. 결과 테이블의 결과 세트, 결과 세트에 있는 실제 행 수, 오류 코드 및 메시지 텍스트를 리턴합니다.

동적 조회를 지원하기 위해, dxxRetrieveXML()은 입력 매개변수 *override*를 사용합니다. 입력 *overrideType*에 따라, 응용프로그램은 DAD 파일에서 SQL 맵핑의 경우 SQL_stmt를, RDB_node 맵핑의 경우 RDB_node에 있는 조건을 겹쳐쓰기할 수 있습니다. 입력 매개변수 *overrideType*은 *override* 유형을 명시하는 데 사용됩니다. *override* 매개변수에 대한 세부사항은 158 페이지의 『DAD 파일에서 동적으로 값 겹쳐쓰기』를 참조하십시오.

dxxRetrieveXML()을 위한 DAD 파일의 요구사항은 dxxGenXML()의 요구사항과 같습니다. 유일한 차이점은 dxxRetrieveXML()의 경우 DAD가 입력 매개변수는 아니지만, 사용 가능한 XML 콜렉션의 이름이라는 것입니다.

```
dxxRetrieveXML(char(collectionName) collectionName,   /* input */
               char(resultTabName) resultTabName,    /* input */
               integer        overrideType,    /* input */
               varchar(1024)  override,        /* input */
               integer        maxRows,         /* input */
               integer        numRows,         /* output */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

매개변수

표 48. dxxRetrieveXML() 매개변수

매개변수	설명	IN/OUT 매개변수
collectionName	사용 가능한 XML 콜렉션 이름.	IN
resultTabName	결과 테이블 이름으로서, 호출하기 전에 있어야 합니다. 테이블에는 XMLVARCHAR 또는 XMLCLOB 유형의 한 컬럼만이 들어갑니다.	IN

표 48. *dxxRetrieveXML() 매개변수 (계속)*

매개변수	설명	IN/OUT 매개변수
overrideType	다음 *override* 매개변수 유형을 나타내는 플래그. • **NO_OVERRIDE**: 겹쳐쓰기 하지 않음. • **SQL_OVERRIDE**: SQL_stmt가 겹쳐쓰기함. • **XML_OVERRIDE**: XPath 기본 조건으로 겹쳐쓰기됨.	IN
override	DAD 파일에 있는 조건을 겹쳐쓰기합니다. 입력값은 *overrideType*을 기초로 합니다. • **NO_OVERRIDE**: 널 문자열. • **SQL_OVERRIDE**: 유효한 SQL문. 이 *overrideType*을 사용하려면 DAD 파일에서 SQL 맵핑이 사용되어야 합니다. 입력 SQL문은 DAD 파일에 있는 SQL_stmt를 겹쳐쓰기합니다. • **XML_OVERRIDE**: "AND"로 구분되고 큰 따옴표로 묶여진 하나 이상의 표현식이 들어있는 문자열. 이 *overrideType*을 사용하려면 DAD 파일에서 RDB_node 맵핑이 사용되어야 합니다.	IN
maxRows	결과 테이블에 있는 최대 행 수.	IN
numRows	결과 테이블에서 생성된 실제 행 수.	OUT
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

예

다음은 dxxRetrieveXML()의 호출 예입니다. 다음 예에서 결과 테이블은 XML_ORDER_TAB 이름으로 작성되고 XMLVARCHAR 유형의 한 컬럼이 있습니다.

```
#include "dxx.h"
#include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
         char    collection[32];  /* dad buffer */
         char    result_tab[32];  /* name of the result table */
              char    override[2];      /* override, will set to NULL*/
              short   overrideType;     /* defined in dxx.h */
         short   max_row;         /* maximum number of rows */
         short   num_row;         /* actual number of rows */
         long    returnCode;      /* return error code */
         char                    returnMsg[1024];  /* error message text */
         short   dadbuf_ind;
         short   rtab_ind;
         short   ovtype_ind;
         short   ov_inde;
         short   maxrow_ind;
         short   numrow_ind;
         short                   returnCode_ind;
         short                   returnMsg_ind;
          EXEC SQL END DECLARE SECTION;

         /* create table */
         EXEC SQL CREATE TABLE xml_order_tab (xmlorder XMLVarchar);
        /* initialize host variable and indicators */
         strcpy(collection,"sales_ord");
         strcpy(result_tab,"xml_order_tab");
     override[0] = '\0';
     overrideType = NO_OVERRIDE;
         max_row = 500;
         num_row = 0;
        returnCode = 0;
         msg_txt[0] = '\0';
         collection_ind = 0;
         rtab_ind = 0;
     ov_ind = -1;
         ovtype_ind = 0;
         maxrow_ind = 0;
         numrow_ind = -1;
        returnCode_ind = -1;
        returnMsg_ind = -1;
        /* Call the store procedure */
         EXEC SQL CALL dxxRetrieve(:collection:collection_ind;
                :result_tab:rtab_ind,
          :overrideType:ovtype_ind,:override:ov_ind,
                :max_row:maxrow_ind,:num_row:numrow_ind,
                      :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

분해 저장 프로시듀어

분해 저장 프로시듀어 dxxInsertXML() 및 dxxShredXML()은 들어오는 XML 문서를 분할하고, 데이터를 새 또는 기존 데이터베이스 테이블에 저장합니다. dxxInsertXML() 저장 프로시듀어는 사용 가능한 XML 콜렉션 이름을 입력으로 사용합니다. dxxShredXML() 저장 프로시듀어는 DAD 파일을 입력으로 사용하며, 사용 가능한 XML 콜렉션이 필요하지 않습니다.

dxxShredXML()

목적

dxxShredXML() 저장 프로시듀어는 dxxGenXML() 저장 프로시듀어와 한 쌍을 이룹니다. dxxShredXML()이 동작하려면, DAD 파일에서 지정된 모든 테이블들이 있어야 하고, DAD 파일에서 지정된 모든 컬럼과 데이터 유형들이 기존 테이블과 일치해야 합니다. 저장 프로시듀어 dxxShredXML()에서는 조인 테이블 사이에서 기본-외부 키 관계가 필요하지 않으며, 이것은 사용 가능 콜렉션 프로세스 동안 XML Extender가 작성합니다. 그러나, 루트 element_node의 RDB_node에서 지정된 조인 조건 컬럼은 테이블에 있어야 합니다.

```
dxxShredXML(CLOB(100K)    DAD,            /* input */
            CLOB(1M)        xmlobj,           /* input */
              long            returnCode,      /* output */
              varchar(1024)   returnMsg)       /* output */
```

매개변수

표 49. dxxShredXML() 매개변수

매개변수	설명	IN/OUT 매개변수
DAD	DAD 파일이 들어있는 CLOB	IN
xmlobj	XMLCLOB 유형의 XML 문서 오브젝트.	IN
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

예

다음은 dxxShredXML() 호출 예입니다.

```
#include "dxx.h"
#include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
         SQL TYPE is CLOB      dad;             /* DAD*/
         SQL TYPE is CLOB_FILE dadFile;         /* DAD file*/
         SQL TYPE is CLOB      xmlDoc;          /* input XML document */
         SQL TYPE is CLOB_FILE xmlFile;          /* input XML file */
         long                  returnCode;       /* error code */
         char                  returnMsg[1024];  /* error message text */
         short                 dad_ind;
         short                 xmlDoc_ind;
         short                 returnCode_ind;
         short                 returnMsg_ind;
          EXEC SQL END DECLARE SECTION;
```

```
/* initialize host variable and indicators */
      strcpy(dadFile.name,"c:\dxx\samples\dad\getstart_xcollection.dad");
      dadFile.name_length=strlen("c:\dxx\samples\dad\getstart_xcollection.dad");
dadFile.file_option=SQL_FILE_READ;
      strcpy(xmlFile.name,"c:\dxx\samples\cmd\getstart.xml");
      xmlFile.name_length=strlen("c:\dxx\samples\cmd\getstart.xml");
xmlFile.file_option=SQL_FILE_READ;
SQL EXEC VALUES (:dadFile) INTO :dad;
SQL EXEC VALUES (:xmlFile) INTO :xmlDoc;
returnCode = 0;
 returnMsg[0] = '\0';
dad_ind = 0;
xmlDoc_ind = 0;
returnCode_ind = -1;
returnMsg_ind = -1;
/* Call the store procedure */
      EXEC SQL CALL db2xml.dxxShredXML(:dad:dad_ind;
               :xmlDoc:xmlDoc_ind,
               :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

dxxInsertXML()

목적

두 개의 입력 매개변수 즉, 사용 가능한 XML 콜렉션 이름과 분해될 XML 문서를 사용하며, 두 개의 출력 매개변수 즉, 리턴 코드와 리턴 메시지를 리턴합니다.

```
dxxInsertXML(char(collectionName) collectionName,   /* input */
             CLOB(1M)       xmlobj,           /* input */
               long           returnCode,      /* output */
               varchar(1024)  returnMsg)       /* output */
```

매개변수

표 50. dxxInsertXML() 매개변수

매개변수	설명	IN/OUT 매개변수
collectionName	사용 가능한 XML 콜렉션 이름.	IN
xmlobj	CLOB 유형의 XML 문서 오브젝트.	IN
returnCode	저장 프로시듀어의 리턴 코드.	OUT
returnMsg	오류가 발생하면 리턴되는 메시지 텍스트.	OUT

예

다음 예에서, dxxInsertXML() 호출은 입력 XML 문서 e:\xml\order1.xml을 분해하고, 이전에 사용할 수 있었던 DAD 파일에서 지정된 맵핑에 따라 SALES_ORDER 콜렉션 테이블로 데이터를 삽입합니다.

```
#include "dxx.h"
#include "dxxrc.h"
       EXEC SQL INCLUDE SQLCA;
        EXEC SQL BEGIN DECLARE SECTION;
         char                collection[64];   /* name of an XML collection */
              SQL TYPE is CLOB_FILE xmlDoc;         /* input XML document */
         long                returnCode;       /* error code */
         char                returnMsg[1024];  /* error message text */
         short               collection_ind;
         short               xmlDoc_ind;
         short               returnCode_ind;
         short               returnMsg_ind;
          EXEC SQL END DECLARE SECTION;

        /* initialize host variable and indicators */
         strcpy(collection,"sales_ord")
              strcpy(xmlobj.name,"c:\dxx\samples\cmd\getstart.xml");
              xmlobj.name_length=strlen("c:\dxx\samples\cmd\getstart.xml");
         xmlobj.file_option=SQL_FILE_READ;
        returnCode = 0;
```

```
 returnMsg[0] = '\0';
 collection_ind = 0;
 xmlobj_ind = 0;
returnCode_ind = -1;
returnMsg_ind = -1;
/* Call the store procedure */
      EXEC SQL CALL db2xml.dxxInsertXML(:collection:collection_ind;
                    :xmlobj:xmlobj_ind,
              :returnCode:returnCode_ind,:returnMsg:returnMsg_ind);
```

제11장 관리 지원 테이블

데이터베이스를 사용할 수 있으면, DTD 참조 테이블 DTD_REF 및 XML_USAGE 테이블이 작성됩니다. DTD_REF 테이블에는 DTD 모두에 관한 정보가 들어갑니다. XML_USAGE 테이블은 XML이 사용할 수 있는 컬럼에 대한 일반 정보를 저장합니다.

지원 테이블에 나열된 매개변수 한계는 317 페이지의 『부록D. XML Extender 한계』에 문서화되어 있습니다.

DTD 참조 테이블

XML Extender가 XML DTD 저장소로서 제공되기도 합니다. 데이터베이스를 사용할 수 있으면, DTD 참조 테이블 DTD_REF가 작성됩니다. 이 테이블의 각 행은 DTD와 함께 추가 메타데이터 정보를 나타냅니다. 사용자는 이 테이블에 액세스하여 자신의 DTD를 삽입할 수 있습니다. DTD_REF 테이블에 있는 DTD는 XML 문서를 확인하고, 응용프로그램이 DAD를 정의하는 데 도움이 되도록 하는 데 사용됩니다. 이것의 스키마 이름은 db2xml입니다. DTD_REF 테이블에는 표 51에 표시된 컬럼이 들어갈 수 있습니다.

표 51. DTD_REF 테이블

컬럼 이름	데이터 유형	설명
DTDID	VARCHAR(128)	기본 키(고유하고 널이 아님). DTD를 식별하는 데 사용됩니다. 기본 키가 DAD 파일에서 지정되면, DAD 파일은 DTD에서 정의된 스키마에 따라야 합니다.
CONTENT	XMLCLOB	DTD 내용.
USAGE_COUNT	INTEGER	DTD를 사용하여 해당 DAD 파일을 정의하는 데이터베이스의 XML 컬럼 및 XML 콜렉션 수.
AUTHOR	VARCHAR(128)	DTD 작성자, 사용자가 입력하는 선택적인 정보.
CREATOR	VARCHAR(128)	처음 삽입하는 사용자 ID. CREATOR 컬럼은 선택적입니다.
UPDATOR	VARCHAR(128)	마지막으로 갱신하는 사용자 ID. UPDATOR 컬럼은 선택적입니다.

제한사항: DTD는 USAGE_COUNT가 0일 경우에만 응용프로그램이 수정할 수 있습니다.

XML 사용 테이블

XML이 사용할 수 있는 각 컬럼에 대한 일반 정보를 저장합니다. XML_USAGE 테이블의 스키마 이름은 db2xml이며, 기본 키는 (*table_name*, *col_name*)입니다. XML_USAGE 테이블은 표52에 나열된 컬럼과 함께 데이터베이스를 사용할 수 있을 때 작성됩니다.

표 52. XML_USAGE 테이블

컬럼 이름	설명
table_schema	XML 컬럼의 경우, XML 컬럼이 들어있는 사용자 테이블의 스키마 이름. XML 콜렉션의 경우, 기본 스키마 이름으로서의 ″DXX_COLL″ 값.
table_name	XML 컬럼의 경우, XML 컬럼이 들어있는 사용자 테이블 이름. XML 콜렉션의 경우, 엔터티를 콜렉션으로서 식별하는 ″DXX_COLLECTION″ 값.
col_name	XML 컬럼이나 XML 콜렉션 이름. table_name과 함께 복합 키의 일부입니다.
DTDID	DTD_REF 테이블에서 DAD 파일을 정의하는 DTD의 ID. 이것이 외부 키입니다.
DAD	컬럼과 연관된 DAD 파일 내용.
default_view	기본 뷰가 있으면 기본 뷰 이름을 저장합니다.
trigger_suffix	널이 아님. 고유한 경우 트리거 이름.
Validation	1은 예, 0은 아니오.
access_mode	XML 콜렉션의 경우 1, XML 컬럼의 경우 0

제한사항: DTD는 USAGE_COUNT가 0일 경우에만 응용프로그램이 수정할 수 있습니다.

제12장 진단 정보

프로그램에 있는 모든 Embedded SQL문과, DB2 XML Extender 사용자 정의 함수(UDF)를 호출하는 호출을 포함하여 DB2 명령 행 인터페이스(CLI) 호출은 Embedded SQL문 또는 DB2 CLI 호출이 성공적으로 실행되었는지 나타내는 코드를 생성합니다.

프로그램에서는 이러한 코드를 보충하는 정보를 검색할 수 있습니다. 여기에는 SQLSTATE 정보와 오류 메시지가 포함됩니다. 이 진단 정보를 사용하여 프로그램의 문제점을 판별하고 수정할 수 있습니다.

때때로 문제점 소스를 쉽게 진단할 수 없을 수도 있습니다. 이러한 경우에는, 소프트웨어 지원센터에 정보를 제공하여 문제점을 진단하고 수정해야 합니다. XML Extender에는 XML Extender 활동을 기록하는 추적 기능이 있습니다. 추적 정보는 IBM 소프트웨어 지원 센터에게는 가치있는 정보가 될 수 있습니다. IBM 지원 센터의 지시하에서만 추적 기능을 사용해야 합니다.

이 장에서는 이 진단 정보에 액세스하는 방법에 대해 설명합니다. 다음에 대해서도 설명합니다.

- XML Extender UDF 리턴 코드 처리법.
- 추적 제어법

XML Extender가 리턴하는 SQLSTATE 코드 및 오류 메시지를 나열하고 이에 대해 설명합니다.

UDF 리턴 코드 처리

SQLCA 구조의 SQLCODE, SQLWARN 및 SQLSTATE 필드에 있는 Embedded SQL문 리턴 코드. 이 구조는 SQLCA INCLUDE 파일에서 정의됩니다(SQLCA 구조 및 SQLCA INCLUDE 파일에 대해서는 *DB2 응용프로그램 개발 안내서*를 참조하십시오).

DB2 CLI 호출은 SQLError 함수를 사용하여 검색할 수 있는 SQLCODE 및 SQLSTATE 값을 리턴합니다(SQLError 함수로 오류 정보를 검색하는 내용에 대해서는 *CLI 안내서 및 참조서*를 참조하십시오).

SQLCODE 값이 0이면 이는 명령문이 성공적으로 실행되었음을 나타냅니다(경고 조건이 있을 수 있음). 양수 SQLCODE 값은 명령문이 성공적으로 실행되었지만 경고가 있었음을 의미합니다(Embedded SQL문은 SQLWARN 필드에 있는 0이나 양수의 SQLCODE 값과 연관된 경고에 대한 정보를 리턴합니다). 음수 SQLCODE 값은 오류가 발생했음을 의미합니다.

DB2는 각각의 SQLCODE 값과 메시지를 연결합니다. XML Extender UDF가 경고나 오류 조건을 발견하면, 연관된 정보를 DB2로 전달하여 SQLCODE 메시지에 포함시킵니다.

SQLSTATE 값에는 SQLCODE 메시지를 보충하는 코드가 들어 있습니다. XML Extender가 리턴하는 SQLSTATE 코드 각각에 대한 설명은 259 페이지의 『SQLSTATE 코드』를 참조하십시오.

DB2 XML Extender UDF를 호출하는 Embedded SQL문 및 DB2 CLI 호출은 이들 UDF에서는 고유한 SQLCODE 메시지 및 SQLSTATE 값을 리턴하지만, DB2는 다른 Embedded SQL문이나 다른 DB2 CLI 호출에서처럼 이들 값을 리턴합니다. 그러므로, 이들 값에 액세스하는 방식은 DB2 XML Extender UDF를 시작하지 않는 Embedded SQL문 또는 DB2 CLI 호출 방식과 같습니다.

XML Extender가 리턴할 수 있는 SQLSTATE 값과 연관 메시지 갯수는 259 페이지의 『SQLSTATE 코드』를 참조하십시오. 각 메시지에 대한 정보는 263 페이지의 『메시지』를 참조하십시오.

저장 프로시듀어 리턴 코드 처리

XML Extender는 저장 프로시듀어로 문제점을 해결하는 데 도움이 되는 리턴 코드를 제공합니다. 저장 프로시듀어로부터 리턴 코드를 받을 때, 다음 파일을 점검하십시오. 이 파일은 XML Extender 오류 메시지 번호 및 기호 상수를 리턴 코드와 일치시킵니다.

```
DXX_INSTALL/include/dxxrc.h
```

263 페이지의 『메시지』에 있는 오류 메시지 번호를 참조하고, 설명에 있는 진단 정보를 사용할 수 있습니다.

SQLSTATE 코드

표53에서는 XML Extender가 리턴한 SQLSTATE 값을 나열하고 이에 대해 설명합니다. SQLSTATE 값 각각에 대한 설명에는 기호 표현도 포함됩니다. 테이블에서는 SQLSTATE 값 각각과 연관된 메시지 번호를 나열합니다. 각 메시지에 대한 정보는 263 페이지의 『메시지』를 참조하십시오.

표 53. SQLSTATE 코드 및 연관 메시지 번호

SQLSTATE	메시지 번호.	설명
00000	DXX*nnnn*I	오류가 발생하지 않았습니다.
01HX0	DXXD003W	경로 표현식에서 지정된 요소 또는 속성이 XML 문서에서 누락되었습니다.
38X00	DXXC000E	XML Extender는 지정된 파일을 열 수 없습니다.
38X01	DXXA072E	XML Extender가 데이터베이스를 사용하기 전에 데이터베이스를 자동으로 바인드하도록 시도했으나 바인드 파일을 찾을 수 없었습니다.
	DXXC001E	XML Extender가 지정된 파일을 찾을 수 없습니다.
38X02	DXXC002E	XML Extender가 지정된 파일에서 데이터를 읽을 수 없습니다.
38X03	DXXC003E	XML Extender가 파일로 데이터를 기록할 수 없습니다.
	DXXC011E	XML Extender가 추적 제어 파일에 데이터를 기록할 수 없습니다.
38X04	DXXC004E	XML Extender가 지정된 위치 지정자를 조작할 수 없습니다.
38X05	DXXC005E	파일 크기가 XMLVarchar 크기보다 크므로, XML Extender가 파일에서 데이터 모두를 가져오기할 수 없습니다.
38X06	DXXC006E	파일 크기가 XMLCLOB 크기보다 크므로, XML Extender가 파일에서 데이터 모두를 가져오기할 수 없습니다.

표 53. SQLSTATE 코드 및 연관 메시지 번호 (계속)

SQLSTATE	메시지 번호.	설명
38X07	DXXC007E	LOB 위치 지정자에 있는 바이트 수가 파일 크기와 같지 않습니다.
38X08	DXXD001E	스칼라 추출 함수가 여러 번 발생하는 위치 경로를 사용했습니다. 스칼라 함수는 여러 번 발생하지 않는 위치 경로만을 사용할 수 있습니다.
38X09	DXXD002E	경로 표현식이 문법적으로 틀렸습니다.
38X10	DXXG002E	XML Extender가 운영 체제에서 메모리를 할당할 수 없습니다.
38X11	DXXA009E	이 저장 프로시듀어는 XML 컬럼만을 위한 것입니다.
38X12	DXXA010E	컬럼을 사용할 수 있도록 설정하는 동안, XML Extender는 DTDID를 찾을 수 없습니다. 이것은 문서 액세스 정의(DAD) 파일에서 DTD에 대해 지정된 식별자입니다.
38X14	DXXD000E	잘못된 문서를 테이블로 저장하려 했습니다. 유효성 검증에 실패했습니다.
38X15	DXXA056E	문서 액세스 정의(DAD) 파일에 있는 유효성 검증 요소가 잘못되었거나 누락되었습니다.
	DXXA057E	문서 액세스 정의(DAD) 파일에 있는 부가 테이블의 이름 속성이 잘못 되었거나 누락되었습니다.
	DXXA058E	문서 액세스 정의(DAD) 파일에 있는 컬럼의 이름 속성이 잘못되었거나 누락되었습니다.
	DXXA059E	문서 액세스 정의(DAD) 파일에 있는 컬럼의 유형 속성이 잘못되었거나 누락되었습니다.
	DXXA060E	문서 액세스 정의(DAD) 파일에 있는 컬럼의 경로 속성이 잘못되었거나 누락되었습니다.
	DXXA061E	문서 액세스 정의(DAD) 파일에 있는 컬럼의 multi_occurrence 속성이 잘못되었거나 누락되었습니다.

표 53. SQLSTATE 코드 및 연관 메시지 번호 (계속)

SQLSTATE	메시지 번호.	설명
	DXXQ000E	기본 요소가 문서 액세스 정의(DAD) 파일에서 누락되었습니다.
38X16	DXXG004E	필수 매개변수의 널 값이 XML 저장 프로시듀어로 전달되었습니다.
38X17	DXXQ001E	문서 액세스 정의(DAD) 파일에 있는 SQL문이나 이를 겹쳐쓰기하는 다른 명령문이 잘못되었습니다. SELECT문은 XML 문서를 생성하는 데 필요합니다.
38X18	DXXG001E	XML Extender가 내부 오류를 발견했습니다.
	DXXG006E	XML Extender가 CLI를 사용하는 중에 내부 오류가 발생했습니다.
38X19	DXXQ002E	시스템을 실행하기에 메모리나 디스크가 부족합니다. 그 결과 XML 문서가 들어갈 공간이 없습니다.
38X20	DXXQ003W	사용자 정의 SQL 조회가 지정된 최대값 이상의 XML 문서를 생성합니다. 지정된 수의 문서만이 리턴됩니다.
38X21	DXXQ004E	지정된 컬럼이 SQL 조회 결과에 있는 컬럼 중 하나가 아닙니다.
38X22	DXXQ005E	XML에 대한 SQL 조회 맵핑이 잘못되었습니다.
38X23	DXXQ006E	문서 액세스 정의(DAD) 파일에 있는 attribute_node 요소에 이름 속성이 없습니다.
38X24	DXXQ007E	문서 액세스 정의(DAD) 파일에 있는 컬럼 요소나 RDB_node가 없습니다.
38X25	DXXQ008E	문서 액세스 정의(DAD) 파일에 있는 text_node 요소에 컬럼 요소가 없습니다.
38X26	DXXQ009E	지정된 결과 테이블을 시스템 카탈로그에서 찾을 수 없습니다.
38X27	DXXQ010E	attribute_node 또는 text_node의 RDB_node에 테이블이 있어야 합니다.
	DXXQ011E	attribute_node 또는 text_node의 RDB_node에 컬럼이 있어야 합니다.

표 53. SQLSTATE 코드 및 연관 메시지 번호 (계속)

SQLSTATE	메시지 번호.	설명
	DXXQ017E	XML Extender가 생성한 XML 문서가 너무 커서 결과 테이블 컬럼에 맞지 않습니다.
38X28	DXXQ012E	XML Extender가 DAD를 처리하는 중에 예상한 요소가 없습니다.
	DXXQ016E	모든 테이블이 문서 액세스 정의(DAD) 파일에 있는 맨 위 요소의 RDB_node에서 정의되어야 합니다. 부속요소 테이블이 맨 위 요소에서 정의된 테이블과 일치해야 합니다. 이 RDB_node에 있는 테이블 이름이 맨 위 요소에 없습니다.
38X29	DXXQ013E	요소 테이블 또는 컬럼의 이름이 문서 액세스 정의(DAD) 파일에 있어야 합니다.
	DXXQ015E	문서 액세스 정의(DAD)의 조건 요소에 있는 조건 형식이 잘못 되었습니다.
38X30	DXXQ014E	문서 액세스 정의(DAD) 파일에 있는 element_node 요소에 이름 속성이 없습니다.
	DXXQ018E	ORDER BY 절이 문서 액세스 정의(DAD) 파일의 SQL문에서 누락되었습니다.
38X31	DXXQ019E	SQL을 XML로 맵핑하는 문서 액세스 정의(DAD) 파일에는 objid 요소의 컬럼 요소가 없습니다.
38X36	DXXA073E	사용자가 데이터베이스를 사용하려고 시도할 때 데이터베이스가 바인드되지 않았습니다.
38X37	DXXG007E	서버 운영 체제 로케일이 DB2 코드 페이지와 일치하지 않습니다.
38X38	DXXG008E	코드 페이지 테이블에 서버 운영 체제 로케일이 없습니다.
38x33	DXXG005E	이 릴리스에서는 이 매개변수가 지원되지 않으므로, 이후 릴리스에서 지원됩니다.
38x34	DXXG000E	잘못된 파일 이름이 지정되었습니다.

메시지

XML Extender는 문제점 판별에 도움이 되는 오류 메시지를 제공합니다.

오류 메시지

XML Extender는 동작을 완료하거나 오류를 발견할 때 다음 메시지를 생성합니다.

DXXA000I <*column_name*> **컬럼을 사용 가능으로 설정 중입니다. 잠시만 기다리십시오.**

설명: 이것은 정보용 메시지입니다.

사용자 응답: 조치가 필요하지 않습니다.

DXXA001S <*build_ID*>, <*file_name*> **파일 그리고** <*line_number*> **행을 빌드하는 동안 예상치 못한 오류가 발생했습니다.**

설명: 예상치 못한 오류가 발생했습니다.

사용자 응답: 이 오류가 계속되면, 소프트웨어 서비스 제공자에게 문의하십시오. 오류를 보고할 때, 모든 메시지 텍스트, 추적 파일 및 문제점이 다시 발생하는 방법에 대한 설명도 함께 보고하십시오.

DXXA002I <*database*> **데이터베이스에 연결 중입니다.**

설명: 이것은 정보용 메시지입니다.

사용자 응답: 조치가 필요하지 않습니다.

DXXA003E <*database*> **데이터베이스에 연결할 수 없습니다.**

설명: 지정된 데이터베이스가 없거나 손상될 수 있습니다.

사용자 응답:

1. 데이터베이스를 정확하게 지정하도록 하십시오.
2. 데이터베이스가 있는지, 액세스할 수 있는지 확인하십시오.
3. 데이터베이스가 손상되었는지 판별하십시오. 데이터베이스가 손상되었으면, 데이터베이스 관리자에게 백업에서 이를 복구하도록 요청하십시오.

DXXA004E <*database*> **데이터베이스를 사용 가능으로 설정할 수 없습니다.**

설명: 데이터베이스가 이미 사용 가능으로 설정되었거나 손상되었습니다.

사용자 응답:

1. 데이터베이스가 사용 가능한지 판별하십시오.
2. 데이터베이스가 손상되었는지 판별하십시오. 데이터베이스가 손상되었으면, 데이터베이스 관리자에게 백업에서 이를 복구하도록 요청하십시오.

DXXA005I <*database*> **데이터베이스를 사용 가능으로 설정하는 중입니다. 잠시만 기다리십시오.**

설명: 이것은 정보용 메시지입니다.

사용자 응답: 조치가 필요하지 않습니다.

DXXA006I *<database>* **데이터베이스가 사용 가능으로 설정되었습니다.**

설명: 이것은 정보용 메시지입니다.

사용자 응답: 조치가 필요하지 않습니다.

DXXA007E *<database>* **데이터베이스를 사용 불능으로 설정할 수 없습니다.**

설명: 데이터베이스에 XML 컬럼이나 콜렉션이 포함되지 않으면 XML Extender가 데이터베이스를 사용 불능으로 설정할 수 없습니다.

사용자 응답: 중요 데이터를 백업하고, 모든 XML 컬럼 또는 콜렉션을 사용 불능으로 설정한 뒤, 데이터베이스에 XML 데이터 유형이 남아있지 않을 때까지 테이블을 갱신하거나 제거하십시오.

DXXA008I *<column_name>* **컬럼을 사용 불능으로 설정 중입니다. 잠시만 기다리십시오.**

설명: 이것은 정보 메시지입니다.

사용자 응답: 조치가 필요하지 않습니다.

DXXA009E Xcolumn 태그가 DAD 파일에서 지정되지 않았습니다.

설명: 이 저장 프로시듀어는 XML 컬럼만을 위한 것입니다.

사용자 응답: Xcolumn 태그가 DAD 파일에서 정확히 지정되었는지 확인하십시오.

DXXA010E DTD ID *<dtdid>*를 **찾지 못했습니다.**

설명: 컬럼을 사용할 수 있도록 설정하는 동안, XML Extender는 DTDID를 찾을 수 없습니다. 이것은 문서 액세스 정의(DAD) 파일에서 DTD에 대해 지정된 식별자입니다.

사용자 응답: DTDID에 대한 정확한 값이 DAD 파일에서 지정되었는지 확인하십시오.

DXXA011E DB2XML.XML_USAGE 테이블에 레코드를 삽입하지 못했습니다.

설명: 컬럼을 사용 가능으로 설정하려고 시도하는 동안, XML Extender는 DB2XML.XML_USAGE 테이블에 레코드를 삽입할 수 없습니다.

사용자 응답: DB2XML.XML_USAGE 테이블에 있는지, 같은 이름의 레코드가 이미 테이블에 없는지 확인하십시오.

DXXA012E DB2XML.DTD_REF 테이블을 갱신하지 못했습니다.

설명: 컬럼을 사용 가능으로 설정하는 동안, XML Extender가 DB2XML.DTD_REF 테이블을 갱신할 수 없습니다.

사용자 응답: DB2XML.DTD_REF 테이블이 있는지 확인하십시오. 테이블이 손상되었는지 또는 관리 사용자 ID에 테이블을 갱신하는 데 올바른 권한이 있는지 판별하십시오.

DXXA013E <*table_name*> **테이블을 변경하지 못했습니다.**

설명: 컬럼을 사용 가능으로 설정하는 동안, XML Extender가 지정된 테이블을 변경할 수 없습니다.

사용자 응답: 테이블을 변경하는 데 필요한 특권을 점검하십시오.

DXXA014E **지정된 루트 ID 컬럼인** <*root_id*> **가** <*table_name*> **테이블의 단일 기본 키가 아닙니다.**

설명: 지정된 루트 ID가 키가 아니거나 *table_name* 테이블의 단일 키가 아닙니다.

사용자 응답: 지정된 루트 ID가 테이블의 단일 기본 키인지 확인하십시오.

DXXA015E **DXXROOT_ID 컬럼이** <*table_name*> **테이블에 이미 있습니다.**

설명: DXXROOT_ID 컬럼이 있지만, XML Extender가 작성하지 않았습니다.

사용자 응답: 컬럼을 사용 가능으로 설정할 때 다른 컬럼 이름을 사용하여 루트 ID 옵션에 대한 기본 컬럼을 지정하십시오.

DXXA016E <*table_name*> **입력 테이블이 없습니다.**

설명: XML Extender가 시스템 카탈로그에서 지정된 테이블을 찾을 수 없습니다.

사용자 응답: 데이터베이스에 테이블이 있는지와, 정확하게 지정되었는지 확인하십시오.

DXXA017E <*column_name*> **입력 컬럼이 지정된 테이블인** <*table_name*>**에 없습니다.**

설명: XML Extender가 시스템 카탈로그에서 컬럼을 찾을 수 없습니다.

사용자 응답: 사용자 테이블에 컬럼이 있는지 확인하십시오.

DXXA018E **지정된 컬럼을 XML 데이터에 사용할 수 없습니다.**

설명: 컬럼을 사용할 수 없게 하는 동안, XML Extender는 DB2XML.XML_USAGE 테이블에서 컬럼을 찾을 수 없습니다. 이는 컬럼이 사용 가능하지 않다는 것을 나타냅니다. XML이 컬럼을 사용할 수 없으면, 이 컬럼을 사용 불능으로 설정할 필요가 없습니다.

사용자 응답: 조치가 필요하지 않습니다.

DXXA019E **컬럼을 사용 가능하게 하는 데 필요한 입력 매개변수가 널입니다.**

설명: enable_column() 저장 프로시듀어에 대한 필수 입력 매개변수가 널입니다.

사용자 응답: enable_column() 저장 프로시듀어에 대한 모든 입력 매개변수를 점검하십시오.

DXXA020E <*table_name*> **테이블에서 컬럼을 찾을 수 없습니다.**

설명: 기본 뷰를 작성하는 동안, XML Extender가 지정된 테이블에서 컬럼을 찾을 수 없습니다.

사용자 응답: 컬럼과 테이블 이름이 제대로 지정되었는지 확인하십시오.

DXXA021E *<default_view>* **기본 뷰를 작성할 수 없습니다.**

설명: 컬럼을 사용 가능으로 설정하는 동안, XML Extender가 지정된 뷰를 작성할 수 없습니다.

사용자 응답: 기본 뷰 이름이 고유한 것인지 확인하십시오. 이 이름의 뷰가 이미 있는 경우, 기본 뷰에 대해 고유한 이름을 지정하십시오.

DXXA022I *<column_name>* **컬럼이 사용 가능으로 설정되었습니다.**

설명: 이것은 정보용 메시지입니다.

사용자 응답: 응답이 필요하지 않습니다.

DXXA023E DAD 파일을 찾을 수 없습니다.

설명: 컬럼을 사용 불능으로 설정하는 동안, XML Extender가 문서 액세스 정의(DAD) 파일을 찾을 수 없습니다.

사용자 응답: 정확한 데이터베이스 이름, 테이블 이름 또는 컬럼 이름을 지정했는지 확인하십시오.

DXXA024E XML Extender가 시스템 카탈로그 테이블에 액세스하는 동안 내부 오류를 발견했습니다.

설명: XML Extender가 시스템 카탈로그에 액세스할 수 없습니다.

사용자 응답: 데이터베이스가 안정된 상태에 있는지 확인하십시오.

DXXA025E *<default_view>* **기본 뷰를 제거할 수 없습니다.**

설명: 컬럼을 사용 불능으로 설정하는 동안, XML Extender가 기본 뷰를 제거할 수 없습니다.

사용자 응답: XML Extender에 대한 관리 사용자 ID가 기본 뷰를 제거하는 데 필요한 특권을 가지고 있는지 확인하십시오.

DXXA026E *<side_table>* **부가 테이블을 제거할 수 없습니다.**

설명: 컬럼을 사용 불능으로 설정하는 동안, XML Extender가 지정된 테이블을 제거할 수 없습니다.

사용자 응답: XML Extender에 대한 관리 사용자 ID가 테이블을 제거하는 데 필요한 특권을 가지고 있는지 확인하십시오.

DXXA027E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA028E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA029E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA030E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA031E 응용프로그램 테이블에 있는 DXXROOT_ID 컬럼 값을 널(NULL)로 재설정할 수 없습니다.

설명: 컬럼을 사용 불능으로 설정하는 동안, XML Extender가 응용프로그램 테이블에서 DXXROOT_ID 값을 NULL로 설정할 수 없습니다.

사용자 응답: XML Extender용 관리 사용자 ID가 응용프로그램 테이블을 변경하는 데 필요한 특권을 가지고 있는지 확인하십시오.

DXXA032E DB2XML.XML_USAGE 테이블에서 USAGE_COUNT를 줄이지 못했습니다.

설명: 컬럼을 사용 불가능화하는 동안, XML Extender가 USAGE_COUNT 컬럼 값을 1만큼 줄일 수 없습니다.

사용자 응답: DB2XML.XML_USAGE 테이블이 있는지, XML Extender용 관리자 사용자 ID가 테이블을 갱신하는 데 필요한 특권을 가지고 있는지 확인하십시오.

DXXA033E DB2XML.XML_USAGE 테이블에서 행을 삭제하지 못했습니다.

설명: 컬럼을 사용 불능으로 설정하는 동안, XML Extender가 DB2XML.XML_USAGE 테이블에서 연관된 행을 삭제할 수 없습니다.

사용자 응답: DB2XML.XML_USAGE 테이블이 있는지, XML Extender에 대한 관리 사용자 ID가 이 테이블을 갱신하는 데 필요한 특권을 가지고 있는지 확인하십시오.

DXXA034I XML Extender가 <*column_name*> 컬럼을 사용 불능으로 설정했습니다.

설명: 이것은 정보용 메시지입니다.

사용자 응답: 조치가 필요하지 않습니다.

DXXA035I XML Extender가 <*database*> 데이터베이스를 사용 불능으로 설정하는 중입니다. 잠시만 기다리십시오.

설명: 이것은 정보용 메시지입니다.

사용자 응답: 아무 조치도 필요하지 않습니다.

DXXA036I XML Extender가 <*database*> 데이터베이스를 사용 불능으로 설정했습니다.

설명: 이것은 정보용 메시지입니다.

사용자 응답: 아무 조치도 필요하지 않습니다.

DXXA037E 지정된 테이블 공간 이름이 18자 이상입니다.

설명: 테이블 공간 이름은 18개의 영숫자 보다 길 수 없습니다.

사용자 응답: 18자 이하의 이름을 지정하십시오.

DXXA038E 지정된 기본 뷰 이름이 18자 이상입니다.

설명: 기본 뷰 이름은 18개의 영숫자 보다 길 수 없습니다.

사용자 응답: 18자 이하의 이름을 지정하십시오.

DXXA039E 지정된 ROOT_ID 이름이 18자 이상입니다.

설명: ROOT_ID 이름은 18개의 영숫자 보다 길 수 없습니다.

사용자 응답: 18자 이하의 이름을 지정하십시오.

DXXA046E <*side_table*> 부가 테이블을 작성할 수 없습니다.

설명: 컬럼을 사용 가능으로 설정하는 동안, XML Extender가 지정된 테이블을 작성할 수 없습니다.

사용자 응답: XML Extender에 대한 관리 사용자 ID가 부가 테이블을 작성하는 데 필요한 특권을 가지고 있는지 확인하십시오.

DXXA047E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA048E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA049E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA050E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA051E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA052E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA053E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA054E 컬럼을 사용 불가능화할 수 없습니다.

설명: 내부 트리거가 실패했으므로 XML Extender는 컬럼을 사용 불가능화할 수 없습니다. 가능한 원인은 다음과 같습니다.

사용자 응답: 추적 파일을 작성하여 문제를 바로잡으려면, 추적 기능을 사용하십시오. 문제가 계속되면, 소프트웨어 서비스 제공자에게 연락하고 추적 파일을 제공하십시오.

DXXA056E DAD 파일에 있는 <*validation_value*> 유효성 검증 값이 올바르지 않습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 유효성 검증 요소가 잘못되었거나 누락되었습니다.

사용자 응답: 유효성 검증 태그가 DAD 파일에서 정확히 지정되었는지 확인하십시오.

DXXA057E DAD에 있는 <*side_table_name*> 부가 테이블 이름이 올바르지 않습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 부가 테이블의 이름 속성이 잘못 되었거나 누락되었습니다.

사용자 응답: 부가 테이블의 이름 속성이 DAD 파일에서 정확히 지정되었는지 확인하십시오.

DXXA058E DAD 파일에 있는 <*column_name*> 컬럼 이름이 올바르지 않습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 컬럼의 이름 속성이 잘못되었거나 누락되었습니다.

사용자 응답: 컬럼의 이름 속성이 DAD 파일에서 정확히 지정되었는지 확인하십시오.

DXXA059E DAD 파일에 있는 <*column_name*> 컬럼의 <*column_type*> 유형이 올바르지 않습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 컬럼의 유형 속성이 잘못되었거나 누락되었습니다.

사용자 응답: 컬럼의 유형 속성이 DAD 파일에서 정확히 지정되었는지 확인하십시오.

DXXA060E DAD 파일에 있는 <*column_name*>의 경로 속성 <*location_path*>이 올바르지 않습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 컬럼의 경로 속성이 잘못되었거나 누락되었습니다.

사용자 응답: 컬럼의 경로 속성이 DAD 파일에 올바로 지정되어 있는지 확인하십시오.

DXXA061E DAD 파일에 있는 <*column_name*>의 <*multi_occurrence*> multi_occurrence 속성이 올바르지 않습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 컬럼의 multi_occurrence 속성이 잘못되었거나 누락되었습니다.

사용자 응답: 컬럼의 multi_occurrence 속성이 DAD 파일에서 정확히 지정되었는지 확인하십시오.

DXXA062E <*table_name*> 테이블에서 <*column_name*>의 컬럼 번호를 검색할 수 없습니다.

설명: XML Extender가 시스템 카탈로그에서 *table_name* 테이블에 있는 *column_name*의 컬럼 이름을 검색할 수 없습니다.

사용자 응답: 응용프로그램 테이블이 잘 정의되었는지 확인하십시오.

DXXA063I <*collection_name*> 콜렉션을 사용 가능으로 설정하는 중입니다. 잠시만 기다리십시오.

설명: 이것은 정보 메시지입니다.

사용자 응답: 조치가 필요하지 않습니다.

DXXA064I <*collection_name*> 콜렉션을 사용 불능으로 설정하는 중입니다. 잠시만 기다리십시오.

설명: 이것은 정보 메시지입니다.

사용자 응답: 조치가 필요하지 않습니다.

DXXA065E 저장 프로시듀어 <*procedure_name*>을 호출하지 못했습니다.

설명: 공유 라이브러리인 db2xml을 점검하고, 사용권한이 정확한지 확인하십시오.

사용자 응답: 클라이언트가 저장 프로시듀어를 실행할 권한을 가지고 있는지 확인하십시오.

DXXA066I XML Extender가 <*collection_name*> 콜렉션을 사용 불능으로 설정했습니다.

설명: 이것은 정보용 메시지입니다.

사용자 응답: 응답이 필요하지 않습니다.

DXXA067I XML Extender가 <*collection_name*> 콜렉션을 사용 가능으로 설정했습니다.

설명: 이것은 정보용 메시지입니다.

사용자 응답: 응답이 필요하지 않습니다.

DXXA068I XML Extender가 추적을 설정했습니다.

설명: 이것은 정보용 메시지입니다.

사용자 응답: 응답이 필요하지 않습니다.

DXXA069I XML Extender가 추적을 해제했습니다.

설명: 이것은 정보용 메시지입니다.

사용자 응답: 응답이 필요하지 않습니다.

DXXA070W 데이터베이스가 이미 사용 가능으로 설정되었습니다.

설명: enable 데이터베이스 명령을 사용할 수 있는 데이터베이스에서 실행했습니다.

사용자 응답: 아무 조치도 필요하지 않습니다.

DXXA071W 데이터베이스가 이미 사용 불능으로 설정되었습니다.

설명: disable 데이터베이스 명령을 사용할 수 없게 된 데이터베이스에서 실행했습니다.

사용자 응답: 아무 조치도 필요하지 않습니다.

DXXA072E XML Extender가 바인드 파일을 찾을 수 없습니다. 데이터베이스를 사용 가능으로 설정하기 전에 이를 바인드하십시오.

설명: XML Extender가 데이터베이스를 사용하기 전에 데이터베이스를 자동으로 바인드하도록 시도했으나 바인드 파일을 찾을 수 없었습니다.

사용자 응답: 데이터베이스를 사용 가능으로 설정하기 전에 이를 바인드하십시오.

DXXA073E 데이터베이스가 바인드되지 않았습니다. 데이터베이스를 사용 가능으로 설정하기 전에 이를 바인드하십시오.

설명: 사용자가 데이터베이스를 사용하려고 시도할 때 데이터베이스가 바인드되지 않았습니다.

사용자 응답: 데이터베이스를 사용 가능으로 설정하기 전에 이를 바인드하십시오.

DXXA074E 잘못된 매개변수 유형입니다. 저장 프로시듀어는 STRING 매개변수를 예상합니다.

설명: 저장 프로시듀어는 STRING 매개변수를 예상합니다.

사용자 응답: 입력 매개변수를 STRING 유형으로 선언하십시오.

DXXA075E 잘못된 매개변수 유형입니다. 입력 매개변수는 LONG 유형이어야 합니다.

설명: 저장 프로시듀어가 입력 매개변수를 LONG 유형으로 예상합니다.

사용자 응답: 입력 매개변수를 LONG 유형으로 선언하십시오.

DXXA076E XML Extender 추적 인스턴스 ID가 유효하지 않습니다.

설명: 제공된 인스턴스 ID로는 추적을 시작할 수 없습니다.

사용자 응답: 인스턴스 ID가 유효한 AS/400 사용자 ID인지 확인하십시오.

DXXC000E 지정된 파일을 열 수 없습니다.

설명: XML Extender는 지정된 파일을 열 수 없습니다.

사용자 응답: 응용프로그램 사용자 ID가 파일에 대해 읽기 및 쓰기 권한을 가지고 있는지 확인하십시오.

DXXC001E 지정된 파일을 찾지 못했습니다.

설명: XML Extender가 지정된 파일을 찾을 수 없습니다.

사용자 응답: 파일이 있는지와 경로가 올바로 지정되어 있는지 확인하십시오.

DXXC002E 파일을 읽을 수 없습니다.

설명: XML Extender가 지정된 파일에서 데이터를 읽을 수 없습니다.

사용자 응답: 응용프로그램 사용자 ID가 파일에 대해 읽기 권한을 가지고 있는지 확인하십시오.

DXXC003E 지정된 파일에 기록할 수 없습니다.

설명: XML Extender가 파일로 데이터를 기록할 수 없습니다.

사용자 응답: 응용프로그램 사용자 ID가 파일에 대해 쓰기 권한을 가지고 있는지 또는 파일 시스템에 공간이 충분한지 확인하십시오.

DXXC004E LOB 위치 지정자를 작동시킬 수 없습니다 : rc=<*locator_rc*>.

설명: XML Extender가 지정된 위치 지정자를 조작할 수 없습니다.

사용자 응답: LOB 위치 지정자가 제대로 설정되었는지 확인하십시오.

DXXC005E 입력 파일 *크기*가 XMLVarchar *크기*보다 큽니다.

설명: 파일 *크기*가 XMLVarchar *크기*보다 크므로, XML Extender가 파일에서 데이터 모두를 가져오기할 수 없습니다.

사용자 응답: XMLCLOB 컬럼 유형을 사용하십시오.

DXXC006E 입력 파일이 DB2 LOB 한계를 초과했습니다.

설명: 파일 *크기*가 XMLCLOB *크기*보다 크므로, XML Extender가 파일에서 데이터 모두를 가져오기할 수 없습니다.

사용자 응답: 파일을 작은 오브젝트로 분해하거나 XML 콜렉션을 사용하십시오.

DXXC007E 파일에서 LOB 위치 지정자로 데이터를 검색할 수 없습니다.

설명: LOB 위치 지정자에 있는 바이트 수가 파일 *크기*와 같지 않습니다.

사용자 응답: LOB 위치 지정자가 제대로 설정되었는지 확인하십시오.

DXXC008E <*file_name*> 파일을 제거할 수 없습니다.

설명: 파일이 공유 액세스가 위반되었거나 계속 열려 있습니다.

사용자 응답: 파일을 닫거나 파일을 보류하고 있는 모든 프로세스를 중지하십시오. DB2를 중지하고 재시작해야 합니다.

DXXC009E <*directory*> 디렉토리로 파일을 작성할 수 없습니다.

설명: XML Extender가 *directory* 디렉토리에서 파일을 작성할 수 없습니다.

사용자 응답: 디렉토리가 있는지와 응용프로그램 사용자 ID에 디렉토리에 대한 쓰기 권한이 있는지, 파일 시스템에 파일 공간이 충분한지 확인하십시오.

DXXC010E <*file_name*> 파일에 기록하는 중에 오류가 발생했습니다.

설명: *file_name* 파일에 기록하는 중에 오류가 발생했습니다.

사용자 응답: 파일 시스템에 파일 공간이 충분한지 확인하십시오.

DXXC011E 추적 제어 파일에 기록할 수 없습니다.

설명: XML Extender가 추적 제어 파일에 데이터를 기록할 수 없습니다.

사용자 응답: 응용프로그램 사용자 ID가 파일에 대해 쓰기 권한을 가지고 있는지 또는 파일 시스템에 공간이 충분한지 확인하십시오.

DXXC012E 임시 파일을 작성할 수 없습니다.

설명: 시스템 임시 디렉토리에 파일을 작성할 수 없습니다.

사용자 응답: 응용프로그램 사용자 ID에 파일 시스템 임시 디렉토리에 대한 쓰기 권한이 있는지 또는 파일 시스템에 파일 공간이 충분한지 확인하십시오.

DXXD000E 올바르지 않은 XML 문서가 거부되었습니다.

설명: 잘못된 문서를 테이블로 저장하려 했습니다. 유효성 검증에 실패했습니다.

사용자 응답: 볼 수 없는 틀린 문자를 볼 수 있는 편집기를 사용하여, 이 DTD를 가진 문서를 점검하십시오. 이 오류가 발생되지 않도록 하려면, DAD 파일에서 유효성 검증을 해제하십시오.

DXXD001E <*location_path*>가 여러 번 발생합니다.

설명: 스칼라 추출 함수가 여러 번 발생하는 위치 경로를 사용했습니다. 스칼라 함수는 여러 번 발생하지 않는 위치 경로만을 사용할 수 있습니다.

사용자 응답: 테이블 함수를 사용하십시오(스칼라 함수 이름의 맨 끝에 's'를 추가하십시오).

DXXD002E 검색 경로에 있는 위치 <*position*> 근처에서 구문 오류가 발생했습니다.

설명: 경로 표현식이 문법적으로 틀렸습니다.

사용자 응답: 조회의 검색 경로 인수를 정정하십시오. 경로 표현식 구문에 대해서는 책자를 참조하십시오.

DXXD003W 경로를 찾지 못했습니다. 널이 리턴됩니다.

설명: 경로 표현식에서 지정된 요소 또는 속성이 XML 문서에서 누락되었습니다.

사용자 응답: 지정된 경로가 맞는지 검증하십시오.

DXXG000E <*file_name*> 파일 이름이 올바르지 않습니다.

설명: 잘못된 파일 이름이 지정되었습니다.

사용자 응답: 정확한 파일 이름을 지정하고 다시 시도하십시오.

DXXG001E <*build_ID*>, <*file_name*> 파일 그리고 <*line_number*> 행을 빌드하는 동안 내부 오류가 발생했습니다.

설명: XML Extender가 내부 오류를 발견했습니다.

사용자 응답: 소프트웨어 서비스 제공자에게 문의하십시오. 오류를 보고할 때, 모든 메시지와 추적 파일 및 오류가 다시 발생하는 방법도 함께 보고하십시오.

DXXG002E 시스템에 메모리가 부족합니다.

설명: XML Extender가 운영 체제에서 메모리를 할당할 수 없습니다.

사용자 응답: 일부 응용프로그램을 닫고 다시 시도하십시오. 그래도 문제점이 계속되면, 도움을 위해 운영 체제 문서를 참조하십시오. 몇몇 운영 체제에서는 사용자가 시스템을 재부트하여 문제점을 정정해야 하는 경우도 있습니다.

DXXG004E 널 매개변수가 올바르지 않습니다.

설명: 필수 매개변수의 널 값이 XML 저장 프로시듀어로 전달되었습니다.

사용자 응답: 저장 프로시듀어 호출에 대한 인수 목록에서 모든 필수 매개변수를 점검하십시오.

DXXG005E 매개변수가 지원되지 않습니다.

설명: 이 릴리스에서는 이 매개변수가 지원되지 않으므로, 이후 릴리스에서 지원됩니다.

사용자 응답: 이 매개변수를 NULL로 설정하십시오.

DXXG006E 내부 오류
CLISTATE=<*clistate*>**,**
RC=<*cli_rc*>**, build** <*build_ID*>**,**
file <*file_name*>**, line**
<*line_number*>
CLIMSG=<*CLI_msg*>**.**

설명: XML Extender가 CLI를 사용하는 중에 내부 오류가 발생했습니다.

사용자 응답: 소프트웨어 서비스 제공자에게 문의하십시오. 이 오류는 올바르지 않은 사용자 입력으로 인해 발생할 수 있습니다. 오류를 보고할 때, 모든 출력 메시지와 추적 로그 및 문제가 다시 발생되는 방법도 함께 보고하십시오. 가능하다면, 적용되는 DAD, XML 문서 및 테이블 정의를 보내십시오.

DXXG007E <*locale*> **로케일이 DB2 코드페이지** <*code_page*>**와 일치하지 않습니다.**

설명: 서버 운영 체제 로케일이 DB2 코드 페이지와 일치하지 않습니다.

사용자 응답: 서버 운영 체제 로케일을 수정한 뒤 DB2를 재시작하십시오.

DXXG008E <*locale*> **로케일은 지원되지 않습니다.**

설명: 코드 페이지 테이블에 서버 운영 체제 로케일이 없습니다.

사용자 응답: 서버 운영 체제 로케일을 수정한 뒤 DB2를 재시작하십시오.

DXXQ000E <요소>**가 DAD 파일에서 누락되었습니다.**

설명: 기본 요소가 문서 액세스 정의(DAD) 파일에서 누락되었습니다.

사용자 응답: 누락된 요소를 DAD 파일로 추가하십시오.

DXXQ001E XML 생성을 위한 SQL문이 올바르지 않습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 SQL 문이나 이를 겹쳐쓰기하는 다른 명령문이 잘못되었

습니다. SELECT문은 XML 문서를 생성하는 데 필요합니다.

사용자 응답: SQL문을 정정하십시오.

DXXQ002E XML 문서를 보관할 저장영역 공간을 생성할 수 없습니다.

설명: 시스템을 실행하기에 메모리나 디스크가 부족합니다. 그 결과 XML 문서가 들어갈 공간이 없습니다.

사용자 응답: 생성될 문서 수를 제한하십시오. 필요없는 요소와 속성 노드를 문서 액세스 정의(DAD) 파일에서 제거하여 각 문서의 크기를 줄이십시오.

DXXQ003W 결과가 최대값을 초과합니다.

설명: 사용자 정의 SQL 조회가 지정된 최대값 이상의 XML 문서를 생성합니다. 지정된 수의 문서만이 리턴됩니다.

사용자 응답: 아무 조치도 필요하지 않습니다. 문서 모두가 필요하면, 문서의 최대값으로 0을 지정하십시오.

DXXQ004E <*column_name*> 컬럼이 조회 결과에 없습니다.

설명: 지정된 컬럼이 SQL 조회 결과에 있는 컬럼 중 하나가 아닙니다.

사용자 응답: 문서 액세스 정의(DAD) 파일에서 지정된 컬럼 이름을 변경하여, SQL 조회 결과에서 컬럼 중 하나가 되도록 하십시오. 또는, 결과에 지정된 컬럼이 포함될 수 있도록 SQL 조회를 변경하십시오.

DXXQ004W DTD ID가 DAD에 없습니다.

설명: DAD에서, VALIDATION이 YES이지만 DTDID 요소가 지정되지 않았습니다. 어떤 유효성 검증도 수행되지 않습니다.

사용자 응답: 아무 조치도 필요하지 않습니다. 유효성 검증이 필요하면, DAD 파일에서 DTDID 요소를 지정하십시오.

DXXQ005E 관계형 맵핑이 잘못되었습니다. <*element_name*> 요소가 이것의 하위 컬럼인 <*column_name*>보다 하위 레벨에 있습니다.

설명: XML에 대한 SQL 조회 맵핑이 잘못되었습니다.

사용자 응답: SQL 조회 결과에 있는 컬럼들이 관계형 계층에서 하향 순서로 되도록 하십시오. 또한 단일 컬럼 후보 키가 각 레벨을 시작하도록 하십시오. 그러한 키를 테이블에서 사용할 수 없으면, 조회에서는 테이블 표현식 및 DB2 내장 함수 generate_unique()를 사용하여 이 테이블에 대해 키를 생성해야 합니다.

DXXQ006E attribute_node 요소에 이름이 없습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 attribute_node 요소에 이름 속성이 없습니다.

사용자 응답: DAD 파일에 있는 모든 attribute_node에 이름이 지정되도록 하십시오.

DXXQ007E attribute_node <*attribute_name*> 에 컬럼 요소나 RDB_node가 없습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 컬럼 요소나 RDB_node가 없습니다.

사용자 응답: DAD 파일에 있는 모든 attribute_node에 컬럼 요소나 RDB_node가 포함되도록 하십시오.

DXXQ008E text_node 요소에 컬럼 요소가 없습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 text_node 요소에 컬럼 요소가 없습니다.

사용자 응답: DAD에 있는 모든 text_node에 컬럼 요소가 포함되도록 하십시오.

DXXQ009E <*table_name*> 결과 테이블이 없습니다.

설명: 지정된 결과 테이블을 시스템 카탈로그에서 찾을 수 없습니다.

사용자 응답: 저장 프로시듀어를 호출하기 전에 결과 테이블을 작성하십시오.

DXXQ010E DAD 파일에 있는 <*node_name*> RDB_node에 테이블이 없습니다.

설명: attribute_node 또는 text_node의 RDB_node에 테이블이 있어야 합니다.

사용자 응답: 문서 액세스 정의(DAD) 파일에서 attribute_node 또는 text_node에 대한 RDB_node 테이블을 지정하십시오.

DXXQ011E DAD 파일에서 <*node_name*> RDB_node에 컬럼이 없습니다.

설명: attribute_node 또는 text_node의 RDB_node에 컬럼이 있어야 합니다.

사용자 응답: 문서 액세스 정의(DAD) 파일에서 attribute_node 또는 text_node에 대한 RDB_node 컬럼을 지정하십시오.

DXXQ012E DAD에 오류가 발생했습니다.

설명: XML Extender가 DAD를 처리하는 중에 예상한 요소가 없습니다.

사용자 응답: DAD가 유효한 XML 문서인지와 DAD DTD에 필요로 하는 모든 요소를 포함하고 있는지 점검하십시오. DAD DTD에 대해서는 XML Extender 책자를 참조하십시오.

DXXQ013E DAD 파일에 있는 테이블 또는 컬럼 요소에 이름이 없습니다.

설명: 요소 테이블 또는 컬럼의 이름이 문서 액세스 정의(DAD) 파일에 있어야 합니다.

사용자 응답: DAD에서 테이블 또는 컬럼 요소의 이름을 지정하십시오.

DXXQ014E element_node 요소에 이름이 없습니다.

설명: 문서 액세스 정의(DAD) 파일에 있는 element_node 요소에 이름 속성이 없습니다.

사용자 응답: DAD 파일에 있는
모든 element_node 요소에 이름이 지정되도록 하십시오.

DXXQ015E 조건 형식이 올바르지 않습니다.

설명: 문서 액세스 정의(DAD)의 조건 요소에 있는 조건 형식이 잘못 되었습니다.

사용자 응답: 조건 형식이 맞는지 확인하십시오.

DXXQ016E 이 RDB_node에 있는 테이블 이름이 DAD 파일의 맨 위 요소에서 정의되지 않았습니다.

설명: 모든 테이블이 문서 액세스 정의(DAD) 파일에 있는 맨 위 요소의 RDB_node에서 정의되어야 합니다. 부속요소 테이블이 맨 위 요소에서 정의된 테이블과 일치해야 합니다. 이 RDB_node에 있는 테이블 이름이 맨 위 요소에 없습니다.

사용자 응답: RDB 노드의 테이블이 DAD 파일의 맨 위 요소에서 정의되었는지 확인하십시오.

DXXQ017E <*table_name*> 결과 테이블에 있는 컬럼이 너무 작습니다.

설명: XML Extender가 생성한 XML 문서가 너무 커서 결과 테이블 컬럼에 맞지 않습니다.

사용자 응답: 결과 테이블을 제거하십시오. 좀 더 큰 컬럼이 있는 다른 결과 테이블을 작성하십시오. 저장 프로시듀어를 다시 실행하십시오.

DXXQ018E ORDER BY 절이 SQL문에서 누락되었습니다.

설명: ORDER BY 절이 문서 액세스 정의(DAD) 파일의 SQL문에서 누락되었습니다.

사용자 응답: DAD 파일을 편집하십시오. 엔터티 식별 컬럼이 포함된 ORDER BY 절을 추가하십시오.

DXXQ019E DAD 파일에는 objid 요소의 컬럼 요소가 없습니다.

설명: SQL을 XML로 맵핑하는 문서 액세스 정의(DAD) 파일에는 objid 요소의 컬럼 요소가 없습니다.

사용자 응답: DAD 파일을 편집하십시오. 키 컬럼을 objid 요소의 하위 요소로 추가하십시오.

DXXQ020I XML이 생성되었습니다.

설명: 요청된 XML 문서가 데이터베이스에서 생성되었습니다.

사용자 응답: 아무 조치도 필요하지 않습니다.

DXXQ021E 테이블 <*table_name*>에 컬럼 <*column_name*>이 없습니다.

설명: 데이터베이스에 있는 테이블에 지정된 컬럼이 없습니다.

사용자 응답: DAD에서 다른 컬럼 이름을 지정하거나, 지정된 컬럼을 테이블 데이터베이스로 추가하십시오.

DXXQ022E <*table_name*>의 컬럼 <*column_name*>은 유형 <*type_name*>을 가져야 합니다.

설명: 컬럼 유형이 틀립니다.

사용자 응답: 문서 액세스 정의(DAD)에서 컬럼 유형을 정정하십시오.

DXXQ023E <*table__name*>**의 컬럼** <*table_name*>**은** <*length*>**보다 길 수 없습니다.**

설명: DAD에서 컬럼에 대해 정의된 길이가 너무 깁니다.

사용자 응답: DAD에 있는 컬럼 길이를 수정하십시오.

DXXQ024E 테이블 <*table_name*>**을 작성할 수 없습니다.**

설명: 지정된 테이블을 작성할 수 없습니다.

사용자 응답: 테이블을 작성 중인 사용자 ID가 데이터베이스에서 테이블을 작성할 수 있는 권한을 가지고 있는지 확인하십시오.

DXXQ025I XML이 분해되었습니다.

설명: XML 문서가 분해되어 콜렉션에 저장되었습니다.

사용자 응답: 아무 조치도 필요하지 않습니다.

DXXQ026E XML 데이터 <*xml_name*>**가 너무 커서 컬럼** <*column_name*>**에 맞지 않습니다.**

설명: XML 문서의 지정된 데이터 조작이 너무 커서 지정된 컬럼에 맞지 않습니다.

사용자 응답: ALTER TABLE문을 사용하여 컬럼의 길이를 늘리거나, XML 문서를 편집하여 데이터의 크기를 줄이십시오.

DXXQ028E XML_USAGE 테이블에 콜렉션 <*collection_name*>**이 없습니다.**

설명: 콜렉션에 대한 레코드가 XML_USAGE 테이블에 없습니다.

사용자 응답: 콜렉션이 사용 가능한지 확인하십시오.

DXXQ029E 콜렉션 <*collection_name*>**에 대한 DAD가 XML_USAGE 테이블에 없습니다.**

설명: 콜렉션에 대한 DAD 레코드가 XML_USAGE 테이블에 없습니다.

사용자 응답: 콜렉션이 올바로 사용 가능한지 확인하십시오.

DXXQ030E 올바르지 않은 XML 겹쳐쓰기 구문.

설명: XML_override 값이 저장 프로시듀어에서 잘못 지정되었습니다.

사용자 응답: XML_override의 구문이 올바른지 확인하십시오.

DXXQ031E 테이블 이름은 DB2에서 허용하는 최대 길이를 초과할 수 없습니다.

설명: DAD에 있는 조건 요소로 지정된 테이블 이름이 너무 깁니다.

사용자 응답: DAD에 있는 테이블 이름의 길이를 정정하십시오.

DXXQ032E 컬럼 이름은 DB2에서 허용하는 최대 길이를 초과할 수 없습니다.

설명: DAD에 있는 조건 요소로 지정된 컬럼 이름이 너무 깁니다.

사용자 응답: DAD에 있는 컬럼 이름의 길이를 정정하십시오.

DXXQ033E <*identifier*>에서 올바르지 않은 식별자가 시작합니다.

설명: 이 문자열은 올바른 DB2 SQL 식별자가 아닙니다.

사용자 응답: DAD에 있는 문자열을 정정하여 DB2 SQL 식별자에 대한 규칙을 따르십시오.

DXXQ034E DAD의 최상위 RDB_node의 조건 요소 <*condition*>이 유효하지 않습니다.

설명: 조건 요소는 결합 AND로 연결되는 조인 조건으로 구성되는 올바른 WHERE 절이어야 합니다.

사용자 응답: DAD에서의 조인 조건의 올바른 구문에 대해서는 XML Extender 문서를 참조하십시오.

DXXQ035E DAD의 최상위 RDB_node의 조인 조건 <*condition*>이 올바르지 않습니다.

설명: DAD가 여러 테이블을 지정할 경우, 맨 위 RDB_node의 조건 요소에 있는 컬럼 이름은 테이블 이름으로 규정되어야 합니다.

사용자 응답: DAD에서의 조인 조건의 올바른 구문에 대해서는 XML Extender 문서를 참조하십시오.

DXXQ036E DAD 조건 태그에서 지정된 스키마 이름이 허용된 것보다 깁니다.

설명: DAD 내의 조건 태그에서 텍스트를 구문 분석하는 중에 오류가 검출되었습니다. 너무 긴 스키마 이름에 의해 규정된 ID가 조건 텍스트에 포함되어 있습니다.

사용자 응답: 문서 액세스 정의(DAD)에서 조건 태그의 텍스트를 정정하십시오.

DXXQ037E 다중 발생하는 <*element*>를 생성할 수 없습니다.

설명: 요소 노드 및 그 하위 구성원들은 데이터베이스에 맵핑되지 않으나, 그 multi_occurrence는 YES로 설정됩니다.

사용자 응답: multi_occurrence를 NO로 설정하거나 그 하위 구성원에서 RDB-node를 작성하여 DAD를 정정하십시오.

진단 추적

XML Extender에는 XML Extender 서버 활동을 기록하는 추적 기능이 포함됩니다. IBM 소프트웨어 지원 센터의 지시하에서만 추적 기능을 사용해야 합니다.

추적 기능은 서버 파일에서 XML Extender 구성요소 시작 또는 종료, XML Extender 구성요소의 오류 코드 리턴과 같은 여러 이벤트 정보를 기록합니다. 많은 이벤트 정보를 기록하므로, 추적 기능은 필요할 때, 예를 들면 오류 조건을 조사할 때와 같은 경우에만 사용되어야 합니다. 뿐만 아니라, 추적 기능을 사용할 때 활동중인 응용프로그램의 수를 제한해야 합니다. 활동 중인 응용프로그램의 수를 제한하면 문제점의 원인을 쉽게 판별할 수 있습니다.

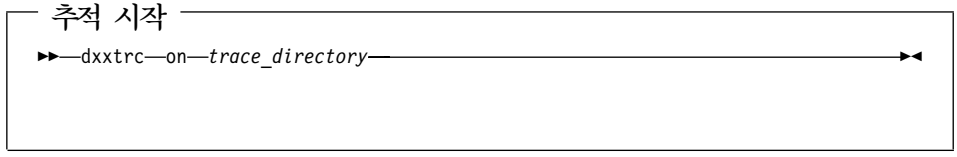
dxxtrc 명령을 사용하여 추적을 시작하거나 중지할 수 있습니다. AIX, Windows NT 또는 Solaris 서버의 명령행에서 명령을 실행할 수 있습니다. 명령을 실행하려면 SYSADM, SYSCTRL 또는 SYSMINT 권한이 있어야 합니다.

추적 시작

목적

XML Extender 서버 활동을 기록합니다. 추적을 시작하려면, 추적 파일을 포함하는 디렉토리 이름과 함께 on 옵션을 **dxxtrc**에 적용하십시오. 추적이 시작되면 dxx*INSTANCE*.trc 파일이 지정된 디렉토리에 위치하게 됩니다. *INSTANCE*는 DB2INSTANCE 값입니다. 각 DB2 인스턴스는 고유한 로그 파일을 가집니다.

형식



매개변수

표 54. 추적 매개변수

매개변수	설명
trace_directory	dxx*INSTANCE*.trc가 대체되는 디렉토리의 이름. 필수, 기본값 없음.

예

다음의 예는 AIX에서 db2inst1 인스턴스에 대한 추적 시작을 보여줍니다. 추적 파일 dxxdb2inst1.trc는 /home/db2inst1/sqllib/log 디렉토리에 위치하게 됩니다.

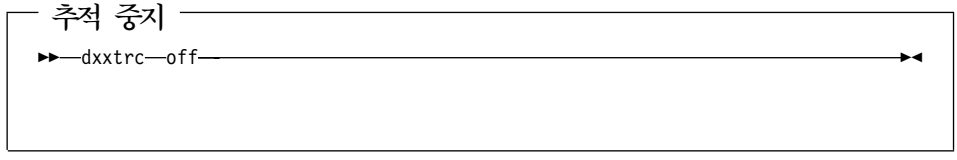
```
dxxtrc on /home/db2inst1/sqllib/log
```

추적 중지

목적

추적을 해제합니다. 아무 추적 정보도 기록되지 않습니다. 추적을 실행하면 성능에 영향을 줄 수 있으므로, 제품 환경에서 추적을 해제하는 것이 좋습니다.

형식



예

이 예에서는 추적이 해제됨을 보여줍니다.

```
dxxtrc off
```

제5부 부록 및 끝머리

부록A. DAD 파일의 DTD

이 절에서는 문서 액세스 정의(DAD) 파일의 문서 유형 선언(DTD)에 대해 설명합니다. DAD 파일 자체는 트리 구조로 된 XML 문서이며, DTD가 필요합니다. DTD 파일 이름은 `dxxdad.dtd`이며, 288 페이지의 그림13에서는 DAD 파일의 DTD를 보여줍니다. 이 파일의 요소에 대해서는 다음 그림에서 설명합니다.

```
<?xml encoding="US-ASCII"?>
  <!ELEMENT DAD (dtdid?, validation, (Xcolumn | Xcollection))>
  <!ELEMENT dtdid (#PCDATA)>
  <!ELEMENT validation (#PCDATA)>
  <!ELEMENT Xcolumn (table*)>
  <!ELEMENT table (column*)>
 <!ATTLIST table name CDATA #REQUIRED
                              key CDATA #IMPLIED
                              orderBy CDATA #IMPLIED>
  <!ELEMENT column EMPTY>
<!ATTLIST column
                     name CDATA #REQUIRED
                     type  CDATA #IMPLIED
                     path  CDATA #IMPLIED
                     multi_occurrence CDATA #IMPLIED>
  <!ELEMENT Xcollection (SQL_stmt?, objids?, prolog, doctype, root_node)>
  <!ELEMENT SQL_stmt (#PCDATA)>
  <!ELEMENT objids (column+)>
  <!ELEMENT prolog (#PCDATA)>
  <!ELEMENT doctype (#PCDATA | RDB_node)*>
  <!ELEMENT root_node (element_node)>
  <!ELEMENT element_node (RDB_node*,
                                        attribute_node*,
                                        text_node?,
                                        element_node*,
                                        namespace_node*,
                                        process_instruction_node*,
                                        comment_node*)>
  <!ATTLIST element_node
                     name CDATA #REQUIRED
                      ID CDATA #IMPLIED
                     multi_occurrence CDATA "NO"
                     BASE_URI CDATA #IMPLIED>
  <!ELEMENT attribute_node (column | RDB_node)>
  <!ATTLIST attribute_node
                     name CDATA #REQUIRED>
  <!ELEMENT text_node (column | RDB_node)>
  <!ELEMENT RDB_node (table+, column?, condition?)>
  <!ELEMENT condition (#PCDATA)>
  <!ELEMENT comment_node (#PCDATA)>
  <!ELEMENT namespace_node (EMPTY)>
  <!ATTLIST namespace_node
                      name CDATA #IMPLIED
                     value CDATA #IMPLIED>
  <!ELEMENT process_instruction_node (#PCDATA)>
```

그림 13. 문서 액세스 정의(DAD)의 DTD

DAD 파일에는 4개의 주요 요소가 있습니다.

- DTDID
- 유효성 검증
- Xcolumn

- Xcollection

Xcolumn 및 Xcollection에는 XML 데이터를 DB2에 있는 관계형 테이블로 맵핑하는 데 도움이 되는 하위 요소 및 속성이 있습니다. 다음 목록에서는 주요 요소 및 이들의 요소와 속성에 대해 설명합니다. 구문 예는 288 페이지의 그림13에 있습니다.

DTDID 요소

DTD_REF 테이블에 저장된 DTD의 ID를 지정합니다. DTDID는 XML 문서에 대해 유효성 검증을 수행하거나 XML 콜렉션 테이블과 XML 문서간의 맵핑을 안내하는 DTD를 지적합니다. DTDID는 XML 콜렉션에 대해 지정되어야 합니다. XML 컬럼의 경우, 이것은 선택적이며, 요소나 속성에 대한 색인화를 위해 부가 테이블을 작성하거나 입력 XML 문서를 검증해야 하는 경우에만 필요합니다. DTDID는 XML 문서의 doctype에서 지정된 SYSTEM ID와 같아야 합니다.

구문: `<!ELEMENT dtdid (#PCDATA)>`

유효성 검증 요소

XML 문서가 DAD에 대한 DTD로 유효성이 검증되었는지 나타냅니다. YES가 지정되면, DTDID도 지정되어야 합니다.

구문: `<!ELEMENT validation(#PCDATA)>`

Xcolumn 요소

XML 컬럼의 색인화 스킴을 정의합니다. 0개 또는 그 이상의 테이블로 구성됩니다.

구문: `<!ELEMENT Xcolumn (table*)>` Xcolumn에는 하나의 하위 요소 즉 테이블이 있습니다.

테이블 요소

XML 문서에 저장된 문서의 요소 또는 속성을 색인화하기 위해 작성된 하나 이상의 관계형 테이블을 정의합니다.

구문:

```
<!ELEMENT table (column+)>
 <!ATTLIST table name CDATA #REQUIRED
key CDATA #IMPLIED
orderBy CDATA #IMPLIED>
```

테이블 요소에는 하나의 속성이 있습니다.

이름 속성

부가 테이블의 이름을 지정합니다.

테이블 요소에는 하나의 하위 요소가 있습니다.

키 속성 테이블의 기본 단일 키.

orderBy 속성

XML 문서를 생성할 때 여러 번 발생하는 요소 텍스트나 속성값의 순서를 판별하는 컬럼 이름.

컬럼 요소

지정된 유형의 위치 경로 값이 들어있는 테이블 컬럼을 지정합니다.

구문:

```
<!ATTLIST column
                      name CDATA #REQUIRED
                      type  CDATA #IMPLIED
                      path  CDATA #IMPLIED
                      multi_occurrence CDATA #IMPLIED>
```

컬럼 요소에는 4개의 속성이 있습니다.

이름 속성

컬럼 이름을 지정합니다. 요소나 속성을 식별하는 위치 경로의 별명 이름입니다.

유형 속성

컬럼의 데이터 유형을 정의합니다. SQL 데이터 유형이 될 수 있습니다.

경로 속성

XML 요소나 속성의 위치 경로를 보여주며, 3.1.a(수정 링크)에서 지정된 것처럼 단순 위치 경로여야 합니다.

multi_occurrence 속성

이 요소나 속성이 XML 문서에서 한번 이상 사용될 수 있는지 여부를 나타냅니다. 값은 예나 아니오가 될 수 있습니다.

Xcollection

XML 문서와, 관계형 테이블의 XML 콜렉션 사이의 맵핑을 정의합니다.

구문: `<!ELEMENT Xcollection(SQL_stmt*, prolog, doctype, root_node)>`
Xcollection에는 다음의 하위 요소가 있습니다.

SQL_stmt

XML Extender가 콜렉션을 정의하는 데 사용하는 SQL문을 지정합니다. 특히, 명령문은 XML 콜렉션 테이블에서 XML 데이터를 선택하고, 데이터를 사용하여 콜렉션에서 XML 문서를 생성합니다. 이 요소 값은 유효한 SQL문이어야 합니다. 이것은 작성시에만 사용되며, 하나의 SQL_stmt만이 허용됩니다. 분해의 경우, 필요한 테이블 작성 및 삽입을 수행하기 위해 SQL_stmt에 대해 하나 이상의 값이 지정될 수 있습니다.

구문: `<!ELEMENT SQL_stmt  #PCDATA >`

prolog XML 프롤로그 텍스트. 전체 콜렉션에 있는 모든 문서에 대해 같은 프롤로그가 지원됩니다. prolog 값은 고정됩니다.

구문: `<!ELEMENT prolog  #PCDATA>`

doctype

XML 문서 유형을 정의하기 위한 텍스트를 정의합니다.

구문: `<!ELEMENT doctype  #PCDATA | RDB_node>` doctype은 다음 방식중 하나로 지정될 수 있습니다.

- 명시적 값 정의. 이 값은 전체 콜렉션에 있는 모든 문서에 지원됩니다.
- 분해를 사용할 경우, 하위 요소인 RDB_node를 지정하십시오. 이것은 테이블의 컬럼 데이터로 맵핑되고 저장될 수 있습니다.

doctype에는 하나의 하위 요소가 있습니다.

RDB_node

아직 구현되지 않습니다.

root_node

가상 루트 노드를 정의합니다. root_node에는 하나의 필수 하위 요소인 element_node가 있어야 하며, 이것은 한번만 사용될 수 있습니다. root_node 아래에 있는 element_node는 실제로 XML 문서의 root_node입니다.

구문: `<!ELEMENT root_node(element_node)>`

element_node

XML 요소를 나타냅니다. 콜렉션에 대해 지정된 DTD에서 정의되어야 합니다. RDB_node 맵핑의 경우, element_node 루트에는 자체 및 하위

노드 모두에 대한 XML 데이터가 포함된 모든 테이블을 지정하는 RDB_node가 있어야 합니다. 0개 또는 그 이상의 attribute_node 및 하위 element_node가 있을 수 있으며, 0개 또는 1개의 text_node가 있을 수 있습니다. 루트 요소가 아닌 요소의 경우, RDB_node는 필요하지 않습니다.

구문:

element_node는 다음 하위 요소에서 정의됩니다.

RDB_node

(선택적) XML 데이터에 대한 테이블, 컬럼 및 조건을 지정합니다. 요소에 대한 RDB_node는 RDB_node 맵핑에 정의되어야 합니다. 이런 경우, 하나 이상의 테이블이 지정되어야 합니다. 요소 내용은 이것의 text_node에서 지정되므로 컬럼이 필요하지 않습니다. 조건은 DTD 및 조회 조건에 따라 선택적입니다.

하위 노드

(선택적) element_node에는 다음 하위 노드가 있을 수 있습니다.

element_node

현재 XML 요소의 하위 요소를 나타냅니다.

attribute_node

현재 XML 요소의 속성들을 나타냅니다.

text_node

현재 XML 요소의 CDATA 텍스트를 나타냅니다.

attribute_node

XML 속성을 나타냅니다. XML 속성과 관계형 테이블에 있는 컬럼 데이터 사이의 맵핑을 정의하는 노드입니다.

구문:

attribute_node에는 name 속성, column 또는 RDB_node 하위 요소에 대한 정의가 있어야 합니다. attribute_node에는 다음 속성이 있습니다.

이름 속성 이름.

attribute_node에는 다음 하위 요소가 있습니다.

컬럼 SQL 맵핑에 사용됩니다. 컬럼은 SQL_stmt의 SELECT 절에서 지정되어야 합니다.

RDB_node

RDB_node 맵핑에 사용됩니다. 노드는 이 속성과 관계형 테이블에 있는 컬럼 데이터간의 맵핑을 정의하며, 테이블과 컬럼이 지정되어야 합니다. 조건은 선택적입니다.

text_node

XML 요소의 텍스트 내용을 나타냅니다. XML 요소 내용과, 관계형 테이블에 있는 컬럼 데이터간의 맵핑을 정의하는 노드입니다.

구문: 컬럼 또는 RDB_node 하위 요소에서 정의되어야 합니다.

컬럼 SQL 맵핑에 필요합니다. 이 경우, 컬럼은 SQL_stmt의 SELECT 절에 있어야 합니다.

RDB_node

RDB_node 맵핑에 필요합니다. 노드는 이 텍스트 내용과 관계형 테이블에 있는 컬럼 데이터간의 맵핑을 정의합니다. 테이블 및 컬럼이 지정되어야 합니다. 조건은 선택적입니다.

부록B. 샘플

이 부록에서는 이 책에 있는 예와 함께 사용되는 샘플 오브젝트를 보여줍니다.

- 『XML DTD』
- 296 페이지의 『XML 문서: getstart.xml』
- 297 페이지의 『문서 액세스 정의 파일』
  - 297 페이지의 『DAD 파일: XML 컬럼』
  - 299 페이지의 『DAD 파일: XML 콜렉션 - SQL 맵핑』
  - 301 페이지의 『DAD 파일: XML - RDB_node 맵핑』

XML DTD

다음 DTD는 이 책 전반에 참조되며 296 페이지의 그림15에 있는 `getstart.xml` 문서에 사용됩니다.

```
<!xml encoding="US-ASCII"?>
<!ELEMENT Order (Customer, Part+)>
<!ATTLIST Order key CDATA #REQUIRED>
<!ELEMENT Customer (Name, Email)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Email (#PCDATA)>
<!ELEMENT Part (key,Quantity,ExtendedPrice,Tax, Shipment+)>
<!ELEMENT key (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT ExtendedPrice (#PCDATA)>
<!ELEMENT Tax (#PCDATA)>
<!ATTLIST Part color CDATA #REQUIRED>
<!ELEMENT Shipment (ShipDate, ShipMode)>
<!ELEMENT ShipDate (#PCDATA)>
<!ELEMENT ShipMode (#PCDATA)>
```

그림 14. 샘플 XML DTD: getstart.dtd

XML 문서: getstart.xml

다음 XML 문서 getstart.xml은 이 책 전반의 예에서 사용되는 샘플 XML 문서입니다. 여기에는 구문 순서를 구성하는 XML 태그가 들어 있습니다.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd">
    <Order key="1">
  <Customer>
    <Name>American Motors</Name>
    <Email>parts@am.com</Email>
  </Customer>
  <Part color="black ">
    <key>68</key>
    <Quantity>36</Quantity>
    <ExtendedPrice>34850.16</ExtendedPrice>
    <Tax>6.000000e-02</Tax>
          <Shipment>
      <ShipDate>1998-08-19</ShipDate>
      <ShipMode>BOAT  </ShipMode>
          </Shipment>
          <Shipment>
      <ShipDate>1998-08-19</ShipDate>
       <ShipMode>AIR   </ShipMode>
          </Shipment>
  </Part>
  <Part color="red   ">
    <key>128</key>
    <Quantity>28</Quantity>
    <ExtendedPrice>38000.00</ExtendedPrice>
    <Tax>7.000000e-02</Tax>
          <Shipment>
      <ShipDate>1998-12-30</ShipDate>
      <ShipMode>TRUCK </ShipMode>
          </Shipment>
  </Part>
</Order>
```

그림 15. 샘플 XML 문서: getstart.xml

문서 액세스 정의 파일

다음 절에는 XML 컬럼 또는 XML 콜렉션 액세스 모드를 사용하여 XML 데이터를 DB2 관계형 테이블로 맵핑하는 문서 액세스 정의(DAD) 파일이 들어 있습니다.

- 『DAD 파일: XML 컬럼』
- 299 페이지의 『DAD 파일: XML 콜렉션 - SQL 맵핑』에서는 SQL 맵핑을 사용하는 XML 콜렉션에 대한 DAD 파일을 보여줍니다.
- 301 페이지의 『DAD 파일: XML - RDB_node 맵핑』에서는 RDB_node 맵핑을 사용하는 XML 콜렉션의 DAD를 보여줍니다.

DAD 파일: XML 컬럼

이 DAD 파일에는 XML 컬럼에 대한 맵핑, XML 데이터가 들어갈 테이블, 부가 테이블 및 컬럼에 대한 맵핑이 들어 있습니다.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dad.dtd">
<DAD>
<dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>
   <Xcolumn>
<table name="order_side_tab">
         <column name="order_key"
            type="integer"
           path="/Order/@key"
          multi_occurrence="NO"/>
  <column name="customer"
          type="varchar(50)"
           path="/Order/Customer/Name"
          multi_occurrence="NO"/>
     </table>
<table name="part_side_tab">
         <column name="price"
          type="decimal(10,2)"
          path="/Order/Part/ExtendedPrice"
            multi_occurrence="YES"/>
     </table>
<table name="ship_side_tab">
         <column name="date"
          type="DATE"
          path="/Order/Part/Shipment/ShipDate"
            multi_occurrence="YES"/>
     </table>
   </Xcolumn>
</DAD>
```

그림 16. XML 컬럼에 대한 샘플 DAD 파일

DAD 파일: XML 콜렉션 - SQL 맵핑

이 DAD 파일에는 XML 데이터가 들어갈 DB2 테이블, 컬럼 및 조건을 지정하는 SQL문이 들어 있습니다.

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<validation>NO</validation>
<Xcollection>
<SQL_stmt>SELECT o.order_key, customer_name, customer_email, p.part_key, color, quantity,
price, tax, ship_id, date, mode from order_tab o, part_tab p,
table (select substr(char(timestamp(generate_unique())),16)
  as ship_id, date, mode, part_key from ship_tab) s
     WHERE o.order_key = 1 and
              p.price > 20000 and
           p.order_key = o.order_key and
           s.part_key = p.part_key
        ORDER BY order_key, part_key, ship_id</SQL_stmt>
     <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
```

그림 17. SQL 맵핑을 사용하는 XML 콜렉션의 샘플 DAD 파일 (1/2)

```
      <root_node>
         <element_node name="Order">
  <attribute_node name="key">
                <column name="order_key"/>
             </attribute_node>
  <element_node name="Customer">
    <element_node name="Name">
      <text_node><column name="customer_name"/></text_node>
    </element_node>
    <element_node name="Email">
      <text_node><column name="customer_email"/></text_node>
    </element_node>
  </element_node>
  <element_node name="Part">
    <attribute_node name="color">
      <column name="color"/>
    </attribute_node>
     <element_node name="key">
      <text_node><column name="part_key"/></text_node>
  </element_node>
    <element_node name="Quantity">
      <text_node><column name="quantity"/></text_node>
  </element_node>
    <element_node name="ExtendedPrice">
      <text_node><column name="price"/></text_node>
  </element_node>
    <element_node name="Tax">
      <text_node><column name="tax"/></text_node>
  </element_node>
    <element_node name="Shipment" multi_occurence="YES">
              <element_node name="ShipDate">
        <text_node><column name="date"/></text_node>
      </element_node>
      <element_node name="ShipMode">
        <text_node><column name="mode"/></text_node>
      </element_node>
      </element_node>
      </element_node>
      </element_node>
</root_node>
  </Xcollection>
</DAD>
```

그림 17. SQL 맵핑을 사용하는 XML 콜렉션의 샘플 DAD 파일 (2/2)

DAD 파일: XML - RDB_node 맵핑

이 DAD 파일에서는 <RDB_node> 요소를 사용하여 XML 데이터가 들어갈 DB2 테이블, 컬럼 및 조건을 정의합니다.

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "c:\dxx\dtd\dad.dtd">
<DAD>
<dtdid>c:\dxx\samples\dtd\getstart.dtd</dtdid>
   <validation>YES</validation>
<Xcollection>
      <prolog>?xml version="1.0"?</prolog>
<doctype>!DOCTYPE Order SYSTEM "c:\dxx\samples\dtd\getstart.dtd"</doctype>
      <root_node>
         <element_node name="Order">
                     <RDB_node>
                     <table name="order_tab"/>
                     <table name="part_tab"/>
                        <table name="ship_tab"/>
                        <condition>
order_tab.order_key = part_tab.order_key AND
                     part_tab.part_key = ship_tab.part_key
                        </condition>
                     </RDB_node>
  <attribute_node name="key">
                     <RDB_node>
                     <table name="order_tab"/>
                  <column name="order_key"/>
                     </RDB_node>
               </attribute_node>
  <element_node name="Customer">
                  <text_node>
                     <RDB_node>
                     <table name="order_tab"/>
                     <column name="customer"/>
                     </RDB_node>
                  </text_node>
               </element_node>
```

그림 18. RDB_node 맵핑을 사용하는 XML 콜렉션의 샘플 DAD 파일 (1/3)

```
<element_node name="Part">
                  <RDB_node>
                  <table name="part_tab"/>
                     <table name="ship_tab"/>
                     <condition>
                  part_tab.part_key = ship_tab.part_key
                     </condition>
                  </RDB_node>
<attribute_node name="key">
                  <RDB_node>
                  <table name="part_tab"/>
                  <column name="part_key"/>
                  </RDB_node>
  </attribute_node>
             <element_node name="Quantity">
               <text_node>
                  <RDB_node>
                  <table name="part_tab"/>
                  <column name="quantity"/>
                  </RDB_node>
               </text_node>
            </element_node>
  <element_node name="ExtendedPrice">
               <text_node>
                  <RDB_node>
                  <table name="part_tab"/>
                     <column name="price"/>
                     <condition>
                        price > 2500.00
                     </condition>
                  </RDB_node>
               </text_node>
            </element_node>
  <element_node name="Tax">
               <text_node>
                  <RDB_node>
                  <table name="part_tab"/>
                  <column name="tax"/>
                  </RDB_node>
               </text_node>
            </element_node>
```

그림 18. RDB_node 맵핑을 사용하는 XML 콜렉션의 샘플 DAD 파일 (2/3)

```
            <element_node name="shipment">
                     <RDB_node>
                        <table name="ship_tab"/>
                        <condition>
                     part_key = part_tab.part_key
                        </condition>
                     </RDB_node>
          <element_node name="ShipDate">
                     <text_node>
                        <RDB_node>
                           <table name="ship_tab"/>
                           <column name="date"/>
                           <condition>
                              date > "1966-01-01"
                           </condition>
                        </RDB_node>
                     </text_node>
                  </element_node>
        <element_node name="ShipMode">
                     <text_node>
                        <RDB_node>
                           <table name="ship_tab"/>
                           <column name="mode"/>
                        </RDB_node>
                     </text_node>
                  </element_node>
                  <element_node name="Comment">
                     <text_node>
                        <RDB_node>
                           <table name="ship_tab"/>
                           <column name="comment"/>
                        </RDB_node>
                     </text_node>
                  </element_node>
               </element_node> <! -- end of element Shipment>
            </element_node> <! -- end of element Part --->
         </element_node> <! -- end of element Order --->
</root_node>
  </Xcollection>
</DAD>
```

그림 18. RDB_node 맵핑을 사용하는 XML 콜렉션의 샘플 DAD 파일 (3/3)

부록C. 코드 페이지 고려사항

XML 문서 및 기타 관련 파일이 파일을 액세스하는 각각의 클라이언트나 서버에 대해 올바로 코드화되었는지 확인하는 것이 중요합니다. 따라서, 파일 처리시의 XML Extender 가정 및 코드 페이지 변환 처리 방법을 이해하는 것이 중요합니다. 기본적인 고려사항은 다음과 같습니다.

- DB2에서 XML 문서를 검색하는 클라이언트의 실제 코드 페이지가 문서의 코드화와 일치하는지 확인합니다.
- 문서가 XML 구문분석기에 의해 처리되는 경우, XML 문서의 코드화 선언이 코드화와도 일치하는지 확인합니다.

다음 섹션은 이러한 고려사항과 관련된 사항, 가능한 문제점에 대해 준비할 수 있는 방법 및 문서가 클라이언트에서 서버로, 그리고 데이터베이스로 전달되는 경우에 XML Extender 및 DB2가 코드 페이지를 지원하는 방법을 설명합니다.

용어

다음 용어가 섹션에서 사용됩니다.

문서 코드화
XML 문서의 실제 코드 페이지.

문서 코드화 선언
XML 선언에 지정된 코드 페이지의 이름. 예를 들면 다음과 같습니다.

```
<?xml version="1.0" encoding="ibm037"?>
```

일치하는 문서
문서 코드화가 문서 코드화 선언 코드 페이지와 일치하는 문서.

불일치하는 문서
문서 코드화가 문서 코드화 선언 코드 페이지와 일치하지 않는 문서.

DB2CODEPAGE 레지스트리 (환경) 변수
데이터베이스 클라이언트 응용프로그램에서 DB2로 표현된 데이터의 코드

페이지를 지정합니다. 이 변수가 설정되지 않는 경우, DB2는 클라이언트의 운영 체제 로케일에서 클라이언트의 코드 페이지를 가져옵니다. DB2에서, 설정시 이 값은 클라이언트 운영 체제 로케일을 대체합니다.

클라이언트 코드 페이지

응용프로그램 코드 페이지. DB2CODEPAGE 변수가 설정된 경우, 클라이언트 코드 페이지는 DB2CODEPAGE의 값입니다. 그 이외의 경우, 클라이언트 코드 페이지는 클라이언트의 운영 체제 로케일입니다.

서버 코드 페이지 또는 서버 운영 체제 로케일 코드 페이지

DB2 데이터베이스가 설치된 운영 체제 로케일.

데이터베이스 코드 페이지:

데이터베이스 작성 시간에 판별된 저장된 데이터의 코드화. USING CODESET 절을 사용하여 명시적으로 지정하지 않으면, 이 값의 기본값은 서버 운영 체제 로케일입니다.

DB2 및 XML Extender 코드 페이지 가정

DB2가 XML 문서를 받거나 보내는 경우, 이는 코드화 선언을 점검하지 않습니다. 오히려, 이는 클라이언트의 코드 페이지를 점검하여 데이터베이스 코드 페이지와 일치하는지 확인합니다. 코드 페이지가 서로 다르면, DB2는 XML 문서에 있는 데이터를 다음의 코드 페이지와 일치하도록 변환합니다.

- 문서 또는 문서 조각을 데이터베이스 테이블로 가져오는 경우, 데이터베이스
- 문서를 하나 이상의 데이터베이스 테이블로 분해할 경우, 데이터베이스
- 데이터베이스에서 문서를 내보내는 경우 및 클라이언트로 문서를 표현하는 경우, 클라이언트
- 서버 파일 시스템에서 파일로 데이터를 리턴하는 UDF로 파일을 처리하는 경우, 서버.

XML 문서를 데이터베이스로 가져오는 경우, 이는 일반적으로 XML 컬럼에 저장되는 XML 문서로 가져오거나, 또는 XML 콜렉션 작성의 분해에 대해 가져옵니다. 여기에서 요소 및 속성 내용은 DB2 데이터로 저장됩니다. 문서를 가져오는 경우, DB2는 문서 코드화를 데이터베이스의 문서 코드화로 변환합니다. DB2는 문서가 아래 테이블에서 『소스 코드 페이지』 컬럼에 지정된 코드 페이지에 있다고

가정합니다. 표55는 XML 문서를 가져오는 경우에 DB2가 수행하는 변환을 요약하고 있습니다.

표 55. XML 파일을 데이터베이스로 가져오는 경우 UDF 및 저장 프로시듀어 사용

처리 메소드	변환을 위한 소스 코드 페이지	변환을 위한 목적지 코드 페이지	주석
DTD_REF 테이블에 DTD 파일 삽입	클라이언트 코드 페이지	데이터베이스 코드 페이지	
컬럼을 사용하거나, DAD 파일을 가져오는 관리 명령이나 콜렉션 저장 프로시듀어를 사용	클라이언트 코드 페이지	데이터베이스 코드 페이지	
UDF: • XMLVarcharFromFile() • XMLCLOBFromFile() • Content(): XMLFILE에서 CLOB로 검색	서버 코드 페이지	데이터베이스 코드 페이지	
분해 저장 프로시듀어	클라이언트 코드 페이지	데이터베이스 코드 페이지	분해될 문서는 클라이언트 코드 페이지에 존재한다고 가정합니다. 분해된 데이터는 데이터베이스 코드 페이지의 테이블에 저장됩니다.

XML 문서를 데이터베이스에서 내보내는 경우, 이는 일반적으로 문서를 표현하는 클라이언트 요청, XML 문서의 내용에 대한 조회를 기반으로 내보내거나, DB2 데이터로부터 문서의 작성을 수행하는 동안에 내보냅니다. 문서를 내보내는 경우, DB2는 요청이 개시된 위치 및 데이터가 표현되는 위치에 따라 문서 코드화를 클라이언트나 서버의 문서 코드화로 변환합니다. 표56은 XML 문서를 내보내는 경우 DB2가 수행하는 변환을 요약하고 있습니다..

표 56. XML 파일을 데이터베이스에서 내보내는 경우 UDF 및 저장 프로시듀어 사용

처리 메소드	변환	주석
UDF • XMLFileFromVarchar() • XMLFileFromCLOB()	데이터가 클라이언트에 표현되는 경우에 클라이언트 코드 페이지에 대한 데이터베이스 코드 페이지	

표 56. XML 파일을 데이터베이스에서 내보내는 경우 UDF 및 저장 프로시듀어 사용 (계속)

처리 메소드	변환	주석
UDF • Content(): XMLVARCHAR에서 외부 서버 파일로 검색	서버 코드 페이지로의 데이터베이스 코드 페이지	
작성 저장 프로시듀어: 조회 및 내보내기가 가능한 결과 테이블에 저장된 결과 테이블.	결과 세트가 클라이언트에 표현되는 경우에 클라이언트 코드 페이지에 대한 데이터베이스 코드 페이지	문서 작성시, XML Extender는 DAD의 태그에 의해 지정된 코드화 선언을 새로 작성된 문서로 복사합니다. 표현시 이는 클라이언트 코드 페이지와 일치해야 합니다.

코드화 선언 고려사항

코드화 선언은 XML 문서의 코드화를 선언하며 XML 선언 명령문에 표시됩니다. XML Extender 사용시, 파일이 위치한 장소에 따라 문서의 코드화가 클라이언트나 서버의 코드 페이지와 일치하는지 확인하는 것이 중요합니다.

법적 코드화 선언

일부 지침에서, XML 문서의 모든 코드화 선언을 사용할 수 있습니다. 이 섹션에서는 지원되는 코드화 선언과 함께 지침이 정의됩니다.

XML 데이터에 대해 권장하는 이식 가능 코드화는 XML 스펙에 따라 UTF-8 및 UTF-16 입니다. 이러한 코드화를 사용하는 경우, 회사간 응용프로그램의 상호 운용이 가능합니다. 309 페이지의 표57에 제시된 코드화를 사용하는 경우, 응용프로그램은 IBM 운영 체제간에 이식이 가능합니다. 다른 코드화를 사용하면 데이터의 이식성을 보장할 수 없습니다.

모든 운영 체제에서 다음의 코드화 선언이 지원됩니다. 다음 목록은 각 컬럼의 의미를 설명합니다.

- **코드화**는 XML 선언에서 사용되는 코드화 문자열을 지정합니다.
- **OS**는 DB2가 주어진 코드 페이지를 지원하는 운영 체제를 표시합니다.
- **코드 페이지**는 주어진 코드화와 연관된 IBM 정의 코드 페이지를 표시합니다.

표 57. XML Extender에 의해 지원되는 코드화 선언

범주	코드화	OS	코드 페이지
유니코드	UTF-8	AIX, SUN, Linux	1208
	UTF-16	AIX, SUN, Linux	1200
ASCII	iso-8859-1	AIX, Linux, Sun	819
	ibm-1252	Windows NT	1252
	iso-8859-2	AIX, Linux, Sun	912
	iso-8859-5	AIX, Linux	915
	iso-8859-6	AIX	1089
	iso-8859-7	AIX, Linux	813
	iso-8859-8	AIX, Linux	916
	iso-8859-9	AIX, Linux	920
MBCS	gb2312	Windows NT	1386
	ibm-932, shift_jis78	AIX	932
	Shift_JIS	AIX 4.3 전용, Windows NT	943
	IBM-eucCN	AIX, Sun	1383
	IBM-eucJP, EUC-JP	AIX, Linux, Sun	954, 33722
	euc-tw, IBM-eucTW	AIX, Sun	964
	euc-kr, IBM-eucKR	AIX	970
	big5	AIX, Sun, Windows NT	950

코드화 문자열은 문서의 목적지에 대한 코드 페이지와 호환되어야 합니다. 문서가 서버로부터 클라이언트로 리턴되는 경우, 문서의 코드화 문자열은 클라이언트의 코드 페이지와 호환되어야 합니다. 호환되지 않는 코드화의 결과에 대해서는 『일치하는 코드화 및 코드화 선언』의 내용을 참조하십시오. XML Extender에서 사용하는 XML 구문분석기가 지원하는 코드 페이지의 목록은 다음의 URL을 참조하십시오.

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

일치하는 코드화 및 코드화 선언

XML 문서가 처리되거나 다른 시스템과 교환된 경우, 코드화 선언이 문서의 실제 코드화와 대응되는지가 중요합니다. 구문 분석기 같은 XML 도구가 선언에 이름

이 지정된 것과 다른 코드화 선언을 포함하는 엔터티에 대해 오류를 생성하므로, 문서의 코드화가 클라이언트와 일치하는지 확인하는 것은 중요합니다.

그림19는 클라이언트가 문서 코드화 및 선언 코드화와 일치하는 코드 페이지를 보유하는지 보여줍니다.

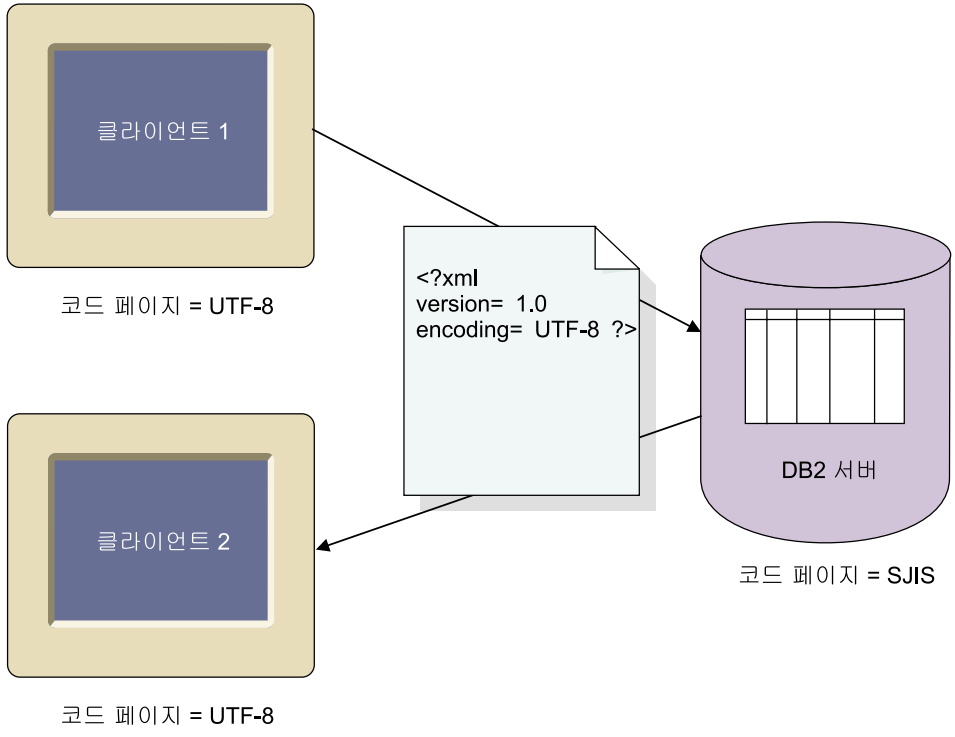


그림 19. 일치하는 코드 페이지를 갖는 클라이언트

계속해서 다른 코드 페이지를 가질 경우, 다음과 같은 상황이 발생할 수 있습니다.

- 소스 코드 페이지가 유니코드이고 목표 코드 페이지가 유니코드가 아닌 경우에는 특히 변환시 데이터가 유실될 수 있습니다. 유니코드에는 문자 변환의 전체 세트가 포함됩니다. UTF-8로부터 문서가 사용하는 모든 문자를 지원하지 않는 코드 페이지로 파일을 변환하는 경우에는, 변환시 데이터가 유실됩니다.
- 문서의 선언된 코드화와 다른 코드 페이지를 사용하는 클라이언트에 의해 문서가 검색되는 경우, XML 문서의 선언된 코드화는 더 이상 실제 문서 코드화와 일치하지 않습니다.

그림20은 클라이언트의 코드 페이지들이 불일치하는 환경을 나타냅니다.

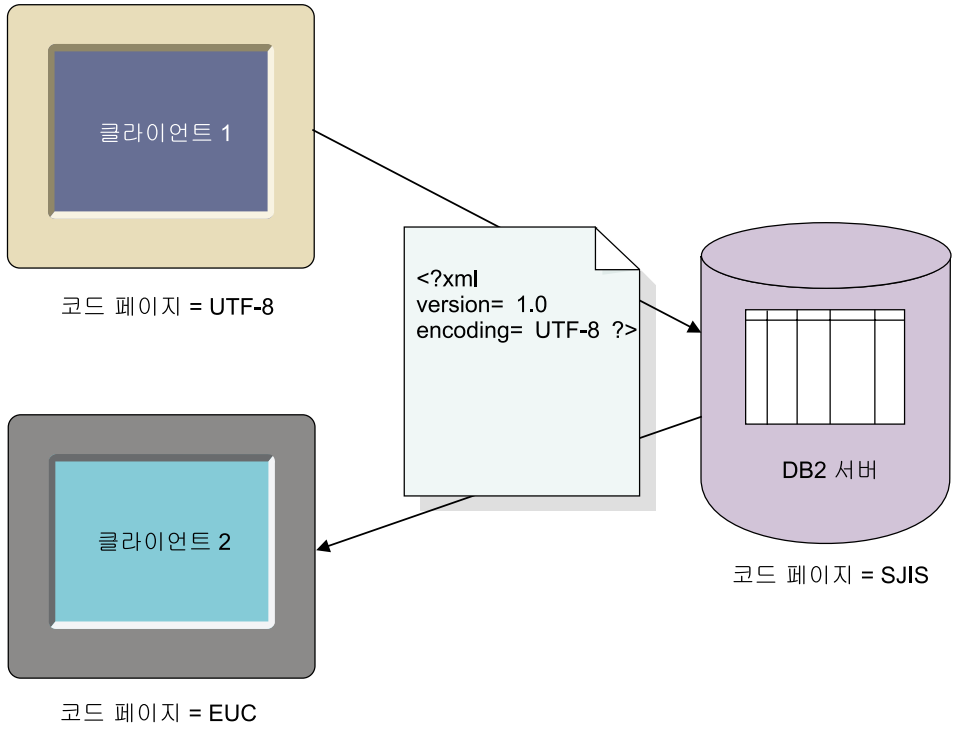


그림 20. 불일치 코드 페이지를 가지는 클라이언트

클라이언트2는 EUC의 문서를 수신하지만, 문서는 UTF-8의 코드화 선언을 가집니다.

코드화 선언

코드화 선언의 기본값은 UTF-8이며, 코드화 선언이 되어 있지 않으면 문서는 UTF-8 형태임을 의미합니다.

코드화 값을 선언하려면, 다음과 같이 하십시오.

XML 문서에서 선언은 코드화 선언을 클라이언트 코드 페이지 이름으로 지정합니다. 예를 들면 다음과 같습니다.

```
<?xml version="1.0" encoding="UTF-8" ?>
```

변환 시나리오

XML Extender는 다음과 같은 경우에 XML 문서를 처리합니다.

- XML 컬럼 저장영역 및 액세스 메소드를 사용하여 XML 컬럼 데이터를 저장하고 검색
- XML 문서 작성 및 분해

클라이언트나 서버로부터 데이터베이스로 전달될 때 문서에 대해 코드 페이지 변환이 수행됩니다. 클라이언트, 서버 및 데이터베이스의 코드 페이지로부터 변환되는 동안 XML 문서의 불일치나 손상이 발생할 확률이 높습니다. 문서의 코드화 선언을 선택하는 경우 및 데이터베이스에서 가져오거나 내보낼 수 있는 클라이언트와 서버를 계획하는 경우, 위의 테이블에서 설명한 변환 및 아래에서 설명하는 시나리오를 고려하십시오.

다음의 시나리오는 일반적으로 발생하는 변환 시나리오입니다.

시나리오 1: 이 시나리오는 일치하는 코드화, 비 DB2 변환 및 서버로부터 가져오는 문서로 구성됩니다. 문서 코드화 선언은 UTF-8이고 서버가 UTF-8이며, 데이터베이스는 UTF-8입니다.

1. XMLClobfromFile UDF를 사용하여 DB2로 문서를 가져옵니다.
2. 서버로 문서를 추출합니다.
3. 서버 코드 페이지와 데이터베이스 코드 페이지가 동일하므로 DB2는 문서를 변환할 필요가 없습니다. 코드화 및 선언이 일치합니다.

시나리오 2: 이 시나리오는 일치하는 코드화, DB2 변환 및 서버로부터 가져오고 클라이언트로 내보내는 문서로 구성됩니다. 문서 코드화 및 선언은 SJIS이고, 클라이언트 코드 페이지는 SJIS이며, 서버와 데이터베이스 코드 페이지는 UTF-8 입니다.

1. XMLClobfromfile UDF를 사용하여 서버로부터 DB2로 문서를 가져옵니다.
2. DB2는 SJIS로부터 문서를 변환하며, 이를 UTF-8에 저장합니다. 코드화 선언 및 코드화는 데이터베이스에서 일치하지 않습니다.
3. SJIS를 사용하는 클라이언트는 웹 브라우저에서의 프리젠테이션을 위한 문서를 요구합니다.

4. DB2는 문서를 SJIS, 클라이언트 코드 페이지로 변환합니다. 문서 코드화 및 선언은 현재 클라이언트에서 일치합니다.

시나리오 3: 이 시나리오는 불일치하는 코드화, DB2 변환 및 서버로부터 가져오고 클라이언트로 내보내는 문서로 구성됩니다. 유입되는 문서에 대해 문서 코드화 선언은 SJIS입니다. 서버 코드 페이지는 SJIS이고, 클라이언트 및 데이터베이스는 UTF-8입니다.

1. 저장영역 UDF를 사용하여 데이터베이스로 문서를 가져옵니다.
2. DB2가 문서를 SJIS로부터 UTF-8로 변환합니다. 코드화 및 선언이 불일치합니다.
3. UTF-8 코드 페이지를 사용하는 클라이언트는 웹 브라우저에서의 프리젠테이션을 위한 문서를 요청합니다.
4. 클라이언트 및 데이터베이스 코드 페이지가 동일하므로 DB2가 변환하지 않습니다.
5. 선언이 SJIS이고 코드화가 UTF-8이므로 문서 코드화 및 선언이 일치하지 않습니다. XML 구문분석기나 기타 XML 처리 도구를 사용하여 문서를 처리할 수 없습니다.

시나리오 4: 이 시나리오는 데이터 유실, DB2 변환 및 UTF-8 서버로부터 가져오는 문서로 구성됩니다. 문서 코드화 선언은 UTF-8이고, 서버가 UTF-8이며, 데이터베이스는 SJIS입니다.

1. XMLClobfromFile UDF를 사용하여 문서를 DB2로 가져옵니다.
2. DB2는 SJIS로 코드화를 변환합니다. 문서를 가져오는 경우, UTF-8로 표현되는 문자가 SJIS로 표현되지 않으므로 데이터베이스에 저장된 문서는 손상됩니다.

시나리오 5:

이 시나리오는 Windows NT 제한사항으로 구성됩니다. Windows NT에서 운영체제 로케일은 UTF-8로 설정될 수 없습니다. 그러나, DB2는 클라이언트가 `db2set DB2CODEPAGE=1208`을 사용하여 코드 페이지를 UTF-8로 설정하도록 허용합니다. 이 시나리오에서, 클라이언트와 서버는 동일한 시스템에 존재합니다. 클라이언트는

UTF-8이지만, 서버는 UTF-8로 설정될 수 없습니다. 이의 코드 페이지는 1252입니다. 문서는 1252로 코드화되며, 코드화 선언은 ibm-1252입니다. 데이터베이스 코드 페이지는 UTF-8입니다.

1. 저장영역 UDF가 문서를 서버에서 가져오며, 1252에서 1208로 변환합니다.
2. 클라이언트가 Content() UDF를 사용하여 DB2에서 문서를 내보냅니다.
3. 클라이언트가 서버와 동일한 시스템에 존재하며 1208로 설정되어 있으므로, 클라이언트가 1208을 예상하더라도 DB2가 문서를 UTF-8에서 1252로 변환합니다.

불일치하는 XML 문서 방지

이전의 섹션에서는 어떻게 XML 문서가 불일치하는 코드화를 보유하는지, 즉 코드화 선언이 문서의 코드화와 충돌하는 경우를 설명했습니다. 불일치하는 코드화로 인해 데이터 및/또는 사용하지 않는 XML 문서가 유실될 수 있습니다.

다음 권장사항 중 하나를 사용하면, 구문분석기와 같은 XML 프로세서로 문서를 처리하기 전에 XML 문서 코드화가 클라이언트 코드 페이지와 일치하도록 보증할 수 있습니다.

- XML Extender UDF를 사용하여 데이터베이스에서 문서를 내보내는 경우, 다음 기술 중 하나를 시도하십시오(가정: XML Extender는 서버 코드 페이지에서 파일을 서버의 파일 시스템으로 내보낸 상태임).
  - 문서를 선언된 코드화 코드 페이지로 변환하십시오.
  - 도구에 겹쳐쓰기 기능이 있는 경우, 선언된 코드화를 겹쳐쓰십시오.
  - 내보낸 문서의 코드화 선언을 문서의 실제 코드화로 수동으로 변경하십시오(즉, 서버 코드 페이지).
- XML Extender 저장 프로시듀어를 사용하여 데이터베이스에서 문서를 내보내는 경우, 다음 기술 중 하나를 시도하십시오(클라이언트가 작성된 문서가 저장된 결과 테이블을 조회하고 있음을 가정함).
  - 문서를 선언된 코드화 코드 페이지로 변환하십시오.
  - 도구에 겹쳐쓰기 기능이 있는 경우, 선언된 코드화를 겹쳐쓰십시오.

- 결과 테이블을 조회하기 전에, 클라이언트가 클라이언트 코드 페이지를 강제하는 환경 변수 DB2CODEPAGE를 XML 문서의 코드화 선언과 호환되는 코드 페이지로 설정하도록 하십시오.
- 내보낸 문서의 코드화 선언을 문서의 실제 코드화로 수동으로 변경하십시오 (즉, 클라이언트 코드 페이지).

• **유니코드 및 Windows NT 사용시 제한사항:** Windows NT에서는 운영 체제 로케일을 UTF-8로 설정할 수 없습니다. 문서를 내보내거나 가져오는 경우에는 다음의 지침을 사용하십시오.

- UTF-8로 코드화된 파일 및 DTD를 가져오는 경우, 다음을 사용하여 클라이언트 코드 페이지를 UTF-8로 설정하십시오.

  ```
  db2set DB2CODEPAGE=1208
  ```

  다음과 같은 경우에 이 기술을 사용하십시오.

  - DTD를 db2xml.DTD_REF 테이블에 삽입
  - 컬럼이나 콜렉션을 사용
  - 저장 프로시듀어 분해

- Content() 또는 XMLxxxfromFile UDF를 사용하여 XML 문서를 가져오는 경우, 문서는 서버의 운영 체제 로케일의 코드 페이지로 코드화 되어야 하며, 이는 UTF-8이 될 수 없습니다.
- 데이터베이스에서 XML 파일을 내보내는 경우. DB2가 UTF-8의 결과 데이터를 코드화하도록 다음 명령을 사용하여 클라이언트 코드 페이지를 설정하십시오.

  ```
  db2set DB2CODEPAGE=1208
  ```

  다음과 같은 경우에 이 기술을 사용하십시오.

  - 작성 이후 결과 테이블을 조회
  - 추출 UDF를 사용하여 XML 컬럼에서 데이터를 추출

- Content() 또는 XMLxxxfromFile UDF를 사용하여 XML 문서를 서버 파일 시스템의 파일로 내보내는 경우, 결과 문서는 서버의 운영 체제 로케일의 코드 페이지로 코드화 되며, 이는 UTF-8이 될 수 없습니다.

부록D. XML Extender 한계

XML Extender DAD 파일, 저장 프로시듀어와 테이블의 한계는 다음과 같습니다.

표 58. XML Extender 한계

값	한계
분해 XML 콜렉션의 테이블	각각의 분해된 XML 문서로부터 1024 행
저장 프로시듀어 매개변수:	
XML 문서 CLOB	1 MB[2]
문서 액세스 정의 (DAD) CLOB	100 KB[2]
collectionName	30
colName	30
dbName	18
defaultView	128
rootID	128
resultTabName	128
tablespace	128
tbName	128[1]
db2xml.DTD_REF 테이블 컬럼	
AUTHOR	128
CREATOR	128
UPDATOR	128
DTDID	128
CLOB	100 KB

주:

1. *tbName*이 스키마 이름에 의해 규정된 경우, 전체 이름(구분자 포함)은 128 문자를 넘지 않아야 합니다.
2. 이 크기는 변경될 수 있습니다. 이에 대한 내용은 233 페이지의 『CLOB 한계 증가』를 참조하십시오.

DB2가 이름을 클라이언트 코드 페이지에서 데이터베이스 코드 페이지로 변환할 때 이름이 확장될 수 있습니다. 이름은 클라이언트에서 크기 한계 내에 들어올 수 있지만, 저장 프로시듀어가 변환된 이름을 확보하는 경우에는 한계가 초과됩니다. 자세한 정보는 *DB2 응용프로그램 개발 안내서*에서 『복잡한 환경에서의 프로그래밍』 장의 『자국 언어 지원 응용프로그램 개발』 섹션을 참조하십시오.

주의사항

이 책은 미국에서 제공되는 제품 및 서비스를 위해 개발되었습니다. IBM은 다른 나라에서는 이 책에서 논의된 제품, 서비스 또는 기능을 제공하지 않을 수 있습니다. 현재 사용자 지역에서 사용 가능한 제품 및 서비스에 대한 정보는 해당 지역의 IBM 영업 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 반드시 IBM 제품, 프로그램 또는 서비스만을 사용하라는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수 있습니다. 그러나 IBM이 아닌 제품, 프로그램 또는 서비스의 운영상의 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대한 특허를 보유하고 있거나 출원 중일 수 있습니다. 이 책을 제공한다고 해서 이 특허에 대한 사용권을 부여하는 것은 아닙니다. 특허 사용권에 대한 문의는 다음 주소로 하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화 번호: 080-023-8080

2바이트(DBCS) 정보에 관한 특허 사용권에 대한 문의는 사용자 국가의 IBM 지적 재산권부나 다음 주소로 서면 문의하십시오.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 사항은 영국이나 이 조항이 현지법과 상충하는 나라에는 적용되지 않습니다.
IBM에서는 이 책을 어떠한 종류의 보증도 없이『현상태대로』제공하므로, 판매 가능성을 보장하거나 특정 목적에 적합한지 여부에 대해서는 책임질 수 없습니다. 일부 국가에서는 특정 거래의 명백한 또는 암시적인 보증을 부인하는 문장은 허용하지 않으므로, 이 사항이 사용자에게 적용되지 않을 수도 있습니다.

이 책은 기술상 부정확한 내용이나 출판상의 오류가 있을 수 있습니다. 이 책의 내용은 정기적으로 변경되며, 이들 변경사항은 개정판에 통합됩니다. IBM에서는 어느 때라도 이 책에 설명된 제품과 프로그램을 개선 및 변경할 수 있습니다.

(i) 독자적으로 작성된 프로그램과 다른 프로그램(이 프로그램 포함)과의 교환과 (ii) 교환된 정보의 공통 사용을 목적으로 이 프로그램에 대한 정보를 필요로 하는 사용권자는 다음 주소로 문의하십시오.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

그러한 정보는 적절한 조항과 조건에 따라 사용이 가능합니다(경우에 따라서는 비용을 지불함).

이 책에 기술된 사용권 프로그램과 이 프로그램에 사용 가능한 모든 사용권 데이터는 IBM 고객 협약, IBM 국제 프로그램 사용권 협약 또는 이와 동등한 모든 협약 조건하에 IBM에서 제공합니다.

IBM 이외의 제품에 관련된 정보는 해당 제품의 공급자, 그 출판된 발표 또는 기타 공적으로 이용 가능한 소스를 통해 확보되었습니다. IBM에서는 이들 제품을 테스트하지 않았으며, 성능의 정확성, 호환성 또는 IBM 이외의 제품과 관련된 다른 요구를 확인할 수 없습니다. IBM 이외의 제품 기능에 대한 문의는 해당 제품 공급자에게 문의해야 합니다.

IBM의 앞으로의 방향 또는 의도에 관한 모든 내용은 통지 없이 변경되거나 철회될 수 있으며, 목표 및 목적만 표시됩니다.

저작권:

이 책은 다양한 운영 플랫폼에서 프로그래밍 기술을 보여주는 샘플 프로그램이 소스 언어로 나와 있습니다. 이들 샘플 프로그램은 IBM에 요금을 지불하지 않고 샘플 프로그램이 작성된 운영 체제 플랫폼의 응용 프로그래밍 인터페이지에 적용하여 응용프로그램 개발, 사용, 판매 또는 배포 목적으로 어떤 형태로든 복사, 수정

및 배포할 수 있습니다. 이들 예제는 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이 프로그램의 신뢰성, 서비스 가능성 및 기능에 대해 보장할 수 없습니다.

등록상표

다음 용어는 IBM사의 등록상표입니다.

DB2DB2 Extender
DB2 Universal DatabaseIBM
IMS
Net.Data
OS/2
OS/390
OS/400
VTAM

Java 및 Java 관련 모든 등록상표와 로고는 Sun Microsystems사의 등록상표입니다.

Microsoft, Windows, Windows NT 및 Windows 로고는 Microsoft Corporation의 등록상표입니다.

UNIX는 미국 및 다른 나라에서 X/Open Company Limited를 통해 독점 허가된 등록상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 등록상표이거나 서비스 상표입니다.

용어집

가

개방형 데이터베이스 연결성 (Open DatabaseConnectivity). 관계형 및 비관계형 데이터베이스 관리 시스템 모두에서 데이터에 액세스하기 위한 표준 응용프로그램 인터페이스. 이 API를 사용하면, 각 데이터베이스 관리 시스템이 다른 데이터 저장영역 형식 및 프로그래밍 인터페이스를 사용하는 경우라도 데이터베이스 응용프로그램이 여러 컴퓨터에 있는 데이터베이스 관리 시스템에 저장된 데이터에 액세스 할 수 있습니다. ODBC는 X/Open SQL 액세스 그룹의 콜 레벨 인터페이스(CLI) 스펙을 기초로 하며, Digital Equipment Corporation(DEC), Lotus, Microsoft 및 Sybase가 개발했습니다. *Java 데이터베이스 연결성*과 대조.

결과 세트. 저장 프로시듀어가 리턴한 행 세트.

결과 테이블. SQL 조회나 저장 프로시듀어 실행 결과로서의 행이 들어있는 테이블.

경로 표현식. *위치 경로* 참조.

관계형 데이터베이스 노드(RDB_node). 테이블, 선택적 컬럼 및 선택적 조건에 대한 하나 이상의 요소 정의가 들어있는 노드. 테이블 및 컬럼은 XML 데이터가 데이터베이스에서 저장되는 방식을 정의하는 데 사용됩니다. 조건에서는 XML 데이터를 선택하는 기준이나 XML 콜렉션 테이블을 조인하는 방식을 지정합니다.

관리 지원 테이블. XML 오브젝트에 대한 사용자 요청을 처리하기 위해 DB2 extender가 사용하는 테이블. 몇몇 관리 지원 테이블에서는 Extender가 사용할 수 있는 사용자 테이블 및 컬럼을 식별합니다. 기타 관리 지원 테이블에는 사용 가능한 컬럼에 있는 오브젝트에 대한 속성 정보가 포함됩니다. 메타데이터 테이블과 동의어.

구별 유형(distinct type). *사용자 정의 유형* 참조.

구조적인 텍스트 색인. DB2 Text Extender를 사용하여, XML 문서의 트리 구조를 기초로 텍스트 문자열에 색인을 지정한 것입니다.

기본 뷰. XML 테이블 및 관련된 모든 부가 테이블이 조인되는 데이터 표현.

기본 유형 변환 함수. SQL 기본 유형을 UDT로 변환합니다.

기본 키. 테이블 정의의 일부인 고유 키. 기본 키는 참조 제한조건 정의의 기본 상위 키입니다.

나

노드(node). 데이터베이스 파티션 지정시에는 데이터베이스 파티션과 동의어.

다

다중 발생. 컬럼 요소 또는 속성이 문서에서 한 번 이상 사용될 수 있는지 여부 표시. 다중 발생은 DAD에서 지정됩니다.

단순 위치 경로. 하나의 슬래시로 연결된 일련의 요소 유형.

단일 자원 위치 지정자(URL). HTTP 서버와 선택적으로 디렉토리 및 파일 이름을 지정하는 주소입니다. 예를 들면: http://www.ibm.com/data/db2/extenders.

대형 오브젝트(LOB). 일련의 바이트, 여기서 길이는 최대 2GB까지 가능합니다. LOB는 세개의 유형 즉, *2진 대*

형 오브젝트(BLOB), *문자 대형 오브젝트(CLOB)* 또는 *2 바이트 문자 대형 오브젝트(DBCLOB)*가 될 수 있습니다.

데이터 교환. 응용프로그램 사이에서 데이터를 공유하는 것입니다. XML은 전용 형식의 데이터를 먼저 변형시키는 프로세스를 처리하지 않고도 데이터 교환을 지원합니다.

데이터 소스. ODBC API를 지원하는 ODBC 드라이버를 통해 데이터 액세스를 지원할 수 있는 지역 또는 원격 관계형 또는 비관계형 데이터 관리 프로그램.

데이터 유형. 컬럼 또는 리터럴의 속성.

라

루트 요소. XML 문서의 맨 위 요소.

루트 ID. 모든 부가 테이블을 응용프로그램 테이블과 연결하는 고유 식별자.

마

맨 위 element_node. DAD에서 XML 문서의 루트 요소를 나타냅니다.

맵핑 스킴. 관계형 데이터베이스에서 XML 데이터가 표현되는 방식을 정의합니다. 맵핑 스킴은 DAD에서 지정됩니다. XML Extender는 두 개의 맵핑 스킴 유형 즉, *SQL 맵핑* 및 *관계형 데이터베이스 노드(RDB_node) 맵핑*을 제공합니다.

메타데이터 테이블. *관리 지원 테이블* 참조.

문서 액세스 정의(DAD). XML 컬럼에 대한 색인화 스킴 또는 XML 콜렉션에 대한 맵핑 스킴을 정의하는 데 사용됩니다. XML 콜렉션의 XML Extender 컬럼을 사용 가능하게 하는 데 사용될 수 있습니다. 이것은 XML 형식으로 되어 있습니다.

문서 유형 정의(DTD). 선언 요소 및 속성에 대한 선언 세트. DTD는 XML 문서에서 사용되는 요소, 이들이 사용되는 순서 및 다른 요소를 포함하는 요소를 정의합니다. DTD를 문서 액세스 정의(DAD) 파일과 연결시켜 XML 문서의 유효성 검증을 수행합니다.

문자 대형 오브젝트(CLOB). 단일 바이트 문자의 문자열, 여기서 문자열은 최대 2GB까지 가능합니다. CLOB에는 연관된 코드 페이지가 있습니다. 단일 바이트 문자가 들어있는 텍스트 오브젝트는 DB2 데이터베이스에서 CLOB로서 저장됩니다.

바

부가 테이블. XML Extender가 XML 컬럼에서 요소나 속성을 검색할 때 성능을 향상시키기 위해 작성하는 추가 테이블.

부속 조회. SQL문의 검색 조건에서 사용되는 전체 SELECT문.

분해. XML 콜렉션에서 XML 문서를 관계형 테이블 콜렉션으로 구분하는 것.

브라우저(browser). *웹 브라우저* 참조.

사

사용자 정의 유형(UDT). 데이터베이스 관리 프로그램에 원래 있지 않은, 사용자가 작성한 데이터 유형입니다. *구별 유형* 참조.

사용자 정의 함수(UDF). 데이터베이스 관리 시스템에 정의되는 함수로서, 이후 SQL 조회에서 참조될 수 있습니다. 이는 다음 함수 중 하나가 될 수 있습니다.

- 외부 함수, 여기에서 함수 본문은 인수가 스칼라 값인 프로그래밍 언어로 기록되며, 각 호출마다 스칼라 결과가 생성됩니다.

- 전래 함수이며, 다른 내장 함수나 이미 DBMS가 알고 있는 사용자 정의 함수로 구현됩니다. 이 함수는 스칼라 함수 또는 컬럼(총계) 함수가 될 수 있으며, 값 세트에서 하나의 값을 리턴합니다(예를 들면 MAX 또는 AVG).

사용자 테이블. 응용프로그램을 위해 작성되고 이 응용프로그램이 사용하는 테이블.

색인(index). 키 값별로 논리적으로 정렬된 포인터 세트. 색인으로 데이터에 빠르게 액세스할 수 있으며, 테이블에 있는 행들이 고유하게 될 수 있습니다.

섹션 검색. 응용프로그램에서 정의할 수 있는 섹션 내에서 텍스트 검색을 제공합니다. 구조적인 텍스트 검색을 지원하려면, 섹션이 Xpath의 축약된 위치 경로에 정의되어야 합니다.

속성. *XML 속성* 참조.

술어. 비교 연산을 표시하거나 의미하는 검색 조건 요소.

스칼라 함수. 다른 값에서 하나의 값을 생성하고, 뒤에 괄호로 묶여진 인수 목록이 오는 함수 이름으로 표현되는 SQL 연산.

스키마. 테이블, 뷰, 색인 또는 트리거와 같은 데이터베이스 오브젝트 콜렉션. 데이터베이스 오브젝트의 논리 클래스를 제공합니다.

아

액세스 및 저장영역 메소드. 두 개의 주요 액세스 및 저장영역 메소드 즉, XML 컬럼 및 XML 콜렉션을 통해 XML 문서를 DB2 데이터베이스에 연결합니다. *XML 컬럼* 및 *XML 콜렉션*도 참조.

양호하게 구성된 문서. DTD가 포함되지 않는 XML 문서. XML 스펙에 있기는 하지만, DTD가 유효한 문서도 잘 구성되어야 합니다.

오브젝트. 객체 지향 프로그래밍에서, 데이터 및 해당 데이터와 연관된 조작으로 구성된 추상적 대상.

외부 키. 참조 제한조건을 정의하는 일부이며, 종속 테이블에 있는 하나 이상의 컬럼으로 구성된 키.

외부 파일(external file). DB2에 대한 외부 파일 시스템에 존재하는 파일.

요소. *XML 요소* 참조.

웹 브라우저. 웹 서버로 요청을 시작하고, 서버가 리턴하는 정보를 표시하는 클라이언트 프로그램.

위치 경로. XML 요소 또는 속성을 나타내는 일련의 XML 태그입니다. 위치 경로는 XML 문서의 구조를 나타내며, 이것은 요소나 속성의 문맥을 나타냅니다. 단일 슬래시(/) 경로는 문맥이 전체 문서임을 나타냅니다. 위치 경로는 추출 UDF에서 사용되어 추출할 요소 및 속성을 나타냅니다. 위치 경로는 DAD 파일에서도 사용되어 XML 컬럼에 대한 색인화 스킴을 정의할 때 XML 요소 또는 속성과 DB2 컬럼 사이의 맵핑을 지정합니다. 뿐만 아니라, 위치 경로는 구조적인 텍스트 검색을 위해 Text Extender가 사용합니다.

위치 지정자. 오브젝트를 찾을 때 사용될 수 있는 포인터. DB2에서, 대형 오브젝트 블록(LOB) 위치 지정자는 LOB가 있는 데이터 유형입니다.

유형 변환(cast) 함수. (소스) 데이터 유형을 다른(목표) 데이터 유형의 인스턴스로 변환하는 데 사용되는 함수. 일반적으로, 유형 변환(cast) 함수의 이름은 목표 데이터 유형의 이름입니다. 유형이 소스 데이터 유형인 단일 인수가 있으며, 이것의 리턴 유형은 목표 데이터 유형입니다.

유효 문서. 연관된 DTD가 있는 XML 문서. 유효하게 되기 위해 XML 문서는 그 DTD에서 지정된 구문론 규칙을 위반할 수 없습니다.

유효성 검증. DTD를 사용하여 XML 문서가 유효한지 확인하고, XML 데이터에 대한 구조화된 검색을 허용하는 프로세스. DTD는 DTD 저장소에 저장됩니다.

응용프로그램 프로그래밍 인터페이스(application programming interface(API)).

(1) 운영 체제 또는 별도로 구입할 수 있는 사용권 프로그램이 지원하는 기능성 인터페이스. API를 사용하면 고급 언어로 작성된 응용프로그램이 운영 체제나 사용권 프로그램의 특정 데이터 또는 기능을 사용할 수 있습니다.

(2) DB2의 경우, 인터페이스 내의 기능으로, 예를 들면 가져오기 오류 메시지 API.

자

작성. XML 콜렉션에 있는 관계형 데이터로 XML 문서를 생성하는 것입니다.

저장 프로시듀어. 데이터베이스에 저장되고 이름으로 호출될 수 있는 절차적 구문 및 Embedded SQL문 블록. 저장 프로시듀어를 사용하면 응용프로그램이 두 파트로 실행될 수 있습니다. 한 파트는 클라이언트에서 실행되고, 다른 파트는 서버에서 실행됩니다. 이를 이용하면 하나의 호출이 데이터베이스에 대해 여러가지로 액세스할 수 있습니다.

전체 텍스트 검색. DB2 Text Extender를 사용하여, 문서 구조에 관계없이 텍스트 문자열을 검색하는 것입니다.

절대 위치 경로. 오브젝트의 전체 경로 이름. 절대 경로 이름은 최상위 레벨 또는 ″root″ 요소로 시작되며, 이것은 슬래시(/)나 백슬래시(\) 문자로 표시됩니다.

정적 SQL. 프로그램 내에 포함되어 있고, 프로그램을 실행하기 전에 프로그램 준비 프로세스를 실행하는 동안 준비되는 SQL문입니다. 준비된 다음에는 명령문에서 지정한 호스트 변수 값이 변경되더라도 정적 SQL문은 변경되지 않습니다.

조건. XML 데이터를 선택하는 기준이나 XML 콜렉션 테이블을 조인하는 방식에 대한 스펙입니다.

조인. 일치하는 컬럼 값을 기초로 둘 이상의 테이블에서 데이터를 검색할 수 있도록 하는 관계형 연산.

조인된 뷰. 하나 이상의 테이블을 함께 조인하는 ″CREATE VIEW″문이 작성한 DB2 뷰.

조회. 특정 조건을 기초로 하는 데이터베이스 정보에 대한 요청. 예를 들면 조회는 고객표에서 잔고가 1000 이상인 모든 고객 목록에 대한 요청이 될 수 있습니다.

지역 파일 시스템. DB2에 있는 파일 시스템.

카

컬럼 데이터. DB2 컬럼 내에 저장된 데이터. 데이터 유형은 DB2가 지원하는 모든 데이터 유형이 될 수 있습니다.

타

테이블 공간. 데이터베이스 오브젝트가 저장되는 추상적인 컨테이너 콜렉션. 테이블 공간에서는 데이터베이스와, 데이터베이스내에 저장되는 테이블 사이에서 간접 레벨을 제공합니다. 테이블 공간에는 다음이 포함됩니다.

- 매체 저장영역 장치에 테이블 공간으로 지정된 공간이 있습니다.
- 이 공간내에서 테이블이 작성됩니다. 이러한 테이블들은 테이블 공간에 속하는 컨테이너의 공간을 사용합니다. 테이블의 데이터, 색인, 긴 필드 및 LOB 부분이 같은 테이블 공간에 저장되거나, 개별적으로 별도의 테이블 공간으로 분할될 수 있습니다.

파

파티션. 고정 크기의 저장영역 부분.

프로시듀어. *저장 프로시듀어* 참조.

숫자

2바이트 문자 대형 오브젝트(DBCLOB). 2바이트 문자로 구성된 문자열 또는 1바이트와 2바이트 문자 결합, 문자열은 최대 2GB까지 가능합니다. DBCLOB에는 연관된 코드 페이지가 있습니다. 2바이트 문자가 포함된 텍스트 오브젝트는 DB2 데이터베이스에 DBCLOB로서 저장됩니다.

A

API. *응용프로그램 프로그래밍 인터페이스(application programming interface)* 참조.

attribute_node. 요소 속성의 표현.

B

B-트리 색인화. DB2 엔진이 제공하는 원시 색인 스킴. B 트리 구조로 색인 항목을 빌드합니다. DB2 기본 데이터 유형을 지원합니다.

C

CLOB. 문자 대형 오브젝트(CLOB)

D

DAD. *문서 액세스 정의* 참조.

datalink. 데이터베이스에서 데이터베이스 외부에 저장된 파일로의 논리 참조를 가능하게 하는 DB2 데이터 유형.

DBCLOB. 2바이트 문자 대형 오브젝트.

DTD. (1) . (2) *문서 유형 정의* 참조.

DTD 저장소. DTD_REF라고 하는 DB2 테이블로서, 테이블의 각 행은 추가 메타데이터 정보와 함께 DTD를 나타냅니다.

DTD 참조 테이블(DTD_REF 테이블). DTD가 포함된 테이블로서, XML 문서를 확인하고 응용프로그램이 DAD를 정의하는 데 도움이 되도록 하는 데 사용됩니다. 사용자들은 자신의 DTD를 DTD_REF 테이블에 삽입할 수 있습니다. 이 테이블은 XML에 데이터베이스를 사용할 수 있을 때 작성됩니다.

DTD_REF 테이블. DTD 참조 테이블.

E

EDI. 전자 데이터 교환.

EDI(Electronic Data Interchange). 비지니스-비지니스(B2B) 응용프로그램을 위한 전자 데이터 교환 표준.

element_node. 요소 표현. element_node는 루트 요소나 하위 요소가 될 수 있습니다.

embedded SQL. 응용프로그램내에서 코딩된 SQL문. *정적 SQL* 참조.

J

Java 데이터베이스 연결성(JDBC). ODBC(Open Database Connectivity)와 같은 특성을 갖지만 Java 데이터베이스 응용프로그램이 사용하도록 특별히 지정된 API. JDBC 드라이버가 없는 데이터베이스의 경우, JDBC에는 ODBC 브릿지에 대한 JDBC가 포함되며, 이것은 JDBC를 ODBC로 변환하는 메카니즘입니다. JDBC는 Java 데이터베이스 응용프로그램에 JDBC API를 제공하며 이를 ODBC로 변환합니다. JDBC는 Sun Microsystems사와 여러 협력업체 및 벤더가 개발했습니다.

JDBC. Java 데이터베이스 연결성.

L

LOB. 대형 오브젝트.

O

ODBC. 개방형 데이터베이스 연결성.

overloaded 함수. 함수의 여러 인스턴스가 존재하는 함수 이름.

R

RDB_node. 관계형 데이터베이스 노드.

RDB_node 맵핑. XML 요소 내용의 위치, 또는 XML 속성 값, 이것은 RDB_node가 정의합니다. XML Extender는 이 맵핑을 사용하여 XML 데이터의 저장 또는 검색 위치를 판별합니다.

S

SQL 맵핑. 하나 이상의 SQL문과 XSLT 데이터 모델을 사용하여 XML 요소 내용이나 XML 속성값과 관계형 데이터 간의 관계를 정의하는 것입니다. XML Extender는 정의를 사용하여 XML 데이터를 저장하거나 검색하는 위치를 판별합니다. SQL 맵핑은 DAD에서 SQL_stmt 요소로 정의됩니다.

T

text_node. 요소의 CDATA 텍스트를 나타냅니다.

U

UDF. *사용자 정의 함수(UDF)* 참조.

UDT. *사용자 정의 유형* 참조.

UNION. 두 SELECT문의 결과를 결합하는 SQL 작업. UNION은 종종 여러 테이블에서 구한 값 목록을 병합하는 데 사용됩니다.

URL. 단일 자원 위치 지정자.

X

XML. 확장 가능한 마크업 언어.

XML UDF. XML Extender가 제공하는 DB2 사용자 정의 함수.

XML UDT. XML Extender가 제공하는 DB2 사용자 정의 유형.

XML 경로 언어. XML 문서의 주소지정 파트를 위한 언어입니다.XML 경로 언어는 XSLT가 사용되도록 디자인된 것입니다. 모든 위치 경로는 XPath에 대해 정의된 구문을 사용하여 표현될 수 있습니다.

XML 속성. DTD에서 XML 요소 하에 있는 ATTLIST가 지정한 속성. XML Extender는 위치 경로를 사용하여 속성을 식별합니다.

XML 오브젝트. XML 문서와 같습니다.

XML 요소. XML DTD에서 지정된 대로의 XML 태그 또는 ELEMENT. XML Extender는 위치 경로를 사용하여 요소를 식별합니다.

XML 컬럼. XML Extender UDT가 사용할 수 있는 응용프로그램 테이블 내의 컬럼.

XML 콜렉션. XML 문서를 작성하거나 XML 문서에서 분해될 데이터를 제공하는 관계형 테이블 콜렉션.

XML 태그. 유효한 XML 마크업 언어 태그, 주로 XML 요소. 태그와 요소라는 용어는 서로 교차되어 사용됩니다.

XML 테이블. 하나 이상의 XML Extender 컬럼이 포함된 응용프로그램 테이블.

XPath. XML 문서의 주소지정 파트를 위한 언어입니다.

XPath 데이터 모델. 노드를 사용하여 XML 문서를 모델화하고 탐색하는 데 사용되는 트리 구조.

XSL. XML 스타일 시트 언어.

XSL(Extensible Stylesheet language). 스타일 시트를 표현하는 데 사용되는 언어. XSL은 두 개의 파트 즉, XML 문서 변환을 위한 언어와 포맷 의미를 지정하기 위한 XML 용어집으로 구성됩니다.

XSLT. XML 스타일 시트 언어 변환.

XSLT(Extensive Stylesheet Language Transformation). XML 문서를 다른 XML 문서로 변환하는 데 사용되는 언어. XSLT는 XSL의 일부로서 디자인되었으며, XSL은 XML의 스타일 시트 언어입니다.

색인

[가]

[나]

[다]

[라]

[마]

[바]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

A

B

C

D

E

F

I

J

M

O

R

S

T

U

W

X

IBM 문의

기술적인 문제점이 있으면, 이를 검토하고 DB2 고객 지원 센터에 문의하기 전에 *문제점 해결 안내서*에서 제시하는 조치를 수행하십시오. 이 안내서는 DB2 고객 지원 센터가 더 나은 지원을 할 수 있도록 사용자가 수집할 수 있는 정보를 제시합니다.

DB2 Universal Database 제품에 대한 정보나 주문에 대해서는 해당 지역 사무소의 IBM 담당자에게 문의하거나 인증된 IBM 소프트웨어 대리점에 문의하십시오.

미국내에 거주하는 경우는 다음 번호 중 하나로 전화할 수 있습니다.

- 고객 지원에 대해서는 1-800-237-5511
- 사용 가능한 서비스 옵션에 대해 알려면 1-888-426-4343

제품 정보

미국내에 거주하는 경우는 다음 번호 중 하나로 전화할 수 있습니다.

- 제품을 주문하거나 일반 정보를 알려면 1-800-IBM-CALL(1-800-426-2255) 또는 1-800-3IBM-OS2(1-800-342-6672).
- 서적을 주문하려면 1-800-879-2755.

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

DB2 WWW 페이지에서는 뉴스, 제품 설명, 교육 일정 등등에 대한 현재 DB2 정보를 제공합니다.

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

DB2 제품 및 서비스 기술 라이브러리에서는 일반적인 질문, 수정사항, 관련 서적 및 최신 DB2 기술 정보에 대한 액세스를 제공합니다.

주: 이 정보는 영어로만 되어 있습니다.

http://www.elink.ibmlink.ibm.com/pbl/pbl/

국제 서적 주문 웹 사이트에서는 책 주문 방법에 대한 정보를 제공합니다.

http://www.ibm.com/education/certify/

IBM 웹 사이트의 전문 인증 프로그램은 DB2를 포함하여 다양한 IBM 제품에 관한 인증 테스트 정보를 제공합니다.

ftp.software.ibm.com

익명으로 로그온하십시오. /ps/products/db2 디렉토리에서, DB2 및 다수의 다른 제품에 관한 데모, 수정 내용, 정보 및 도구를 찾을 수 있습니다.

comp.databases.ibm-db2, bit.listserv.db2-l

이들 인터넷 뉴스 그룹은 사용자가 DB2 제품에 대한 경험을 논의할 때 사용할 수 있습니다.

Compuserve 상에서: GO IBMDB2

이 명령을 입력하여 IBM DB2 계열 포럼에 액세스하십시오. 모든 DB2 제품은 이들 포럼을 통해 지원됩니다.

미국 외에 있는 IBM에 연락하는 방법에 대해서는 *IBM 소프트웨어 지원 핸드북*의 부록 A를 참조하십시오. 이 문서에 액세스하려면, 웹 페이지 http://www.ibm.com/support/에 가서 이 페이지의 맨 아래 쪽에 있는 IBM 소프트웨어 지원 핸드북 링크를 선택하십시오.

주: 일부 국가에서, IBM 인증 딜러는 IBM 지원 센터 대신 자국의 딜러 지원 조직에 문의해야 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
XML Extender 관리 및 프로그래밍
버전 7

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.