IBM® DB2® Universal Database

Image, Audio, Video Extender 관리 및 프로그래밍

버전 7

SA30-1043-00

IBM® DB2® Universal Database

Image, Audio, Video Extender 관리 및 프로그래밍

버전 7

SA30-1043-00

이 책의 정보와 지원하는 제품을 사용하기 전에 반드시 653 페이지의 『부록C. 주의사항』을 읽으십시오.

목차

그림

표

이 책에 관하여

이 책에서는 DB2Extender를 사용하여 이미지, 오디오 또는 비디오 데이터용 DB2® 데이터베이스를 준비하고 유지보수하는 방법에 대해 설명합니다. 여기서는 또한 사용자 정의 함수(UDF)와 DB2Extender에서 제공하는 응용프로그램 프로그래밍 인터페이스(API)를 사용하여 이러한 데이터 유형에 액세스하고 조작하는 방법을 설명합니다. 사용하는 프로그램의 SQL문에 UDF를 통합하고 API를 통합함으로써, 이미지와 비디오 클립과 같은 비전통적 데이터와 전통적 숫자 데이터 및 문자 데이터에 액세스할 수 있습니다.

이 책에서의 ″DB2″는 DB2 UDB를 의미합니다.

이 책의 사용자

이 책은 DB2 관리 개념, 툴 및 기법에 익숙한 DB2 데이터베이스 관리자를 위한 것입니다.

이 책은 또한 DB2 응용프로그램에 사용할 수 있는 SQL과 하나 이상의 프로그래밍 언어에 익숙한 DB2 응용프로그램 작성자를 위한 것입니다.

이 책은 DB2 Image, Audio 및 Video Extender 사용자를 위한 것입니다. Text Extender 사용자는 *DB2 Text Extender 관리 및 프로그래밍*을 참조해야 합니다.

이 책의 구성

이 책은 다음과 같은 구조로 되어 있습니다.

『제1부. 소개』

제1부에서는 DB2 Extender 개요를 제공합니다. DB2 Extender로 관리하거나 프로그래밍하는 것이 생소하다면 이 부분을 읽으십시오.

『제2부. 이미지, 오디오 및 비디오 데이터 관리』

제2부에서는 이미지, 오디오 및 비디오 데이터용 DB2 데이터베이스를 준비하고 유지보수하는 방법에 대해 설명합니다. 이미지, 오디오 또는 비디오 데이터가 있는 DB2 데이터베이스를 관리해야 할 경우에 이 부분을 읽으십시오.

『제3부. 이미지, 오디오 또는 비디오 데이터용 프로그래밍』

제3부에서는 DB2 Extender UDF 및 API를 사용하여 이미지, 오디오 또는 비디오 데이터에 대한 조작을 요청하는 방법에 대해 설명합니다. DB2 응용프로그램에서 이미지, 오디오 또는 비디오 데이터에 액세스하고 조작해야 한다면 이 부분을 읽으십시오.

『제4부. 참조 정보』

제4부에서는 DB2 Extender UDF, API, 관리 명령 및 진단 정보(메시지 및 코드)에 대한 참조 정보를 제공합니다. DB2 Extender 개념과 타스크에 익숙하지만 특정 DB2 Extender UDF, API, 명령, 메시지 또는 코드에 관한 정보가 필요한 경우에는 이 부분을 읽으십시오.

『부록』

부록에서는 다음을 설명합니다.

- DB2 Extender가 이미지, 오디오 및 비디오 오브젝트의 표시나 재생 프로그램을 식별하고 파일을 찾는 데 사용하는 환경 변수의 설정 방법
- Extender와 함께 제공된 샘플 프로그램과 미디어 파일의 설치 및 사용법

플랫폼 고유 정보

DB2 Extender는 DB2 Universal Database의 단일 파티션 데이터베이스 환경이나 DB2 Universal Database Enterprise-Extended Enterprise Edition의 다중 파티션 데이터베이스 환경과 결합하여 사용할 수 있습니다.

이 책에는 이러한 두 환경에서 DB2 Extender를 사용하는 방법에 대한 정보가 들어 있습니다. DB2 Universal Database Enterprise-Extended Enterprise Edition의 다중 파티션 환경에서 Extender를 사용하는 것에만 관련된 정보는 "**EEE 전용**"으로 표시됩니다. DB2 Universal Database의 단일 파티션 환경에서 DB2 Extender를 사용하는 것에만 관련된 정보는 "**비 EEE 전용**"으로 표시됩니다. 특정 환경과 관련된 것으로 표시되지 않는 정보는 두 환경 모두에 적용됩니다.

강조표시 규약

이 책은 다음 규약을 사용합니다.

굵은체 굵은체는 새로운 용어 정의를 나타냅니다.

기울임체

기울임체는 값으로 대체될 변수 매개변수를 나타내거나 텍스트에 사용된 단어를 강조합니다.

대문자 대문자체는 다음을 표시합니다.

- 데이터 유형
- 디렉토리 이름
- 필드 이름
- API 호출
- 명령
- 키워드
- 변수 이름

예시 예시 텍스트는 시스템 메시지나 입력한 값을 표시합니다. 또한 예시 텍스트는 코딩 예에도 사용됩니다.

구문 도표 읽는 방법

이 책에서 명령과 SQL 구문은 구문 도표를 사용하여 설명됩니다. 다음과 같이 구문 도표를 읽으십시오.

- 구문 도표를 줄의 경로에 따라 왼쪽에서 오른쪽으로, 위에서 아래로 읽으십시오.

  ▶▶── 기호는 명령문의 시작을 표시합니다.

  ──▶ 기호는 명령문 구문이 다음 줄에서 계속됨을 표시합니다.

  ▶── 기호는 명령문이 이전 줄에서 계속됨을 표시합니다.

  ──▶◀ 기호는 명령문의 종료를 표시합니다.

- 필수 항목은 수평선(주 경로)에 나타납니다.

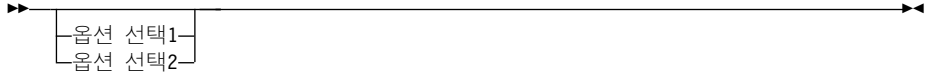
  ▶▶─필수 항목──────────────────────────────────▶◀

- 선택적 항목은 주 경로의 아래 부분에 나타납니다.

- 둘 이상의 항목에서 선택할 수 있다면 그것들은 스택에 나타납니다.

  항목 중 하나를 *반드시 선택해야* 한다면, 스택에 있는 항목 중 하나가 주 경로에 나타납니다.

  항목을 하나도 선택하지 않을 수 있다면, 전체 스택은 주 경로 아래 부분에 나타납니다.



  스택의 반복 화살표는 스택의 항목에서 둘 이상을 선택할 수 있음을 나타냅니다.

- 키워드는 대문자로 표시됩니다(예를 들면, `/DB2IMAGE:`). 그것은 표시된 그대로 정확히 사용되어야 합니다. 변수는 소문자로 나타납니다(예를 들면, `srcpath`). 그것은 구문에서 사용자 제공 이름과 값을 나타냅니다.
- 구두점이나 괄호, 산술 연산자 또는 다른 기호가 나타난다면, 구문의 일부분으로 그것들을 입력해야 합니다.

관련 정보

DB2 Universal Database

빠른 시작, GA30-0982(OS/2®), GA30-0985(Windows®), GA30-0984(UNIX). 이 책에서는 해당 플랫폼에서의 DB2 Universal Database 계획, 설치, 구성 및 이주 방법에 대해 설명합니다.

DB2 Universal Database Enterprise-Extended Edition 빠른 시작, GA30-0978(AIX®), GA30-0977(Windows). 이 책에서는 해당 플랫폼에서의 DB2 Universal Database Enterprise-Extended Edition 계획, 설치 및 구성 방법에 대해 설명합니다.

관리 안내서: 계획, (볼륨 1), SA30-0990. 이 책에서는 데이터베이스의 개념에 대한 개요, 논리적 및 물리적인 데이터베이스 설계와 같은 설계 관련 문제에 대한 정보, 그리고 고가용성에 대한 논의 정보를 제공합니다.

관리 안내서: 구현, (볼륨 2), SA30-0988. 이 책에서는 사용자의 설계 구현, 데이터베이스 액세스, 감사, 백업 및 복구와 같은 구현 관련 문제에 대한 정보를 제공합니다.

관리 안내서: 성능, (볼륨 3), SA30-0989. 이 책에서는 데이터베이스 환경, 응용프로그램 성능 평가 및 성능 조정에 대한 정보를 제공합니다.

응용프로그램 개발 안내서, SA30-0992. 이 책에서는 Embedded SQL 또는 JDBC를 사용하여 DB2 데이터베이스에 액세스하는 응용프로그램을 개발하는 방법을 설명합니다. 또한 저장 프로시듀어 및 사용자 정의 함수를 작성하는 방법과, 사용자 정의 유형을 정의하는 방법 및 트리거 사용 방법을 설명합니다.

CLI Guide and Reference, SC09-2950. 이 책에서는 Microsoft ODBC 스펙과 호환 가능한 호출 가능 SQL 인터페이스인 DB2 콜 레벨 인터페이스(CLI)를 사용하여 DB2 데이터베이스에 액세스하는 응용프로그램을 개발하는 방법을 설명합니다.

Command Reference, SC09-2951. 이 책에서는 DB2 명령행 처리기를 사용하는 방법을 설명하고 DB2 명령에 관한 참조 정보를 제공합니다.

메시지 참조서, GA30-0986 및 GA30-0987. 이 책에서는 DB2에서 사용하는 메시지와 코드를 나열하고 설명하며, 사용자가 지정된 오류 또는 문제점에서 복구하기 위해 취할 수 있는 조치를 설명합니다.

DB2 Universal Database Text Extender

DB2 Universal Database Text Extender 관리 및 프로그래밍 버전 7, SA30-1044. 이 책은 텍스트 데이터용 DB2 데이터베이스의 관리 방법을

설명합니다. 이것은 또한, 텍스트 데이터의 액세스와 조작을 위해 DB2 Text Extender가 제공하는 응용프로그램 프로그래밍 인터페이스의 사용법을 설명합니다.

DB2 Universal Database XML Extender

DB2 Universal Database XML Extender 관리 및 프로그래밍. 이 책에서는 XML 문서용 DB2 데이터베이스의 관리 방법을 설명합니다. 또한, XML 문서 및 데이터에 액세스하고 조작하기 위해 DB2 XML Extender에서 제공되는 응용프로그램 프로그래밍 인터페이스를 사용하는 방법에 대해서도 설명합니다.

DB2 Universal Database Spatial Extender

Spatial Extender 사용자 안내 및 참조서, SA30-1045. 이 책에서는 Spatial Extender의 설치, 구성, 관리, 프로그래밍 및 문제 해결에 대한 정보를 제공합니다. 또한, 공간 데이터 개념에 대한 설명을 제공하고 Spatial Extender에만 고유하게 적용되는 참조 정보(메시지 및 SQL)를 제공합니다.

OS/390용 DB2 Universal Database Image, Audio, Video Extender

DB2 Universal Database for OS/390 Version 6 Image, Audio, and Video Extenders Administration and Programming, SC26-9650. 이 책에서는 이미지, 오디오 및 비디오 데이터를 위해 OS/390용 DB2 데이터베이스 서버를 관리하는 방법을 설명합니다. 또한 OS/390용 DB2 Image, Audio 및 Video Extender에서 제공하는 응용프로그램 프로그래밍 인터페이스와 사용자 정의 함수를 사용하여 이미지, 오디오 및 비디오 데이터에 액세스하고 조작하는 방법을 설명합니다.

OS/390용 DB2 Universal Database Text Extender

DB2 Universal Database for OS/390 Version 6 Text Extender Administration and Programming, SC26-9651. 이 책에서는 텍스트 데이터를 위해 OS/390용 DB2 데이터베이스 서버를 관리하는 방법에 대해 설명합니다. 여기서는 또한 OS/390용 DB2 Text Extender에서 제공하는 사용자 정의 함수와 응용프로그램 프로그래밍 인터페이스를 사용하여 텍스트 데이터에 액세스하고 조작하는 방법을 설명합니다.

월드 와이드 웹(www)

DB2 Extender 웹 사이트. 이 웹 사이트에는 DB2 Extender에 관한 정보와 Extender와 관련된 기술에 관한 정보가 들어 있습니다. DB2 Extender 홈 페이지의 URL은 다음과 같습니다.

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

의견을 보내는 방법

고객의 피드백을 통해 IBM은 고품질의 정보를 제공할 수 있습니다. 이 책 또는 그 밖의 DB2 Extender 문서에 관하여 가지고 계신 의견을 보내주십시오. 다음과 같은 방법 중 하나를 사용하여 의견을 보내실 수 있습니다.

- 웹에서 의견을 보낼 경우에는 다음의 웹 사이트를 방문하십시오.

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

  웹 사이트에 의견을 입력하고 전송할 수 있는 피드백 페이지가 있습니다.
- 전자 우편을 이용할 경우에는 comments@vnet.ibm.com으로 의견서를 보내십시오. 제품 이름, 제품의 버전 번호 및 책 이름과 부품 번호(적용 가능한 경우)를 반드시 적으십시오. 특정 텍스트에 관한 의견이 있는 경우, 텍스트의 위치(예를 들어, 장, 절 제목, 표 번호, 페이지 번호 또는 도움말 주제 제목)를 적으십시오.
- 우편을 이용할 경우에는 다음 주소로 의견서를 보내십시오.

  135-270
  서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
  한국 아이.비.엠 주식회사
  고객만족센터
- 팩스 번호 02)3781-5200으로 의견서 제출.
- IBM 영업대표에게 의견서 제출.

IBM은 독자가 제공한 정보가 타당한 경우, 적절한 방식으로 사용하거나 배포할 수 있으며 제공한 독자는 이에 대해 책임을 지지 않습니다.

IBM에 연락하는 방법에 대해서는 677 페이지의 『IBM에 문의』에서 자세한 정보를 참조하십시오.

제1부 소개

제1장 개요

DB2(DB2) Universal Database(UDB)는 강력한 오브젝트 관계형 데이터베이스 관리 프로그램입니다. 이것은 크고 복잡한 오브젝트(LOB)뿐 아니라, 전통적인 숫자와 문자 데이터까지도 저장하고 보호합니다. DB2 Extender는 DB2의 오브젝트 관계형 기능을 활용하는 데 도움이 됩니다. Extender는 이미지, 오디오, 비디오 및 텍스트 오브젝트용 구별 데이터 유형과 특수 함수를 정의합니다. 이렇게 하여 Extender는 응용프로그램에서 이러한 데이터 유형과 함수를 정의하는 시간을 절약하고 수고를 덜게 합니다. 데이터 유형과 함수는 SQL을 통해 사용 가능합니다. 이러한 이유로 Extender는 응용프로그램에 전통적인 숫자 및 문자 데이터와 기타 모든 데이터 유형으로의 단일 액세스 포인트를 제공합니다. 또한, Extender는 응용프로그램에 정보를 탐색하는 새로운 방법을 제공합니다. 예를 들어, 응용프로그램은 색상이나 텍스처의 시각적 예를 사용하여 시각적 특성별로 이미지를 탐색할 수 있습니다.

DB2 이용

DB2 Extender는 DB2의 객체 지향 특성들을 이용합니다. 특히, DB2로 다음을 수행할 수 있습니다.

- DB2 데이터베이스에 최대 2기가바이트까지의 LOB을 저장합니다.
- 이들 크고 복잡한 오브젝트에 대한 구별 데이터 유형을 정의합니다. 이미지나 오디오와 같은 오브젝트가 나타내는 데이터 오브젝트를 식별하기 위해 사용자 정의 유형(UDT)을 사용합니다.
- 사용자 정의 유형의 데이터에 요청될 수 있는 세부 함수를 정의합니다. 예를 들어, 이미지 색상 수를 세는 함수를 정의하거나 오디오 샘플링율을 확보하는 함수를 정의할 수 있습니다. 다른 SQL 함수와 동일한 방법으로 SQL문에서 이러한 사용자 정의 함수(UDF)를 요청합니다.

DB2 Extender

DB2 Extender는 이미지, 오디오, 비디오 및 텍스트 오브젝트에 대한 UDT와 UDF를 작성합니다. UDT와 UDF는 다음과 같은 면에서 사용자를 돕는 중요한 지원 도구입니다.

- 응용프로그램 개발. Extender가 데이터 유형과 함수를 정의하기 때문에 응용프로그램에 그것들을 정의할 필요가 없습니다.
- 일관성 유지. 동일한 일련의 Extender UDT와 UDF들이 응용프로그램 모두에 사용 가능합니다. 이것은 대형 오브젝트를 처리하는 응용프로그램들 사이에서 다른 방법으로는 유지하기 어려울 수 있는 적정 수준의 일관성 레벨을 제공합니다.
- 강력한 조회 작성. UDF가 다른 SQL 함수와 동일한 방법으로 호출되기 때문에, 응용프로그램은 멀티 데이터 유형 조회를 포함할 수 있습니다. 하나의 SQL 문으로 이미지, 오디오, 비디오 및 텍스트 오브젝트를 전통적인 숫자와 문자 데이터와 함께 액세스할 수 있습니다. DB2 콜 레벨 인터페이스(DB2 CLI)에서 뿐만 아니라, Embedded SQL문에 UDF와 UDT를 지정할 수 있습니다.

그리고 Extender가 처리하는 오브젝트를 DB2 데이터베이스에 저장할 수 있기 때문에, 데이터베이스에 저장된 전통적인 데이터 유형에 대해서와 마찬가지로 동일한 보안, 무결성 및 복구 보호 기능이 이러한 오브젝트에 대해서도 적용됩니다.

또한, DB2 Extender는 DB2 Universal Enterprise Edition의 파티션된 데이터베이스 환경을 이용합니다. 파티션을 통해 응용프로그램이 단일 컴퓨터에서 사용하기에 너무 큰 데이터베이스를 사용할 수 있습니다. 또한 파티션을 통해 SQL 조작을 병렬로 수행할 수 있으므로, SQL 조회나 유틸리티의 속도가 빨라집니다.

강력한 정보 탐색의 새로운 방법

DB2 Extender는 응용프로그램에 정보 탐색에 있어 많은 융통성을 제공합니다. 응용프로그램은 데이터베이스에 저장된 전통적인 데이터 유형과 연관된 오브젝트를 탐색할 수 있습니다. 예를 들어, 그것들은 설명과 기록된 날짜로 오디오 클립을 탐색할 수 있습니다. 응용프로그램은 비디오 클립의 재생 시간과 같은 자체의 고유한 특성으로 오브젝트를 탐색할 수도 있습니다. Extender는 탐색에 사용된 이러한 특성을 자동으로 결정하고 저장합니다.

응용프로그램은 내용으로 이미지를 탐색할 수도 있습니다. 이미지를 탐색하는 데 시각적 예시를 사용하는 응용프로그램을 상상해 보십시오. 이런 응용프로그램으로, 예시 이미지를 선택할 수 있고 예시와 유사한 색상이나 텍스처를 갖는 다른 이미지를 찾을 수 있습니다. DB2 Extender의 이미지 내용별 조회(QBIC) 기능을 사용하면, 이러한 시각적 방법으로 이미지를 탐색하는 응용프로그램을 작성할 수 있습니다.

DB2 Extender

DB2 Extender는 별도의 Image Extender, Audio Extender, Video Extender 및 Text Extender로 구성됩니다.

이 책은 Image, Audio, Video Extender에 대한 내용을 다룹니다. 이 책에서 『Extender』나 『DB2 Extender』는 특별한 지시가 없는 한 Image, Audio, Video Extender를 의미합니다. Text Extender에 대한 자세한 정보는 *Text Extender 관리 및 프로그래밍*을 참조하십시오. XML Extender에 대해서는 *XML Extender 관리 및 프로그래밍*을 참조하십시오.

SDK 및 런타임 환경

DB2 Extender 설치 패키지는 소프트웨어 개발 프로그램 킷(SDK)과 클라이언트 및 서버 런타임 환경을 제공합니다. DB2 Extender SDK를 설치한 클라이언트 또는 서버 머신에서 DB2 Extender 응용프로그램을 개발할 수 있습니다.

DB2 Extender 클라이언트 런타임 코드 및 서버 런타임 코드를 포함하는 서버 머신에서 DB2 Extender 응용프로그램을 수행할 수 있습니다. (클라이언트 런타임 코드는 서버 런타임 코드를 설치할 때 자동으로 설치됩니다.) 또한 DB2 Extender 클라이언트 런타임 코드가 설치된 클라이언트 머신에서 DB2 Extender 응용프로그램을 수행할 수도 있습니다. 클라이언트 머신에서 Extender 응용프로그램을 수행할 경우, 서버에 연결할 수 있는지 확인해야 합니다.

Extender 사용

DB2 응용프로그램에서 Extender UDF를 호출하거나 DB2 명령행 처리기를 사용하여 대화식으로 이들을 호출할 수 있습니다.

Extender는 또한 다음과 같은 응용프로그램 프로그래밍 인터페이스(API)를 제공합니다.

- 이미지, 오디오 및 비디오 데이터용 데이터베이스를 준비하고 유지보수하는 관리 API.
- 이미지를 표시하고 비디오와 오디오 클립을 재생하는 표시 및 재생 API.
- 이미지를 준비하고 내용에 의한 탐색을 요청하는 QBIC API(내용 탐색은 UDF를 통해 요청될 수도 있습니다).
- 비디오의 장면 변경에 따라 프레임의 순서를 식별하는 비디오 컷 검출 API.

DB2 Extender는 또한 사용자가 관리 명령을 발생하는 데 사용하는 명령행 처리기도 제공합니다. Extender에서 제공하는 명령행 처리기를 DB2에서 제공하는 명령행 처리기와 구분하기 위해, 이 책에서는 전자를『db2ext 명령행 처리기』, 후자를『DB2 명령행 처리기』라고 합니다.

예시

한 광고 대행사가 광고의 DB2 데이터베이스를 유지보수합니다. 예전에, 이 광고 대행사는 고객의 이름 및 광고가 완성된 날짜 등 각각의 광고 캠페인에 대해 숫자 및 문자 데이터를 저장했습니다. 이제 DB2 UDB 및 DB2 Extender를 설치하여, 이 대행사에서는 데이터베이스에 광고 내용을 저장할 수도 있게 되었습니다. 광고 내용은 인쇄 광고의 이미지, TV 광고 비디오 및 라디오 광고 녹음을 포함합니다. 7 페이지의 그림1에서와 같이, 모든 광고 관련 정보는 ADS라는 하나의 데이터베이스 테이블에 저장됩니다.

그림 1. 멀티미디어 데이터베이스 테이블. 테이블에는 전통적인 데이터 유형 뿐만 아니라 이미지, 오디오 및 비디오 데이터가 들어 있습니다. 비디오, 오디오 및 이미지가 보입니다.

예시 1: 특성에 의한 비디오 검색

광고 대행사에 있는 회계 관리자는 광고 시간이 30초 이하이면서, 1997년 IBM 회계용으로 작성된 비디오 광고를 찾아야 합니다.

8 페이지의 그림2는 비디오에 액세스하는 조회를 표시합니다. 조회에 있는 Video Extender UDF명 `Filename`과 `Duration`을 주목하십시오.

```
SELECT FILENAME(ADS_VIDEO)
FROM ADS
WHERE CLIENT='IBM' AND
SHIP_DATE>='01/01/1997' AND
DURATION(ADS_VIDEO) <=30
```

그림 2. 비디오에 액세스하는 조회

조회는 원하는 비디오의 파일 이름을 리턴합니다. 그러면 회계 관리자는 자신이 선호하는 비디오 플레이어를 시작하고 각각의 비디오 파일의 내용을 재생할 수 있습니다.

그림2는 회계 관리자가 대화식으로 실행할 수 있는 조회의 예입니다. 일반적으로, 회계 관리자는 비디오를 찾고 재생하는 응용프로그램을 사용합니다. 예를 들어, 그림3은 C로 코딩된 그런 응용프로그램의 일부 키 요소를 나타냅니다. 응용프로그램은 hvVid_fname이라는 DB2 호스트 변수에서 비디오 파일 이름을 검색합니다. 또한, 응용프로그램은 이름이 DBvPlay인 재생 API를 사용하여 비디오를 재생합니다.

```
#include <dmbvideo.h>

int count = 0;

EXEC SQL BEGIN DECLARE SECTION;
char hvClient[30];                      /*client name*/
char hvCampaign[30];                    /*campaign name*/
char hvSdate[8];                        /*ship date*/
char hvVid_fname [251]                  /*video file name*/
EXEC SQL END DECLARE SECTION;


EXEC SQL DECLARE c1 CURSOR FOR
     SELECT CLIENT, CAMPAIGN, SHIP_DATE, FILENAME(ADS_VIDEO)
       FROM ADS
       WHERE CLIENT='IBM' AND
         SHIP_DATE≥'01/01/1997' AND
         DURATION(ADS_VIDEO)≤30
FOR FETCH ONLY;
```

그림 3. 비디오에 액세스하고 재생하는 응용프로그램 (1/2)

```
EXEC SQL OPEN c1;
for (;;){
        EXEC SQL FETCH c1 INTO :hvClient, :hvCampaign,
                          :hvSdate, :hvVid_fname;

    if (SQLCODE != 0)
              break;

    printf("\nRecord %d:\n", ++count);
    printf("Client = '%s'\n", hvClient);
    printf("Campaign = '%s'\n", hvCampaign);
    printf("Sdate = '%s'\n", hvSdate);

    rc=DBvPlay(NULL,MMDB_PLAY_FILE,hvVid_fname,MMDB_PLAY_WAIT);
}
EXEC SQL CLOSE c1;
```

그림 3. 비디오에 액세스하고 재생하는 응용프로그램 (2/2)

예시 2: 내용별 이미지 검색

광고 대행사에 있는 그래픽 일러스트레이터는 고객을 위해 새 인쇄 광고를 개발중입니다. 일러스트레이터는 광고 배경에 청색의 특정 음영의 사용을 원하고, 그 대행사가 이전에 작성한 인쇄 광고에 그 색이 사용되었는지 확인하려 합니다. 그렇게 하기 위해, 그래픽 일러스트레이터는 내용에 의해 이미지를 탐색하는 응용프로그램을 수행합니다. 이미지는 데이터베이스 테이블에 저장되어 있습니다(7 페이지의 그림1을 참조하십시오). 응용프로그램은 사용자가 시각적 예시, 즉 관심있는 색을 보여 주는 이미지를 제공하도록 요구합니다. 그러면 응용프로그램은 예시의 색을 분석하고 예시와 가장 비슷한 색의 이미지를 찾습니다.

10 페이지의 그림4는 시각적 예시와 해당 예시와 색이 가장 일치하는 검색된 이미지를 나타냅니다.

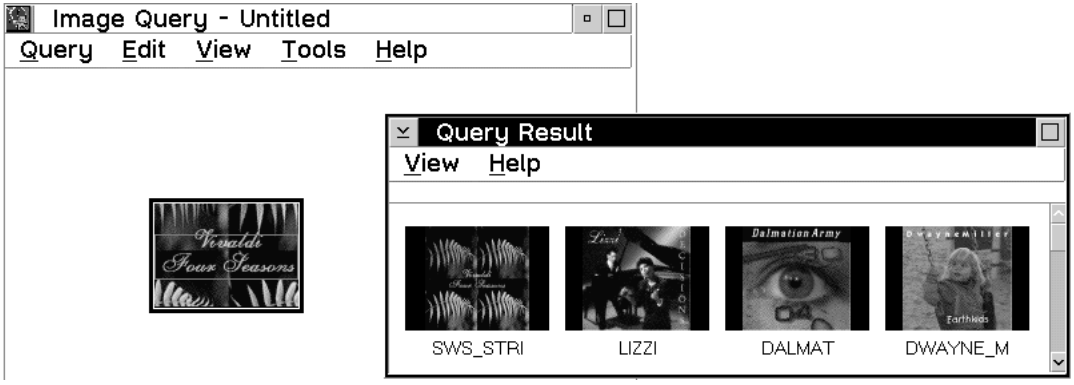


그림 4. 내용으로 이미지 찾기. 시각적 예시는 평균 색으로 이미지를 탐색하는 데 사용됩니다.

11 페이지의 그림5는 응용프로그램의 일부 키 요소를 나타냅니다. 응용프로그램은 QBIC 조회를 작성하기 위해 `QbQueryCreate`라는 QBIC API를, 색상 선택을 조회에 추가하기 위해 `QbQueryAddFeature` 및 `QbQuerySetFeatureData`를, 조회를 발행하기 위해 `QbQuerySearch`를, 그리고 조회를 삭제하기 위해 `QbQueryDelete`를 사용한다는 것에 주의하십시오. 응용프로그램은 또한 검색된 이미지를 표시하기 위해 `DBiBrowse`라는 그래픽 API를 사용합니다.

```
#include <dmbqbqpi.h>

#define MaxQueryReturns  10

static  SQLHENV    henv;
static  SQLHDBC    hdbc;
static  SQLHSTMT   hstmt;
static  SQLRETURN  rc;

void main(int argc, char* argv[])
{
 char           line[4000];
 char*          handles[MaxQueryReturns];
 QbQueryHandle  qHandle=0;
 QbResult       results[MaxQueryReturns];
 SQLINTEGER     count;
 SQLINTEGER     resultType=qbiArray;

 SQlAllocEnv(&henv);
 SQLAllocConnect(henv, &hdbc);
 rc = SQLConnect(hdbc, (SQLCHAR*)"qtest", SQL_NTS,
              (SQLCHAR*)"", SQL_NTS, (SQLCHAR*)"", SQL_NTS);

 if (argc !=2) {
 printf("usage: query colorname\n");
          exit(1);
 }

 QbImageSource is;
 is.type = qbiSource_AverageColor;

 /* run the get color subroutine */
 getColor(argv[1], is.average.Color);

 QbQueryCreate(&qhandle);
 QbQueryAddFeature(qhandle, "QbColorFeatureClass");
 QbQuerySetFeatureData(qhandle, "QbColorFeatureClass",&is);
 QbQuerySearch(qhandle, "ADS", "ADS_IMAGE", 10, 0, resultType
                              &count, results);
 for (int j = 0; j <count; j++) {
 printf(j,":\n");

 DBiBrowse("usr/local/bin/xv %s", MMDB_PLAY_HANDLE, handles[j],
                MMDB_PLAY_WAIT);
 }
```

그림 5. 내용으로 이미지를 찾는 응용프로그램 (1/2)

```
QbQueryDelete(qhandle);

 SQLDisconnect(hdbc);
 SQLFreeConnect(hdbc);
 SQLFreeEnv(henv);
}
```

그림 5. 내용으로 이미지를 찾는 응용프로그램 (2/2)

운영 환경

DB2 Extender 버전 7은 클라이언트/서버 환경에서 DB2 Universal Database 버전 7.1(또는 이상)과 함께 작동합니다. 지원되는 플랫폼에 대해 요구되는 최소 버전 및 릴리스 레벨은 DB2 Universal Database 버전 7.1에 대해 요구되는 것과 동일합니다.

지원되는 클라이언트 플랫폼은 OS/2, AIX, Windows NT® 버전 4.0 이상, Windows 95, Windows 98, Solaris Operating Environment, HP-UX입니다.

지원되는 서버는 OS/2, AIX, Windows NT 버전 4.0 이상, Solaris Operating Environment, HP-UX입니다.

13 페이지의 그림6은 지원되는 플랫폼을 보여줍니다.

또다른 DB2 Extender 제품인, OS/390용 DB2 Universal Database Extender는 OS/390 클라이언트와 서버를 지원합니다. OS/390용 DB2 Universal Database Extender에 대해서는 *DB2 Universal Database for OS/390 Image, Audio, and Video Extenders Administration and Programming* 또는 *DB2 Universal Database for OS/390 Text Extender Administration and Programming*을 참조하십시오.

DB2 Extender는 단일 파티션 데이터베이스 환경에서 작동할 수 있습니다.

EEE 전용: Extender는 또한 AIX, Solaris Operating Environment, Windows NT 버전 4.0 이상 플랫폼에서는 다중 파티션 데이터베이스 환경에서도 작동할 수 있습니다.

DB2 Extender를 다중 파티션 데이터베이스 환경에서 운영하려면 DB2 Universal Database Enterprise-Extended Edition와 함께 사용해야 합니다.

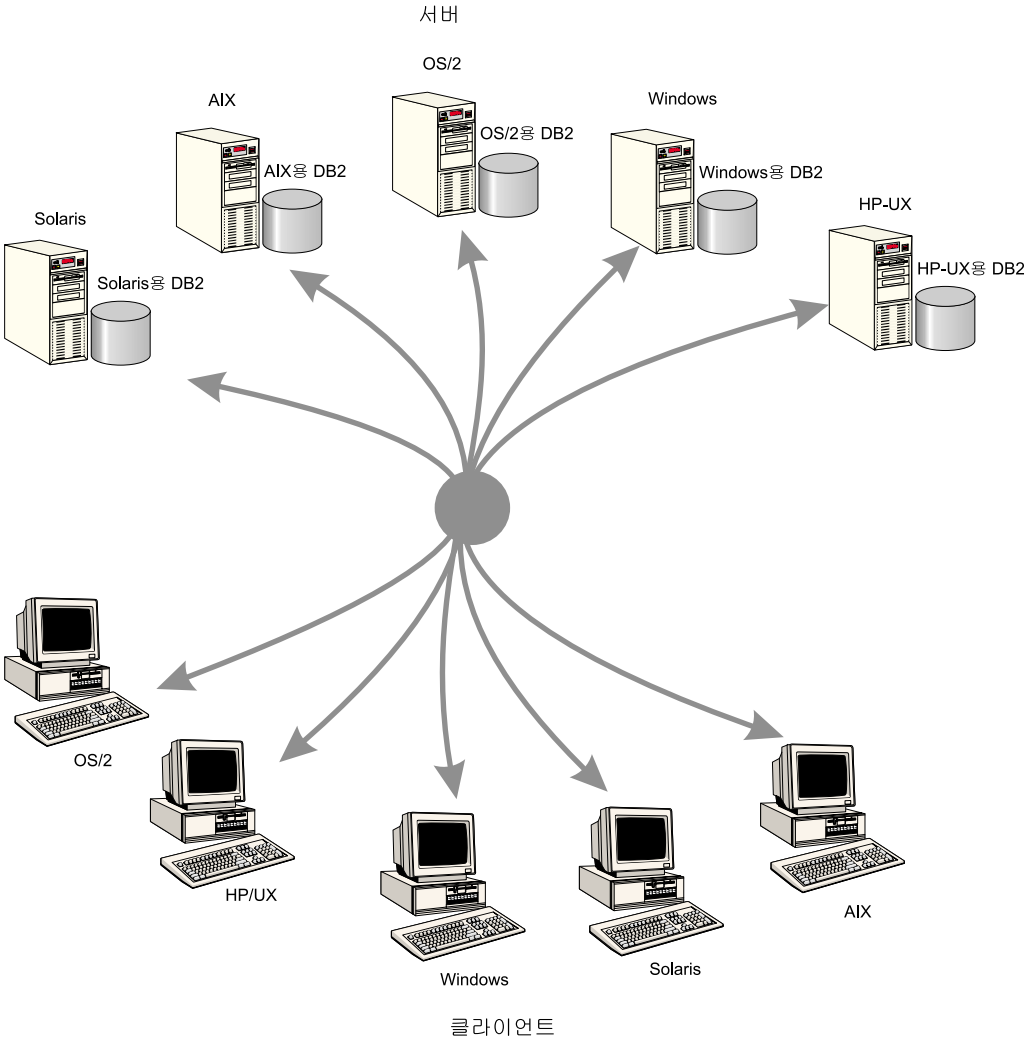


그림 6. DB2 Extender 플랫폼

제2장 DB2 Extender 개념

이 장은 DB2 Extender를 사용하기 전에 이해해야 하는 개념을 설명합니다.

객체 지향 개념에 대해서는 *DB2 응용프로그램 개발 안내서*를 참조하십시오.

객체 지향 개념

DB2는 실제적이거나 추상적인 모든 것을 조작과 데이터 값의 세트로 구성되는 하나의 오브젝트로 응용프로그램에서 표현할 수 있는 **객체 지향**이라는 개념을 지원합니다. 예를 들어, 문서는 문서 데이터와 파일링, 송신 및 인쇄와 같이 문서에서 수행될 수 있는 조작으로 구성된 문서 오브젝트로서 표현될 수 있습니다. 비디오 데이터와 비디오 클립은 비디오 클립 재생이나 특정 비디오 프레임 찾기와 같은 조작으로 구성된 비디오 오브젝트로 표현될 수 있습니다. 실제 오브젝트와 같이, 대표 오브젝트도 속성을 갖습니다. 예를 들어, 비디오 오브젝트는 압축 유형과 샘플링율과 같은 속성을 부여받을 수 있습니다.

오브젝트는 유형으로 함께 그룹화될 수 있습니다. 동일 유형의 오브젝트는 동일 속성을 갖고 같은 방식으로 행동합니다. 즉 그것들은 동일 조작에 연관됩니다. 예를 들어, 비디오 유형이 압축 유형 속성을 갖도록 정의된다면, 비디오 유형의 오브젝트는 모두 그러한 속성을 갖습니다. 비디오 유형의 한 오브젝트가 재생될 수 있다면, 비디오 유형의 오브젝트는 모두 재생될 수 있습니다.

객체 지향에 대한 DB2의 지원을 통해 테이블 컬럼에 오브젝트 유형의 인스턴스를 저장하고, SQL문에서 함수를 사용하여 그것들을 조작할 수 있습니다. 예를 들어, 테이블 컬럼에 비디오 오브젝트를 저장하고 SQL 함수를 사용하여 그것들을 조

작할 수 있습니다. 또한, 응용프로그램 간에 저장된 오브젝트의 속성과 행동을 공유할 수 있습니다. 응용프로그램 모두는 동일 오브젝트 유형에 대한 속성과 행동의 동일 세트를『참조합니다』.

비디오 오브젝트는 일반적으로 크고 복잡합니다. 이미지와 오디오 오브젝트도 역시 그렇습니다. 객체 지향에 대한 지원의 일부로서, DB2는 데이터베이스에 대형 오브젝트(LOB)를 저장할 수 있도록 합니다. 또한 사용자 정의 유형(UDT), 사용자 정의 함수(UDF) 및 트리거를 통해 LOB을 정의하고 조작하는 방법을 제공합니다.

대형 오브젝트

DB2로 데이터베이스에 다음과 같이 **대형 오브젝트**(LOB)를 저장할 수 있습니다.

- 2진 대형 오브젝트(BLOB)
- 문자 대형 오브젝트(CLOB)
- 2바이트 문자 대형 오브젝트(DBCLOB)

BLOB은 2진 문자열입니다. 이미지, 오디오 및 비디오 오브젝트는 DB2 데이터베이스에 BLOB으로 저장됩니다. CLOB은 연관된 코드 페이지의 단일 바이트 문자로 구성된 문자열입니다. 이 데이터 유형은 단일 바이트 문자가 들어 있는 텍스트 오브젝트용으로 사용됩니다. DBCLOB은 연관된 코드 페이지의 2바이트 문자로 구성된 문자열입니다. 이 데이터 유형은 2바이트 문자가 사용되는 텍스트 오브젝트용으로 사용됩니다.

각각의 LOB은 길이가 최대 2기가바이트까지 가능합니다. 그러나, DB2는 테이블당 많은 LOB 컬럼을 허용합니다. 행당 최대 24GB의 LOB 공간과 테이블당 최대 4TB의 LOB 공간을 저장할 수 있습니다.

크기로 인해, LOB 내용을 사용자 테이블에 직접 저장하지 않습니다. 대신에, 각각의 LOB은 대형 오브젝트 설명자에 의해 테이블에서 식별됩니다. 설명자는 디스크 상의 임의의 장소에 저장된 대형 오브젝트에 액세스하는 데 사용됩니다.

DB2 Extender는 파일에 LOB 내용을 유지하고 데이터베이스에서 그것을 지시하게 하여 추가 융통성을 제공합니다. 사용자는 DB2 Extender 사용시 이것을 지정하여 오브젝트를 저장합니다.

사용자 정의 유형

이미지, 비디오 및 오디오 오브젝트는 데이터베이스에 BLOB으로 표현됩니다. **구별 유형**이라고도 하는 **사용자 정의 유형**(UDT)은 BLOB을 다른 것과 구별하는 방법을 제공합니다. 예를 들어, 한 UDT는 이미지 오브젝트용으로 작성될 수 있고 또다른 것은 오디오 오브젝트용으로 작성될 수 있습니다. BLOB으로 저장되더라도, 이미지와 오디오 오브젝트는 BLOB과 각각의 다른 것에 대해 구별되는 유형으로 다루어집니다.

사용자는 UDT를 SQL CREATE DISTINCT TYPE문으로 작성합니다. 예를 들어, 맵에서 지리학적인 특성을 처리하는 응용프로그램을 개발중이라고 가정하십시오. 다음과 같이 맵 오브젝트에 대한 map이라는 구별 유형을 작성할 수 있습니다.

```
CREATE DISTINCT TYPE map AS BLOB (1M)
```

맵 유형 오브젝트는 길이 1MB의 BLOB으로서 내부적으로 표현되나, 구별 유형의 오브젝트로 취급됩니다.

테이블 컬럼에 저장된 데이터를 설명하기 위해 SQL 내장 함수와 같이 UDT를 사용할 수 있습니다. 다음의 예시에서, 테이블은 맵 유형 데이터를 보유하기 위해 고안된 컬럼으로 작성됩니다.

```
CREATE TABLE places
  (locid     INTEGER NOT NULL,
   location  CHAR (50),
   grid      map)
```

각각의 DB2 Extender는 자신의 유형 즉, 이미지, 오디오 및 비디오에 대해 UDT를 작성합니다.

사용자 정의 함수

사용자 정의 함수(UDF)는 SQL 함수를 작성하는 방법으로, 이것은 DB2에 제공된 내장 함수 세트로 추가됩니다. 특히, 이미지, 오디오 및 비디오 오브젝트에만 고유한 조작을 수행하는 UDF를 작성할 수 있습니다. 예를 들어, 비디오의 압축 형식을 확보하거나 오디오의 샘플링율을 리턴하는 UDF를 작성할 수 있습니다. 이것은 특정 유형 오브젝트의 행동을 정의하는 방법을 제공합니다. 예를 들어, 비디오 오브젝트는 비디오 유형에 대해 작성된 함수 구문으로 행동하고, 이미지 오브젝트는 이미지 유형에 대해 작성된 함수 구문으로 행동합니다.

사용자는 SQL CREATE FUNCTION문으로 UDF를 작성합니다. 명령문은 다른 것들 사이에서, UDF가 적용될 수 있는 데이터 유형을 지정합니다. 예를 들어, 다음의 명령문은 맵의 축척을 계산하는 map_scale이라는 UDF를 작성합니다. UDF가 자신이 적용될 수 있는 데이터 유형으로서 map을 식별함을 유의하십시오. 함수를 구현하는 코드는 C로 작성되고 EXTERNAL NAME절에서 식별됩니다.

```
CREATE FUNCTION map_scale (map)
  RETURNS SMALLINT
  EXTERNAL NAME 'scale!map'
  LANGUAGE C
  PARAMETER STYLE DB2SQL
  NO SQL
  DETERMINISTIC
  NO EXTERNAL ACTION
```

UDF는 내장 함수와 동일한 방법으로 SQL문에 사용될 수 있습니다. 다음의 예에서, map_scale UDF는 grid라는 테이블 컬럼에 저장된 맵 축척을 리턴하는 SQL SELECT문에 사용됩니다.

```
SELECT map_scale (grid)
  FROM places
  WHERE location='SAN JOSE, CALIFORNIA'
```

각각의 DB2 Extender는 자신의 유형, 즉, 이미지 특정, 오디오 특정 및 비디오 특정 UDF에 대한 UDF 세트를 작성합니다. 테이블에 이미지 저장, 비디오의 프레임율 확보 또는 오디오에 대한 주석 추가 등의 Extender 함수를 요청하는 SQL문에 이러한 UDF를 사용합니다.

UDF 및 UDT 이름

DB2 함수의 전체 이름은 *schema-name.function-name*이며, 여기서 *schema-name*은 SQL 오브젝트에 논리적 그룹화를 제공하는 식별자입니다. DB2 Extender UDF용 스키마 이름은 MMDBSYS입니다. MMDBSYS 스키마 이름은 또한 DB2 Extender UDT에 대한 규정자입니다.

UDF나 UDT를 참조하는 어떠한 위치에서도 전체 이름을 사용할 수 있습니다. 예를 들어, MMDBSYS.CONTENT는 스키마 이름이 MMDBSYS이고 그 함수 이름이 CONTENT인 UDF를 식별합니다. MMDBSYS.DB2IMAGE는 스키마가 MMDBSYS이고 구별 유형 이름이 DB2IMAGE인 UDF를 식별합니다. UDF 또

는 UDT를 참조할 때 스키마 이름을 생략할 수도 있습니다. 이 경우, DB2는 함수 경로를 사용하여 원하는 함수 또는 구별 데이터 유형을 판별합니다.

함수 경로

함수 경로는 스키마 이름의 순서화된 목록입니다. DB2는 함수와 구별 데이터 유형에 대한 참조를 해석하기 위해 목록에 있는 스키마 이름의 순서를 사용합니다. SQL문 SET CURRENT FUNCTION PATH를 지정하여 함수 경로를 지정할 수 있습니다. 이것은 CURRENT FUNCTION PATH 특수 레지스터에 함수 경로를 설정합니다.

DB2 Extender의 경우, mmdbsys 스키마를 함수 경로에 추가하는 것이 좋습니다. 이렇게 하면 mmdbsys 접두부를 지정할 필요 없이 DB2 Extender UDF 및 UDT 이름을 입력할 수 있습니다. 다음은 mmdbsys 스키마를 함수 경로에 추가하는 예입니다.

```
SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH
```

mmdbsys로 로그온하는 경우, **mmdbsys를 함수 경로에서 첫번째 스키마로 추가하지 마십시오.** mmdbsys 사용자 ID로 로그온하는 경우, 함수 경로에서 첫번째 스키마가 mmdbsys로 설정됩니다. 그런 다음, SET CURRENT FUNCTION PATH 문에서 함수 경로의 첫번째 스키마를 mmdbsys로 설정하려 하면, 함수 경로가 두 개의 mmdbsys 스키마로 시작됩니다. 이는 오류 조건입니다.

오버로드된 함수 이름

함수 이름은 **오버로드**될 수 있습니다. 이것은 복수 UDF가 심지어 동일 스키마에서도 동일한 이름을 가질 수 있음을 의미합니다. 단, 두 개의 함수가 동일 **시그니처**를 가질 수는 없습니다. 시그니처는 정의된 데이터 유형의 모든 함수 매개변수로 병합된 규정 함수 이름입니다.

트리거

트리거는 테이블 변경으로서 활성화되는 조치 세트를 정의합니다. 트리거는 입력 데이터 유효성 확인, 새로 삽입된 행에 대한 값의 자동 생성, 상호 참조 목적으로 다른 테이블에서 읽기 또는 감사 목적으로 다른 테이블에 쓰기 등과 같은 조치를 수행하는 데 사용될 수 있습니다. 트리거는 종종 업무 규칙을 강제하거나, 무결성 점검에 사용됩니다.

SQL CREATE TRIGGER문을 사용하여 트리거를 작성합니다. 다음의 명령문은 부품 목록에 대해 업무 규칙을 강제하는 트리거를 작성합니다. 트리거는 가지고 있는 수가 비축된 최대 수의 10 퍼센트 이하인 경우 부품을 재주문합니다.

```
CREATE TRIGGER reorder
       AFTER UPDATE OF on_hand, max_stocked ON parts
       REFERENCING NEW AS n_row
       FOR EACH ROW MODE DB2SQL

       WHEN (n_row.on_hand < 0.10 * n_row.max_stocked)
       BEGIN ATOMIC
              VALUES(issue_ship_request(n_row.max_stocked -
                                  n_row.on_hand,
                                  n_row.partno));
       END
```

DB2 Extender는 데이터베이스에 저장된 이미지, 오디오 및 비디오 데이터에 관한 정보를 기록하기 위해 관리 지원 테이블을 작성하고 유지보수합니다(이러한 테이블에 대한 자세한 정보는 『관리 지원 테이블』을 참조하십시오). 데이터베이스에서 이미지, 오디오 또는 비디오 데이터가 삽입되거나, 갱신되거나 또는 삭제될 때, Extender는 트리거를 사용하여 이러한 테이블을 갱신합니다.

Extender 데이터 구조

Image, Audio 및 Video Extender는 관리 지원 테이블 및 핸들을 작성하고 사용하여 이미지, 오디오 및 비디오 데이터를 저장하고 액세스합니다. Image Extender는 또한 QBIC 카탈로그를 작성하고 사용하여 내용에 의해 이미지에 액세스합니다. 또한 Video Extender는 색인 파일 및 컷 카탈로그를 사용하여 비디오의 장면 변경에 대한 정보를 액세스합니다.

관리 지원 테이블

메타데이터 테이블이라고도 하는 **관리 지원 테이블**에는 Extender가 이미지, 오디오 및 비디오 오브젝트에 대한 사용자 요청을 처리하는 데 필요한 정보가 들어 있습니다. 관리 지원 테이블에 있는 정보는 종종 『메타데이터』로 참조됩니다.

22 페이지의 그림7에 나타난 것과 같이, 일부 관리 지원 테이블은 Extender용으로 사용 가능한 사용자 테이블 및 컬럼을 식별합니다. 이러한 테이블은 사용 가능한 컬럼 내의 오브젝트에 관한 속성 정보를 보유하기 위해 작성된 다른 관리 지원

테이블을 참조합니다. 이러한 테이블에서, Extender는 특정 Extender 정의 데이터 유형에 고유한 속성에 관한 정보와 Extender 데이터 유형 간에 공통되는 속성에 관한 정보를 유지보수합니다. 예를 들어, Image Extender는 이미지의 폭, 높이 및 색상 수에 관한 정보와 오브젝트를 데이터베이스로 가져왔거나 오브젝트를 마지막으로 갱신한 사람의 ID와 같은 이미지, 오디오 및 비디오 오브젝트에 공통되는 속성에 관한 정보를 유지보수합니다.

관리 지원 테이블은 BLOB 형식으로 저장된 오브젝트 내용을 포함할 수도 있습니다. 또한, 오브젝트는 파일에 보관되고 관리 지원 테이블에 의해 참조될 수 있습니다. 예를 들어, 비디오 클립은 관리 지원 테이블에 BLOB으로 저장되거나, 테이블이 참조하는 파일에 보관될 수 있습니다.

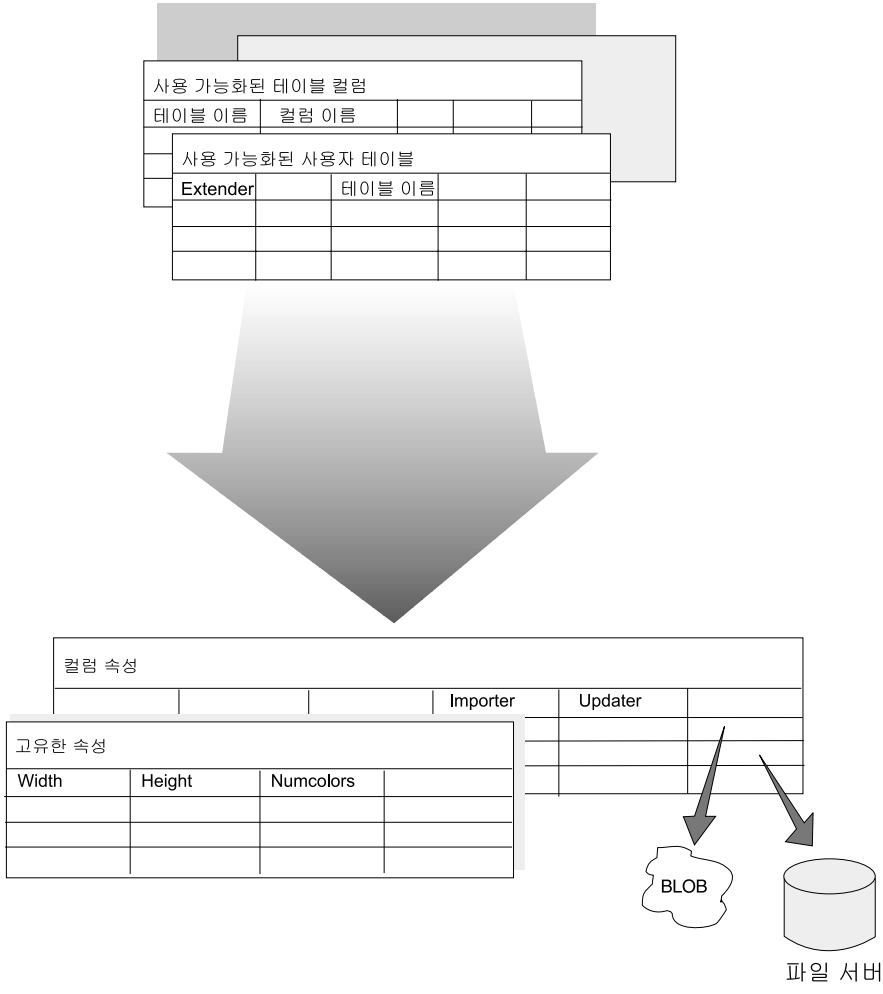


그림 7. 관리 지원 테이블

핸들

사용자 테이블에 이미지, 오디오 또는 비디오 오브젝트 저장시, 오브젝트는 테이블에 실제로 저장되지 않습니다. 그 대신에, Extender는 오브젝트를 표현하기 위해 **핸들**이라고 하는 문자열을 작성하고 테이블에 핸들을 저장합니다. Extender는 관리 지원 테이블에 오브젝트를 저장하거나, 파일에 오브젝트 내용을 보관하는 경우 관리 지원 테이블에 파일 식별자를 저장합니다. 또한 관리 지원 테이블에 오브젝

트 속성과 핸들을 저장합니다. 이러한 방법으로, Extender는 관리 지원 테이블에 저장된 오브젝트 정보에 사용자 테이블에 저장된 핸들을 링크할 수 있습니다. 그림8은 사용자 테이블에서 두 개의 이미지에 대해 저장된 정보를 보여 줍니다.

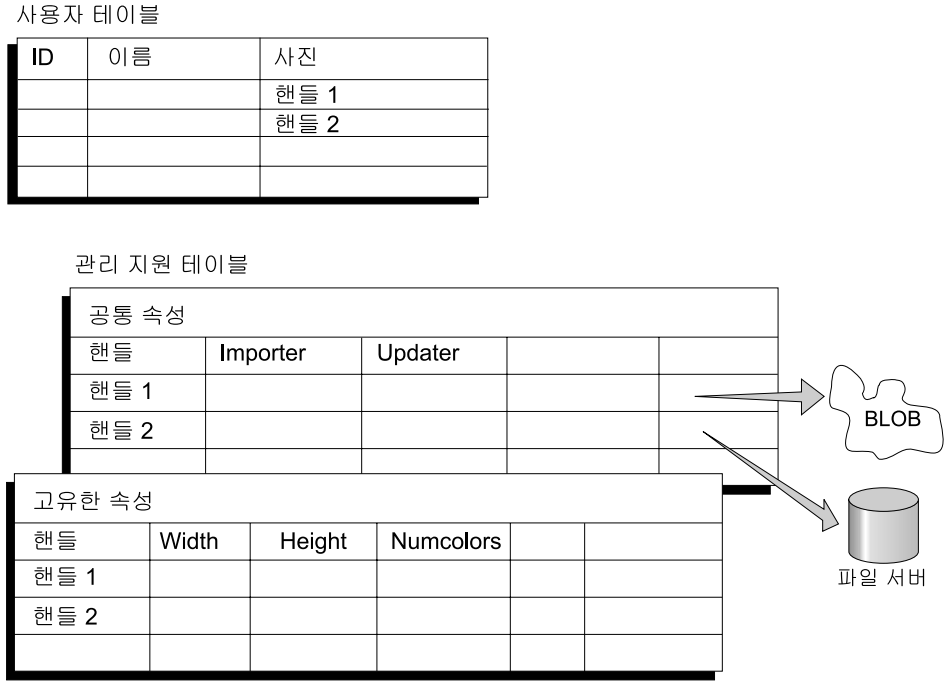


그림 8. 핸들

QBIC 카탈로그

QBIC 카탈로그는 이미지의 시각적 특성에 관한 데이터를 보유하는 파일 세트입니다. Image Extender는 이 데이터를 내용으로 이미지를 검색하는 데 사용합니다.

내용으로 검색하려는 사용자 테이블에 각각의 이미지 컬럼에 대한 QBIC 카탈로그를 작성하십시오. QBIC 카탈로그 작성시 Image Extender로 분석 및 저장하고, 나중에 조회하려는 데이터에 대한 특성을 식별하십시오. 카탈로그가 작성된 후, QBIC 카탈로그에서 특성을 추가하거나 삭제할 수도 있습니다.

QBIC 카탈로그는 다음의 이미지 특성에 대한 데이터를 보유할 수 있습니다.

평균 색상 이미지 픽셀 번호에 의해 구분된 이미지 내 모든 픽셀에 대한 색상값의 합(픽셀은 색상과 명암이 할당될 수 있는 최소한의 이미지 요소입니다). 예를 들어, 이미지 50%가 파란 픽셀로 구성되어 있고 나머지 부분은 빨간 픽셀이라면, 이미지는 보라색의 평균 색상값을 갖습니다. 평균 색상은 주 색상이 있는 이미지를 검색하는 데 사용됩니다. 이미지가 주 색상을 갖는다면, 평균 색상은 주 색상과 유사할 것입니다.

히스토그램 색상

64 색상의 스펙트럼에 대해 이미지의 색상 분포를 측정합니다. 64 색상 각각에 대해, 히스토그램 색상은 이미지에서 그러한 색상이 있는 픽셀의 비율을 식별합니다. 예를 들어, 이미지의 히스토그램 색상은 흰색 40%, 파란색 50%, 빨간색 10% 픽셀일 수 있습니다. 이미지 픽셀 중 아무 것도 히스토그램 스펙트럼의 나머지 색상을 갖지 않습니다. 히스토그램 색상은 다양한 색을 갖는 이미지를 검색하는 데 사용됩니다.

위치 색상 이미지에서 지정된 영역에 있는 픽셀의 평균 색상값. 예를 들어, 이미지의 상위 오른쪽 구석에 밝은 노란색 태양이 나타날지 모릅니다. 이러한 이미지 영역의 위치 색상은 밝은 노란색입니다. 위치 색상은 특정 영역에 주 색상이 있는 이미지를 검색하는 데 사용됩니다.

텍스처 이미지의 거친 정도, 대비 및 방향성을 측정합니다. 거친 정도는 이미지에서 반복 항목의 *크기*를 표시합니다(예를 들어, 자갈 대 둥근 돌). 대비는 이미지에서 밝기 변화를 식별합니다(밝음 대 어두움). 방향성은 방향이 이미지에서 우세한지(피켓 울타리의 수직 방향처럼) 우세하지 않은지(모래 이미지처럼) 표시합니다. 텍스처는 특정 패턴이 있는 이미지를 검색하는 데 사용됩니다.

내용으로 검색하는 데 이미지를 사용 가능하게 하려면, 이미지를 카탈로그화해야 합니다. 이미지 카탈로그화시, Image Extender는 이미지에 대한 특성값을 계산하여 이미지를 분석하고 값을 QBIC 카탈로그에 저장합니다.

내용으로 이미지 검색시, 조회는 탐색용 특성(평균 색상과 같은), 각 특성용 소스(예시 이미지와 같은) 및 카탈로그화된 이미지의 목표 세트를 식별합니다. Image Extender는 소스의 특성값을 계산하고 그것을 목표 이미지에 대해 카탈로그화된 특성값과 비교합니다. 그 다음 목표 이미지의 특성값이 소스와 얼마나 유사한지 표시하는 스코어를 계산합니다.

Image Extender를 소스와 가장 유사한 특성 이미지를 리턴해야 할 수 있습니다. Image Extender는 각각의 이미지 핸들과 이미지 스코어를 리턴할 것입니다. 또한 Image Extender가 단일 이미지의 스코어만을 리턴하게 할 수도 있습니다.

비디오 색인

비디오 색인은 Video Extender가 비디오 클립에서 특정 컷이나 프레임을 찾는 데 사용할 파일입니다.

Video Extender는 비디오에서 장면 변경을 검출할 수 있습니다. **장면 변경**은 하나의 비디오 클립에서 두 개의 연속된 프레임 간의 상당한 차이점이 있는 한 지점입니다. 이것은, 예를 들어, 카메라가 비디오의 녹화시 뷰 포인트를 변경할 때 발생합니다. 두 장면 변경 사이의 프레임은 **컷(shot)**을 구성합니다.

비디오 클립에서 컷 또는 심지어 개별 프레임을 찾으려면, Video Extender 장면 검출 기능을 사용할 수 있습니다. 이렇게 하려면, Extender는 컷이나 프레임에 대한 색인 정보가 필요합니다. 이 색인 정보는 **색인 파일**에 저장됩니다.

컷 카탈로그

컷 카탈로그는 비디오 클립에 컷에 대한 데이터를 저장하는 데 사용됩니다. 컷 카탈로그는 데이터베이스나 파일에 저장될 수 있습니다.

파일에 저장된 컷 카탈로그에는 다음의 컷 관련 데이터가 들어 있습니다.

- 컷 카탈로그 파일 이름
- Video Extender가 컷을 검출하는 방법을 제어하는 값. 예를 들어, 컷에서 프레임의 최소 번호
- 컷용 대표 프레임으로 저장될 프레임 수와 프레임 종류를 제어하는 값
- 컷 번호

- 시작 프레임 번호
- 끝 프레임 번호
- 대표 프레임 번호
- 대표 프레임의 내용이 있는 파일 이름

컷 카탈로그 파일의 데이터나 데이터베이스에 저장된 컷 카탈로그의 뷰(view)를 액세스할 수 있습니다. 뷰에는 다음의 컷 관련 데이터에 대한 컬럼이 있습니다.

- 컷 핸들
- 비디오 테이블 이름
- 비디오 컬럼 이름
- 비디오 핸들
- 비디오 파일 이름
- 시작 프레임 번호
- 끝 프레임 번호
- 대표 프레임 번호
- 대표 프레임 데이터
- 주석

파티션된 데이터베이스 개념(EEE 전용)

DB2 Extender는 DB2 Extended Enterprise Edition으로 조작할 수 있으며, 이러한 방법에 의해 DB2 Extended Enterprise Edition에서 제공하는 파티션된 데이터베이스 지원을 활용할 수 있습니다.

파티션된 데이터베이스는 둘 이상의 독립 머신 간에 분산되어 있는 데이터베이스입니다. 일반 사용자와 응용프로그램 개발자들에게 데이터베이스는 단일 머신 상의 단일 데이터베이스처럼 보입니다. 응용프로그램은 너무 커서 하나의 머신으로는 처리가 불가능한 데이터베이스를 파티션에 의해 효율적으로 사용할 수 있습니다.

파티션된 데이터베이스는 둘 이상의 파티션으로 구성됩니다. 각 파티션은 자체의 **데이터베이스 파티션 서버**에 의해 관리됩니다. 데이터베이스 파티션 서버에는 데이터베이스 관리 프로그램과 이 프로그램이 관리하는 데이터 및 시스템 자원 집합이 포

함됩니다. 일반적으로, 각 머신에는 하나의 데이터베이스 파티션 서버가 지정됩니다. 그러나, 단일 머신 상에 여러 개의 데이터베이스 파티션 서버를 가질 수 있습니다. 각 데이터베이스 파티션 서버는 전체 데이터베이스의 일부를 보유합니다. 데이터베이스 파티션 서버를 **노드**라고도 합니다.

28 페이지의 그림9에 나와 있는 바와 같이, 데이터베이스 파티션은 논리적으로 그룹화하여 이름을 지정할 수 있습니다. 데이터베이스 파티션의 각 그룹을 **노드 그룹**이라고 합니다. 노드 그룹을 정의함으로써, 예를 들어, 응용프로그램 조회를 선택된 데이터베이스 파티션으로 제한하고, 이로써 트랜잭션 시간을 단축시킬 수 있습니다. 각 노드 그룹에는 하나의 데이터베이스 파티션만이 포함되거나 여러 개의 데이터베이스 파티션이 포함될 수 있습니다. 노드 그룹에 여러 개의 데이터베이스 파티션이 포함될 경우, 이를 **다중 파티션 노드 그룹**이라고 합니다. 하나의 다중 파티션 노드 그룹으로 이름 지정되는 모든 데이터베이스 파티션은 동일한 데이터베이스에 상주해야 합니다.

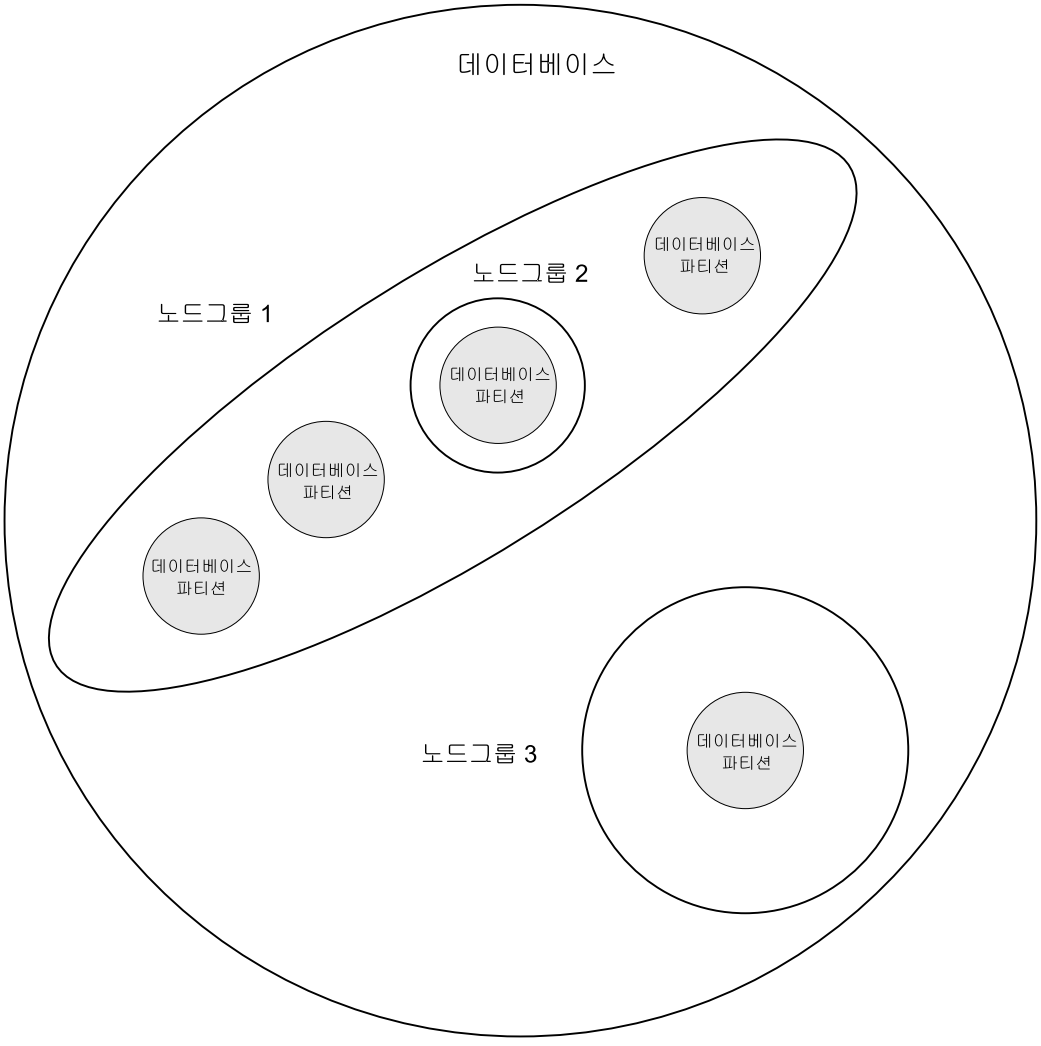


그림 9. 데이터베이스 내의 노드 그룹

파티션된 데이터베이스 시스템에서 Extender를 사용하면 다음이 가능합니다.

- 여러 파티션 간에 데이터를 분산함으로써 입/출력 및 처리 병목 현상을 줄일 수 있습니다.
- 머신을 더 추가하고 머신 간에 데이터를 재분산함으로써 데이터베이스 크기를 증가시킬 수 있습니다.

병렬 처리

파티션된 데이터베이스는 여러 개의 CPU를 사용하여 정보 요청에 응답할 수 있습니다. 검색 및 갱신 요청은 자동으로 부속 요청으로 할당되어 각 머신 상의 데이터베이스 파티션 서버에서 병렬로 실행됩니다.

파티션된 데이터베이스 시스템에서의 처리 능력에 대한 설명으로서, 단일 파티션 데이터베이스에서 스캔하고자 하는 레코드가 100,000,000개라고 가정하십시오. 이를 스캔하려면, 단일 데이터베이스 관리 프로그램이 100,000,000개의 레코드를 검색해야 합니다. 이제 이들 레코드가 20개의 데이터베이스 파티션 서버에 고루 분산된다고 가정하십시오. 각 데이터베이스 관리 프로그램은 단지 5,000,000개의 레코드만을 스캔하면 됩니다. 각 데이터베이스 관리 프로그램은 동시에 동일한 속도로 스캔하며, 스캔을 완료하는 데 요구되는 시간은 단일 파티션 시스템이 이 타스크를 처리하는 데 요구되는 시간의 약 5%에 불과할 것입니다.

규모 확장 가능성

데이터베이스 크기가 증가함에 따라, 데이터베이스 시스템에 데이터베이스 파티션 서버를 추가하여 성능을 향상시킬 수 있습니다. 이러한 프로세스를 데이터베이스 시스템 **규모 확장**(scaling)이라고 합니다.

데이터베이스를 규모 확장할 때, 데이터베이스 파티션 서버를 추가하면, 이 서버가 다시 데이터베이스 시스템에 있는 기존의 각 데이터베이스에 데이터베이스 파티션을 추가합니다. 그러면 그 데이터베이스에 대한 기존의 노드 그룹에 새로운 데이터베이스 파티션을 지정할 수 있습니다. 마지막으로, 이 노드 그룹의 데이터를 재분산하여 새로운 데이터베이스 파티션을 활용할 수 있습니다.

파티션된 데이터베이스 환경에서의 DB2 Extender 사용

파티션된 데이터베이스 환경에서 DB2 Extender를 사용함으로써, 특히 LOB 조작을 훌륭히 지원하는 기능을 활용할 수 있습니다. 데이터베이스를 다수의 머신에 분산시킬 수 있으므로, 대형 저장소인 LOB(각각 최대 2기가바이트에 달함)을 하나의 데이터베이스에 저장할 수 있습니다.

또한, DB2 Extender는 DB2 Extended Enterprise Edition에 의해 관리되는 SQL 조작의 병렬 처리에 참여합니다. DB2 Extended Enterprise Edition이 병렬 조회를 수행하면, 이 조회 내의 DB2 Extender UDF 역시 개별 데이터베이스 파티션에서 병렬로 수행됩니다.

보안과 복구

DB2 데이터베이스에 BLOB으로 저장된 이미지, 오디오 및 비디오 오브젝트에는 전통적 숫자와 문자 데이터에 제공된 동일한 보안과 복구 보호 기능이 적용됩니다. 또한, 메타데이터 테이블에 이러한 오브젝트용으로 정보가 저장됩니다. 사용자는 오브젝트를 선택, 삽입 또는 갱신하기 위해 필요한 특권을 가져야 합니다.

사용자는 사용자 테이블에서 오브젝트를 선택, 삽입, 갱신 또는 삭제는 UDF를 발행합니다. 요청된 조작을 수행하려면, UDF는 오브젝트에 대한 속성 정보를 보유하는 관리 지원 테이블을 액세스하고, 필요에 따라 갱신할 수 있어야 합니다. Extender는 UDF를 사용하여 사용자가 사용자 테이블에 관한 적합한 특권을 갖는 경우 관리 지원 테이블에 이러한 조작을 수행할 수 있습니다.

일부 Extender 관련 관리 조작에는 DBADM 권한이 필요합니다. DB2 Extender 관리 API가 요구하는 권한에 대해서는 291 페이지의 『제16장 응용프로그램 프로그래밍 인터페이스(API)』을 참조하십시오. DB2 Extender 관리 명령이 요구하는 권한에 대해서는 291 페이지의 『제16장 응용프로그램 프로그래밍 인터페이스(API)』을 참조하십시오.

이미지, 오디오 또는 비디오의 내용이 데이터베이스에서 참조되는 파일에 저장될 때, 오브젝트용 메타데이터는 DB2에 의해 보호됩니다. 파일은 PUBLIC, 즉 모든 사용자가 읽을 수 있는 디렉토리에 있어야 합니다.

BLOB과 메타데이터는 DB2의 다른 데이터와 동일한 방법으로 백업되고 복구될 수 있습니다. 파일에 저장된 오브젝트 내용은 DB2에 없는 툴을 사용하여 백업되고 복구될 수 있습니다. 또한, QBIC 카탈로그와 비디오 색인은 DB가 없는 툴을 사용하여 백업되고 복구될 수 있습니다. QBIC 카탈로그의 백업에 대한 정보는 149 페이지를 참조하십시오. 비디오 색인의 백업에 대한 정보는 200페이지를 참조하십시오.

제3장 Extender 작동 방법

DB2 Extender는 이미지, 오디오 및 비디오 데이터 요청을 처리하기 위한 여러 작업을 수행합니다. Extender 작동 방법을 설명하는 좋은 방법은 이것을 사용할 때 수행하는 것들을 조사하는 것입니다. 이 장은 Image 및 Audio Extender가 있는 시나리오를 설명합니다. 사용자가 수행하는 조작과 Extender 응답 방법에 대해 논의합니다.

Extender 시나리오

회사 인사부에서 각 직원의 사진이 들어 있는 직원 데이터베이스를 (AIX용 DB2에서) 작성하려고 합니다.

사진이 있는 데이터베이스: 그림10에 나타난 것처럼 데이터베이스의 직원 테이블은 직원 사진뿐 아니라 각 직원의 이름과 ID를 포함합니다.

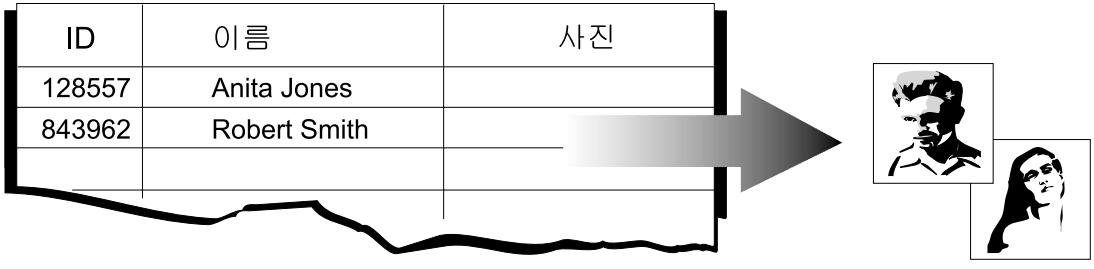


그림 10. 직원 테이블

이미지 처리를 위해 직원 데이터베이스를 준비하려면, 시스템 관리자, 즉 SYSADM 권한을 가지고 있는 누군가가 Extender 서비스를 시작합니다. 그런 다음, 시스템 관리자는 데이터베이스를 작성하고 그것을 Image Extender에서 사용할 수 있도록 해야 합니다.

데이터베이스 관리자(DBA) 또는 동등한 권한을 갖는 사람은 직원 테이블을 작성한 후, 그것과 직원 사진 컬럼을 Image Extender에서 사용할 수 있도록 합니다.

음향이 있는 데이터베이스: 직원 데이터베이스와 직원 테이블이 이미지 프로세스용으로 준비된 후, 인사부는 각 직원에 대한 오디오 레코딩을 테이블에 추가할 것을 결정합니다. 이것은 그림11에 나타납니다.

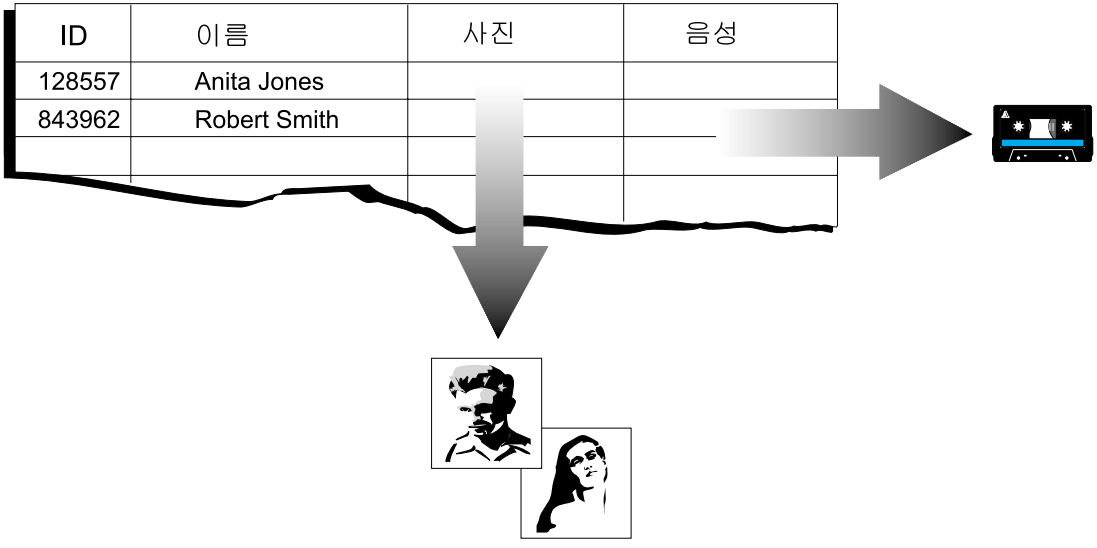


그림 11. 오디오 컬럼이 추가된 직원 테이블

시스템 관리자는 새로운 컬럼을 추가하여 테이블을 변경하고, 데이터베이스, 테이블 및 컬럼을 Audio Extender에서 사용할 수 있도록 합니다.

그 다음 인사부의 사용자는 테이블에 데이터를 입력하고 데이터를 선택 및 표시하며 데이터를 갱신하고 데이터를 삭제합니다.

Extender 서비스 시작

Extender는 그 조작의 일부로서 서버에서 서비스를 사용합니다. 이러한 서비스가 서버의 표준『시작』조작 기능으로 이미 사용 가능화되지 않았다면, 시스템 관리자가 이를 시작합니다.

시스템 관리자의 수행 작업: 시스템 관리자는 Extender 인스턴스 소유자로서 AIX로 로그온합니다. 그런 다음, 시스템 관리자는 서버에서 다음의 명령을 발행합니다.

```
DMBSTART
```

결과: Extender 서비스는 서버에서 Extender 인스턴스용으로 시작됩니다. DMBSTART 명령은 이미 수행중이 아닌 경우, DB2 인스턴스도 시작합니다.

데이터베이스 준비

시스템 관리자는 직원 데이터베이스를 작성하고 Image Extender에서 사용할 수 있게 합니다.

시스템 관리자가 수행하는 작업: 시스템 관리자는 다음과 같은 SQL문을 사용하여 AIX용 DB2에서 직원 데이터베이스를 작성합니다.

```
CREATE DATABASE personnl           /*name of the database*/
       ON /persdb                  /*name of the database directory*/
       WITH "Personnel database"   /*comment*/
```

시스템 관리자는 데이터베이스에 연결하여 Image Extender에서 사용할 수 있게 합니다. 시스템 관리자는 db2ext 명령행 처리기를 사용하여 다음과 같은 명령을 발행합니다.

```
CONNECT TO personnl
ENABLE DATABASE FOR DB2IMAGE
```

결과: ENABLE DATABASE 명령에 대한 응답으로, Image Extender는 다음을 수행합니다.

- 이미지 오브젝트에 대해 DB2IMAGE라는 사용자 정의 유형을 작성합니다.
- 이미지 오브젝트용 관리 지원 테이블을 작성합니다.
- 이미지 오브젝트용 사용자 정의 함수를 작성합니다. UDF는 표1에 나열됩니다.

표 1. Image Extender로 작성된 사용자 정의 함수

UDF 이름	설명
Comment	사용자 주석을 확보하거나 갱신합니다.
Content	이미지 내용을 확보하거나 갱신합니다.
DB2Image	이미지 내용을 저장합니다.
Filename	이미지가 있는 파일 이름을 확보합니다.
Format	이미지 형식을 확보합니다(예: GIF).
Height	픽셀 단위로 이미지 높이를 확보합니다.
Importer	이미지 반입자의 사용자 ID를 확보합니다.

표 1. Image Extender로 작성된 사용자 정의 함수 (계속)

UDF 이름	설명
ImportTime	이미지를 가져올 때의 시간소인을 확보합니다.
NumColors	이미지에 있는 색상 수를 확보합니다.
QbScoreFromName	이름 지정된 조회를 사용하여 이미지의 유사 스코어를 확보합니다.
QbScoreFromStr	조회 문자열를 사용하여 이미지의 유사 스코어를 확보합니다.
QbScoreTBFromName	이름 지정된 조회를 사용하여 이미지 컬럼용 유사 스코어 테이블을 확보합니다.
QbScoreTBFromStr	조회 문자열을 사용하여 이미지 컬럼용 유사 스코어 테이블을 확보합니다.
Replace	이미지에 대한 내용과 사용자 주석을 갱신합니다.
Size	바이트 단위로 이미지 크기를 확보합니다.
Thumbnail	이미지의 썸네일 크기 버전을 확보합니다.
Updater	이미지 갱신자의 사용자 ID를 확보합니다.
UpdateTime	이미지 갱신시의 시간소인을 확보합니다.
Width	픽셀 단위로 이미지 폭을 확보합니다.

테이블 준비

DBA는 직원 테이블을 작성하고 그것과 사진 컬럼을 Image Extender에서 사용할 수 있도록 합니다.

DBA가 수행하는 작업: 편의상, DBA는 다음과 같은 SQL문을 사용하여 mmdbsys 스키마를 현재 함수 경로에 추가합니다.

```
SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH
```

이것은 UDT와 UDF 이름을 mmdbsys 스키마 이름으로 시작하지 않고 지정할 수 있도록 합니다.(mmdbsys 스키마는 함수 경로의 첫번째 스키마일 필요가 없습니다.) UDT와 UDF 이름에 대한 자세한 정보는 18 페이지의 『UDF 및 UDT 이름』을 참조하십시오.

DBA는 직원 테이블을 작성합니다. DBA는 DB2 명령행 처리기를 사용하여 다음 SQL문을 발행합니다.

```
CREATE TABLE   employee            /*name of the table*/
       (id      CHAR(6)            /*employee identification*/
       name    VARCHAR(40)         /*employee name*/
       picture DB2IMAGE)           /*employee picture*/
```

그 다음 DBA는 db2ext 명령행 처리기를 사용하여 아래의 명령을 발행합니다.

```
ENABLE TABLE employee FOR DB2IMAGE
ENABLE COLUMN employee picture FOR DB2IMAGE
```

결과: ENABLE TABLE 명령에 대한 응답으로, Image Extender는 다음을 수행합니다.

- 사용할 직원 테이블을 식별합니다.
- 사용 가능한 컬럼에 이미지 오브젝트에 대한 속성 정보를 보유하는 관리 지원 테이블을 작성합니다.

ENABLE COLUMN 명령에 대한 응답으로, Image Extender는 다음을 수행합니다.

- 사용할 사진 컬럼을 식별합니다.
- 트리거를 작성합니다. 이러한 트리거는 직원 테이블에 대한 삽입, 갱신 및 삭제 조작에 대한 응답으로 여러 관리 지원 테이블을 갱신합니다.

테이블 변경

DBA는 오디오 컬럼을 직원 테이블에 추가하고 그것을 Audio Extender에서 사용할 수 있도록 합니다.

DBA가 수행하는 작업: DBA는 db2ext 명령행 처리기를 사용하여 직원 데이터베이스를 Audio Extender에서 사용할 수 있게 합니다.

```
ENABLE DATABASE FOR DB2AUDIO
```

그런 다음 DBA는 다음과 같은 SQL문을 발행하여 직원 테이블을 변경합니다. DBA는 DB2 명령행 처리기를 사용하여 SQL문을 발행합니다.

```
ALTER TABLE employee               /*name of the table*/
      ADD voice DB2AUDIO           /*employee audio recording*/
```

DBA는 Audio Extender로 db2ext 명령행 처리기를 사용하여 직원 테이블과 음성 컬럼을 사용할 수 있도록 합니다.

```
ENABLE TABLE employee FOR DB2AUDIO
ENABLE COLUMN employee voice FOR DB2AUDIO
```

결과: ENABLE DATABASE 명령에 대한 응답으로, Audio Extender는 다음을 수행합니다.

- 오디오 오브젝트에 대해 DB2AUDIO라는 사용자 정의 유형을 작성합니다.
- 오디오 오브젝트용 관리 지원 테이블을 작성합니다.
- 오디오 오브젝트용 사용자 정의 함수를 작성합니다. UDF는 표2에 나열됩니다.

표 2. Audio Extender로 작성된 사용자 정의 함수

UDF 이름	**설명**
AlignValue	오디오의 샘플값 당 바이트를 확보합니다.
BitsPerSample	오디오를 나타내는 데 사용되는 비트 수를 확보합니다.
BytesPerSec	오디오의 평균 초당 바이트 수를 확보합니다.
Comment	사용자 주석을 확보하거나 갱신합니다.
Content	오디오 내용을 확보하거나 갱신합니다.
DB2Audio	오디오 내용을 저장합니다.
Duration	오디오 재생 시간을 확보합니다.
Filename	오디오가 있는 파일 이름을 확보합니다.
FindInstrument	오디오에 특정 악기를 레코드하는 오디오의 트랙 수를 확보합니다.
FindTrackName	오디오 레코딩에 이름 지정된 트랙의 트랙 수를 확보합니다.
Format	오디오 형식을 확보합니다.
GetInstruments	오디오에 레코드된 악기 이름을 확보합니다.
GetTrackNames	오디오의 트랙 이름을 확보합니다.
Importer	오디오 반입자의 사용자 ID를 확보합니다.
ImportTime	오디오를 가져올 때의 시간소인을 확보합니다.
NumAudioTracks	오디오에 레코드된 트랙 수를 확보합니다.
1NumChannels	오디오의 채널 수를 확보합니다.
Replace	오디오 레코딩에 대한 내용과 사용자 주석을 갱신합니다.
SamplingRate	오디오의 샘플링율을 확보합니다.
Size	바이트 단위로 오디오 크기를 확보합니다.
TicksPerQNote	오디오 레코딩에 사용된 1/4 노트 당 시계 틱 수를 확보합니다.

표 2. Audio Extender로 작성된 사용자 정의 함수 (계속)

UDF 이름	설명
TicksPerSec	오디오 레코딩에 사용되는 초당 시계 틱 수를 확보합니다.
Updater	오디오 갱신자의 사용자 ID를 확보합니다.
UpdateTime	오디오 갱신시의 시간소인을 확보합니다.

ENABLE TABLE 명령에 대한 응답으로, Audio Extender는 다음을 수행합니다.

- 사용할 직원 테이블을 식별합니다.
- 사용 가능한 컬럼에 오디오 오브젝트에 대한 속성 정보를 보유하는 관리 지원 테이블을 작성합니다.

ENABLE COLUMN 명령에 대한 응답으로, Audio Extender는 다음을 수행합니다.

- 사용할 음성 컬럼을 확인합니다.
- 트리거를 작성합니다. 이러한 트리거는 직원 테이블에 삽입, 갱신 및 삭제 조작에 대한 응답으로 여러 관리 지원 테이블을 갱신합니다.

테이블에 데이터 삽입

사용자는 직원 테이블에 Anita Jones에 대한 레코드를 삽입합니다. 레코드에는 Anita의 ID(128557), 이름, 사진 및 음성 레코딩이 있습니다. 소스 이미지와 오디오 내용은 서버의 파일에 있습니다. 이미지는 BLOB으로 테이블에 저장됩니다. 오디오 내용은 서버 파일에 남습니다(테이블 항목이 서버 파일을 참조합니다).

사용자 수행 작업: 사용자는 38 페이지의 그림12에 나타난 명령문을 포함하는 응용프로그램을 사용하여 직원 테이블에 레코드를 삽입합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvInt_Stor;
long hvExt_Stor;
EXEC SQL END DECLARE SECTION;

hvInt_Stor = MMDB_STORAGE_TYPE_INTERNAL;
hvExt_Stor = MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                           /*Image Extender UDF*/
           CURRENT SERVER,                  /*database server name in*/
                                            /*CURRENT SERVER register*/
           '/employee/images/ajones.bmp'    /*image source file*/
           'ASIS',                          /*keep the image format*/
           :hvInt_Stor,                     /*store image in DB as BLOB*/
           'Anita''s picture'),             /*comment*/
        DB2AUDIO(                           /*Audio Extender UDF*/
           CURRENT SERVER,                  /*database server name in*/
                                            /*CURRENT SERVER register*/
           '/employee/sounds/ajones.wav',   /*audio source file*/
           'WAVE',                          /* audio format */
           :hvExt_Stor,                     /*retain content in server file*/
           'Anita''s voice')                /*comment*/
      );
```

그림 12. 테이블에 데이터 삽입

결과: INSERT문에 있는 DB2Image UDF에 대한 응답으로, Image Extender는 다음을 수행합니다.

- 높이, 폭 및 색상 수와 같은 이미지 속성을 소스 이미지 파일 헤더에서 읽습니다.
- 이미지에 대해 고유한 핸들을 작성하고 관리 지원 테이블에 기록합니다.
  - 이미지용 핸들
  - 시간소인
  - 바이트 단위의 이미지 *크기*
  - 내용『Anita's picture』
  - 이미지 내용

  이미지 소스가 ajones.bmp라는 서버 파일에 있습니다. 파일 내용은 BLOB으로 관리 지원 테이블 레코드에 삽입됩니다. 저장된 이미지의 형식은 소스 이미지와 같습니다. 형식 변환이 수행되지 않습니다.
- 관리 지원 테이블에 레코드를 저장합니다. 레코드에는 이미지의 썸네일 *크기* 버전뿐 아니라 이미지에 있는 색상 수와 같은 이미지 지정 속성이 있습니다.

INSERT문에서 DB2Audio UDF에 대한 응답으로, Audio Extender는 다음을 수행합니다.

- 오디오 파일 헤더에서 오디오의 트랙 및 채널 수와 같은 오디오 속성을 읽습니다.
- 오디오에 대해 고유한 핸들을 작성합니다.
- 관리 지원 테이블에 레코드를 저장합니다. 레코드에는 다음이 포함됩니다.
  - 오디오용 핸들
  - 시간소인
  - 바이트 단위의 오디오 *크기*
  - 내용『Anita's voice』

  오디오 내용은 ajones.wav라는 서버 파일에 있습니다. 관리 지원 테이블 레코드가 파일을 참조합니다.
- 다른 관리 지원 테이블에 레코드를 저장합니다. 레코드에는 오디오 샘플링율과 같은 오디오 지정 속성이 있습니다.

트리거는 다양한 관리 지원 테이블에 이미지 및 오디오 속성 데이터를 삽입합니다.

테이블에서 데이터 선택

사용자는 Robert Smith의 이미지와 음성 레코딩이 얼마나 최근에 직원 테이블에 저장되었는지에 대한 정보를 검색합니다.

사용자 수행 작업: 사용자는 40 페이지의 그림13에 표시된 SQL문을 포함하는 응용프로그램을 사용하여 정보를 확보합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
char[255]  hvImg_Time;
char[255]  hvAud_Time;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT IMPORTTIME(PICTURE),     /*when image was stored*/
       IMPORTTIME(VOICE)                 /*when audio was stored*/
       INTO :hvImg_Time, :hvAud_Time
       FROM EMPLOYEE
       WHERE NAME='Robert Smith';
```

그림 13. 테이블에서 데이터 선택

결과: PICTURE 컬럼용 ImportTime UDF에 대한 응답으로, Image Extender는 이미지가 저장된 날짜와 시간을 포함하는 시간소인을 리턴합니다. VOICE 컬럼용 ImportTime UDF에 대한 응답으로, Audio Extender는 음성 레코딩이 저장된 날짜와 시간을 포함하는 시간소인을 리턴합니다.

오브젝트의 표시 및 재생

사용자가 Robert Smith의 이미지를 표시하고 Robert Smith의 음성 레코딩을 재생합니다. 이미지는 BLOB으로 직원 테이블에 저장됩니다. 음성 레코딩에 대한 내용은 서버 파일에 있습니다.

사용자 수행 작업: 사용자는 41 페이지의 그림14에 표시된 SQL문을 포함하는 응용프로그램을 사용하여 이미지를 표시하고 음성 레코딩을 재생합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
char hvImg_hdl [251];
char hvAud_hdl [251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT PICTURE,                /*Get image handle*/
                VOICE                   /*Get audio handle*/
       INTO :hvImg_hdl, :hvAud_hdl
       FROM EMPLOYEE
       WHERE NAME='Robert Smith';

rc=DBiBrowse(
        NULL,                           /*Use default image browser*/
        MMDB_PLAY_HANDLE,               /*Use handle*/
        hvImg_hdl,                      /*Image handle*/
        MMDB_PLAY_NO_WAIT);             /*Run browser independently*/

rc=DBaPlay(
        NULL,                           /*Use default audio player*/
        MMDB_PLAY_HANDLE,               /*Use handle*/
        hvAud_hdl,                      /*Audio handle*/
        MMDB_PLAY_WAIT);                /*Wait for player to end*/
                                        /*before continuing*/
```

그림 14. 오브젝트 표시와 재생

결과: DB2는 Robert Smith의 이미지와 음성 레코딩 핸들을 검색합니다. DBiBrowse API에 대한 응답으로, Image Extender는 검색된 이미지 핸들과 연관된 이미지 내용을 확보합니다. Image Extender는 데이터베이스에서 이미지 내용을 검색하고 그것을 이미지 브라우저를 사용하여 표시할 수 있도록 임시 클라이언트 파일에 둡니다. NULL 매개변수는 사용자 시스템용 기본 이미지 브라우저가 사용될 것임을 표시합니다. 브라우저는 호출 프로그램과 독립적으로 수행되며, 이는 호출 프로그램이 계속하기 전에 이미지 브라우저의 종료를 대기하지 않을 것임을 의미합니다.

DBaPlay API에 대한 응답으로, Audio Extender는 검색된 오디오 핸들과 연관된 오디오 파일 이름을 확보하고 그 파일 이름을 오디오 플레이어에 전달합니다. NULL 매개변수는 사용자 시스템용 기본 오디오 플레이어가 사용될 것임을 표시합니다. 호출 프로그램은 계속하기 전에 오디오 플레이어가 종료하기를 대기할 것입니다.

테이블의 데이터 갱신

Anita Jones는 직원 테이블에서 자신의 사진을 최신 사진으로 바꿉니다. 더욱 새로운 사진 내용은 서버 파일에 있습니다.

사용자 수행 작업: 사용자는 그림15에 표시된 SQL문을 포함하는 응용프로그램을 사용하여 직원 테이블의 사진을 대체합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  char hvComment [16385];
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

strcpy(hvComment, "Picture taken at Anita's promotion");
hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL UPDATE EMPLOYEE
       SET PICTURE=REPLACE(
              PICTURE,                      /*image handle*/
              '/myimages/newone.bmp',       /*source image content*/
              'BMP',                        /*source format*/
              :hvStorageType,               /*store image in table as BLOB*/
              :hvComment)                   /*replace comment*/
       WHERE NAME='Anita Jones';
```

그림 15. 테이블에 데이터 갱신

결과: UPDATE문의 Replace UDF에 대한 응답으로, Image Extender가 새로운 이미지의 속성을 읽습니다. Image Extender는 새로운 이미지의 속성을 사용하여 기존 이미지에 대한 관리 지원 테이블에 저장된 속성을 갱신합니다. 이미지 소스는 newone.bmp라는 서버 파일에 있습니다. 파일 내용은 이전 이미지의 BLOB 내용을 대체하며 BLOB으로서 관리 지원 테이블 레코드에 삽입됩니다.

트리거는 다양한 관리 지원 테이블에 있는 이미지 속성 데이터를 대체합니다.

테이블에서 데이터 삭제

직원 테이블에서 Anita Jones의 레코드를 삭제합니다.

사용자 수행 작업: 사용자는 다음과 같은 SQL문을 포함하는 응용프로그램을 사용하여 직원 테이블에서 레코드를 삭제합니다.

```
DELETE FROM EMPLOYEE
       WHERE NAME='Anita Jones';
```

결과: 트리거는 여러 관리 지원 테이블에서 Anita Jones에 대한 항목을 삭제합니다.

제2부 이미지, 오디오 및 비디오 데이터 관리

제4장 관리 개요

이 장에서는 DB2 Extender를 사용하는 응용프로그램 작성시 포함되는 관리 타스크의 개요를 제공합니다.

DB2 Extender는 대부분의 관리 타스크를 수행하기 위한 두 가지 방법을 제공합니다.

- 관리 응용프로그램 프로그래밍 인터페이스(API). C 언어 프로그램에 DB2 Extender API를 포함할 수 있습니다. 이러한 API에 대한 참조 정보는 291 페이지의 『제16장 응용프로그램 프로그래밍 인터페이스(API)』를 참조하십시오.
- 관리 명령. 관리 명령을 db2ext 명령행 처리기에 제출할 수 있습니다.이러한 명령은 DB2 명령행에서 실행되지 않습니다. 관리 명령 입력에 대한 지시사항과 추가 참조 정보에 대해서는 525 페이지의 『제17장 클라이언트용 관리 명령』을 참조하십시오.

DB2 Extender로 수행 가능한 관리 타스크

다음과 같은 5가지 범주의 관리 타스크가 있습니다.

- Extender 서비스 관리. DB2 Extender는 DB2 상단의 자체 서버에서 실행합니다. 응용프로그램이 Extender 데이터를 사용하려면, 시스템 관리자가 Extender 서비스를 시작하고, 사용자가 Extender 데이터를 보유하는 데이터베이스에 연결합니다.
- Extender 데이터에 대한 데이터 오브젝트 준비. 데이터베이스, 테이블 및 컬럼을 사용할 수 있게 하여 Extender 데이터를 보유할 수 있게 준비합니다. 데이터 오브젝트를 사용 가능하게 할 때, Extender는 Extender 데이터를 관리하기 위해 관리 지원 테이블(메타데이터 테이블이라고도 함)을 만들고 관리합니다.
- **EEE 전용.** 파티션된 환경에서의 Extender 데이터 재분산. 파티션된 데이터베이스에서 파티션을 추가하거나 삭제할 때, 새로운 노드 구성을 이용하기 위해 데이터를 재분산할 수 있습니다.

- 데이터 오브젝트와 미디어 파일 추적. DB2 Extender를 사용하는 응용프로그램을 디버그하므로, Extender 데이터에 대해 사용 가능한 데이터 오브젝트를 판별하는 데 유용합니다. 또한 사용자 테이블과 외부 미디어 파일 간의 상관 관계를 이해하는 데도 유용합니다.
- 관리 지원 테이블 정리. DB2 Extender로 작업함에 따라, 결과적으로 사용하지 않는 항목은 관리 지원 테이블에 모일 수 있습니다. 사용하지 않는 메타데이터를 삭제하여 성능을 향상시키고 저장 공간을 회수할 수 있습니다.

49 페이지의 표3은 Extender 데이터에 관련된 모든 타스크를 나열합니다. 테이블은 각각의 타스크를 수행하기 위해 제공되는 툴의 종류와 자세한 정보를 찾을 수 있는 장소를 지정합니다.

Extender API 컬럼에서 각각의 API문의 세 번째 문자는 x로 나타납니다. 이 문자는 사용중인 Extender에 따라 변경됩니다.

문자	**Extender**
a	Audio
i	Image
v	Video

예를 들어, 이미지 데이터에 대해 테이블을 사용 가능화하는 API는 DB**i**EnableTable이고, 오디오에 대해 테이블을 사용 가능화하는 API는 DB**a**EnableTable이며, 비디오에 대해 테이블을 사용 가능화하는 API는 DB**v**EnableTable입니다. Extender API 컬럼의 값이 아니오이면, 해당 타스크에 대한 Extender API가 없음을 나타냅니다. Extender 명령 컬럼의 값이 아니오이면 해당 타스크에 대한 Extender 명령이 없음을 나타냅니다.

QBIC에는 추가 관리가 필요합니다. Image Extender의 이미지 내용별 조회(QBIC) 기능을 사용하려고 한다면, QBIC 카탈로그의 작성과 같이 추가 관리 타스크를 수행해야 합니다. 이러한 타스크에 대한 정보는 145 페이지의 『제13장 내용으로 이미지 조회』를 참조하십시오.

표 3. DB2 Extender용 관리 타스크와 기능

타스크	Extender API	Extender 명령	참조
Extender 서비스 관리			
Extender 서비스 시작	아니오	DMBSTART	53페이지
Extender 서비스의 상태 확보	아니오	DMBSTAT	57 페이지
Extender 서비스 정지	아니오	DMBSTOP	55페이지
데이터베이스에 연결	아니오	CONNECT	53페이지
데이터베이스용 Extender 서비스 시작	아니오	START SERVER	55페이지
데이터베이스용 Extender 서비스 상태 확보	아니오	GET SERVER STATUS	57 페이지
데이터베이스용 Extender 서비스 정지	아니오	STOP SERVER	55페이지
Extender 인스턴스 작성 및 관리	아니오	DMBICRT, DMBILIST, DMBIDROP, DMBIMIGR	57페이지
멀티미디어 데이터용 데이터 오브젝트 준비			
데이터베이스 사용 가능화	DBxEnableDatabase	ENABLE DATABASE	61페이지
데이터베이스 사용 불가능화	DBxDisableDatabase	DISABLE DATABASE	69페이지
테이블 사용 가능화	DBxEnableTable	ENABLE TABLE	65페이지
테이블 사용 불가능화	DBxDisableTable	DISABLE TABLE	69페이지
컬럼 사용 가능화	DBxEnableColumn	ENABLE COLUMN	68페이지
컬럼 사용 불가능화	DBxDisableColumn	DISABLE COLUMN	69페이지
파티션된 환경에서의 Extender 데이터 재분산(EEE 전용)			
새로운 노드 그룹 구성에 기초한 Extender 데이터 재분산	DMBRedistribute	REDISTRIBUTE NODEGROUP	71페이지
데이터 오브젝트와 미디어 파일 추적			
데이터베이스가 사용 가능한지 확인	DBxIsDatabaseEnabled	GET EXTENDER STATUS	73페이지
테이블이 사용 가능한지 확인	DBxIsTableEnabled	GET EXTENDER STATUS	73페이지
컬럼이 사용 가능한지 확인	DBxIsColumnEnabled	GET EXTENDER STATUS	73페이지
규정자가 현재 사용자 ID인 테이블에서 파일을 참조하는 테이블 항목 찾기	DBxIsFileReferenced	아니오	74페이지

표 3. DB2 Extender용 관리 타스크와 기능 (계속)

타스크	Extender API	Extender 명령	참조
지정 규정자의 모든 테이블이나 데이터베이스에 관련된 모든 테이블에서 파일을 참조하는 테이블 항목 찾기	DBxAdminIsFileReferenced	아니오	74페이지
규정자가 현재 사용자 ID인 테이블에서 테이블 항목이 참조하는 파일 찾기	DBxGetReferencedFiles	GET REFERENCED FILES	76페이지
지정 규정자의 모든 테이블이나 데이터베이스에 관련된 모든 테이블에서 테이블 항목이 참조하는 파일 찾기	DBxAdminGetReferencedFiles	GET REFERENCED FILES	76페이지
규정자가 현재 사용자 ID인 모든 테이블에서 테이블 항목이 참조하는 액세스 불가능한 파일 찾기	DBxGetInaccessibleFiles	GET INACCESSIBLE FILES	77페이지
지정 규정자의 모든 테이블이나 데이터베이스에 관련된 모든 테이블에서 테이블 항목이 참조하는 액세스 불가능한 파일 찾기	DBxAdminGetInaccessibleFiles	GET INACCESSIBLE FILES	77페이지
관리 지원(메타데이터) 테이블 정리			
지정 사용자 테이블이나 규정자가 현재 사용자 ID인 모든 사용자 테이블에 대한 메타데이터 테이블 정리	DBxReorgMetadata	REORG	79페이지
지정 규정자의 모든 테이블이나 데이터베이스에 관련된모든 사용자 테이블에 대한 메타데이터 테이블 정리	DBxAdminReorgMetadata	REORG	79페이지

관리 타스크의 순서: 다음 목록은 처음으로 Extender 사용시 수행하는 관리 타스크의 순서 요약입니다. DB2 명령 또는 명령문을 사용하여 일부 타스크를 수행합니다. DB2 Extender를 사용하여 기타 타스크를 수행합니다. 이 순서는 DB2 시스템이 실행중임을 가정합니다.

필수 타스크:

1. Extender 서비스를 시작합니다.

2. 데이터베이스를 작성합니다(DB2 사용).
3. 데이터베이스에 연결합니다.
4. 데이터베이스를 사용 가능하게 합니다.
5. 테이블 및 컬럼을 작성합니다(DB2 사용).
6. 데이터베이스에서 테이블을 사용 가능하게 합니다.
7. 테이블에 있는 컬럼을 사용 가능하게 합니다.

선택적 타스크:

1. 데이터 오브젝트와 미디어 파일을 추적합니다.
2. 함수 경로를 설정합니다(DB2 사용).
3. 관리 지원 테이블을 정리합니다.

예시: 다음 5개의 장에서는 시스템 관리자(SYSADM) 또는 데이터베이스 관리자(DBA)가 타스크를 수행 중이라고 가정합니다. 일부 타스크는 DBA나 SYSADM 권한을 필요로 하지 않습니다.

예시에서는 DBA가 현재 함수 경로에 MMDBSYS 스키마를 추가한 것으로 가정합니다. 이것은 DBA가 MMDBSYS 스키마 이름의 접두부 없이 UDT 이름을 지정하도록 합니다. UDT 이름에 대한 자세한 정보는 18 페이지의 『UDF 및 UDT 이름』을 참조하십시오.

이 절의 많은 API 예시는 Extender가 제공하는 샘플 응용프로그램 코드에 기초합니다. 샘플 코드는 클라이언트에 있는 SAMPLES 서브디렉토리에 있습니다.

제5장 Extender 서버 관리

DB2 Extender는 DB2 클라이언트/서버 환경에서 실행합니다. 이러한 환경은 하나의 데이터베이스 서버와 하나 이상의 원격 데이터베이스 클라이언트로 구성됩니다. DB2 Extender 서비스는 서버에서 실행합니다. 이러한 서비스에 액세스하려면, 먼저 이들을 시작해야 합니다.

환경 설정 후, 클라이언트에서 Extender 서비스를 정지하고 다시 시작할 수 있습니다. 클라이언트 또는 서버에서 Extender 상태를 확보할 수 있습니다.

EEE 전용: 다중 파티션 환경에서는 데이터베이스 파티션을 추가 및 삭제할 수도 있습니다.

Extender 환경 설정

Extender 서비스를 시작하려면 서버의 운영 체제 명령행에서 DMBSTART 명령을 입력하십시오.

```
dmbstart
```

DMBSTART 명령은 Extender 데이터를 보유할 수 있도록 사용 가능화된 데이터베이스 모두에 대해 Extender 서비스를 시작합니다. 명령은 또한 DB2가 수행중이 아닌 경우, DB2를 시작합니다. 명령을 수행하려면 SYSADM, SYSCTRL 또는 SYSMAINT 권한이 필요합니다. AIX에서, 사용자는 Extender 인스턴스 소유자로서 로그온되어야 합니다.

여기에서 응용프로그램이 데이터베이스에 연결한다면, C 언어 응용프로그램은 API를 통해 Extender 서비스에 액세스할 수 있습니다. 마찬가지로 db2ext 명령행을 사용하려면, 작업하려는 데이터베이스에 연결해야 합니다. db2ext 명령행은 DB2 명령행에서 사용하는 것과 다른 독립 연결을 요청합니다.

클라이언트에서 db2ext 명령행 처리기를 열고 DB2 Extender CONNECT 명령을 수행하십시오. 다음의 예시에서, 명령은 PERSONNL 데이터베이스에 연결합니다. 이것은 ANPASS 암호를 사용하여 ANITAS 규정자로 테이블에 액세스합니다.

```
connect to personnl user anitas using anpass
```

EEE 전용: 파티션된 데이터베이스 환경에서 DB2 Extender를 사용할 경우, DMBSTART 명령은 해당 인스턴스에 대해 정의된 모든 데이터베이스 파티션 서버에서 Extender 서비스를 시작합니다. 하나의 데이터베이스 파티션 서버에서만 Extender 서비스를 시작하려면, 명령에 시작하고자 하는 노드를 지정하십시오. 아래의 예시는 노드 번호 2에서 Extender 서비스를 시작할 때의 입력을 보여줍니다.

```
dmbstart nodenum 2
```

EEE 전용: 단일 데이터베이스 파티션 서버를 시작하려면, 먼저 해당 노드에서 DB2를 시작해야 합니다.

이제 525 페이지의 『제17장 클라이언트용 관리 명령』에 나열된 DB2 Extender 명령의 나머지를 수행할 수 있습니다.

데이터베이스 파티션 추가 및 삭제 (EEE 전용)

파티션된 데이터베이스 환경에서 Extender를 사용하려면, Extender에 대해 정의된 파티션이 DB2에 대해 정의된 파티션과 일치해야 합니다. DMBSTART 명령은 현재 인스턴스에 대해 정의된 각각의 노드에서 Extender 서버를 시작합니다. 이 서버는 자신이 수행중인 노드가 최근에 작성되었는지의 여부를 자동으로 검출하여 필요한 초기화를 수행합니다. 노드가 DB2에서 삭제된 경우에는 그 노드와 연관된 Extender 파일을 수동으로 삭제해야 합니다.

파티션을 추가 및 삭제하기 위한 DB2 명령에 대한 자세한 정보는 *IBM DB2 Universal Database Extended Enterprise Edition 빠른 시작*을 참조하십시오.

다음은 데이터베이스 파티션을 추가하는 데 필요한 단계들입니다.

1. DB2NCRT 명령 또는 DB2START ADDNODE 명령을 사용하여 DB2용 파티션을 작성하십시오.

2. Extender 명령 DMBSTART NODENUM을 사용하여 Extender용 파티션을 작성하십시오.
3. DB2 명령 REDISTRIBUTE NODEGROUP으로 새로운 노드 구성을 이용하도록 DB2 데이터를 재분산하십시오.
4. Extender 명령 REDISTRIBUTE NODEGROUP으로 새로운 노드 구성을 이용하도록 Extender 데이터를 재분산하십시오.

다음은 데이터베이스 파티션을 삭제하는 데 필요한 단계들입니다.

1. DB2 명령 REDISTRIBUTE NODEGROUP으로 삭제할 파티션에서 DB2 데이터를 제거하도록 DB2 데이터를 재분산하십시오.
2. Extender 명령 REDISTRIBUTE NODEGROUP으로 삭제할 파티션에서 Extender 데이터를 제거하도록 Extender 데이터를 재분산하십시오.
3. DB2 명령 DB2NDROP 또는 DB2STOP DROP을 사용하여 DB2용 파티션을 삭제하십시오.
4. Extender 명령 DMBSTART NODENUM을 사용하여 Extender용 파티션을 삭제하십시오.
5. 삭제된 파티션과 연관된 Extender 파일을 수동으로 제거하십시오.

데이터베이스 파티션의 Extender 데이터는 node*num*이라고 하는 서브디렉토리에 있으며, *num*은 데이터베이스 파티션에 해당하는 노드 번호입니다. 이 서브디렉토리는 DB2MMDATAPATH 환경 변수의 값으로 지정된 디렉토리에 있습니다. 삭제된 데이터베이스 파티션의 Extender 데이터를 제거하려면, 해당 node*num* 서브디렉토리와 그 아래에 있는 모든 서브디렉토리를 삭제하십시오. (DB2MMDATAPATH에 관한 자세한 내용은 635 페이지의 『DB2MMDATAPATH 환경 변수의 사용법(EEE 전용)』을 참조하십시오.)

Extender 서버의 정지 및 시작

Extender 서비스를 사용하는 응용프로그램의 정지시, 서버는 그것을 명시적으로 정지할 때까지 또는 서버 컴퓨터가 다시 시작할 때까지 활동 상태로 남습니다. 서버 컴퓨터에서 운영 체제용 명령행에 DMBSTOP 명령을 입력하여 Extender 서버 모두를 정지할 수 있습니다.

클라이언트에서 Extender 서비스를 정지하고 다시 시작하려면, db2ext 명령행에서 STOP SERVER와 START SERVER 명령을 수행하십시오. 이러한 명령은 현재의 데이터베이스에 대해 Extender 서비스를 정지하고 시작합니다.

EEE 전용: 파티션된 데이터베이스 환경에서, DMBSTART는 인스턴스에 대해 정의된 모든 데이터베이스 파티션 서버나 단일 데이터베이스 파티션 서버만을 시작하는 데 사용할 수 있습니다. 매개변수가 없는 DMBSTART는 모든 데이터베이스 파티션 서버를 시작합니다. 하나의 데이터베이스 파티션 서버만을 시작하려면, 아래와 같이 시작하려고 하는 노드를 명령에 지정하십시오.

```
dmbstart nodenum 2
```

특정 노드에서 서버를 시작하였으면, 해당 서버를 데이터베이스에 다시 연결해야 합니다. 다음과 같이 Extender 명령 RECONNECT SERVER를 사용하십시오.

```
reconnect server at nodenum 2
```

EEE 전용: 파티션된 데이터베이스 환경에서 DB2 Extender를 사용할 경우, 매개변수가 없는 DMBSTOP 명령은 인스턴스에 대해 정의된 모든 데이터베이스 파티션 서버를 정지합니다. 하나의 데이터베이스 파티션 서버만을 정지하려면, 먼저 데이터베이스에서 해당 서버를 연결해제해야 합니다. 다음과 같이 Extender 명령 DISCONNECT SERVER를 사용하십시오.

```
disconnect server at nodenum 2
```

그런 다음, 정지할 노드를 지정하여 DMBSTOP 명령을 수행할 수 있습니다. 아래의 예는 노드 번호 2에서 Extender 서비스를 정지하기 위해 서버 명령행에 입력하는 명령을 보여줍니다.

```
dmbstop nodenum 2
```

EEE 전용: 데이터베이스가 유지보수 모드에서 수행중이 아닌 경우에는 특정 노드에서 DMBSTOP를 수행하지 마십시오. 또한, 노드가 작동 중지 상태일 때는 이 노드에서 Extender 활동이 트리거되지 않도록 해야 합니다. 그러지 않을 경우, 예기치 못한 동작이 발생할 수 있습니다.

서버 상태 표시

서버에서, DMBSTAT 명령으로 Extender 서버 상태를 표시할 수 있습니다. 예를 들어, 다음의 명령은 사용 가능한 데이터베이스와 Extender가 시작되고 수행중인지를 나열합니다. 이러한 명령을 수행하기 전에 서버에 연결하십시오.

```
dmbstat
```

클라이언트에서, GET SERVER STATUS 명령을 사용하여 데이터베이스용 Extender 서버의 상태를 확보할 수 있습니다. 예를 들어, 다음의 명령은 personnl 데이터베이스의 상태를 나열합니다.

```
get server status personnl
```

여러 서버 인스턴스 작성 및 관리

DB2 Extender 서버의 여러 인스턴스를 작성하고 사용할 수 있습니다. DB2 서버의 여러 인스턴스를 작성하였으면 여러 인스턴스를 작성해야 합니다. DB2 Extender 서버의 각 인스턴스는 DB2 서버의 인스턴스와 연관되며 이름이 같습니다. 또한, 시스템에서 사용할 수 있는 DB2 Extender 서버의 인스턴스들을 나열하고, 여러 인스턴스를 동시에 실행하며, 인스턴스를 제거할 수도 있습니다.

여러 DB2 Extender 서버 인스턴스 작성

초기 또는 기본 DB2 Extender 인스턴스는 DB2 Extender를 설치할 때 작성되며, 기본 DB2 인스턴스와 같은 이름이 지정됩니다. Windows 및 OS/2에서, 기본 DB2 Extender 인스턴스의 이름은 DB2입니다. UNIX에서, 기본 DB2 Extender 인스턴스의 이름은 초기 기본 DB2 인스턴스에 지정된 것과 같습니다. DB2 Extender 서버의 추가 인스턴스를 작성하려면, SYSADMIN 권한이 있어야 하며, UNIX에서는 루트 권한이 있어야 합니다.

DB2 Image, Audio, Video Extender 서버의 추가 인스턴스를 작성하려면 DMBICRT 명령을 사용하십시오. DB2 인스턴스 DEVINST에 대해 DB2 Extender 서버 인스턴스를 작성하려면, 운영 체제 명령행에서 다음을 입력하십시오.

```
dmbicrt  devinst
```

DMBICRT 명령을 실행할 때 인스턴스의 서브디렉토리가 작성되고, 그 인스턴스는 DB2 Extender에 의해 유지보수되는 인스턴스 목록에 추가됩니다.

EEE 전용:

- 기본 DB2 Extender 서버 인스턴스의 이름은 Windows에서 DB2MPP입니다.
- DMBICRT를 사용하여 DB2 Image, Audio, Video Extender 서버의 추가 인스턴스를 작성할 때, 파티션된 데이터베이스 환경에서 다양한 조작에 대해 DB2 Extender에서 사용하는 디렉토리를 지정해야 합니다. 이 디렉토리는 UNIX에서 DB2MMDATAPATH 환경 변수에, 그리고 Windows에서는 레지스트리 항목에 지정한 디렉토리입니다. 이것은 공유 디렉토리여야 하며 인스턴스의 모든 노드에 존재해야 합니다.
- 또한, Windows에서는 사용할 TCP/IP 포트의 범위도 지정해야 합니다. UNIX에서는 포트 범위가 `/etc/services` 파일에 추가되어야 합니다(572 페이지의 『DMBICRT』 참조).

인스턴스 나열

시스템에서 사용할 수 있는 DB2 Extender 서버의 모든 인스턴스들을 나열하려면 DMBILIST 명령을 사용하십시오. 사용 중인 인스턴스를 찾으려면, 다음 명령을 입력하십시오.

```
echo  %DB2INSTANCE%     (Windows 또는 OS/2에서)

echo $DB2INSTANCE       (UNIX에서)
```

동시에 여러 인스턴스 실행

DB2 Extender 서버의 여러 인스턴스를 동시에 실행하려면, 다음 단계를 수행하십시오.

Windows 또는 OS/2에서

명령행에서:

1. 다음을 입력하여 DB2INSTANCE 변수를 시작할 인스턴스의 이름으로 설정하십시오.

   ```
   set db2instance=instanceName
   ```

2. Extender 서비스를 시작하십시오.

UNIX에서

1. 인스턴스 소유자나 인스턴스에 대한 시스템 관리 권한을 갖는 사용자로 로그인하십시오.
2. 환경을 설정하십시오.
3. 데이터베이스 관리 프로그램을 시작하십시오.

현재 인스턴스 설정

인스턴스에 대한 서비스를 시작하거나 정지하는 명령을 실행할 때, 그 명령은 현재 인스턴스에 적용됩니다. DB2INSTANCE 변수를 인스턴스 이름으로 설정하여 사용할 DB2 Extender 서버의 인스턴스를 지정합니다.

인스턴스 제거

DB2 Extender 인스턴스를 제거하려면, 다음 단계를 수행하십시오.

1. 현재 인스턴스를 사용 중인 모든 응용프로그램을 중지하십시오.
2. DMBSTOP 및 db2ext TERMINATE 명령을 사용하여 Extender 서비스와 모든 db2ext 명령행 처리기 세션을 중지하십시오.
3. QBIC 카탈로그 파일처럼, 저장할 DB2 Extender 인스턴스 디렉토리에 있는 파일들을 백업하십시오. 이 디렉토리의 파일들은 인스턴스가 제거될 때 제거됩니다.
4. 제거할 인스턴스에 대해 DMBIDROP 명령을 입력하십시오. 예를 들어, DEVINST 인스턴스를 제거하려면 다음을 입력하십시오.

```
dmbidrop devinst
```

DMBIDROP 명령을 사용하여 DB2 Extender의 인스턴스를 제거할 경우 연관되는 DB2 인스턴스는 제거되지 않습니다. 연관되는 DB2 인스턴스는 별도로 제거해야 합니다. DB2 Extender와 연관되는 DB2 인스턴스를 제거할 경우, DB2 Extender 인스턴스는 제거되지 않습니다. 그러나, 이를 사용할 수는 없습니다.

인스턴스 이주

UNIX 시스템에서, DB2 UDB 및 DB2 Extender의 새 버전을 설치하고 나면, DB2 Extender 인스턴스를 이주해야 합니다.

이전 버전에서 작성된 기존의 DB2 Extender 인스턴스를 이주하려면 다음을 수행하십시오.

1. DB2 Extender 인스턴스와 연관되는 DB2 UDB 인스턴스를 이주하십시오.
2. DMBIMIGR 명령을 사용하여 그 인스턴스를 이주하십시오. 예를 들어, OLDINST 인스턴스를 이주하려면 다음을 입력하십시오.

```
dmbimigr  oldinst
```

제6장 Extender 데이터용 데이터 오브젝트 준비

데이터베이스, 테이블 및 컬럼을 사용 가능화하여 Extender 데이터를 보유할 수 있게 준비합니다. 먼저, 데이터베이스를 사용할 수 있게 하십시오. 그리고 나서, 그 데이터베이스에서 테이블을 사용할 수 있게 하십시오. 마지막으로, 테이블에 있는 컬럼을 사용할 수 있게 하십시오.

데이터 오브젝트에서 Extender 데이터를 더 이상 원하지 않을 때, 오브젝트를 사용 불가능하게 할 수 있습니다.

C 언어 프로그램에서 API를 사용하거나 db2ext 명령행에서 오브젝트를 사용 가능하게 하거나 사용 불가능하게 할 수 있습니다. 이 장에서, 예시는 각각의 방법에 대해 제공됩니다.

데이터베이스 사용 가능화

DBxEnableDatabase API(여기서 x는 오디오의 경우 a, 이미지의 경우 i 또는 비디오의 경우 v) 또는 ENABLE DATABASE 명령을 사용하여 DB2 Extender에 대하여 데이터베이스를 사용할 수 있게 하십시오..

데이터베이스를 사용할 수 있게 할 때, Extender는 다음을 수행합니다.

- xxxxx가 Image, Audio 또는 Video인 데이터 오브젝트에 대해 DB2xxxxx로 이름 지정된 사용자 정의 유형(UDT)을 작성합니다. UDT는 해당 유형의 오브젝트에 대한 핸들을 보유하는 사용자 테이블에 컬럼을 정의하는 데 사용됩니다.
- 데이터베이스에 대한 관리 지원 테이블(메타데이터 테이블이라고도 함)을 작성합니다. 이러한 테이블은 사용자 테이블(사용자가 업무 데이터를 저장한 테이블)이 아닙니다. Extender는 이를 사용하여 Extender 데이터를 관리합니다. 이것들을 수동으로 편집하지 마십시오.
- Extender와 연관된 사용자 정의 함수(UDF)를 작성합니다. UDF는 220 페이지의 『사용자 정의 함수』에 나열됩니다.

데이터베이스를 사용 가능화하려면, 데이터베이스용 관리 지원 테이블(및 해당 색인)을 보유할 테이블 공간도 지정해야 합니다. 지정된 하나 이상의 테이블 공간이 널(NULL) 값을 가질 수 있으며, 이 경우 기본 테이블 공간이 사용됩니다.

사용자는 데이터베이스의 사용을 가능하게 하기 위해 DBA 권한이 필요합니다.

EEE 전용: 파티션된 환경에서 Extender에 대해 데이터베이스를 사용 가능화할 때, 지정하는 테이블 공간은 파티션된 데이터베이스 시스템 내의 모든 노드가 포함되는 노드 그룹에 정의되어 있어야 합니다. 또한, 지정된 테이블 공간은 사용자 테이블과 동일한 노드 그룹에 있어야 합니다.

예시

다음의 예시에서, 데이터베이스는 기본 테이블 공간을 사용 중인 이미지 데이터를 보유하기 위해 사용 가능화됩니다.

API 사용: 63 페이지의 그림16의 코드는 먼저 기존의 데이터베이스에 연결하고 이를 사용할 수 있게 합니다. 이 예시는 DB2 콜 레벨 인터페이스를 사용하여 작성됩니다. 여기에는 일부 설정 및 오류 점검 코드가 포함됩니다. 완전한 샘플 프로그램은 SAMPLES 서브디렉토리의 ENABLE.C 파일에 있습니다.

```
/*---- Set-up -----------------------------------------------*/
#include <stdio.h>
#include <stlib.h>
#include <string.h>
#include "dmbimage.h"       /* image extender function prototypes (DBi) */
#include "utility.h"        /* utility functions */

#define MMDB_ERROR_MSG_TEXT_LEN         1000
#define SERVER_IS_DB2390 (strcmp(dbms,"DB2")==0 || strcmp(dmbs,"DSN06010")==0)

int
main(int argc, char *argv[])
{
        SQLHENV henv = SQL_NULL_HENV;
        SQLHDBC hdbc = SQL_NULL_HDBC;
        SQLHSTMT hstmt = SQL_NULL_HSTMT;
        SQLCHAR uid[18+1];
        SQLCHAR pwd[30+1];
        SQLCHAR dbname[SQL_MAX_DSN_LENGTH+1];
        SQLCHAR buffer[500];
        SQL SMALLINT dbms_sz = 0;
        char    dbms[20];

        SQLRETURN rc = SQL_SUCCESS;
        SQLINTEGER sqlcode = 0;
        char errorMsgText[MMDB_ERROR_MSG_TEXT_LEN+1];
        char *program = "enable;
        char *step;
```

그림 16. 데이터베이스의 사용을 가능하게 하는 샘플 코드 (1/3)

```
/*---- Prompt for database name, userid, and password ----*/
        if (argc > 5) ||  (argc >=2 && strcmp(argv[1],"?")== 0))
          {
            printf("Syntax for enable - enabling a DB2 UDB database: \n"
                   "   enable database_name userid password\n");
            exit(0);
          }

        if (argc == 4) {
          strcpy((char *)dbname, argv[1]);
          strcpy((char *)uid   , argv[2]);
          strcpy((char *)pwd   , argv[3]);
          }
        else {
          printf("Enter database name:\n");
          gets((char *) dbName);
          printf("Enter userid:\n");
          gets((char *) uid);
          printf("Enter password:\n");
          gets((char *) pwd);
          }
/*----- connect to the database ------------------------------------------------*/
         rc = cliInitialize(&henv, &hdbc, dbname, uid, pwd);
         cliCheckError(henv, hdbc, SQL_NULL_HSTMT, rc);
         if (rc < 0) goto SERROR;

/*----- find out if application is connected to DB2/UDB or DB2/390?------------*/
         rc = SQLGetInfo(hdbc, SQL_DBMS_NAME, (SQLPOINTER) &dbms,
                         sizeof(dbms), &dbms_sz);
         cliCheckError(henv, hdbc, SQL_NULL_HSTMT, rc);
         if (rc < 0) goto SERROR;
```

그림 16. 데이터베이스의 사용을 가능하게 하는 샘플 코드 (2/3)

```
/****** enable server for image extender ***************************************/
         if (!SERVER_IS_DB2390)
           {
            printf("%s: Enabling database......\n", program);
           }
         printf("%s: This may take a few minutes, please wait......\n", program);

         if (!SERVER_IS_DB2390)
           {
            step="DBiEnableDatabase with NULL tablespace"
            rc=DBiEnableDatabase(NULL);
           }
         if (rc < 0) {
            printf ("%s: %s failed!\n", program, step);
            printMsg(rc);
            DBiGetError(&sqlcode, errorMsgText);
            if (sqlcode)
              printf("sqlcode=%i, ",sqlcode);
         }else if (rc > 0) {
            printf("%s: %s, warning detected.\n", program, step);
            printMsg(rc);
            DBiGetError(&sqlcode, errorMsgText);
            printf("warning MsgText=%s\n", errorMsgText);
         } else
            printf("%s: %s OK\n", program, step);
/****** end of enable server ****************************************************/
```

그림 16. 데이터베이스의 사용을 가능하게 하는 샘플 코드 (3/3)

db2ext 명령행 사용: 이 예시에서, 데이터베이스는 이미 연결되어 있습니다.

```
enable database for db2image
```

테이블 사용 가능화

DBxEnableTable API(여기서 x는 오디오의 경우 a, 이미지의 경우 i 또는 비디오의 경우 v) 또는 ENABLE TABLE 명령을 사용하여 DB2 Extender에 대한 테이블을 사용 가능화하십시오.

사용자 테이블을 사용 가능화하려면, 관리 지원 테이블(및 해당 색인)을 보유할 테이블 공간도 지정해야 합니다. 지정된 하나 이상의 테이블 공간이 널(NULL) 값을 가질 수 있으며, 이 경우 기본 테이블 공간이 사용됩니다.

EEE 전용: 파티션 환경에서 Extender에 대해 테이블을 사용 가능화할 때, 지정하는 테이블 공간은 파티션된 데이터베이스 시스템 내의 모든 노드가 포함되는 노드 그룹에 정의되어 있어야 합니다. 또한, 지정된 테이블 공간은 사용자 테이블과 동일한 노드 그룹에 있어야 합니다.

EEE 전용: 파티션된 데이터베이스 환경에서 DB2 Extender 컬럼을 파티션 키 컬럼으로 사용할 수 없습니다.

사용자 테이블에 대한 제어 또는 변경 권한이 필요합니다. 데이터베이스에 있는 테이블을 사용 가능화하려면, 먼저 데이터베이스를 사용 가능화해야 합니다.

다음의 예시에서, 테이블은 기본 테이블 공간을 사용하여 이미지를 보유하도록 사용이 가능화되어 있습니다. 데이터베이스는 이미 사용 가능합니다.

API 사용: 67 페이지의 그림17에서 테이블을 사용 가능화하기 전에 코드는 테이블을 작성하고 변경사항을 확약합니다. 예시에는 일부 오류 점검 코드가 포함됩니다. 완전한 샘플 프로그램은 SAMPLES 서브디렉토리의 ENABLE.C 파일에 있습니다.

```
        SQLCHAR szCreate_DB2UDB[]="CREATE TABLE %s(%s mmdbsys.DB2Image,
        %s mmdbsys.DB2Video, %s mmdbsys.DB2Audio, artist varchar(25), title varchar(25)
        stock_no char(11), tw char(10), price char(10))";

        SQLRETURN rc = SQL_SUCCESS;
        SQLINTEGER sqlcode = 0;
        char errorMsgText[MMDB_ERROR_MSG_TEXT_LEN+1];
        char tableName[8+18+1] = "sobay_catalog";
        char audioColumn[18+1] = "music";
        char imageColumn[18+1] = "covers";
        char videoColumn[18+1] = "video";
        char *program = "enable";
        char *step;

/*-----create table ---------------------------------------------------*/
       printf("%s: Creating table ......\n", program);
       if (!SERVER_IS_DB2390)
         sprintf((char*) buffer, (char*) szCreate_DB2UDB,
                   tableName, imageColumn, videoColumn, audioColumn):

       rc = SQLAllocStmt(hdbc, &hstmt);
       cliCheckError(SQL_NULL_HENV, hdbc, SQL_NULL_HSTMT, rc);
       rc = SQLExecDirect(hstmt, buffer, SQL_NTS);
       cliCheckError(SQL_NULL_HENV, SQL_NULL_HDBC, hstmt, rc);

/*---- enable table for image extender ----------------------------------*/
     printf("%s: Enabling table......\n", program);
     step="DBiEnableTable";
     if (!SERVER_IS_DB2390)
         rc = DBiEnableTable(NULL,tableName);
           }
         if (rc < 0) {
                printf("%s: %s failed!\n", program, step);
                printMsg(rc);
                DBiGetError(&sqlcode, errorMsgText);
                if (sqlcode)
                  printf("sqlcode=%i, "sqlcode");
                printf("errorMsgText=%s\n", errorMsgText);
         } else if (rc > 0) {
                printf("%s: %s, warning detected.\n", program, step);
                printMsg(rc);
                DBiGetError(&sqlcode, errorMsgText);
                printf("warningMsgText=%s\n", errorMsgText);
         } else
                printf("%s: %s OK\n", program, step)
/*---- end of enable table ------------------------------------------*/
```

그림 17. 테이블 사용을 가능하게 하는 샘플 코드

db2ext 명령행 사용: 이 예시에서, 테이블은 이미 존재하며 데이터베이스가 사용 가능합니다.

```
enable table employee for db2image
```

컬럼 사용 가능화

DBxEnableColumn API(여기서 x는 오디오의 경우 a, 이미지의 경우 i 또는 비디오의 경우 v) 또는 ENABLE COLUMN 명령을 사용하여 DB2 Extender에 대해 컬럼을 사용 가능화하십시오. API 또는 명령을 실행할 때, 관련 테이블 및 컬럼을 지정합니다.

컬럼을 사용 가능하게 하면, Extender가 정보를 사용자 테이블에 속한 관리 지원 테이블에 추가합니다. 컬럼이 있는 사용자 테이블에 대한 제어 또는 변경 권한이 필요합니다. 컬럼을 사용 가능화하기 전에 데이터베이스와 테이블을 둘다 사용 가능화해야 합니다.

다음의 예시에서, EMPLOYEE 테이블에 있는 PICTURE 컬럼은 이미지 데이터를 보유하기 위해 사용 가능화됩니다. 데이터베이스와 테이블은 이미 사용 가능합니다.

API 사용: 이 예시에는 일부 오류 점검 코드가 포함됩니다. 완전한 샘플 프로그램은 SAMPLES 서브디렉토리의 ENABLE.C 파일에 있습니다.

```
char imageColumn[18+1] = "covers";

/*---- enable column for image extender ----*/
     printf("%s: Enabling columns......\n", program);
     step="DBiEnableColumn";
     rc = DBiEnableColumn(tableName, imageColumn);
     if (rc < 0) {
            printf("%s: %s failed!\n", program, step);
            printMsg(rc);
           DBiGetError(&sqlcode, errorMsgText);
           if (sqlcode)
              printf("sqlcode=%i, ", sqlcode);
           printf("errorMsgText=%s\n", errorMsgText)

    } else if (rc > 0) {
           printf("%s: %s, warning detected.\n", program, step);
           printMsg(rc);
           DBiGetError(&sqlcode, errorMsgText);
           printf("warningMsgText=%s\n", errorMsgText);
    } else
           printf("%s: %s OK\n", program, step);
/*---- enable column for image extender ----*/
```

그림 18. 컬럼 사용을 가능하게 하는 샘플 코드

db2ext 명령행 사용: 이 예시에서, 컬럼은 이미 존재하며 데이터베이스 테이블이 사용 가능합니다.

```
enable column employee picture for db2image
```

데이터 오브젝트 사용 불가능화

Extender 데이터를 데이터베이스, 테이블 또는 컬럼에서 제거하는 경우, 더이상 사용 가능할 필요가 없습니다. 데이터 오브젝트의 사용을 불가능하게 하는 두 가지 방법은 DISABLE 명령과 API입니다. Extender 명령에 관한 자세한 내용은 525 페이지의 『제17장 클라이언트용 관리 명령』을 참조하십시오. Extender API에 관한 자세한 내용은 291 페이지의 『제16장 응용프로그램 프로그래밍 인터페이스 (API)』를 참조하십시오.

Extender 데이터가 있는 테이블이나 데이터베이스를 삭제하기 전에, 그것을 사용 불가능하게 하고 그 데이터베이스에 대한 서버를 정지하십시오.

제7장 파티션된 데이터베이스 시스템에서 Extender 데이터 재분산(EEE 전용)

DB2 Extended Enterprise Edition은 파티션된 데이터베이스 환경에서 데이터베이스 파티션 서버(노드라고도 함)를 추가하고 삭제할 수 있도록 합니다. 노드를 추가한 후에는(또는 노드를 삭제하기 전), 기존의 데이터를 재분산하여 새로운 구성을 이용할 수 있습니다.

Extender 데이터를 재분산하려면 두 가지 단계를 수행해야 합니다. 먼저, DB2 데이터를 재분산해야 합니다. 그런 다음, DB2 Extender 데이터를 재분산해야 합니다.

DB2 데이터 재분산

데이터를 재분산하기 전, DB2 명령 REDISTRIBUTE NODEGROUP을 사용하여 DB2 데이터를 재분산해야 합니다.

DB2 데이터 재분산에 관한 자세한 내용은 *DB2 관리 안내서*를 참조하십시오.

Extender 데이터 재분산

DB2 데이터를 재분산한 후에는 Extender 데이터를 재분산할 수 있습니다. Extender 명령 REDISTRIBUTE NODEGROUP을 입력하여 Extender 데이터 재분산을 시작하십시오.

```
redistribute nodegroup
```

REDISTRIBUTE NODEGROUP 명령은 Audio, Image 및 Video Extender 데이터와 QBIC 특성 데이터를 재분산하여 연관되는 사용자 데이터와 동일한 노드에 놓습니다.

재분산 프로세스가 오류를 리턴하는 경우, 명령을 재수행할 수 있습니다. 명령 응답에서 제공하는 지시사항에 따라, CONTINUE 매개변수를 사용하거나 사용하지

않고 명령을 재수행할 수 있습니다. 이 옵션은 시스템에 처음부터 시작하지 말고 정지된 부분에서 계속하도록 지시합니다. DB2의 REDISTRIBUTE NODEGROUP 명령 수행 후 처음으로 REDISTRIBUTE NODEGROUP 명령을 수행할 때에는 CONTINUE 매개변수를 사용할 수 없습니다.

데이터 무결성을 유지하려면, 한번에 한 개의 노드 그룹을 재분산하십시오. 하나의 노드 그룹이 재분산을 완료한 후에 다른 노드 그룹의 재분산을 시작하십시오.

이 명령을 사용하기 전에 데이터베이스에 연결해야 합니다.

이 명령을 수행하려면 SYSADM 또는 DBADM 권한이 있어야 합니다.

제8장 데이터 오브젝트와 미디어 파일 추적

DB2 Extender를 사용하는 응용프로그램을 작성 및 디버그할 때, Extender 데이터에 대해 사용 가능한 데이터 오브젝트를 아는 것이 유용합니다. 예를 들어, 특정 테이블이 이미지 데이터에 대해 사용 가능한지 판별할 수 있다면, 응용프로그램은 이미지 파일을 그 테이블에 성공적으로 저장할 수 있습니다.

또한, 사용자 테이블과 외부 미디어 파일 간의 상관 관계를 이해하는 데 유용합니다(예를 들어, 특정 파일을 참조하는 테이블 또는 특정 테이블에서 참조하는 파일). 또한 테이블이 시스템에 더이상 존재하지 않는 파일을 참조하는지 아는 것도 유용합니다.

적합한 특권이 필요합니다. 테이블의 데이터를 추적하려면 테이블에 대한 액세스 권한이 있어야 합니다. 데이터베이스의 모든 사용자 테이블에 있는 항목 중에서 파일을 참조하는 항목 찾기와 같은, 포괄적인 추적 조작을 수행하려면, 검색되는 모든 사용자 테이블과 이에 연관되는 관리 지원 테이블에 있는 사용 가능한 컬럼에 관한 SYSADM 권한, DBADM 권한 또는 SELECT 특권이 필요합니다. 모든 테이블에 대한 액세스 권한이 없는 경우, Extender는 사용자가 액세스할 수 있는 테이블에 대한 추적 정보만을 리턴합니다. 또한 필요한 테이블 중 일부에 대한 액세스 권한이 없음을 나타내는 코드를 리턴합니다.

데이터 오브젝트의 상태 점검

데이터베이스, 테이블 및 컬럼이 Extender 데이터를 보유할 수 있는지 확인할 수 있습니다. 다음 예시는 현재 데이터베이스가 Image Extender에 대해 사용 가능한지를 판별합니다. 데이터베이스는 이미 연결되어 있습니다. 완전한 샘플 프로그램은 SAMPLES 서브디렉토리의 API.C 파일에 있습니다.

API 사용: 74 페이지의 그림19의 샘플 코드에는 일부 오류 점검 코드가 포함됩니다.

```
/*---- Query the database using DBiIsDatabaseEnabled API. ----------*/
step="DBiIsDatabaseEnabled API";
rc = DBiIsDatabaseEnabled(&status);
if (rc < 0) {
     printf("%s: %s FAILED!\n", argv[0], step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
     fail = TRUE;
} else if (rc > 0) {
     printf("%s: %s, warning detected.\n", argv[0], step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else {
     if (status == 1) {
         printf("%s: \"%s\" database is enabled for Image Extender\n",
                argv[0], dbName);
         printf("%s: %s PASSED\n\n", argv[0], step);
     } else if (status == 0) {
         printf("%s: \"%s\" database is not enabled for Image Extender\n",
                argv[0], dbName);
         printf("%s: %s PASSED\n\n", argv[0], step);
     } else
         printf("%s: %s FAILED, invalid status!\n", argv[0], step);
}
```

그림 19. 데이터베이스가 사용 가능한지 점검하는 샘플 코드

db2ext 명령행 사용:

```
get extender status
```

사용자 테이블과 컬럼의 상태를 점검하는 것은 데이터베이스 상태를 점검하는 것과 유사합니다. DBxIsTableEnabled 및 DBxIsColumnEnabled API 또는 GET EXTENDER STATUS 명령을 사용하십시오.

파일을 참조하는 테이블 항목 찾기

사용자 테이블에서 외부 미디어 파일을 참조하는 항목을 점검할 수 있습니다. DBxAdminIsFileReferenced API를 사용하여 현재 데이터베이스의 전체 또는 일부 사용자 테이블에 있는 외부 미디어 파일을 참조하는 항목을 점검하십시오. DBxIsFileReferenced API를 사용하여 특정 사용자 테이블에 있는 외부 미디어 파일을 참조하는 항목을 점검하십시오.

API 사용: 그림20의 샘플 코드는 파일이 참조되는 장소와 참조되는 횟수를 리턴합니다. 이것은 일부 오류 점검 코드를 포함합니다. 완전한 샘플 프로그램은 SAMPLES 서브디렉토리의 API.C 파일에 있습니다.

```
/*---- Query the database using DBiAdminIsFileReferenced API. ------*/
step="DBiAdminIsFileReferenced API";
rc = DBiAdminIsFileReferenced((char*) uid, filename, &count, &filelist);
if (rc < 0) {
     printf("%s: %s FAILED!\n", program, step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else if (rc > 0) {
     printf("%s: %s, warning detected.\n", program, step);
     printMsg(rc);
     DBiGetError(&sqlcode, errorMsgText);
     printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsg Text);
} else {
     if (count == 0)
          printf("%s: \"%s\" file is not referenced\n",
                 program, filename);
     else {
          printf("%s: \"%s\" file is referenced %d times\n",
                 program, filename);
          for (i=0; i < count; i++)
             {
                 /* filename is NULL for any IsFileReferenced APIs */

                  printf ("filename = %s\n", filelist[i].filename);
                  printf ("\tqualifier = %s\n", filelist[i].tqualifier);
                  printf ("\ttable = %s\n", filelist[i].tname);
                  printf ("\thandle = %s\n", filelist[i].handle);
                  printf ("\tcolumn = %s\n", filelist[i].column);
                  if (filelist[i].filename)
                          free (filelist[i].filename);
             }
          }
          if (filelist)
                free (filelist);
          printf("%s: %s PASSED\n\n", argv[0], step);
}
```

그림 20. 사용자 테이블이 파일을 참조하는지 점검하는 샘플 코드

테이블 항목이 참조하는 파일 찾기

DBxAdminGetReferencedFiles API 또는 GET REFERENCED FILES 명령을 사용하여 현재 데이터베이스의 전체 또는 일부 사용자 테이블에서 참조하는 외부 미디어 파일을 나열하십시오. DBxGetReferencedFiles API 또는 GET REFERENCED FILES 명령을 사용하여 특정 테이블에서 참조되는 외부 미디어 파일을 나열하십시오.

API 사용: 그림21의 샘플 코드는 그것이 찾는 파일 수와 파일 목록을 리턴합니다. 완전한 샘플 프로그램은 SAMPLES 서브디렉토리의 API.C 파일에 있습니다.

```
/*---- Query the database using DBiAdminGetReferencedFiles API. ----------*/
step="DBiAdminGetReferencedFiles API"
rc = DBiAdminGetReferencedFiles((char*) uid, &count, &filelist);
if (rc < 0) {
   printf("%s: %s FAILED!\n", program, step);
   printMsg(rc);
   DBiGetError(&sqlcode, errorMsgText);
   printf{"sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else if (rc > 0) {
   printf("%s: %s, warning detected.\n", program, step);
   printMsg(rc);
   DBiGetError(&sqlcode, errorMsgText);
   printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else {
   if (count == 0)
     printf("%s: no referenced files\n", program);
  else {
     printf("%s: %d referenced files\n", program, count);
     for (i=0; i < count; i++)
       {
   printf ("filename = %s\n", filelist[i].filename);
   printf ("\tqualifier = %s\n", filelist[i].tqualifier);
   printf ("\ttable = %s\n", filelist[i].tname);
   printf ("\thandle = %s\n", filelist[i].handle);
   printf ("\tcolumn = %s\n", filelist[i].column);
       if (filelist[i].filename)
                free (filelist[i].filename);
     }
   }
   if (filelist)
       free (filelist);
   printf("%s: %s PASSED\n\n", argv[0], step);
}
```

그림 21. 참조되는 파일 목록을 확보하는 샘플 코드

db2ext 명령행 사용:

```
get referenced files user anitas for db2image
```

미디어 파일이 있는지 점검

누군가 미디어 파일을 참조하는 사용자 테이블을 갱신하지 않고 시스템에서 미디어 파일을 삭제한다고 가정하십시오. 사용자는 사용자 테이블이 참조하는 모든 액세스할 수 없는 미디어 파일을 나열하려 할 수 있습니다.

DBxAdminGetInaccessibleFiles API 또는 GET INACCESSIBLE FILES 명령을 사용하여 현재 데이터베이스의 전체 또는 일부 사용자 테이블에서 참조하는 액세스할 수 없는 미디어 파일을 나열하십시오. DBxGetInaccessibleFiles API 또는 GET INACCESSIBLE FILES 명령을 사용하여 특정 테이블에서 참조하는 액세스할 수 없는 미디어 파일을 나열하십시오.

제9장 관리 지원 테이블 정리

DB2 Extender로 작업할 때, 사용하지 않는 항목들이 관리 지원 테이블에 모일 수 있습니다. 누군가 데이터베이스에서 미디어 파일은 삭제했으나 이에 대한 참조는 삭제하지 않았을지 모릅니다. 사용하지 않는 메타데이터를 삭제하여 성능을 향상시키고 저장 공간을 회수할 수 있습니다.

API 사용: 그림22의 샘플 코드는 ANTITAS에서 소유하는 모든 사용자 테이블에 대한 이미지 메타데이터를 정리합니다. 이것은 일부 오류 점검 코드를 포함합니다. 완전한 샘플 프로그램은 SAMPLES 서브디렉토리의 API.C 파일에 있습니다.

```
/*---- query database using DBiAdminReorgMetadata API ----*/
step="DBiAdminReorgMetadata API";
rc = DBiAdminReorgMetadata("anitas");
if (rc < 0) {
        printf("%s: %s FAILED!\n", argv[0], step);
        printMsg(rc);
        DBiGetError(&sqlcode, errorMsgText);
        printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
        fail = TRUE;
} else if (rc > 0) {
        printf("%s: %s, warning detected.\n", argv[0], step);
        printMsg(rc);
        DBiGetError(&sqlcode, errorMsgText);
        printf("sqlcode=%i, errorMsgText=%s\n", sqlcode, errorMsgText);
} else
        printf("%s: %s PASSED\n\n", argv[0], step);

/*---- end of query using DBiAdminReorgMetadata API ----*/
```

그림 22. 관리 지원 테이블을 정리하는 샘플 코드

db2ext 명령행 사용:

```
reorg database user anitas for db2image
```

DBA는 아니지만 CONTROL 권한을 갖는 사용자라면, DBxReorgMetadata API 또는 REORG 명령을 사용하여 소유한 테이블에 대한 메타데이터를 정리할 수 있습니다.

제3부 이미지, 오디오 및 비디오 데이터 프로그래밍

제10장 프로그래밍 개요

이 장에서는 DB2 Extender에 대한 프로그래밍의 개요를 제공합니다. Extender에 대한 프로그래밍을 시작하기 전에 필요한 정보와 Extender의 코딩 방법을 설명하는 샘플 응용프로그램을 제시합니다.

Extender UDF 및 API 사용

DB2 Extender는 데이터베이스에 이미지, 오디오 및 비디오 데이터를 저장하고, 액세스하며, 처리하는 사용자 정의 함수를 제공합니다. 사용자는 SQL 내장 함수를 요청한 방법과 동일하게 SQL문을 사용하여 응용프로그램에 이러한 UDF에 대한 요청을 코딩합니다. 내장 함수와 같이, UDF는 데이터베이스 서버에서 수행됩니다.

C 응용프로그램에 있는 다음의 SQL문은 데이터베이스 테이블에 이미지를 저장하기 위한 DB2Image라는 Image Extender UDF를 요청합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*/
           CURRENT SERVER,                 /*database */
           '/employee/images/ajones.bmp', /*image content*/
           'ASIS',                         /*keep the image format*/
           :hvStorageType,                 /*store image in DB as BLOB*
           'Anita''s picture')             /*comment*/
      );
```

Extender 응용프로그램 프로그래밍 인터페이스를 사용하여 이미지를 표시하고 오디오나 비디오 오브젝트를 재생합니다. 클라이언트 함수 호출을 사용하는 이러한 API를 C로 코딩합니다. 함수는 데이터베이스 클라이언트 워크스테이션에서 수행됩니다.

다음의 C문은 DBiBrowse라는 API를 포함합니다. API는 데이터에서 이미지 핸들을 검색하여 이미지를 표시하기 위한 브라우저를 시작합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  char hvImg_hdl[251];
EXEC SQL END DECLARE SECTION

EXEC SQL SELECT PICTURE INTO :hvImg_hdl
   WHERE NAME= 'Robert Smith';

rc=DBiBrowse(
             "ib %s",                 /*image browser*/
             MMDB_PLAY_HANDLE,        /*use image handle*/
             hvImg_hdl,               /*image handle*/
             MMDB_PLAY_NO_WAIT);      /*run browser independently*/
```

UDF는 인스턴스의 사용자 ID 하에서 수행해야 합니다. DB2 Extender UDF는 DB2 Extender 인스턴스와 동일한 사용자 ID 하에서 수행해야 합니다. 또한, DB2 Extender 인스턴스를 작성하거나 기존의 DB2 Extender 인스턴스를 사용할 경우, UDF는 DB2 인스턴스와 동일한 사용자 ID 하에서 수행해야 합니다.

DB2를 적절히 구성해야 합니다. DB2 Extender(특히 DB2 Extender UDF)가 제대로 작동하려면, DB2를 적절히 구성해야 합니다. 특히, APP_CTL_HEAP_SZ 데이터베이스 구성 매개변수를 적절히 설정해야 합니다.

Extender UDF 및 API로 수행 가능한 타스크

표4는 Extender UDF 및 API로 수행할 수 있는 타스크를 나열하고, 각 타스크에 대한 설명이 있는 페이지를 보여줍니다.

표 4. DB2 Extender API로 수행할 수 있는 타스크

타스크	참조
이미지, 오디오 또는 비디오 오브젝트 저장	100페이지
이미지, 오디오 또는 비디오 오브젝트 검색	114페이지
이미지, 오디오 및 비디오 속성을 검색하고 사용	120페이지
이미지, 오디오 또는 비디오 오브젝트와 연관된 주석 검색	123페이지
이미지, 오디오 또는 비디오 오브젝트 갱신	123페이지
이미지 오브젝트 표시	139페이지
썸네일 크기 이미지 또는 비디오 프레임 표시	143페이지

표 4. DB2 Extender API로 수행할 수 있는 타스크 (계속)

타스크	참조
오디오 또는 비디오 오브젝트 재생	144페이지
내용별 이미지 조회	145페이지
비디오 장면 변경 검출	189페이지

Extender 예시용 샘플 테이블

이 장에서는 DB2 Extender를 사용하는 프로그래밍 예시가 다뤄집니다. 예시는 직원 정보가 있는 EMPLOYEE라는 데이터베이스 테이블을 작성하였음을 가정합니다. 테이블에는 직원의 ID와 이름에 대한 컬럼이 있습니다. Extender에 따라, 테이블에는 직원 사진, 음성 인사말 및 비디오 클립에 대한 컬럼도 있을 수 있습니다.

86 페이지의 그림23은 직원 테이블의 구조를 설명하고 테이블을 작성하는 데 사용되는 SQL문을 보여줍니다.

```
CREATE TABLE  employee(
    id          CHAR(6),
    name        VARCHAR(40),
    picture     DB2Image,

    sound       DB2Audio,

    video       DB2Video
    );
```

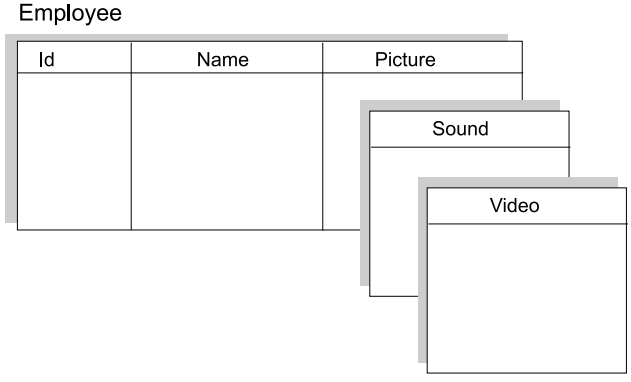


그림 23. DB2 Extender 프로그래밍 예시에 사용되는 테이블

DB2 Extender용 프로그래밍을 시작하기 전에

DB2 Extender를 사용하는 프로그램을 개발하려면, *DB2 응용프로그램 개발 안내서*에 설명된 대로 DB2 응용프로그램 개발 프로세스와 프로그래밍 기법에 친숙해야 합니다. DB2 Extender를 사용하는 프로그램 개발용 프로세스는 전통적인 DB2 응용프로그램용 프로세스와 본질적으로 동일합니다.

응용프로그램 코드는 전통적인 DB2 응용프로그램과는 다른데, 그것은 Extender가 정의한 새로운 데이터 유형과 함수 때문입니다. 예를 들어, 88 페이지의 그림24는 데이터베이스 테이블에 저장된 GIF 이미지를 식별하기 위해 Image Extender를 사용하는 C로 코딩된 응용프로그램을 나타냅니다. 이미지 식별 후, 프로그램은 이미지를 나타내기 위해 이미지 브라우저를 호출합니다.

예시에서 설명하는 것처럼, DB2 Extender를 사용하는 응용프로그램은 다음 기능을 수행해야 합니다.

1 Extender 정의를 포함하십시오. 예시의 `dmbimage.h` 파일은 Image Extender용 Include(헤더) 파일입니다. Include 파일은 Extender용 상수, 변수 및 함수 프로토타입을 정의합니다.

2 UDF에서의 입출력 또는 API 호출로의 입력을 포함하기 위해 필요한 호스트 변수를 정의하십시오. 예시에서, `hvFormat`, `hvSize`, `hvWidth`, `hvHeight` 및 `hvComment`는 Image Extender UDF가 검색하는 데이터를 포함하는 데 사용되는 호스트 변수입니다. 호스트 변수 `hvImg_hdl`은 Image Extender API 호출에 대한 입력으로 지정된 이미지 핸들을 포함하기 위해 사용됩니다.

3 UDF 요청을 필요한 대로 지정하십시오. 예시에서, `SIZE`, `WIDTH`, `HEIGHT`, `COMMENT` 및 `FORMAT`은 Image Extender UDF입니다.

4 API 호출을 필요한 대로 지정하십시오. 예시에서, `DBiBrowse`는 테이블에서 검색되는 핸들의 이미지를 표시하는 국지 C 함수로의 API 호출입니다.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sqlenv.h>
#include <sqlcodes.h>
#include <dmbimage.h>  1

int count=0;

long
main(int argc,char *argv[])
{
EXEC SQL BEGIN DECLARE SECTION;  2
         char hvImg_hdl[251];                    /* image handle */
         char hvDBName[19];                      /* database name */
         char hvName[40];                        /* employee name */
         char hvFormat[9];                       /* image format */
         long hvSize;                            /* image size */
         long hvWidth;                           /* image width */
         long hvHeight;                          /* image height */
         struct  {
                  short len;
                  char data[32700]
         } hvComment;                            /* comment about the image */
 EXEC SQL END DECLARE SECTION;

/*  Connect to database  */

strcpy(hvDBName, argv[1]);                       /* copy the database name */

EXEC SQL CONNECT TO :hvDBName IN SHARE MODE;
/*
 * Set current function path
*/
EXEC SQL SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH;
```

그림 24. DB2 Extender를 사용하는 응용프로그램 (1/2)

```
/*
 * Select (query) using Image Extender UDF
 *
 * The SQL statement below finds all images in GIF format.
 */
EXEC SQL DECLARE c1 CURSOR FOR
         SELECT PICTURE, NAME,                 3
                SIZE(PICTURE), WIDTH(PICTURE),
                HEIGHT(PICTURE), COMMENT(PICTURE)
         FROM EMPLOYEE
         WHERE PICTURE IS NOT NULL AND
               FORMAT(PICTURE) LIKE 'GIF%'
FOR FETCH ONLY;

EXEC SQL OPEN c1;
for (;;) {
         EXEC SQL FETCH c1 INTO :hvImg_hdl, :hvName, :hvSize,
                                :hvWidth, :hvHeight, :hvComment
;
         if (SQLCODE != 0)
                   break;

         printf("\nRecord %d:\n", ++count);
         printf("employee name = '%s'\n", hvName);
         printf("image size = %d bytes, width=%d, height=%d\n",
                                                 hvSize, hvWidth, hvHeight);
         hvComment.data[Comment.len]='\0';
         printf("comment len = %d\n", hvComment.len);
 printf("comment = %s\n", hvComment.data);
/*
* The API call below displays the images
*/
 4  rc=DBiBrowse ("ib %s",MMDB_PLAY_HANDLE,hvImg_hdl,
                         MMDB_PLAY_WAIT);
}

EXEC SQL CLOSE c1;

/* end of program */
```

그림 24. DB2 Extender를 사용하는 응용프로그램 (2/2)

Extender 정의 포함

사용한 각각의 Extender에 대한 응용프로그램에 Include(헤더) 파일이 필요합니다. 각각의 Include 파일은 Extender가 사용하는 상수, 변수 및 함수 프로토타입을 정의합니다. Include 파일의 이름은 다음과 같습니다.

Include 파일	**Extender**
dmbimage.h	Image
dmbqbapi.h	Image(이미지 내용별 조회)
dmbaudio.h	Audio
dmbvideo.h	Video
dmbshot.h	Video(장면 변경 검출)

Include 파일을 #Include 지시문으로 C 프로그램에 전달합니다. 예를 들어, 다음의 지시문은 Image Extender용 Include 파일을 가져옵니다.

```
#include <dmbimage.h>
```

UDF 및 UDT 이름 지정

DB2 Extender UDF의 전체 이름은 mmdbsys.function-name입니다. DB2 Extender UDT의 전체 이름은 mmdbsys가 함수 또는 구별 유형의 스키마 이름일 때 mmbdsys.type-name입니다. 예를 들어, Content UDF의 전체 이름은 mmdbsys.Content이고, Image Extender가 작성하는 DB2Image 데이터 유형의 전체 이름은 mmdbsys.DB2Image입니다. 이전에 현재 함수 경로를 예를 들어 다음과 같이 mmdbsys로 설정한 경우, mmdbsys 스키마 이름을 생략할 수 있습니다.

```
SET CURRENT FUNCTION PATH = mmdbsys, CURRENT FUNCTION PATH
SET CURRENT PATH = mmdbsys, CURRENT PATH
```

대형 오브젝트 전송

응용프로그램과 DB2 데이터베이스 간에 이미지, 오디오 클립 및 비디오 클립과 같은 대형 오브젝트를 다양한 방법으로 전송할 수 있습니다. 사용자가 사용할 방법은 오브젝트가 파일 또는 메모리 버퍼로/로부터 전송되는지에 따라 다릅니다. 또한 파일이 클라이언트 머신에 있는지 또는 데이터베이스 서버 머신에 있는지에 따라 다릅니다.

오브젝트를 테이블과 서버 파일 간에 전송하는 경우

데이터베이스 테이블과 서버 파일 간에 오브젝트 전송시, 적절한 Extender UDF 요청에 파일 경로를 지정하십시오. Extender UDF와 파일이 모두 서버에 있기 때문에 Extender는 파일을 찾을 수 있을 것입니다. 예를 들어, 다음의 SQL문에서 서버 파일에 있는 내용의 이미지는 데이터베이스 테이블에 저장되어 있습니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2Image(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'ASIS',
          :hvStorageType,
          'Anita''s picture')
       );
```

오브젝트를 클라이언트 버퍼 간에 전송하는 경우

Extender는 메모리 버퍼에 직접 액세스할 수 없습니다. 클라이언트 컴퓨터의 버퍼 상에서 오브젝트를 전송하려면, 버퍼 위치를 지정하는 것과는 다른 수행 방법이 필요합니다. 오브젝트를 버퍼로(에서) 전송하는 한가지 방법은 호스트 변수를 통하는 것입니다. 이것은 일반적으로 응용프로그램과 DB2 데이터베이스 간에 오브젝트를 전송하는 방법입니다.

사용자는 전통적인 문자 및 숫자 오브젝트와 동일한 방법으로 대형 오브젝트에 대한 호스트 변수를 정의하고 사용합니다. DECLARE 섹션에 호스트 변수를 선언하고, 그것에 전송할 값을 할당하거나 또는 그것에 전송되는 값을 액세스합니다.

이미지, 오디오 또는 비디오 데이터용 호스트 변수의 선언시 BLOB 데이터 유형을 지정하십시오. 오브젝트 저장, 검색 또는 갱신하기 위해 UDF를 사용할 때 사용자는 UDF 요청에 대한 인수로서 적절한 호스트 변수를 지정합니다. SQL문에 지정하는 다른 호스트 변수에 대해서와 동일한 형식을 사용하십시오.

예를 들어, 다음 SQL문은 hvaudio라는 호스트 변수를 선언하고 이를 사용하여 오디오 클립을 데이터베이스에 전송합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB (2M) hvaudio;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2Audio(
          CURRENT SERVER,
          :hvaudio,
          'WAVE',
          CAST(NULL as LONG VARCHAR),
          'Anita''s voice')
       );
```

LOB 위치 지정자 사용

오디오 및 비디오 클립과 같은 대형 오브젝트는 아주 커질 수 있고, 호스트 변수의 사용은 그것들을 조작하는 가장 효과적인 방법이 아닐 수 있습니다. **LOB 위치 지정자**가 응용프로그램에서 LOB을 조작하는 더 나은 방법일 수 있습니다.

LOB 위치 지정자는 프로그램이 DB2 데이터베이스에서 더 큰 LOB을 참조하는 데 사용할 수 있는 호스트 변수에 저장된 작은(4바이트) 값입니다. LOB 위치 지정자를 사용하여, 프로그램은 LOB이 일반 호스트 변수에 저장된 것처럼 LOB을 조작할 수 있습니다. 차이점은 클라이언트 컴퓨터 상의 데이터베이스 서버와 응용프로그램 사이에 LOB을 전송할 필요가 없다는 것입니다. 예를 들어, 데이터베이스 테이블에서 LOB 선택시, LOB은 서버에 남고 LOB 위치 지정자는 클라이언트로 이동합니다.

사용자는 DECLARE 섹션에 LOB 위치 지정자를 선언하고 그것을 호스트 변수와 동일한 방법으로 사용합니다. 이미지, 오디오 또는 비디오 데이터에 대한 LOB 위치 지정자 선언시 BLOB_LOCATOR의 데이터 유형을 지정하십시오. 예를 들어, 다음 SQL문은 video_loc로 이름 지정된 LOB 위치 지정자를 선언하고 사용하여 데이터베이스 테이블에서 비디오 클립을 검색합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR video_loc;
EXEC SQL END DECLARE SECTION;
```

```
EXEC SQL SELECT CONTENT(VIDEO)
     INTO :video_loc
     FROM EMPLOYEE
     WHERE NAME='Anita Jones';
```

UDF는 LOB 위치 지정자를 사용합니다. 이미지, 오디오 및 비디오 오브젝트를 저장하고 검색하며 갱신하는 DB2 Extender UDF는 LOB 위치 지정자를 사용합니다. DB2 Extender V1에서 이러한 UDF는 LOB 위치 지정자를 사용하지 않았고, 이 때문에 이것은 2MB를 넘는 오브젝트를 처리할 수 없습니다. 이러한 제한사항은 2MB를 넘는 오브젝트를 세그먼트 단위로 전송하게 합니다. 이러한 UDF가 LOB 위치 지정자를 사용하기 때문에 2MB 제한사항은 제거됩니다.

오브젝트가 클라이언트 파일 간에 전송되는 경우

클라이언트에 있는 파일로(에서) 오브젝트를 전송하려면 파일 참조 변수를 사용하십시오. 파일 참조 변수의 사용으로, 응용프로그램에 대형 오브젝트용 버퍼 공간을 할당하지 않아도 됩니다. UDF에 파일 참조 변수 사용시, DB2는 BLOB 내용을 파일과 UDF 사이에 직접 전달합니다.

사용자는 DECLARE 섹션에 파일 참조 변수를 선언하고 그것을 호스트 변수와 동일한 방법으로 사용합니다. 이미지, 오디오 또는 비디오 데이터용 파일 참조 변수 선언시, BLOB_FILE의 데이터 유형을 지정하십시오. 그러나 오브젝트 내용을 포함하는 호스트 변수와는 달리, 파일 참조 변수는 파일 이름을 포함합니다. 파일 크기는 UDF에 대해 정의되는 BLOB 크기 이하가 될 수 있습니다.

사용자는 입출력용 파일 참조 변수의 사용 방법에 대해 다양한 옵션을 갖습니다. 사용자는 프로그램의 파일 참조 변수 구조에 FILE_OPTIONS 필드를 설정하여, 원하는 옵션을 선택합니다. 다음의 옵션에서 선택 가능합니다.

입력용 옵션

SQL_FILE_READ. 이 파일을 열고, 읽고, 닫을 수 있습니다. 파일의 데이터 길이(바이트)는 파일을 열 때 결정됩니다. 파일 참조 변수 구조의 data_length 필드는 파일 길이(바이트)를 보유합니다.

출력용 옵션

SQL_FILE_CREATE. 이 옵션은 해당 파일이 존재하지 않을 경우에 새 파일을 작성합니다. 파일이 이미 존재한다면, 오류 메시지가 리턴됩니다. 파일 참조 변수 구조의 data_length 필드는 파일 길이(바이트)를 보유합니다.

SQL_FILE_OVERWRITE. 이 옵션은 해당 파일이 존재하지 않을 경우에 새 파일을 작성합니다. 파일이 이미 존재한다면, 새 데이터는 기존 파일의 데이터를 겹쳐 씁니다. 파일 참조 변수 구조의 data_length 필드는 파일 길이(바이트)를 보유합니다.

SQL_FILE_APPEND. 이 옵션은 파일이 이미 존재하는 경우, 파일에 출력을 추가합니다. 파일이 존재하지 않는다면, 새 파일을 작성합니다. 파일 참조 변수 구조의 data_length 필드는 파일의 총 길이가 아닌, 파일에 추가된 데이터의 길이(바이트 단위)를 보유합니다.

예를 들어, 다음 명령문은 Img_file이라는 파일 참조 변수를 선언하고 이를 사용하여 내용이 클라이언트 파일에 있는 이미지를 데이터베이스 테이블로 저장합니다. FILE_OPTIONS 필드의 SQL_FILE_READ 지정을 주의하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Img_file;
  EXEC SQL END DECLARE SECTION;

strcpy (Img_file.name,"/employee/images/ajones.bmp");
Img_file.name_length=strlen(Img_file.name);
Img_file.file_options=SQL_FILE_READ;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2Image(
          CURRENT SERVER,
          :Img_file,
          'ASIS',
          CAST(NULL as LONG VARCHAR),
          'Anita''s picture')
       );
```

오브젝트 전송시 파일 이름 지정

DB2 Extender는 오브젝트의 저장, 검색 또는 갱신시 파일 이름을 지정하는 방법에 대한 융통성을 제공합니다.

저장, 검색 및 갱신 조작을 위해 완전 규정된 파일 이름(즉, 파일 이름이 뒤따르는 완전한 경로)을 지정할 수도 있지만, 상대 파일 이름을 지정하는 편이 더욱 좋습니다. AIX, HP-UX 및 Solaris에서, 상대 파일 이름이란 슬래시로 시작하지 않는 파일 이름입니다. OS/2 및 Windows에서, 상대 파일 이름이란 드라이브명 다음에 콜론 및 백슬래시로 시작하지 않는 모든 파일 이름입니다.

상대 파일 이름을 지정하는 경우, Extender는 파일 이름을 해결하기 위한 탐색 경로로 다양한 클라이언트와 서버 환경 변수의 디렉토리 스펙을 사용합니다. 전체 경로 이름은 일반적으로 마운트 지점과 관련된 선두 부분과 필요한 파일을 고유하게 식별하는 후미 경로 이름으로 이루어집니다. 후미 경로 이름은 UDF에 지정됩니다. 환경 변수는 상대 파일 이름을 해결할 때 탐색할 선두 경로 이름 목록을 제공합니다. DB2 Extender가 파일 이름을 해결하는 데 사용하는 환경 변수에 대해서는 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』에서 자세한 정보를 참조하십시오.

Extender는 파일 이름 형식도 적절하게 변환합니다. 파일 이름의 서버 전달시, 이것은 서버의 운영 체제에 대해 적절한 형식으로 변환됩니다. 예를 들어, AIX 서버로 전달될 때에는 c:\dir1\abc.bmp와 같은 OS/2 파일 이름이 /dir1/abc.bmp로 변환됩니다.

리턴 코드 처리

DB2 Extender UDF를 호출하는 것을 포함하여 프로그램에 있는 Embedded SQL 문 또는 DB2 CLI 호출은 모두 Embedded SQL문이나 DB2 CLI 호출이 성공적으로 수행되었는지를 표시하는 코드를 생성합니다. 관리 API 같은 다른 DB2 Extender API 또한 성공이나 실패를 표시하는 코드를 리턴합니다. 프로그램은 Embedded SQL문, CLI 호출 및 API에서 리턴하는 코드를 점검하고 이에 응답합니다.

이러한 리턴 코드의 처리에 대한 자세한 정보는 585 페이지의 『제19장 진단 정보』를 참조하십시오.

Extender API가 작업 단위를 성공적으로 완료할 수 없는 경우, 구간 복원(rollback) 조작이 수행됩니다. 또한 API는 오류 코드를 리턴합니다. 데이터베이스를 이전 일

관성 지점으로 리턴할 수 있도록 구간 복원 조작을 수행합니다. 자세한 내용은 291 페이지의 『제16장 응용프로그램 프로그래밍 인터페이스(API)』를 참조하십시오.

유니코드 지원

Image, Audio, Video Extender의 유니코드 지원에 대해 다음 사항을 유의하십시오.

- 유니코드 문자열이 될 수 있는 매개변수는 다음 UDF의 주석 필드뿐입니다.
  - mmdbsys.db2image( ) import an image
  - mmdbsys.db2audio( ) import an audio
  - mmdbsys.db2video( ) import a video
  - mmdbsys.replace( ) replace an image, an audio, or a video
  - mmdbsys.comment( ) comment update
- 유니코드 데이터베이스에 액세스할 계획이면, DB2 Extender 인스턴스를 사용하여 유니코드를 지원하도록 설정해야 합니다. 유니코드 인스턴스는 유니코드 데이터베이스만 처리할 것입니다.

  Extender 인스턴스가 유니코드를 지원하도록 하려면, DMBSTART를 호출하기 전에 환경 변수 DB2CODEPAGE를 1208로 설정합니다.

제11장 오브젝트 저장, 검색 및 갱신

이 장에서는 DB2 Extender 사용자 정의 함수를 사용하여 이미지, 오디오 또는 비디오를 저장, 검색 및 갱신하는 방법에 대해 설명합니다.

이미지, 오디오 또는 비디오 형식

표5는 이미지, 오디오 또는 비디오 오브젝트를 저장하거나 검색하거나 갱신할 수 있는 형식을 나열합니다. 이미지 오브젝트에 대해서만, 사용자는 저장, 검색, 갱신시 Image Extender에서 이미지 형식을 변환하도록 할 수 있습니다(오디오와 비디오 형식은 저장, 검색, 갱신시 변환될 수 없습니다).

표에서 읽기 및 쓰기 컬럼은 어떤 형식이 읽히고 어떤 형식이 작성시 변환될 수 있는지를 표시합니다. 표에 있는 읽기 컬럼의 항목이 x인 경우, 대응하는 오브젝트 형식을 저장, 검색 또는 갱신시 사용할 수 있습니다. 쓰기 컬럼의 항목이 x인 경우, 오브젝트(이미지 전용)는 저장, 검색 또는 갱신시 대응하는 형식으로 변환될 수 있습니다. 예를 들어, BMP 형식의 이미지를 저장, 검색 또는 갱신시 GIF 형식으로 변환할 수 있습니다. JPG 형식의 이미지는 TIF 형식으로 변환할 수 있습니다. 그러나 TIF 형식의 이미지는 JPG 형식으로 변환할 수 없습니다.

표에는 대문자로 나열되지만, 저장, 검색 또는 갱신 요청에 있는 형식 스펙은 대소문자 구별이 없습니다. 예로, 스펙 GIF와 gif는 동일합니다.

표 5. DB2 Extender가 처리할 수 있는 형식

형식	설명	읽기	쓰기
	이미지 형식		
_IM	PS/2 Audio Video Connection(AVC)	x	
BMP	OS/2 - Microsoft Windows 비트맵[1]	x	x
EPS	Encapsulated PostScript		x
EP2	Encapsulated level 2 PostScript		x
GIF	Compuserve GIF89a(animated GIF[2] 포함) 및 87	x	x
IMG	IOCA 이미지	x	x

표 5. DB2 Extender가 처리할 수 있는 형식 (계속)

형식	설명	읽기	쓰기
IPS	Brooktrout FAX card file	x	x
JPG	JPEG[3](JFIF 형식)	x	
PCX	PC paint file(회색 전용)	x	x
PGM	Portable gray map(PBMPLUS에서)	x	x
PS	PostScript		x
PSC	Compressed PostScript 이미지		x
PS2	PostScript level 2(칼라)		x
TIF	모든 TIFF 5.0 형식	x	x
YUV	YUV용 Digital video	x	x
	오디오 형식		
AIF 또는 AIFF	Audio Interchange File Format	x	
AIFFC	Audio Interchange File Format Compressed	x	
AU	Sun 오디오 파일 형식	x	
MIDI	Musical Instrument Digital Interface	x	
MPG1 또는 MPEG1	Moving Pictures Expert Group 1	x	
WAV 또는 WAVE	Wave	x	
	비디오 형식		
AVI	Audio/Video Interleaved	x	
MPG1 또는 MPEG1	Motion Picture Coding Expert Group 1	x	
MPG2 또는 MPEG2	Motion Picture Coding Expert Group 2	x	
QT	Quicktime (AVI)	x	

이미지 변환 옵션

표6에는 저장, 검색, 갱신시 이미지에 대해 지정할 수 있는 변환 옵션(형식 변환 외에)이 나열됩니다. Image Extender는 스펙을 목표 이미지에 적용합니다. 소스 이미지는 변경되지 않습니다.

각각의 변환 옵션은 매개변수/값 쌍으로 지정됩니다. 각각의 매개변수에 대해 허용된 값은 표에 나열됩니다.

표 6. 이미지 변환 옵션

매개변수	설명	값
-b	각 이미지 샘플의 표현에 사용되는 비트 수	1 또는 8 비트
-s[4]	규모 확장 요소	0보다 큰 임의의 십진수 값. 규모 확장 요소는 변환된 이미지의 원래 *크기*에 대한 *크기* 비율을 지정합니다. 예를 들어, 0.5의 규모 확장 요소는 원래 *크기*의 반으로 이미지를 변환시킵니다. 2.0의 규모 확장 요소는 이미지를 원래 *크기*의 두 배로 변환시킵니다.
-p	광도 측정(이미지 변환). 이 옵션은 지정된 값에 따라, 이미지 해석을 변경합니다. 이미지 자체는 변경하지 않습니다. 이 옵션은 흑백 또는 회색 이미지에만 적용되며, GIF 형식 이미지에는 적용되지 않습니다.	0 = 검은색입니다. 1 = 흰색입니다.
-n	광도 측정(이미지 변환). 이 옵션은 검정색을 흰색으로, 흰색을 검정색으로 반전시켜서 이미지를 변경합니다. 흑백 또는 회색 이미지에만 적용됩니다.	없음
-r[4]	회전	0 = 0도(회전없음) 1 = 90도(시계 반대 방향) 2 = 90도(시계 방향) 3 = 180도

1. 읽기는 OS/2 버전 1, OS/2 버전 2, Windows 버전 2, Windows 버전 3 및 Windows NT BMP 형식에 대해 지원됩니다. 쓰기는 OS/2 버전 1 BMP 형식에 대해 지원됩니다.

2. DB2 Image Extender는 animated GIF 파일의 처음 이미지 전용 속성 정보를 저장합니다.

3. 지원은 Independent JPEG Group 작업의 일부분에 기초한 소프트웨어를 사용합니다.

표 6. 이미지 변환 옵션 (계속)

매개변수	설명	값
-x[4]	픽셀 폭	픽셀 수
-y[4]	픽셀 높이	픽셀 수
-c	압축 유형	0 = IBM MMR 1 = CCITT Group 3 1-D 2 = CCITT Group 3 2-D (k=2) 3 = CCITT Group 3 2-D (k=4) 4 = CCITT Group 4 6 = TIFF Type 2 10 = Uncompressed 14 = LZW 15 = TIFF Packbits 25 = JBIG

이미지, 오디오 또는 비디오 오브젝트 저장

데이터베이스에 이미지, 오디오 또는 비디오 오브젝트를 저장하려면, SQL INSERT 문에서 DB2Image, DB2Audio 또는 DB2Video를 사용하십시오.

소스가 클라이언트 머신에 있는 버퍼나 파일에 있거나 서버 파일에 있는 오브젝트를 저장할 수 있습니다. 임의의 이러한 소스에 대해, 오브젝트를 BLOB으로 데이터베이스 테이블에 저장하거나 데이터베이스 서버의 파일에 저장할 수 있습니다.

UDF 요청시, 다음을 지정해야 합니다.

- 현재 연결된 데이터베이스 서버명. 이는 CURRENT SERVER 특수 레지스터에 있습니다.
- 오브젝트 내용의 소스. 이것은 클라이언트 버퍼나 클라이언트 파일 또는 서버 파일에 있습니다.
- 내용을 BLOB으로 데이터베이스 테이블에 저장할지 또는 파일 서버에 저장할지 여부.
- 소스 형식.

4. Interlaced GIF 이미지에 대해 이 옵션을 지정한다면, LZW의 압축 유형도 지정해야 합니다.

- 오브젝트와 함께 저장할 주석(주석을 저장하지 않을 경우 널(NULL) 값 또는 빈 문자열).

Image, Audio 및 Video Extender를 사용하여 오브젝트의 형식을 인식하지 못하는 경우에도 오브젝트를 저장할 수 있습니다. 형식이 인식되지 않는 경우, 오브젝트 속성을 지정해야 합니다. 사용자 제공 속성을 사용하여 이미지 또는 비디오를 저장할 때, 썸네일을 또한 저장할 수 있습니다. 썸네일은 이미지 또는 비디오를 표현하는 축소된 이미지입니다.

이미지 전용의 경우, 사용자는 저장시 이미지 형식으로 변환할 수 있는 옵션을 갖습니다. 형식 변환을 요청할 경우, 이미지의 소스와 목표 형식 모두를 지정해야 합니다. 형식 변환 요청에서, 이미지를 단절하거나 회전시키는 것과 같이 이미지에 심화된 변경사항을 지정할 수도 있습니다. 변환 옵션을 지정하여 이러한 변경사항을 표시합니다.

저장 조작 확약: 데이터베이스에 이미지, 오디오 또는 비디오 오브젝트 저장 후 작업 단위를 확약하십시오. 이렇게 하면, Extender가 보유하는 잠금을 해제하여 저장된 오브젝트에 관한 갱신 조작을 수행할 수 있습니다.

DB2Image, DB2Audio 및 DB2Video UDF 형식

DB2Image, DB2Audio 및 DB2Video UDF는 오버로드됩니다. 즉, UDF가 사용되는 용도에 따라 다른 형식을 갖습니다. 각각의 UDF는 다음과 같은 형식을 갖습니다(형식에 표시된 xxxxx는 Image, Audio 또는 Video가 될 수 있습니다).

형식 1: 클라이언트 버퍼 또는 클라이언트 파일에서 오브젝트를 저장합니다.

```
DB2xxxxx(
     CURRENT SERVER,       /* database name name in CURRENT SERVER REGISTER */
     content,              /* object content */
     format,               /* source format */
     target_file,          /* target file name for storage in file server */
                           /* or NULL for storage in table as BLOB */
     comment               /* user comment */
  );
```

형식 2: 서버 파일에서 오브젝트를 저장합니다.

```
DB2xxxxx(
     CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
     source_file,          /* source file name */
     format,               /* source format */
     stortype,             /* MMDB_STORAGE_TYPE_EXTERNAL=store */
```

```
                          /* in file server*/
                          /* MMDB_STORAGE_TYPE_INTERNAL=store */
                          /* as a BLOB*/
    comment               /* user comment */
  );
```

형식 3: 클라이언트 버퍼나 클라이언트 파일에서 사용자 제공 속성으로 오브젝트를 저장합니다.

```
DB2xxxxx(
     CURRENT SERVER,        /* database name in CURRENT SERVER REGISTER */
     content,               /* object content */
     target_file,           /* target file name for storage in file server */
                            /* or NULL for storage in table as BLOB */
     comment,               /* user comment */
     attrs,                 /* user-supplied attributes */
     thumbnail              /* thumbnail (image and video only) */
  );
```

형식 4: 서버 파일에서 사용자 제공 속성으로 오브젝트를 저장합니다.

```
DB2xxxxx(
     CURRENT SERVER,           /* database name in CURRENT SERVER REGISTER */
     source_file,              /* source file name */
     stortype,                 /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                               /* in file server*/
                               /* MMDB_STORAGE_TYPE_INTERNAL=store */
                               /* as a BLOB*/
     comment,                  /* user comment */
     attrs,                    /* user-supplied attributes */
     thumbnail                 /* thumbnail (image and video only) */
  );
```

DB2Image UDF에는 다음의 추가 형식이 있습니다.

형식 5: 형식 변환을 사용하여 클라이언트 버퍼 또는 클라이언트 파일에서 이미지를 저장합니다.

```
DB2Image(
     CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
     content,              /* object content */
     source_format,        /* source format */
     target_format,        /* target format */
     target_file,          /* target file name for storage in file server */
                           /* or NULL for storage in table as BLOB */
     comment               /* user comment */
  );
```

형식 6: 형식 변환을 사용하여 서버 파일에서 이미지를 저장합니다.

```
DB2Image(
     CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
     source_file,          /* server file name */
     source_format,        /* source format */
```

```
    target_format,        /* target format */
    target_file,          /* target file name for storage in file server */
                          /* or NULL for storage in table as BLOB */
    comment               /* user comment */
  );
```

형식 7: 형식 변환 및 추가 변경을 사용하여 클라이언트 버퍼 또는 클라이언트 파일에서 이미지를 저장합니다.

```
DB2Image(
    CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
    content,              /* object content */
    source_format,        /* source format */
    target_format,        /* target format */
    conversion_options,   /* Conversion options */
    target_file,          /* target file name for storage in file server */
                          /* or NULL for storage in table as BLOB */
    comment               /* user comment */
  );
```

형식 8: 형식 변환 및 추가 변경을 사용하여 서버 파일에서 이미지를 저장합니다.

```
DB2Image(
    CURRENT SERVER,       /* database name in CURRENT SERVER REGISTER */
    source_file,          /* server file name */
    source_format,        /* source format */
    target_format,        /* target format */
    conversion_options    /* conversion options */
    target_file,          /* target file name for storage in file server */
                          /* or NULL for storage in table as BLOB */
    comment               /* user comment */
  );
```

예를 들어, C 응용프로그램에 있는 다음의 명령문은 이미지가 있는 행을 직원 테이블로 삽입합니다. 소스 이미지는 ajones.bmp라는 서버 파일입니다. 이미지는 직원 테이블에 BLOB으로 저장됩니다. (이것은 앞에서 설명한 형식 2에 대응합니다.)

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*/
          CURRENT SERVER,                  /*database*/
          '/employee/images/ajones.bmp',   /*source file */
          'ASIS',                          /*keep the image format*/
          :hvStorageType                   /*store image in DB as BLOB*/
          'Anita''s picture')              /*comment */
       );
```

C 응용프로그램에 있는 다음의 명령문은 이전 예시에 나온 동일한 행을 직원 테이블에 저장합니다. 그러나 이러한 이미지의 저장시, BMP는 GIF 형식으로 변환됩니다. (이것은 앞에서 설명한 형식 6에 해당됩니다.)

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',                          /*id*/
        'Anita Jones',                     /*name*/
        DB2IMAGE(                          /*Image Extender UDF*
          CURRENT SERVER,                  /*database*/
          '/employee/images/ajones.bmp',   /*source file */
          'ASIS',                          /*source image format*/
          'GIF',                           /*target image format*/
          'Anita''s picture')              /*comment*/
       );
```

이미지, 오디오 또는 비디오 오브젝트의 저장시, Extender는 이미지에서 사용된 색상 수나 오디오 재생 시간 또는 압축 형식과 같은 속성을 계산합니다. 인식되지 않는 형식으로 오브젝트를 저장하는 경우, 이러한 속성을 입력으로 UDF에 제공해야 합니다. Extender는 오브젝트에 대한 주석 및 오브젝트를 저장한 사용자 ID와 같은 다른 속성들과 함께 데이터베이스에 이러한 속성을 저장합니다. 이러한 속성은 조회에서 사용 가능합니다.

클라이언트에 상주하는 오브젝트 저장

이미지, 오디오 또는 비디오 오브젝트 내용을 클라이언트 버퍼나 클라이언트 파일에서 서버로 전송하는 호스트 변수 또는 파일 참조 변수를 사용하십시오.

오브젝트가 클라이언트 파일에 있다면, 저장할 오브젝트의 내용을 서버로 전송하기 위해 파일 참조 변수를 사용하십시오. 예를 들어, C 응용프로그램에 있는 다음의 명령문은 Audio_file이라는 파일 참조 변수를 정의하고 이를 사용하여 내용이 클라이언트 파일에 있는 오디오 클립을 전송합니다. 오디오 클립은 서버의 데이터베이스 테이블에 저장됩니다. 파일 참조 변수의 file_option 필드가 입력용 SQL_FILE_READ로 설정되었음을 유의하십시오. 또한 파일 참조 변수가 내용 인수로 DB2Audio UDF에 사용되었음을 유의하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Audio_file;
EXEC SQL END DECLARE SECTION;

strcpy (Audio_file.name, "/employee/sounds/ajones.wav");
Audio_file.name_length= strlen(Audio_file.name);
Audio_file.file_options= SQL_FILE_READ;
```

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2AUDIO(
          CURRENT SERVER,
          :Audio_file,          /* file reference variable */
          'WAVE',
          CAST(NULL as LONG VARCHAR),

          'Anita''s voice')
      );
```

오브젝트가 클라이언트 버퍼에 있다면, BLOB 또는 BLOB_LOCATOR로 정의된 호스트 변수를 사용하여 서버에 저장할 내용을 전송하십시오. 다음의 C 응용프로그램 명령문에서는 Video_loc라는 호스트 변수를 사용하여 서버에 저장할 비디오 클립의 내용을 전송합니다. 비디오 클립은 BLOB으로 데이터베이스 테이블에 저장됩니다. 호스트 변수가 내용 인수로 DB2Video UDF에 사용되었음을 유의하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR Video_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2VIDEO(
          CURRENT SERVER,
          :Video_loc,         /* host variable */
          'MPEG1',
          '',
          'Anita''s video')
       );
```

UDF 메모리가 충분한지 확인하십시오. 내용이 클라이언트 버퍼에 있는 오브젝트를 저장할 때, 데이터베이스 관리 프로그램 구성의 UDF_MEM_SZ 매개변수가 4MB 이상으로 설정되었는지 확인하십시오. DB2 명령 UPDATE DATABASE MANAGER CONFIGURATION을 사용하여 UDF_MEM_SZ 매개변수를 갱신할 수 있습니다. UPDATE DATABASE MANAGER 명령에 관한 자세한 내용은 *DB2 Command Reference*를 참조하십시오.

서버에 상주하는 오브젝트 저장

저장하려는 이미지, 오디오 또는 비디오가 서버 파일에 있을 때, UDF로의 내용 인수로 경로를 지정하십시오. 예를 들어, C 응용프로그램에 있는 다음의 명령문은 이미지가 있는 행을 데이터베이스에 저장합니다. 이미지 내용은 서버의 파일에 있습니다. 저장된 이미지는 서버 파일에 남고, 데이터베이스에서 지시됩니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp', /*source in server file */
          'BMP',
          :hvStorageType,
          'Anita''s picture')
       );
```

정확한 경로 지정: 서버 파일에 소스가 있는 오브젝트 저장시, 파일의 완전히 규정된 이름이나 상대 이름을 지정할 수 있습니다. 상대 이름 지정시, DB2 서버의 적절한 환경 변수가 파일에 대해 정확한 경로를 포함하도록 해야 합니다. 이러한 환경 변수의 설정에 대한 정보는 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』을 참조하십시오.

데이터베이스 또는 파일 저장 지정

BLOB으로 데이터베이스 테이블이나 서버 파일에 이미지, 오디오 또는 비디오 오브젝트를 저장할 수 있습니다. 서버 파일에 오브젝트를 저장한다면, 데이터베이스는 파일을 지시합니다.

클라이언트 버퍼 또는 클라이언트 파일에 오브젝트를 저장할 때, target_file 매개변수에 지정한 것의 결과로 BLOB 또는 서버 파일 저장을 표시합니다. 파일 이름을 지정한다면, 서버 파일에 오브젝트를 저장하려는 것을 나타냅니다. 널(NULL) 값 또는 빈 문자열을 지정하는 경우, 이는 오브젝트를 데이터베이스 테이블에 BLOB으로 저장하려는 것을 나타냅니다. target_file 매개변수의 데이터 유형은

LONG VARCHAR입니다. 널(NULL) 값을 지정하는 경우, 그것을 LONG VARCHAR 데이터 유형에 제공해야 함을 기억하십시오.

예를 들어, C 응용프로그램에 있는 다음 명령문은 이미지가 있는 행을 데이터베이스 테이블에 저장합니다. 이미지 소스는 클라이언트 버퍼에 있습니다. 이미지는 서버 파일에 저장됩니다. 데이터베이스 테이블은 다음과 같은 서버 파일을 지시합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR Img_buf
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          :Img_buf,
          'ASIS',
          '/employee/images/ajones.bmp',    /* store image in server file */
          'Anita''s picture')
       );
```

서버 파일에 오브젝트를 저장한다면, 오브젝트를 BLOB으로 데이터베이스 테이블에 저장하는 상수 MMDB_STORAGE_TYPE_INTERNAL을 지정하십시오. 오브젝트를 저장하고 그것의 내용을 서버 파일에 남겨두려 한다면, 상수 MMDB_STORAGE_TYPE_EXTERNAL을 지정하십시오. MMDB_STORAGE_TYPE_INTERNAL은 정수값 1을 갖고, MMDB_STORAGE_TYPE_EXTERNAL은 정수값 0을 갖습니다.

예를 들어, 다음의 C 응용프로그램에서 오디오 클립은 서버 파일에 저장됩니다. 소스 오디오 내용은 이미 서버 파일에 있습니다. 저장 조작은 데이터베이스 파일 이름을 위치시키는데, 이것은 SQL문을 통해 파일을 액세스 가능하게 합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2AUDIO(
          CURRENT SERVER,
```

```
        '/employee/sounds/ajones.wav',
        'WAVE',
        :hvStorageType,              /* store audio in server file */
        'Anita''s voice')
    );
```

저장 형식 식별

오브젝트 저장시, 형식을 식별해야 합니다. 지정할 수 있는 형식은 97 페이지의 표 5에 나열됩니다. Extender는 이미지, 오디오 또는 비디오 오브젝트를 소스와 동일 형식으로 저장할 것입니다. 이미지 오브젝트의 경우에만, Image Extender가 저장된 이미지 형식을 변환시키는 옵션을 갖습니다. 이미지 형식을 변환하도록 선택하면, 소스 이미지 형식과 목표 이미지 형식을 지정해야 합니다. 목표 이미지는 저장될 때 이미지입니다.

변환 없이 저장시 형식 식별

형식 변환 없이 오브젝트 저장시, 소스 이미지, 오디오 또는 비디오 오브젝트의 형식을 지정하십시오. 예를 들어, C 응용프로그램에 있는 다음 명령문은 데이터베이스 테이블에 비트맵(BMP) 이미지를 저장합니다. 소스 내용은 서버 파일에 있습니다. 목표 이미지는 소스와 동일한 형식을 갖습니다.

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'BMP',                          /*image in BMP format */
          '',
          'Anita''s picture')
       );
```

널(Null) 값이나 빈 문자열을 형식, 또는 Image Extender의 경우에 한해 문자열 ASIS를 지정할 수도 있습니다. 그런 다음 Extender는 소스를 검색하여 형식을 결정할 것입니다.

인식 가능한 형식에 대한 널(NULL)값이나 ASIS 사용: 형식이 Extender에 인식 가능한 경우에만 즉, 97 페이지의 표5에 Extender용으로 나열된 형식의 하나인 경우에 널(NULL)값이나 빈 문자열 또는 ASIS를 지정하십시오. 그렇지 않으면 Extender는 오브젝트를 저장할 수 없습니다.

형식 변환을 사용하여 저장시 형식과 변환 옵션 식별

형식 변환을 사용하여 이미지 저장시, 소스와 목표 이미지 형식 모두를 지정하십시오. 97 페이지의 표5에는 허용되는 형식 변환을 나열합니다.

또한 저장된 이미지를 적용하려는 추가 변경사항(회전이나 압축 같은)을 식별하는 변환 옵션을 지정할 수 있습니다. 매개변수와 관련 값으로 각각의 변환 옵션을 지정합니다. 매개변수와 허용값은 99 페이지의 표6에 나열됩니다. 복수의 매개변수/값 쌍을 지정하여 저장된 이미지에 다양한 변경사항을 요청할 수 있습니다.

다음 예시에서, 서버 파일에 내용이 있는 비트맵(BMP) 이미지는 데이터베이스 테이블에 저장시 GIF 형식으로 변환됩니다.

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'BMP',                          /* source format */
          'GIF',                          /* target format */
          '',
          'Anita''s picture')
       );
```

다음 예시에서, 이전 예시의 이미지는 데이터베이스 테이블에 저장시 GIF 형식으로 변환됩니다. 또한 이미지는 저장시 110 픽셀 폭과 150 픽셀 높이로 잘리고 LZW 압축을 사용하여 압축됩니다.

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'BMP',                          /* source format */
          'GIF',                          /* target format */
          '-x 110 -y 150 -c 14',          /* conversion options */
          '/employee/images/ajones.gif',
          'Anita''s picture')
       );
```

사용자 제공 속성으로 오브젝트 저장

이미지, 오디오 또는 비디오 오브젝트의 저장시, Extender가 이해하는 형식으로 제한되지 않습니다. 사용자 자신의 형식을 지정할 수 있습니다. Extender가 형식을 이해하지 못한다면, 소스 오브젝트의 속성을 지정해야 합니다. 속성 구조에 속성값을 지정하십시오. 속성 구조는 UDF에 있는 LONG VARCHAR FOR BIT DATA 변수의 데이터 필드에 저장해야 합니다.

서버의 UDF 코드는 항상 『big endian 형식』의 데이터를 예상합니다. Big endian 형식은 대부분의 UNIX 플랫폼에서 사용하는 형식입니다. 『little endian 형식』으로 오브젝트를 저장중인 경우, 서버의 UDF 코드가 제대로 처리할 수 있도록 사용자 제공 속성 데이터를 준비해야 합니다. Little endian 형식은 Intel® 및 기타 마이크로프로세서 플랫폼에서 일반적으로 사용하는 형식입니다. (little endian 형식으로 오브젝트를 저장하지 않는 경우에도, 사용자 제공 속성 데이터를 준비하는 것이 좋습니다.) 이미지 오브젝트의 속성을 준비하려면 DBiPrepareAttrs API를 사용하십시오. 오디오 오브젝트의 속성을 준비하려면 DBaPrepareAttrs API를 사용하십시오. 비디오 오브젝트의 속성을 준비하려면 DBvPrepareAttrs API를 사용하십시오.

예를 들어, C 응용프로그램에 있는 다음 명령문은 데이터베이스 테이블에 이미지가 있는 행을 저장합니다. 서버 파일에 있는 소스 이미지는 사용자 정의 형식으로, 640 픽셀의 높이와 480 픽셀의 폭을 갖습니다. 이미지가 저장되기 전에 속성이 준비됨을 유의하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[400];
       }hvImgattrs;
EXEC SQL END DECLARE SECTION;

DB2IMAGEATTRS    *pimgattr;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

pimgattr = (DB2IMAGEATTRS *) hvImgattrs.data;
strcpy(pimgattr→format,"FormatI");
pimgattr→width=640;
pimgattr→height=480;
```

```
hvImgattrs.len=sizeof(DB2IMAGEATTRS);

DBiPrepareAttrs(pimgattr);

DBEXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          :hvStorageType,
          'Anita''s picture',
          :hvImgattrs,                    /* user-specified attributes */
          CAST(NULL as LONG VARCHAR)
       );
```

C 응용프로그램에 있는 다음 명령문은 데이터베이스 테이블에 오디오 클립이 있는 행을 저장합니다. 서버 파일에 있는 소스 오디오 클립에는 사용자 정의 형식으로, 44.1 kHz의 샘플링율과 두 개의 레코드된 채널이 있습니다. 오디오 클립이 MIDI가 아니므로, 트랙 이름과 악기에 빈 문자열을 지정합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct (
       short len;
       char data[600];
}hvAudattr;
EXEC SQL END DECLARE SECTION;

MMDBAudioAttrs          *paudiattr;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

paudioattr=(MMDBAudioAttrs *) hvAudattr.data;
strcpy(paudioAttr→cFormat,"FormatA");
paudioAttr→ulSamplingRate=44100;
paudioAttr→usNumChannels=2;
hvAudattrs.len=sizeof(MMDBAudioAttrs);

DBaPrepareAttrs(paudioAttr);

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2AUDIO(
          CURRENT SERVER,
```

```
            '/employee/sounds/ajones.aud',
            :hvStorageType,
            'Anita''s voice',
            :hvAudattr)                    /* user-specified attributes */
        );
```

썸네일 저장(이미지와 비디오 전용)

사용자 자신의 형식의 이미지 저장시, 이미지의 소형 버전인 **썸네일**을 저장할 수도 있습니다. 사용자는 썸네일의 *크기*와 형식을 제어합니다. Image Extender가 인식하는 형식에 이미지 저장시, 오브젝트으로 대해 썸네일을 자동으로 생성하고 저장합니다. Image Extender는 112 x 84 픽셀 *크기*의 GIF 형식으로 썸네일을 만듭니다.

사용자 자신의 형식의 비디오 오브젝트 저장시, 비디오 오브젝트를 상징하는 썸네일을 저장할 수도 있습니다. Video Extender가 인식하는 형식으로 비디오 오브젝트 저장시, 오브젝트에 대한 일반 썸네일을 자동으로 저장합니다. Video Extender는 108 x 78 픽셀 *크기*의 GIF 형식으로 썸네일을 만듭니다.

사용자 제공 속성을 사용하여 이미지 또는 비디오 오브젝트를 저장할 때 썸네일을 저장하지 않으려면, 썸네일에 널(NULL) 값 또는 빈 문자열을 지정하십시오.

프로그램에서 썸네일을 생성하십시오. Extender는 썸네일을 생성하는 API를 제공하지 않습니다. 프로그램에 썸네일용 구조를 만들어 UDF 내에 썸네일 구조를 지정하십시오.

C 응용프로그램에 있는 다음 명령문은 데이터베이스 테이블에 비디오 클립이 있는 행을 저장합니다. 내용이 서버 파일에 있는 소스 비디오 클립은 사용자 정의 형식으로 되어 있습니다. 비디오 내용은 서버에 남고 테이블에서 지시됩니다. 대표 비디오 프레임의 썸네일도 저장됩니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
struct {
       short len;
       char data[4000];
       }hvVidattrs;
struct {
       short len;
       char data[10000];
```

```
      }hvThumbnail;
EXEC SQL END DECLARE SECTION;

MMDBVideoAttrs         *pvideoAttr;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

pvideoAttr=(MMDBVideoAttrs *) hvVidattrs.data;
strcpy(pvideoAttr->cFormat,"Formatv");
hvVidattrs.len=sizeof(MMDBVideoAttrs);

/* Generate thumbnail and assign data in video structure */

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2VIDEO(
          CURRENT SERVER,
          '/employee/videos/ajones.vid',
          :hvStorageType,
          'Anita''s video',
          :hvVidattrs,
          :hvThumbnail)                    /* Thumbnail*/
       );
```

주석 저장

UDF 요청에서 주석을 지정하여 이미지, 오디오 또는 비디오 오브젝트와 함께 주석을 저장하십시오. 주석은 최대 32,700 바이트까지 가능한 LONG VARCHAR 데이터 유형의 무형식 텍스트입니다. 오브젝트를 저장할 때 주석을 저장하지 않으려면, 널(NULL) 값이나 빈 문자열을 주석 대신 지정하십시오. 널(NULL) 값을 지정하는 경우, 그것을 LONG VARCHAR 데이터 유형으로 제공해야 함을 기억하십시오.

예를 들어, C 응용프로그램에 있는 다음 명령문은 비디오 클립과 함께 주석을 저장합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
```

```
        'Anita Jones',
        DB2VIDEO(
          CURRENT SERVER,
          '/employee/videos/ajones.mpg',
          'MPEG1',
          :hvStorageType,
          'Anita''s video')                /* comment */
       );
```

C 응용프로그램에 있는 다음 명령문은 주석 없이 이미지를 저장합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          'GIF',
          :hvStorageType,
          Cast(NULL as LONG
VARCHAR)            /* no comment */
       );
```

이미지, 오디오 또는 비디오 오브젝트 검색

데이터베이스 테이블에서 이미지, 오디오 또는 비디오 오브젝트를 검색하려면, SQL SELECT문에 Content UDF를 사용하십시오. 오브젝트를 클라이언트 버퍼, 클라이언트 파일 또는 서버 파일로 검색할 수 있습니다.

검색을 위한 Content UDF 형식

Content UDF는 오버로드되며, UDF를 사용하는 방법에 따라 서로 다른 형식을 갖게 되는 것을 의미합니다. 형식은 다음과 같습니다.

형식 1: 오브젝트를 클라이언트 버퍼 또는 클라이언트 파일로 검색합니다.

```
Content(
     handle,                        /* object handle */
  );
```

형식 2: 오브젝트 세그먼트를 클라이언트 버퍼 또는 클라이언트 파일로 검색합니다.

```
Content(
     handle,                        /* object handle */
     offset,                        /* offset where retrieval begins */
     size                           /* number of bytes to retrieve */
  );
```

형식 3: 오브젝트를 서버 파일로 검색합니다.

```
Content(
     handle,                    /* object handle */
     target_file,               /* server file name */
     overwrite                  /* 0=Do not overwrite target file if it exists */
                                /* 1=Overwrite target file */

  );
```

또한 Content UDF는 이미지 오브젝트의 경우에 한해 다음 형식을 포함합니다.

형식 4: 형식 변환을 사용하여 이미지를 클라이언트 버퍼 또는 파일로 검색합니다.

```
Content(
     handle,                        /* object handle */
     target format                  /* target format */
       );
```

형식 5: 형식 변환을 사용하여 오브젝트를 서버 파일로 검색합니다.

```
Content(
     handle,                    /* object handle */
     target_file,               /* server file name */
     overwrite,                 /* 0=Do not overwrite target file if it exists */
                                /* 1=Overwrite target file */
     target format              /* target format */

  );
```

형식 6: 형식 변환 및 추가 변경을 사용하여 오브젝트를 클라이언트 버퍼 또는 파일로 검색합니다.

```
Content(
     handle,                        /* object handle */
     target format,                /* target format */
     conversion_options             /* conversion options */
  );
```

형식 7: 형식 변환 및 추가 변경을 사용하여 오브젝트를 서버 파일로 검색합니다.

```
Content(
     handle,                /* object handle */
     target_file,           /* server file name */
     overwrite,             /* 0=Do not overwrite target file if it exists */
                            /* 1=Overwrite target file */
     target format,         /* target format */
     conversion_options     /* conversion options */
  );
```

예를 들어, 다음의 명령문은 직원 테이블에서 서버에 있는 파일로 이미지를 검색합니다. (이것은 형식 3에 대응합니다.)

```
EXEC SQL SELECT CONTENT(                /* retrieval UDF */
       PICTURE,                         /* image handle */
       '/employee/images/ajones.bmp',   /* target file */
       1)                               /* overwrite target file */
FROM EMPLOYEE
WHERE NAME = 'Anita Jones';
```

C 응용프로그램에 있는 다음 명령문은 직원 테이블에서 서버에 있는 파일로 이미지를 검색합니다. 이미지 형식은 검색될 때 변환됩니다. (이것은 형식 5에 대응합니다.)

```
EXEC SQL BEGIN DECLARE SECTION;
  char hvImg_fname[255];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(                  /* retrieval UDF */
       PICTURE,                           /* image handle */
       '/employee/images/ajones.bmp',     /* target file */
       1,                                 /* overwrite target file */
       'GIF')                             /* target format */
INTO :hvImg_fname
FROM EMPLOYEE
WHERE NAME = 'Anita Jones';
```

오브젝트를 클라이언트로 검색

형식 변환 없이 이미지, 오디오 또는 비디오 오브젝트를 클라이언트 버퍼나 클라이언트 파일로 검색하려면, Content UDF를 사용할 수 있습니다. 또한 사용자는 검색시 이미지 형식을 변환시키는 Image Extender 옵션을 갖습니다.

형식 변환 없이 오브젝트를 클라이언트로 검색

이미지, 오디오 또는 비디오 오브젝트를 클라이언트 버퍼로 검색하거나 LOB을 검색하려면 LOB 위치 지정자를 사용하십시오. 이미지, 오디오 또는 비디오 오브젝트를 클라이언트 파일로 검색하려면, 파일 참조 변수를 사용하십시오.

이미지, 오디오 또는 비디오 오브젝트를 호스트 변수를 사용하여 클라이언트 버퍼로 검색하거나, 파일 참조 변수를 사용하여 클라이언트 파일로 검색하는 것은 오브젝트의 내용을 데이터베이스 테이블에 BLOB으로 저장할 때 적합합니다. 내용이 서버 파일에 있다면, 그것은 서버 파일에서 클라이언트 파일로 내용을 복사하는 편이 더 효율적일 수 있습니다.

오브젝트의 핸들을 지정하십시오. 선택적으로, 사용자는 또한 검색이 시작되는 바이트 1에서 시작하는 오프셋과 검색하려는 바이트 수를 지정할 수 있습니다.

C 응용프로그램에 있는 다음의 명령문은 audio_loc라는 LOB 위치 지정자를 사용하여 오디오 클립을 클라이언트 버퍼로 검색합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_LOCATOR audio_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(
        SOUND)                                  /* audio handle */
     INTO :audio_loc
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

UDF 메모리가 충분한지 확인하십시오. 오브젝트를 클라이언트 버퍼로 검색할 때, 데이터베이스 관리 프로그램 구성에 있는 UDF_MEM_SZ 매개변수가 4MB 이상으로 설정되었는지 확인해야 합니다. DB2 명령 UPDATE DATABASE MANAGER CONFIGURATION을 사용하여 UDF_MEM_SZ 매개변수를 갱신할 수 있습니다. UPDATE DATABASE MANAGER 명령에 대한 정보는 *DB2 Command Reference*를 참조하십시오.

변환하여 이미지를 클라이언트로 검색

저장된 이미지를 형식 변환을 사용하여 클라이언트 버퍼로 검색하거나 LOB을 검색하려면 LOB 위치 지정자를 사용하십시오. 저장된 이미지를 형식 변환을 사용하여 클라이언트 파일로 검색하려면 파일 참조 변수를 사용하십시오.

이미지를 호스트 변수를 사용하여 클라이언트 버퍼로 검색하거나 파일 참조 변수를 사용하여 클라이언트 파일로 검색하는 것은 이미지의 내용이 데이터베이스 테이블에 BLOB으로 저장될 때 적합합니다. 내용이 서버 파일에 있다면, 그것은 서버 파일에서 클라이언트 파일로 내용을 복사하는 편이 더 효율적일 수 있습니다.

형식 변환을 사용하여 이미지 검색시, 목표 형식(즉, 변환된 형식)을 지정해야 합니다. 97 페이지의 표5는 허용되는 형식 변환을 식별합니다. 검색된 이미지를 적용하려는 추가 변경사항(회전이나 배율 같은)을 식별하는 변환 옵션도 지정할 수 있습니다. 99 페이지의 표6은 지정할 수 있는 변환 옵션을 나열합니다.

예를 들어, C 응용프로그램에 있는 다음 명령문은 이미지를 클라이언트 파일로 검색합니다. 소스 이미지는 비트맵 형식이고 데이터베이스 테이블에 BLOB으로 저장됩니다. 검색된 이미지는 GIF로 변환되고 원래 *크기*의 3배로 조정됩니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Img_file;
EXEC SQL END DECLARE SECTION;

strcpy (Img_file.name, "/employee/images/ajones.gif");
Img_file.name_length= strlen(Img_file.name);
Img_file.file_options= SQL_FILE_CREATE;

EXEC SQL SELECT CONTENT(
        PICTURE,                              /* image handle */
        'GIF',                                /* target format */
        '-s 3.0')                             /* conversion options */
     INTO :Img_file,
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

오브젝트를 서버 파일로 검색

형식 변환 없이 서버 파일로 이미지, 오디오 또는 비디오 오브젝트를 검색하려면, Content UDF를 사용할 수 있습니다. 또한 형식 변환을 사용하여 서버 파일로 이미지를 검색하려면, Content UDF를 사용할 수 있습니다.

변환 없이 서버에 있는 파일로 이미지, 오디오 또는 비디오 오브젝트 검색시, 오브젝트 핸들, 목표 파일 이름 및 겹쳐쓰기 표시기를 지정해야 합니다. 목표 파일이

서버에 이미 존재한다면, 겹쳐쓰기 표시기는 검색된 데이터로 목표 파일을 겹쳐쓸 것인지 Extender에게 지시합니다. 목표 파일이 존재하지 않는 경우, Extender는 서버에 목표 파일을 작성합니다.

겹쳐쓰기 표시기 값 1을 지정한다면, Extender는 검색된 데이터로 목표 파일을 겹쳐씁니다. 겹쳐쓰기 표시기 값 0을 지정한다면, Extender는 목표 파일을 겹쳐쓰지 않고 데이터는 검색되지 않습니다.

겹쳐쓰기 표시기는 검색될 오브젝트가 BLOB으로 데이터베이스 테이블에 저장된다면 무시됩니다. 목표 파일은 겹쳐쓰기 표시기에 대해 무엇이 지정되든지 상관없이 작성되거나 겹쳐씁니다.

서버 파일로 오브젝트 검색시, 이것은 서버 파일 이름을 리턴합니다. 예를 들어, C 응용프로그램에 있는 다음 명령문은 서버에 있는 파일로 비디오를 검색합니다. 서버 파일의 파일 이름은 호스트 변수 hvVid_fname에 저장됩니다.

```
EXEC SQL BEGIN DECLARE SECTION;
struct{
       short len;
       char data[250];
      }hvVid_fname[;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(
        VIDEO,                                  /* video handle */
        '/employee/videos/ajones.mpg',          /* server file */
        1)                                      /* overwrite target file */
     INTO :hvVid_fname;
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

변환 없이 서버 파일로 오브젝트를 검색하는 Content UDF의 사용은 오브젝트가 BLOB으로 데이터베이스 테이블에 저장될 때 적절합니다. 오브젝트가 서버 파일에 있다면, 소스 파일의 내용을 목표 파일로 복사하는 편이 더 효율적일 수 있습니다.

형식 변환을 사용하여 이미지를 서버 파일로 검색할 때, 이미지 핸들, 목표 파일 이름, 겹쳐쓰기 목표 표시기 및 목표 형식을 지정하십시오. 97 페이지의 표5는 허용되는 형식 변환을 나열합니다. 또한 목표 형식에 널(NULL) 값 또는 빈 문자열이나 문자열 ASIS를 지정할 수도 있습니다. 이 경우, 검색되는 이미지는 소스와 같은 형식을 갖습니다.

예를 들어, C 응용프로그램에 있는 다음 명령문은 서버에 있는 파일로 이미지를 검색합니다. 소스 이미지는 비트맵 형식이고 BLOB으로 데이터베이스 테이블에 저장됩니다. 검색된 이미지는 GIF 형식으로 변환됩니다. 서버 파일의 파일 이름은 호스트 변수 hvImg_fname에 저장됩니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  struct{
        short len;
        char [400];
        }hvImg_fname[;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT(
        PICTURE,                              /* image handle */
        '/employee/images/ajones.gif',        /* target file */
        1,                                    /* overwrite target file */
        'GIF')                                /* target format */
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

서버 파일이 액세스 가능해야 합니다. 오브젝트를 서버 파일로 검색할 때, 목표 파일의 완전히 규정된 이름을 지정해야 합니다. 또다른 방법으로, DB2IMAGEEXPORT, DB2AUDIOEXPORT 및 DB2VIDEOEXPORT 환경 변수가 불완전한 파일 이름 스펙을 제대로 해결할 수 있도록 설정되었는지 확인해야 합니다.

속성의 검색과 사용

데이터베이스에 이미지, 오디오 또는 비디오 오브젝트 저장시, Extender는 데이터베이스에 오브젝트 속성 또한 저장합니다. 오브젝트 갱신시, Extender는 데이터베이스에 저장된 오브젝트의 속성을 갱신합니다. 이러한 속성은 조회에서 사용 가능합니다.

Extender는 관리하는 각각의 속성에 대한 UDF를 만듭니다. 그 결과, 사용자는 오브젝트 속성을 액세스하고 사용하기 위해 SQL문에 UDF를 지정할 수 있습니다. 121 페이지의 표7은 Extender가 관리하는 속성과 자신의 UDF를 나열합니다. 이것은 또한 각각의 속성에 대한 오브젝트 유형을 표시합니다. 오브젝트 형식 및 파일 이름과 같은, 일부 속성은 모든 오브젝트 유형에 공통입니다. 이들 속성은 이

미지, 오디오 및 비디오 오브젝트와 연관됩니다. 샘플링율이나 압축 유형 같은 그 밖의 속성은 오디오와 비디오 같은 특정 오브젝트 유형에 지정됩니다.

표 7. DB2 Extender가 관리하는 속성. 이것의 UDF로 각각의 속성에 액세스할 수 있습니다.

속성	UDF	Image	Audio	Video
오브젝트가 저장된 서버 파일 이름	Filename	x	x	x
오브젝트를 저장한 사람의 사용자 ID	Importer	x	x	x
오브젝트 저장시 날짜와 시간	ImportTime	x	x	x
오브젝트의 바이트 크기	Size	x	x	x
오브젝트를 최근 갱신한 사람의 사용자 ID	Updater	x	x	x
오브젝트의 최근 갱신한 날짜와 시간	UpdateTime	x	x	x
오브젝트 형식(예를 들어, GIF 또는 MPEG1)	Format	x	x	x
오브젝트에 대한 주석	Comment	x	x	x
오브젝트의 높이(픽셀)	Height	x		x
오브젝트의 폭(픽셀)	Width	x		x
오브젝트의 색상 수	NumColors	x		
오브젝트의 썸네일 크기 이미지	Thumbnail	x		x
오디오 또는 비디오의 오디오 트랙에서 샘플당 리턴되는 바이트 수	AlignValue		x	x
각각의 샘플의 표현에 사용되는 비트 수	BitsPerSample		x	x
레코드된 채널 수	NumChannels		x	x
지속기간(초 단위)	Duration		x	x
샘플링율(초당 샘플)	SamplingRate		x	x
전송 시간 초당 평균 바이트	BytesPerSec		x	
악기용 오디오 트랙 번호	FindInstrument		x	
이름 지정된 트랙의 트랙 번호	FindTrackName		x	
레코드된 악기 이름	GetInstruments		x	
레코드된 악기의 트랙 번호와 이름	GetTrackNames		x	
오디오의 초당 시계 틱 수	TicksPerSec		x	
오디오의 1/4 노트당 시계 틱 수	TicksPerQNote		x	
영상비	AspectRatio			x
비디오 압축 형식(예: MPEG1)	CompressType			x

표 7. DB2 Extender가 관리하는 속성 (계속). 이것의 UDF로 각각의 속성에 액세스할 수 있습니다.

속성	UDF	Image	Audio	Video
처리량의 초당 프레임	FrameRate			x
최대 처리량(초당 바이트)	MaxBytesPerSec			x
오디오 트랙 수	NumAudioTracks		x	x
프레임 수	NumFrames			x
비디오 트랙 수	NumVideoTracks			x

속성 UDF는 SQL문 SELECT절 표현식 또는 WHERE절 탐색 조건에서 사용할 수 있습니다. UDF 요청시, 오브젝트 핸들이 있는 데이터베이스 테이블에 컬럼 이름을 지정합니다.

예를 들어, 다음의 명령문은 SQL SELECT문의 SELECT절에서 Updater UDF를 사용하여 직원 테이블에서 이미지를 가장 최근에 갱신한 사람의 사용자 ID를 검색합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
char hvUpdatr[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT UPDATER(PICTURE)
     INTO :hvUpdatr
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

다음의 명령문은 SELECT문의 SELECT절에 Filename UDF와 WHERE절에 NumAudioTracks UDF를 사용하여, 오디오 트랙이 있는 직원 테이블에 저장된 비디오를 찾습니다.

```
EXEC SQL BEGIN DECLARE SECTION;
char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NUMAUDIOTRACKS(VIDEO)>0;
```

주석 검색

이미지, 오디오 또는 비디오 오브젝트로 저장된 주석을 검색하려면, Comment UDF를 사용하십시오. 오브젝트에 대한 주석 검색시, 오브젝트 핸들이 있는 데이터베이스 테이블에서 컬럼을 지정합니다. 예를 들어, 다음의 명령문은 직원 테이블에 오디오 클립을 사용하여 저장된 주석을 검색합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
struct {
       short len;
       char data[32700];
       }hvComment
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT COMMENT(SOUND)
     INTO :hvComment
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

Comment UDF를 SQL 조회의 WHERE절에서 술어처럼 사용할 수도 있습니다. 예를 들어, 다음의 명령문은 『touched up』이라는 주석이 있는 직원 테이블의 모든 이미지 파일 이름을 검색합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
struct {
       short len;
       char data[250];
       }hvImg_fname
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE COMMENT(PICTURE)
           LIKE '%touch%up';
```

이미지, 오디오 또는 비디오 오브젝트 갱신

데이터베이스 테이블에 있는 이미지, 오디오 또는 비디오 오브젝트를 갱신하려면, SQL UPDATE문에서 Content UDF를 사용하십시오. 데이터베이스 테이블에 있는 이미지, 오디오 또는 비디오를 갱신하고 오브젝트와 연관된 주석을 갱신하려면 SQL UPDATE문에 Replace UDF를 사용하십시오. 두 경우 모두, Extender는 오브젝트와 연관된 속성을 갱신합니다.

BLOB으로 데이터베이스 테이블에 저장되었거나 서버 파일에 저장된 (그리고 데이터베이스에서 지시되는) 오브젝트를 갱신할 수 있습니다. 갱신 소스는 버퍼, 클라이언트 파일 또는 서버 파일이 될 수 있습니다.

97 페이지의 표5는 이미지, 오디오 또는 비디오 오브젝트를 갱신할 수 있는 형식을 나열합니다. 그러나 Extender가 인식하지 못하는 형식의 오브젝트도 갱신할 수 있습니다. 이 경우, 오브젝트 속성은 오브젝트 저장시 사용자가 지정합니다. 사용자가 지정하는 속성을 사용하여 오브젝트를 갱신할 때, 오브젝트의 갱신된 속성을 지정해야 합니다. SQL UPDATE문에 ContentA UDF를 사용하여 데이터베이스에서 사용자 제공 속성을 사용하여 이미지, 오디오 또는 비디오 오브젝트를 갱신하십시오. 데이터베이스 테이블에서 사용자 제공 속성을 사용하여 이미지, 오디오 또는 비디오를 갱신하고 오브젝트와 연관된 주석을 갱신하려면, SQL UDPDATE문에서 ReplaceA UDF를 사용하십시오. 사용자 지정 속성으로 오브젝트를 갱신할 때, 오브젝트의 속성, 형식 및 비디오 오브젝트의 경우에 한해 압축 형식을 지정해야 합니다.

저장된 이미지나 비디오에 대한 썸네일도 갱신할 수 있습니다.

갱신 조작 확약: 데이터베이스에서 이미지, 오디오 또는 비디오 오브젝트를 갱신한 후 작업 단위(UOW)를 확약하십시오. 이것은 Extender가 보유하는 잠금을 해제하여 저장된 오브젝트에 관한 후속 갱신 조작을 수행할 수 있습니다.

갱신을 위한 Content UDF 형식

Content UDF는 오버로드되며, UDF를 사용하는 방법에 따라 서로 다른 형식을 갖게 되는 것을 의미합니다. 형식은 다음과 같습니다.

형식 1: 클라이언트 버퍼 또는 클라이언트 파일에서 오브젝트를 갱신합니다.

```
Content(
     handle,                    /* object handle */
     content,                   /* object content */
     source_format,             /* source format */
     target_file                /* target file name for storage in file */
                                /* server or NULL for storage in table as BLOB */
  );
```

형식 2: 서버 파일에서 오브젝트를 갱신합니다.

```
Content(
     handle,                    /* object handle */
     source_file,               /* server file name */
     source_format,             /* source format */
     stortype                   /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server */
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
  );
```

형식 3: 클라이언트 버퍼나 클라이언트 파일에서 사용자 제공 속성으로 오브젝트를 갱신합니다.

```
Content(
     handle,                  /* object handle */
     content,                 /* object content */
     target_file,             /* target file name for storage in file server */
                              /* or NULL for storage in table as BLOB */
     attrs,                   /* user-supplied attributes */
     thumbnail                /* thumbnail (image and video only) */
  );
```

형식 4: 서버 파일에서 사용자 제공 속성으로 오브젝트를 갱신합니다.

```
Content(
     handle,                    /* object handle */
     source_file,               /* source file name */
     stortype,                  /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server*/
                                /* MMDB_STORAGE_TYPE_INTERNAL=store */
                                /* as a BLOB*/
     attrs,                     /* user-supplied attributes */
     thumbnail                  /* thumbnail (image and video only) */
  );
```

이미지 오브젝트의 경우에 한해 Content UDF는 다음의 추가 형식을 갖습니다.

형식 5: 형식 변환을 사용하여 클라이언트 버퍼 또는 클라이언트 파일에서 이미지를 갱신합니다.

```
Content(
     handle,                    /* object handle */
     content,                   /* object content */
     source format,             /* source format */
     target format,             /* target format */
     target_file                /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
  );
```

형식 6: 형식 변환을 사용하여 서버 파일에서 오브젝트를 갱신합니다.

```
Content(
     handle,                    /* object handle */
     source_file,               /* server file name */
```

```
    source format,              /* source format */
    target format,              /* target format */
    target_file                 /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
 );
```

형식 7: 형식 변환 및 추가 변경을 사용하여 클라이언트 버퍼 또는 클라이언트 파일에서 이미지를 갱신합니다.

```
Content(
    handle,                     /* object handle */
    content,                    /* object content */
    source format,              /* source format */
    target format,              /* target format */
    conversion_options,         /* conversion options */
    target_file                 /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
 );
```

형식 8: 형식 변환 및 추가 변경을 사용하여 서버 파일에서 오브젝트를 갱신합니다.

```
Content(
    handle,                     /* object handle */
    source_file,                /* server file name */
    source format,              /* source format */
    target format,              /* target format */
    conversion_options,         /* conversion options */
    target_file                 /* target file name for storage in file server */
                                /* or NULL for storage in table as BLOB */
 );
```

예를 들어, C 응용프로그램에 있는 다음 명령문은 직원 테이블의 이미지를 갱신합니다. 갱신에 대한 소스 내용은 ajones.bmp라는 서버 파일에 있습니다. 갱신된 이미지는 직원 테이블에 BLOB으로 저장됩니다. (이것은 앞에서 설명한 형식 2에 대응합니다.)

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
               PICTURE,                        /*image handle*/
               '/employee/newimg/ajones.bmp',  /*source file */
               'ASIS',                         /*keep the image format*/
               '');                            /*store image in DB as BLOB*/
   WHERE NAME='Anita Jones';
```

예를 들어, C 응용프로그램에 있는 다음 명령문은 이전 예시와 동일한 이미지를 갱신합니다. 그러나 이러한 이미지는 갱신할 경우 BMP에서 GIF 형식으로 변환됩니다. (이것은 앞에서 설명한 형식 6에 해당됩니다.)

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
               PICTURE,                        /*image handle*/
               '/employee/newimg/ajones.bmp',  /*source file */
```

```
                'BMP',                          /*source format*/
                'GIF',                          /*target format*/
                '');                            /*store image in DB as BLOB*/
  WHERE NAME='Anita Jones';
```

갱신을 위한 Replace UDF 형식

Replace UDF는 오버로드되고 UDF의 사용 방법에 따라 다른 형식을 갖습니다. 형식은 다음과 같습니다.

형식 1: 클라이언트 버퍼 또는 클라이언트 파일에서 오브젝트를 갱신하고 해당 주석을 갱신합니다.

```
Replace(
     handle,                    /* object handle */
     content,                   /* object content */
     source_format,             /* source format */
     target_file,               /* target file name for storage in file */
     comment                    /* user comment */
  );
```

형식 2: 서버 파일에서 오브젝트를 갱신하고 해당 주석을 갱신합니다.

```
Replace(
     handle,                  /* object handle */
     source_file,             /* server file name */
     source_format,           /* source format */
     stortype,                /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                              /* in file server*/
                              /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment                  /* user comment */
  );
```

형식 3: 클라이언트 버퍼 또는 클라이언트 파일에서 사용자 제공 속성을 사용하여 오브젝트를 대체하고 주석을 갱신합니다.

```
Replace(
     handle,                    /* object handle */
     content,                   /* object content */
     target_file,               /* target file name for storage in file */
                                /* or NULL for storage in table as BLOB */
     comment,                   /* user comment */
     attrs,                     /* user-supplied attributes */
     thumbnail                  /* thumbnail */
  );
```

형식 4: 서버 파일에서 사용자 제공 속성으로 오브젝트를 저장합니다.

```
Replace(
     handle,                  /* object handle */
     source_file,             /* server file name */
     stortype,                /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                              /* in file server*/
```

```
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
    comment,                    /* user comment */
    attrs,                      /* user-supplied attributes */
    thumbnail                   /* thumbnail */
  );
```

이미지 오브젝트의 경우에 한해 Replace UDF는 다음의 추가 형식을 갖습니다.

형식 5: 형식 변환을 사용하여 클라이언트 버퍼 또는 클라이언트 파일에서 이미지를 갱신하고 해당 주석을 갱신합니다.

```
Replace(
    handle,                    /* object handle */
    content,                   /* object content */
    source_format,             /* source format */
    target_format,             /* target format */
    target_file,               /* target file name for storage in file server */
                               /* or NULL for storage in table as BLOB */
    comment                    /* user comment */
  );
```

형식 6: 형식 변환을 사용하여 서버 파일에서 오브젝트를 갱신하고 해당 주석을 갱신합니다.

```
Replace(
    handle,                     /* object handle */
    source_file,                /* server file name */
    source_format,              /* source format */
    target_format,              /* target format */
    target_file,                /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server */
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
    comment                     /* user comment */
  );
```

형식 7: 형식 변환 및 추가 변경을 사용하여 클라이언트 버퍼 또는 클라이언트 파일에서 이미지를 갱신하고 해당 주석을 갱신합니다.

```
Replace(
    handle,                    /* object handle */
    content,                   /* object content */
    source_format,             /* source format */
    target_format,             /* target format */
    conversion_options,        /* conversion options */
    target_file,               /* target file name for storage in file server */
                               /* or NULL for storage in table as BLOB */
    comment                    /* user comment */
  );
```

형식 8: 형식 변환 및 추가 변경을 사용하여 서버 파일에서 오브젝트를 갱신하고 해당 주석을 갱신합니다.

```
Replace(
     handle,                    /* object handle */
     source_file,               /* server file name */
     source_format,             /* source format */
     target_format,             /* target format */
     conversion_options,        /* conversion options */
     target_file,               /* MMDB_STORAGE_TYPE_EXTERNAL=store */
                                /* in file server */
                                /* MMDB_STORAGE_TYPE_INTERNAL=store as a BLOB*/
     comment                    /* user comment */
  );
```

예를 들어, C 응용프로그램에 있는 다음 명령문은 직원 테이블에 있는 오디오 클립과 연관된 주석을 갱신합니다. 갱신할 소스 내용은 ajones.wav라는 서버 파일에 있습니다. 갱신된 오디오 클립은 형식 변환 없이 BLOB으로 직원 테이블에 저장됩니다(Audio Extender는 형식 변환을 지원하지 않습니다). 이것은 앞에서 설명한 형식 2에 대응합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET SOUND=REPLACE(
                SOUND,                          /*audio handle*/
                '/employee/newaud/ajones.wav',  /*source file */
                'WAV',                          /*keep the audio format*/
                :hvStorageType,                 /*store audio in DB as BLOB*/
                'Anita''s new greeting')        /*user comment*/
   WHERE NAME='Anita Jones';
```

다음의 예시에서 이미지와 관련 주석이 갱신됩니다. 갱신할 소스 내용은 서버 파일에 있습니다. 갱신된 이미지는 BLOB으로 직원 테이블에 저장되고 BMP에서 GIF 형식으로 변환됩니다. (이것은 앞에서 설명한 형식 6에 해당됩니다.)

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=REPLACE(
               PICTURE,                          /*image handle*/
               '/employee/newimg/ajones.bmp',    /*source file */
               'BMP',                            /*source format*/
               'GIF',                            /*target format*/
               ''                                /*store image in DB as BLOB*/
               'Anita''s new picture')
   WHERE NAME='Anita Jones';                     /* user comment */
```

클라이언트에서 오브젝트 갱신

이미지, 오디오 또는 비디오 오브젝트를 클라이언트 버퍼 또는 클라이언트 파일에서 갱신하려면 호스트 변수 또는 파일 참조 변수를 사용하십시오.

갱신할 소스가 클라이언트 파일에 있다면, 내용을 전송하기 위해 파일 참조 변수를 사용하십시오. 예를 들어, C 응용프로그램에 있는 다음의 명령문은 Audio_file로 이름 지정된 파일 참조 변수를 정의하고, 그것을 사용하여 BLOB으로 데이터베이스 테이블에 저장된 오디오 클립을 갱신합니다. 갱신용 소스는 클라이언트 파일에 있습니다. 파일 참조 변수의 file_options 필드가 입력용 SQL_FILE_READ으로 설정됨을 유의하십시오. 또한 파일 참조 변수가 내용 인수로 Content UDF에 사용됨을 유의하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB_FILE Audio_file;
EXEC SQL END DECLARE SECTION;

strcpy (Audio_file.name, "/employee/newsound/ajones.wav");
Audio_file.name_length= strlen(Audio_file.name);
Audio_file.file_options= SQL_FILE_READ;

EXEC SQL UPDATE EMPLOYEE
   SET SOUND=CONTENT(
                SOUND,
                :Audio_file                 /*file reference variable*/
                'WAVE',                     /*keep the image format*/
                CAST(NULL as LONG VARCHAR))

   WHERE NAME='Anita Jones';
```

오브젝트가 클라이언트 버퍼에 있다면, 그것의 갱신용 내용을 전송하기 위해 호스트 변수를 사용하십시오. 다음의 C 응용프로그램 예시에서 Video_seg라고 이름 지정된 호스트 변수가 갱신용으로 비디오 클립의 내용을 전송하는 데 사용됩니다. 비디오 클립과 연관된 주석도 갱신됩니다. 비디오 클립은 BLOB으로 데이터베이스 테이블에 저장됩니다. 호스트 변수가 내용 인수로 Replace UDF에 사용됨을 유의하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB (2M) Video_seg
EXEC SQL END DECLARE SECTION;

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=REPLACE(
                VIDEO,
                :Video_seg                  /*host variable*/
                'MPEG1',
                CAST(NULL as LONG VARCHAR),

                'Anita''s new video')
   WHERE NAME='Anita Jones';
```

UDF 메모리가 충분한지 확인하십시오. 내용이 클라이언트 버퍼에 있는 오브젝트를 갱신할 때, 데이터베이스 관리 프로그램 구성의 UDF_MEM_SZ 매개변수가 4MB 이상으로 설정되었는지 확인하십시오. DB2 명령 UPDATE DATABASE MANAGER CONFIGURATION을 사용하여 UDF_MEM_SZ 매개변수를 갱신할 수 있습니다.

서버에서 오브젝트 갱신

이미지, 오디오 또는 비디오 오브젝트 갱신을 위한 소스 내용이 서버 파일에 있을 때, 파일 경로를 UDF로의 내용 인수로 지정하십시오. 예를 들어, C 응용프로그램에 있는 다음 명령문은 데이터베이스의 이미지를 갱신합니다. 이미지 내용은 서버 파일에 있습니다. 데이터베이스는 서버 파일을 가리킵니다. 갱신용 소스도 역시 서버 파일에 있습니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,                                /* image handle */
                '/employee/newimg/ajones.bmp',          /* source file */
                'ASIS',
                :hvStorageType)
   WHERE NAME='Anita Jones';
```

정확한 경로 지정: 서버 파일에 있는 소스의 오브젝트 갱신시, 파일의 완전히 규정된 이름이나 상대 이름을 지정할 수 있습니다. 상대 이름 지정시, DB2 서버의 해당 환경 변수가 파일에 대해 정확한 경로를 포함하도록 해야 합니다. 이러한 환경 변수의 설정에 대한 정보는 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』을 참조하십시오.

갱신을 위한 데이터베이스 또는 파일 저장 지정

BLOB으로 데이터베이스 테이블에, 또는 서버 파일에 저장되는 (그리고 데이터베이스에서 지시되는) 이미지, 오디오 또는 비디오를 갱신할 수 있습니다.

클라이언트 버퍼 또는 클라이언트 파일에서 오브젝트를 갱신할 경우, 파일 이름 매개변수에 지정한 것의 결과로 BLOB 또는 서버 파일 저장을 표시합니다. 파일 이

름을 지정한다면, 그것은 서버 파일에 내용이 있는 오브젝트를 저장하려는 것을 나타냅니다. 널(Null) 파일 이름을 지정한다면, 이것은 데이터베이스 테이블에 BLOB으로 저장된 오브젝트의 갱신을 하려는 것을 나타냅니다.

예를 들어, C 응용프로그램에 있는 다음 명령문은 서버 파일에 내용이 있는 이미지를 갱신합니다. 갱신 소스는 클라이언트 버퍼에 있습니다. 이미지 주석도 갱신됩니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  SQL TYPE IS BLOB (2M) Img_buf
EXEC SQL END DECLARE SECTION;

EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=REPLACE(
                  PICTURE,
                  :Img_buf,
                  'ASIS',
                  '/employee/newimg/ajones.bmp',      /*update image in*/
                                                      /*server file*/
                  'Anita''s new picture')
   WHERE NAME='Anita Jones';
```

서버 파일에서 오브젝트를 갱신하려면, 데이터베이스 테이블에 BLOB으로 저장된 오브젝트를 갱신하기 위해 MMDB_STORAGE_TYPE_INTERNAL을 지정하십시오. 서버 파일에 있는 내용의 오브젝트를 갱신하려면, MMDB_STORAGE_TYPE_EXTERNAL을 지정하십시오.

예를 들어, 다음의 C 응용프로그램에서 오디오 클립이 갱신됩니다. 오디오 클립의 내용은 서버 파일에 있습니다. 갱신용 소스도 역시 서버 파일에 있습니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET SOUND=CONTENT(
                 SOUND,
                 '/employee/newimg/ajones.wav',
                 'WAVE',
                 :hvStorageType)        /*update audio in server file*/
   WHERE NAME='Anita Jones';
```

갱신 형식 식별

오브젝트 갱신시 형식을 식별해야 합니다. Extender는 갱신되는 이미지, 오디오 또는 비디오 오브젝트를 소스와 동일 형식으로 저장할 것입니다. 이미지 오브젝트의 의 경우에 한해, 갱신된 이미지 형식을 변환시키는 Image Extender 옵션이 있습니다. 이미지 형식을 변환하려면, 갱신 소스 형식과 목표 이미지 형식을 지정해야 합니다. 목표 이미지는 저장될 때 갱신된 이미지입니다.

변환 없이 갱신 형식 식별

형식 변환 없이 오브젝트 갱신시, 소스 이미지, 오디오 또는 비디오 오브젝트의 형식을 지정하십시오. 예를 들어, C 응용프로그램에 있는 다음 명령문은 서버 파일에 내용이 있는 비트맵(BMP) 이미지를 갱신합니다. 갱신된 이미지 형식은 변환되지 않을 것입니다.

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,
                '/employee/newimg/ajones.bmp',
                'BMP',                          /*image format*/
                '')
   WHERE NAME='Anita Jones';
```

널(Null) 값이나 빈 문자열을 형식으로, 또는 Image Extender의 경우에 한해 문자열 ASIS를 지정할 수도 있습니다. 그런 다음 Extender는 소스를 검색하여 형식을 결정할 것입니다.

인식 가능한 형식에 대한 널(NULL) 값이나 ASIS 사용: 형식이 Extender에 인식 가능한 경우에만, 즉, 97 페이지의 표5에 Extender용으로 나열된 형식의 하나인 경우에 널(NULL)값이나 빈 문자열 또는 ASIS를 지정하십시오. 그렇지 않다면 Extender는 오브젝트를 갱신할 수 없습니다.

형식 변환을 사용하여 갱신시 형식과 변환 옵션

형식 변환을 사용하여 이미지 갱신시, 소스와 목표 이미지의 형식 모두를 지정하십시오. 97 페이지의 표5에는 허용되는 형식 변환을 나열합니다.

또한, 갱신된 이미지를 적용하려는 추가 변경사항(회전이나 압축 같은)을 식별하는 변환 옵션을 지정할 수 있습니다. 매개변수와 관련 값으로 각각의 변환 옵션을 지

정합니다. 매개변수와 허용값은 99 페이지의 표6에 나열됩니다. 복수의 매개변수/값 묶음을 지정하여 갱신된 이미지에 다양한 변경사항을 요청할 수 있습니다.

다음의 예시에서, 서버 파일에 내용이 있는 이미지가 갱신됩니다. 갱신 소스는 비트맵(BMP) 형식입니다. 갱신시 형식은 BMP에서 GIF로 변환됩니다.

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
                PICTURE,
                '/employee/newimg/ajones.bmp',
                'BMP',                                  /*source format*/
                'GIF',                                  /*target format*/
                '')
   WHERE NAME='Anita Jones';
```

다음의 예시에서, 동일 이미지가 갱신시 GIF 형식으로 변환됩니다. 또한 갱신시 이미지는 시계 방향으로 90도 회전합니다.

```
EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=CONTENT(
               PICTURE,
               '/employee/newimg/ajones.bmp',
               'BMP',                                   /*source format*/
               'GIF',                                   /*target format*/
               '-r 1',                                  /* conversion options */
               '')
   WHERE NAME='Anita Jones';
```

사용자 제공 속성으로 오브젝트 갱신

사용자 제공 속성으로 저장된 이미지, 오디오 또는 비디오 오브젝트 갱신시 내용을 갱신하는 속성을 지정해야 합니다. 속성 구조에 속성값을 지정하십시오. 속성 구조가 UDF에서 LONG VARCHAR FOR BIT DATA 변수의 데이터 필드에 저장되어야 합니다.

서버의 UDF 코드는 항상 『big endian 형식』의 데이터를 예상합니다. Big endian 형식은 대부분의 UNIX 플랫폼에서 사용하는 형식입니다. 『little endian 형식』으로 오브젝트를 저장중인 경우, 서버의 UDF 코드가 제대로 처리할 수 있도록 사용자 제공 속성 데이터를 준비해야 합니다. Little endian 형식은 Intel 및 기타 마이크로프로세서 플랫폼에서 일반적으로 사용하는 형식입니다. (little endian 형식으로 오브젝트를 저장하지 않는 경우에도, 사용자 제공 속성 데이터를 준비하는 것이 좋습니다.) 이미지 오브젝트의 속성을 준비하려면 DBiPrepareAttrs API를 사

용하십시오. 오디오 오브젝트의 속성을 준비하려면 DBaPrepareAttrs API를 사용하십시오. 비디오 오브젝트의 속성을 준비하려면 DBvPrepareAttrs API를 사용하십시오.

예를 들어, C 응용프로그램에 있는 다음 명령문은 서버 파일에 내용이 있는 이미지를 갱신합니다. 이미지는 사용자 정의 형식으로, 640 픽셀 높이와 480 픽셀 폭을 갖습니다. 이미지가 갱신되기 전에 속성이 준비됨을 유의하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
struct {
       short len;
       char data[400];
       }hvImgattrs;
EXEC SQL END DECLARE SECTION;

DB2IMAGEATTRS    *pimgattr;

hvStorageType=MMDB_STORAGE_TYPE_INTERNAL;

pimgattr = (DB2IMAGEATTRS *) hvImgattrs.data;
strcpy(pimgattr→Format,"FormatI");
pimgattr→width=640;
pimgattr→height=480;
hvImgattrs.len=sizeof(DB2IMAGEATTRS);

DBiPrepareAttrs(pimgattr);

EXEC SQL UPDATE EMPLOYEE
   SET PICTURE=REPLACE(
                PICTURE,
                '/employee/newimg/ajones.bmp',
                :hvStorageType,
                'Anita''s new picture',
                :ImgAttrs,                      /*user-supplied attributes*/
                CAST(NULL as LONG VARCHAR))
   WHERE NAME='Anita Jones';
```

썸네일 갱신(이미지와 비디오 전용)

이미지 또는 비디오 오브젝트용으로 저장된 썸네일을 갱신하려면, Thumbnail UDF를 사용하십시오(저장된 이미지나 비디오에 연관된 것이 없다면, 썸네일을 추가하십시오). Thumbnail UDF의 사용시, 갱신 중인 썸네일의 오브젝트 핸들과 갱신된 (또는 새로운) 썸네일의 내용을 지정하십시오.

프로그램에서 썸네일을 생성하십시오. Extender는 썸네일을 생성하는 API를 제공하지 않습니다. 사용자는 갱신하는 썸네일의 크기와 형식을 제어합니다. 프로그램에 썸네일용 구조를 만들어 UDF에 있는 썸네일의 구조를 지정하십시오.

예를 들어, C 응용프로그램에 있는 다음 명령문은 저장된 비디오 클립과 연관된 썸네일을 갱신합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
     struct {
       short len;
       char data[10000];
     }hvThumbnail;
EXEC SQL END DECLARE SECTION;

/*Create thumbnail and store in hvThumbnail*/

EXEC SQL UPDATE employee
   SET picture=Thumbnail(
                picture,
                :hvThumbnail)
   WHERE name='Anita Jones';
```

사용자 제공 속성으로 이미지나 비디오 오브젝트 갱신시, 썸네일도 갱신할 수 있습니다. 사용자 제공 속성으로 이미지나 비디오를 갱신한다면, 입력으로 썸네일을 지정해야 합니다. 오브젝트 갱신시 썸네일을 갱신하고 싶지 않다면, 썸네일 스펙에 널(NULL) 값이나 빈 문자열을 지정하십시오.

C 응용프로그램에 있는 다음 명령문은 사용자 제공 속성으로 비디오 클립을 갱신하고 비디오와 연관된 썸네일을 갱신합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
   struct {
       short len;
       char data[400];
   }hvVidattrs;
   struct {
       short len;
       char data[10000];
   }hvThumbnail;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

MMDBVideoAttrs          *pvideoAttr;
pvideoAttr=(MMDBVideoAttrs *)hvVidattrs.data;
strcpy(pvideoAttr→cformat,"Formatv");
hvVidattrs.len=sizeof(MMDBVideoAttrs);
```

```
/* Update video content and thumbnail */

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=REPLACE(
                VIDEO,
                '/employee/newvid/ajones.mpg',
                :hvStorageType,
                'Anita''s new video',
                :VidAttrs,
                :hvThumbnail)              /*thumbnail*/
   WHERE NAME='Anita Jones';
```

주석 갱신

별도로 주석을 갱신하거나, 연관된 오브젝트 갱신시 주석을 갱신할 수 있습니다.

주석을 갱신하려면, Comment UDF를 사용하십시오. 오브젝트 핸들이 있는 테이블 컬럼과 함께, 갱신된 주석의 내용을 지정하십시오. 내용을 서버로 전송하려면 호스트 변수를 사용하십시오. 예를 들어, 다음의 명령문은 hvRemarks로 이름 지정된 호스트 변수를 선언하고, 그것을 사용하여 저장된 비디오 클립에 대한 기존 주석을 갱신합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  struct {
         short len;
         char data [40];
  }hvRemarks;
EXEC SQL END DECLARE SECTION;

/* Get the old comment */

EXEC SQL SELECT COMMENT(VIDEO)
     INTO :hvRemarks
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';

/* Append to old comment */

hvRemarks.data[Remarks.len]='\0';
hvRemarks.len=strlen(hvRemarks.data);
strcat (hvRemarks.data, "Updated video");
EXEC SQL UPDATE EMPLOYEE
     SET VIDEO=COMMENT(VIDEO, :hvRemarks)
     WHERE NAME = 'Anita Jones';
```

연관된 오브젝트 갱신시, 주석을 갱신하려면 Replace UDF를 사용하십시오. 예를 들어, 다음의 명령문은 관련된 주석과 함께 서버 파일에 저장된 비디오 클립을 갱신합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType=MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=REPLACE(
                VIDEO,
                '/employee/newvid/ajones.mpg',
                'MPEG1',
                :hvStorageType,
                'Anita''s new video')        /*updated comment*/
   WHERE NAME='Anita Jones';
```

제12장 이미지, 오디오 또는 비디오 오브젝트 표시 또는 재생

이 장에서는 DB2 Extender 응용프로그램 프로그래밍 인터페이스를 사용하여 데이터베이스에 저장된 이미지, 오디오 또는 비디오 오브젝트를 표시하거나 재생하는 방법을 설명합니다.

표시 또는 재생 API 사용

데이터베이스에 저장된 이미지나 비디오 프레임을 표시하려면 Extender API를 사용할 수 있습니다. 이미지나 비디오 프레임의 썸네일 크기 버전이나 전체 크기 이미지 버전을 표시할 수 있습니다. 데이터베이스에 저장된 이미지나 비디오 오브젝트를 재생하는 데도 Extender API를 사용할 수 있습니다.

오브젝트를 표시하거나 재생하려면 다음의 API를 사용하십시오.

API	용도
DBiBrowse	이미지 또는 비디오 프레임 표시
DBaPlay	오디오 클립 재생
DBvPlay	비디오 클립 재생 또는 비디오 프레임 표시

이러한 API 요청시, 다음을 지정해야 합니다.

- 표시 또는 재생 프로그램 이름
- 표시되거나 재생되는 오브젝트가 BLOB으로 데이터베이스 테이블에 저장되는지 또는 테이블에서 지시하는 파일에 있는지
- 소스 파일 이름 또는 데이터베이스 테이블에 저장된 핸들
- 응용프로그램에서 계속하기 전에 사용자가 표시나 재생 프로그램을 닫기를 기다리게 할 것인지

표시 또는 재생 프로그램 식별

사용하려는 이미지 브라우저, 오디오 플레이어 또는 비디오 플레이어 이름을 지정하십시오. 이름 뒤에 %s를 입력하십시오. Extender는 오브젝트 내용을 보유하는

파일로 %s를 대체할 것입니다. 예를 들어, C 응용프로그램에 있는 다음 명령문은 이미지를 표시하는 OS/2 이미지 브라우저(ib)를 시작합니다.

```
rc = DBiBrowse(
     "ib %s",                    /* image display program */
     MMDB_PLAY_FILE,
     "/employee/images/ajones.bmp",
     MMDB_PLAY_NO_WAIT
     );
```

특정 표시나 재생 프로그램을 지정하는 대신, 널(Null) 값을 지정해도 됩니다. 이런 경우, Extender는 DB2IMAGEBROWSER, DB2AUDIOPLAYER 또는 DB2VIDEOPLAYER 환경 변수에서 이름 지정된 기본 이미지 브라우저, 오디오 플레이어 또는 비디오 플레이어를 시작합니다. DB2 Extender의 환경 변수 사용 방법에 대한 자세한 정보는 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』을 참조하십시오.

예를 들어, C 응용프로그램에 있는 다음 명령문은 DB2AUDIOPLAYER 환경 변수에서 식별되는 기본 오디오 플레이어를 시작합니다.

```
rc = DBaPlay(
     NULL,                                /* use default audio player */
     MMDB_PLAY_FILE,
     "/employee/sounds/ajones.wav",
     MMDB_PLAY_NO_WAIT
     );
```

환경 변수는 프로그램을 이름 지정해야 합니다. 기본 표시 또는 재생 프로그램을 널(Null) 값을 지정하여 요청한다면, 적절한 환경 변수가 표시 또는 재생 프로그램을 지정하도록 하십시오. 프로그램이 지정되지 않는다면, API는 오류 코드를 리턴할 것입니다.

BLOB이나 파일 내용 지정

BLOB으로 데이터베이스 테이블에 저장된 오브젝트나 파일에 내용이 저장된 오브젝트(또한 데이터베이스 테이블에서 지시된 오브젝트)를 표시하거나 재생할 수 있습니다. 오브젝트가 BLOB으로 저장된다면, MMDB_PLAY_HANDLE을 지정하십시오. 오브젝트 내용이 파일에 저장된다면, MMDB_PLAY_FILE을 지정하십시오. MMDB_PLAY_HANDLE 및 MMDB_PLAY_FILE는 Extender가 정의하는 상수입니다.

예를 들어, C 응용프로그램에 있는 다음 명령문은 서버 파일에 있는 비디오의 내용을 갱신합니다.

```
rc = DBvPlay(
     "explore %s",
     MMDB_PLAY_FILE,                    /* content in file */
     "/employee/videos/ajones.mpg",
     MMDB_PLAY_NO_WAIT
     );
```

표시 및 재생 프로그램은 일반적으로 파일로부터 입력을 허용합니다. MMDB_PLAY_FILE을 지정하는 경우, Extender는 환경 변수의 값을 사용하여 파일의 상대 파일 이름 및 경로를 해결합니다. 그런 다음 Extender는 열람 프로그램을 시작하고 여기에 파일 이름을 전달합니다. MMDB_PLAY_HANDLE을 지정하는 경우, Extender는 핸들에서 파일 이름을 발췌합니다(파일 이름이 널(NULL)이 아닌 경우). 핸들의 파일 이름이 널(NULL)인 경우, 오브젝트가 BLOB으로 저장됩니다. Extender는 클라이언트에 임시 파일을 작성하여 데이터베이스 테이블에서 오브젝트의 내용을 클라이언트 파일에 복사합니다. 그런 다음 Extender가 프로그램을 시작하여 여기에 내용을 보유하는 파일 이름(또는 임시 파일)을 전달합니다.

예를 들어, C 응용프로그램에 있는 다음 명령문은 BLOB으로 저장된 이미지 핸들을 확보하여 이미지를 표시하는 데 이 핸들을 사용합니다.

```
EXEC SQL BEGIN DECLARE SECTION;
char hvImg_hdl[251];
EXEC SQL END DECLARE SECTION;

rc = DBiBrowse(
     "ib %s",
     MMDB_PLAY_HANDLE,                  /* content is BLOB */
     hvImg_hdl,
     MMDB_PLAY_NO_WAIT
     );
```

내용은 액세스 가능해야 합니다. 표시 프로그램이나 재생 프로그램이 오브젝트 내용에 액세스할 수 있는지 확인하십시오. 내용은 서버 파일에 있는데 프로그램이 클라이언트에 있는 내용을 요청한다면, 클라이언트 파일로 파일을 복사하거나 Content UDF를 사용하십시오. 내용이 BLOB으로 저장된다면, Extender는 자동으로 이를 클라이언트로 검색할 것입니다.

대기 표시기 지정

응용프로그램이 계속되기 전에(즉, DBiBrowse, DBaPlay 또는 DBvPlay API가 코드를 리턴하기 전에) 사용자가 표시나 재생 프로그램을 종료하기를 대기할 것인지 여부를 지정할 수 있습니다. 응용프로그램이 대기하기를 원한다면, MMDB_PLAY_WAIT를 지정하십시오. 응용프로그램이 대기하기를 원하지 않는다면, MMDB_PLAY_NO_WAIT를 지정하십시오. MMDB_PLAY_WAIT 및 MMDB_PLAY_NO_WAIT는 Extender가 정의하는 상수입니다.

MMDB_PLAY_WAIT를 지정한다면, 표시나 재생 프로그램은 응용프로그램과 같은 쓰레드나 프로세스에서 수행할 것입니다. MMDB_PLAY_NO_WAIT를 지정한다면, 표시나 재생 프로그램은 응용프로그램에 독립적인 고유의 쓰레드나 프로세스에서 수행할 것입니다.

예를 들어, 다음 명령문의 결과로, 응용프로그램은 응용프로그램이 계속되기 전에 사용자가 이미지 브라우저를 닫을 때까지 대기할 것입니다.

```
rc = DBiBrowse (
     "explore %s",
     MMDB_PLAY_FILE,
     "/employee/images/ajones.bmp",
     MMDB_PLAY_WAIT                    /* wait for browser to close */
     );
```

DBxPlay 및 MMDB_PLAY_NO_WAIT를 지정할 경우에는 주의하십시오. DBaPlay 또는 DBvPlay를 발행하면, Extender가 다음 항목 중에서 참인 것이 있는 경우 임시 파일을 작성합니다.

- 오브젝트가 BLOB으로 저장된 경우
- 환경 변수 값을 사용하여 상대 파일 이름을 해결할 수 없는 경우
- 클라이언트 머신에서 파일을 액세스할 수 없는 경우

임시 파일은 TMP 환경 변수가 지정한 디렉토리에서 작성됩니다. MMDB_PLAY_WAIT를 지정한다면, Extender는 오브젝트 재생 후, 임시 파일을 삭제합니다. 단, MMDB_PLAY_NO_WAIT를 지정하면 임시 파일이 삭제되지 않습니다. 사용자는 임시 파일을 직접 삭제해야 합니다.

썸네일 크기 이미지 또는 비디오 프레임 표시

썸네일은 저장 이미지 또는 비디오 프레임의 소형 버전입니다. 데이터베이스에 이미지 저장시, Image Extender는 속성 테이블에 이미지의 썸네일을 저장합니다. 데이터베이스에 비디오를 저장할 때, Video Extender는 비디오 오브젝트를 상징하는 일반 썸네일을 속성 테이블에 저장합니다.

기본적으로, Image Extender가 자동으로 작성하는 썸네일의 *크기*는 약 112 x 84 픽셀입니다. Video Extender가 삽입하는 일반 비디오 썸네일의 *크기*는 108 x78 픽셀입니다. 썸네일과 일반 비디오 썸네일은 둘다 GIF 형식으로 저장됩니다. 이미지나 비디오 프레임의 데이터 밀도에 따라, 이것은 데이터의 약 4.5KB에서 5KB까지에 대응합니다. 사용자 제공 속성으로 이미지 또는 비디오를 저장하거나 갱신한다면, 선택한 *크기*와 형식의 썸네일을 지정할 수 있습니다.

데이터베이스에서 썸네일을 검색하려면 SQL SELECT문에 Thumbnail UDF를 사용하십시오. 썸네일을 파일에 전송하려면 파일 참조 변수를 사용하십시오. UDF 지정시, 이미지나 비디오 핸들이 있는 데이터베이스 테이블에 컬럼 이름을 지정해야 합니다. 그런 다음, 이미지나 비디오 프레임 썸네일을 표시하려면, DBiBrowse API를 사용하십시오.

예를 들어, 다음의 명령문은 썸네일 이미지를 검색하여 그것을 표시합니다.

```
long rc, outCount;
char Thumbnail_filename[254];
FILE *file_handle;

EXEC SQL BEGIN DECLARE SECTION;
  struct {
         short len
         char data[10000];
         }Thumbnail_buffer;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT THUMBNAIL(PICTURE)
     INTO :Thumbnail_buffer
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';

strcpy (Thumbnail_filename,"/tmp/ajones.tmb");
file_handle=fopen(Thumbnail_filename,"wb+");
outCount=fwrite(Thumbnail_buffer.data, 1, Thumbnail_buffer.len, file_handle);
fclose(file_handle);
rc = DBiBrowse (
               NULL,                          /* use the default display program */
               MMDB_PLAY_FILE,                /* thumbnail image in file */
               Thumbnail_filename,            /* thumbnail image content */
               MMDB_PLAY_WAIT);               /* wait for user to finish */
```

전체 크기 이미지 또는 비디오 프레임 표시

데이터베이스 테이블에 저장된 이미지를 표시하려면 DBiBrowse API를 사용하십시오. 이 API의 사용에 대한 세부사항은 139 페이지의 『표시 또는 재생 API 사용』을 참조하십시오.

전체 크기 비디오 프레임을 확보하려면, DBvGetNextFrame API 또는 DBvSeekFrame API를 사용하십시오. 프레임은 버퍼에 YUV 형식으로 저장되고 DBvFrameDatato24BitRGB API를 사용하여 RGB 형식으로 변환될 수 있습니다. 헤더를 변환된 프레임에 추가하고(예를 들어, BMP 파일 유형 헤더를 추가) 헤더와 프레임 데이터를 파일에 쓰십시오. 그런 다음, 파일 내용을 표시하려면 DBiBrowse API를 사용하십시오. DBvGetNextFrame, DBvSeekNextFrame 및 DBvFrameDatato24BitRGB API 사용이나 비디오 프레임 표시에 대한 세부사항은 189 페이지의 『제14장 비디오 장면 변경 검출』을 참조하십시오.

오디오 또는 비디오 재생

데이터베이스 테이블에 저장된 오디오를 재생하려면 DBaPlay API를 사용하십시오. 데이터베이스 테이블에 저장된 비디오를 재생하려면 DBvPlay API를 사용하십시오. 이러한 API 사용에 대한 세부사항은 139 페이지의 『표시 또는 재생 API 사용』을 참조하십시오.

제13장 내용으로 이미지 조회

그림25는 사용자가 시각적 예시를 탐색 범주로 사용하여 데이터베이스에서 이미지 즉, 주색이나 텍스처 패턴을 표시하는 이미지를 탐색할 수 있게 해주는 응용프로그램을 보여줍니다. 이러한 응용프로그램으로, 사용자는 탐색에 입력으로 이미지를 제공할 수 있습니다. 그런 다음 응용프로그램은 소스 이미지의 색 또는 텍스처를 저장된 이미지의 색 또는 텍스처와 대응시키고, 색 또는 텍스처가 입력에 가장 근접하게 일치하는 이미지를 리턴합니다.

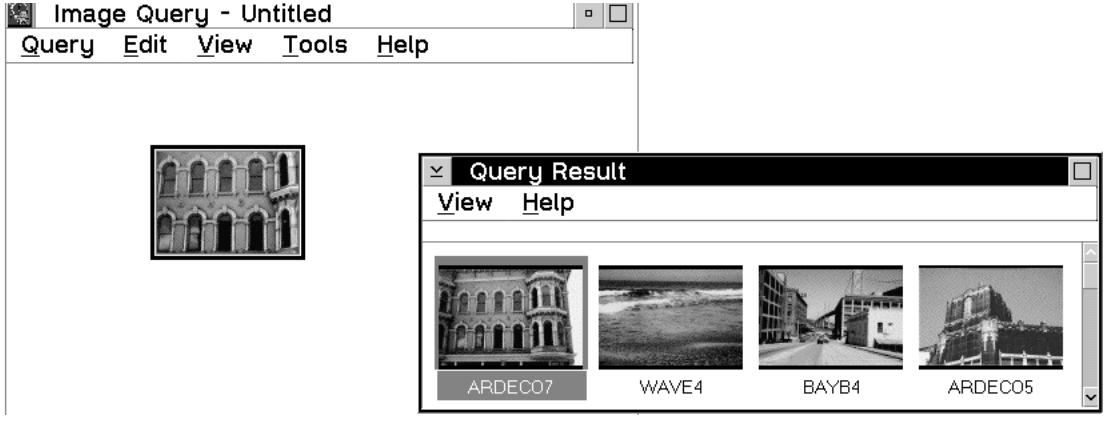


그림 25. 이미지 내용별 조회. 시각적 예시의 색이나 텍스처는 데이터베이스 테이블에 저장된 이미지를 탐색하는 데 이용됩니다.

시각적 특성으로 이미지를 조회하는 기능은 **이미지 내용별 조회(QBIC)**라 불립니다[5]. 이 장에서는 지금 설명한 대로 응용프로그램을 빌드하기 위해 Image Extender에 제공되는 API와 UDF를 사용하는 방법을 설명합니다. 또한 Image Extender에 제공되는 명령 및 API를 사용하여 QBIC 관리 타스크를 수행하는 방법을 설명합니다.

5. Image Extender는 캘리포니아 대학, 버클리 대학 및 기부자에 의해 개발된 소프트웨어를 포함합니다.

내용별 이미지 조회 방법

이미지를 내용별로 조회하려면:

1. 이미지용 QBIC 카탈로그를 작성하십시오.
2. 이미지를 카탈로그화하십시오. 이것은 이미지에 대한 항목을 카탈로그에 추가하고 이미지 특성에 대한 값을 저장하는 것을 의미합니다.

   QBIC 카탈로그와 이미지 특성의 설명에 대해서는 23 페이지의 『QBIC 카탈로그』를 참조하십시오.
3. 조회를 빌드하십시오. 조회는 탐색 기준, 값 및 가중치(즉, 각각의 특성에 위치되는 강조사항)로 사용되는 특성을 식별합니다. 조회 문자열을 호출한 문자열에 이러한 조회 속성을 지정할 수 있습니다. 또다른 방법으로, 조회 오브젝트를 작성하고 이러한 속성을 조회 오브젝트와 관련시킬 수 있습니다. 그러면, 그 조회 문자열을 저장했다가 다시 사용할 수 있습니다.
4. 조회를 수행하십시오. 조회 수행시, 조회 문자열을 입력으로 지정하거나 조회에 대한 조회 오브젝트를 식별합니다. 두 가지 경우 모두에, 탐색되는 이미지를 식별하기도 합니다. 두 가지 경우 모두에, DB2 명령행이나 프로그램 내에서 조회를 제출할 수 있습니다.

응답으로, Image Extender는 조회용 특성값을 계산합니다. 그 값을 목표 이미지에 대해 QBIC 카탈로그에 저장된 특성값과 비교합니다. 그런 다음 Image Extender는 각 목표 이미지의 특성값과 소스와의 유사성을 표시하는 스코어를 계산합니다.

Image Extender가 소스와 가장 유사한 특성값을 갖는 이미지를 리턴하리라는 것을 알 수 있습니다. Image Extender가 하나 이상의 이미지 스코어를 리턴할 것이라는 것도 알 수 있습니다.

QBIC 카탈로그 관리

이미지가 내용별로 조회되기 전에, QBIC 카탈로그에 카탈로그화되어야 합니다. QBIC 카탈로그는 이미지의 시각적 특성에 관한 데이터를 보유합니다.

내용별로 조회할 수 있기를 바라는 사용자 테이블의 각각의 이미지 컬럼에 대해 QBIC 카탈로그를 작성합니다. 사용자 테이블의 각각의 이미지 컬럼에 대해 오직 하나의 QBIC만이 있을 수 있고, 여러 컬럼이 동일 QBIC 카탈로그를 공유할 수 없습니다.

QBIC 카탈로그 작성시, Image Extender가 데이터를 저장하기를 원하는 특성을 식별합니다. Image Extender가 자동으로 이미지를 카탈로그화하기를 원하는지도 표시합니다. 자동 카탈로그화는 사용자 테이블에 이미지가 저장될 때 Image Extender가 자동으로 카탈로그에 그 이미지의 항목을 작성함을 의미합니다. 이미지가 자동으로 카탈로그화되지 않는다면, 수동으로 그것을 카탈로그화해야 합니다. 이는 Image Extender로 하여금 카탈로그에 그 이미지에 대한 항목을 작성하도록 명시적으로 지시하는 것을 의미합니다.

QBIC 카탈로그 작성 후, 다음을 수행할 수 있습니다.

- 그에 대한 후속 조치용 카탈로그 열기
- 자동 카탈로그화 설정을 수동 카탈로그화로나 수동에서 자동으로 변경
- 카탈로그에 특성 추가. 이렇게 하면 Image Extender가 데이터를 저장할 특성을 식별합니다.
- 카탈로그에서 특성 제거
- 사용자 테이블 이름과 카탈로그와 연관된 컬럼 이름 또는 데이터가 카탈로그에 저장된 특성과 같은 카탈로그에 대한 정보 검색
- 수동으로 카탈로그에 이미지 카탈로그화
- 이미지 카탈로그화 해제(즉 카탈로그에서 이미지에 대한 항목 제거)
- 이미지 다시 카탈로그화
- 카탈로그 재분산(즉, 파티션된 데이터베이스 시스템에 노드를 추가하거나 삭제할 때)
- 카탈로그 닫기
- 카탈로그 삭제

Image Extender에서 제공하는 API를 사용하여 QBIC 카탈로그 작성을 포함한, 이들 타스크를 수행할 수 있습니다. 또한 db2ext 명령행 처리기를 사용하여 이들 중 많은 타스크를 수행할 수 있습니다.

QBIC 카탈로그 작성

QBIC 카탈로그를 작성하려면, QbCreateCatalog API나 CREATE QBIC CATALOG 명령을 사용하십시오. 카탈로그를 작성하려면, 해당 이미지를 카탈로그화할 사용자 테이블의 소유자여야 합니다. 또한, 카탈로그가 있을 데이터베이스에 대해 CREATE TABLE 권한을 가져야 합니다. 사용자 테이블과 이미지 컬럼은 그러한 컬럼의 이미지용 QBIC 카탈로그를 작성하기 전에 Image Extender용으로 사용 가능해야 합니다.

QBIC 카탈로그 작성시 다음을 수행합니다.

- 카탈로그화되는 이미지가 있는 테이블과 컬럼을 이름 지정합니다.
- 이미지가 자동으로 카탈로그화 되는지를 표시합니다. 자동 카탈로그화는 이미지를 사용자 테이블에 저장할 때 Image Extender가 이미지를 카탈로그하는 것을 의미합니다. Extender는 이미지가 카탈로그화를 대기중인지 알기 위해 주기적으로 점검합니다. 환경 변수 DB2CATALOGDELAY의 값을 설정하여 주기를 초 단위로 지정합니다. 값은 1초에서 최대 값 사이 범위로 설정할 수 있습니다. 기본값은 60초입니다.

  수동 카탈로그화는 사용자가 Image Extender에 명시적으로 이미지 카탈로그화를 요청하는 것을 의미합니다. (이미지의 수동 카탈로그화 정보는 155 페이지의 『수동으로 이미지 카탈로그화』를 참조하십시오.)

사용자 테이블과 컬럼의 사용이 가능해야 합니다. 사용자 테이블과 컬럼은 그 컬럼에 이미지용 QBIC 카탈로그를 작성하기 전에 Image Extender에 대해 사용이 가능해야 합니다(Image Extender에 대해 사용자 테이블과 컬럼을 사용 가능화하는 데 대한 사용 정보는 61 페이지의 『제6장 Extender 데이터용 데이터 오브젝트 준비』를 참조하십시오).

API 사용: QbCreateCatalog API 사용시, 자동 카탈로그화 값을 지정하여 자동 또는 수동 카탈로그화를 표시합니다. 값 1은 자동 카탈로그화를, 값 0은 수동 카탈로그화를 나타냅니다.

예를 들어, 다음의 명령문은 직원 테이블의 사진 컬럼에 이미지용 QBIC 카탈로그를 작성합니다. 이미지는 직원 테이블에 저장될 때 자동으로 카탈로그화될 것입니다.

```
SQLINTEGER autoCatalog=1;                          /* automatic cataloging */

rc=QbCreateCatalog(
                  "employee",                      /* user table */
                  "picture",                       /* image column */
                  autoCatalog);                    /* auto catalog setting */
```

명령행 사용: CREATE QBIC CATALOG 명령 발행시, ON을 지정하여 자동 카탈로그화를 나타냅니다. OFF를 지정하면 수동 카탈로그화를 나타냅니다. OFF가 기본값입니다.

예를 들어 다음의 명령은 API 예시에서와 동일한 QBIC 카탈로그를 작성합니다.

```
CREATE QBIC CATALOG employee picture on
```

QBIC 카탈로그를 백업하십시오. Image Extender는 파일에 QBIC 카탈로그를 저장합니다. 카탈로그를 복구해야 하는 경우, 이 파일을 주기적으로 백업해야 합니다. AIX, HP-UX 또는 Sun Solaris 서버에서, 파일은 /home/*instance_owner*/dmb/qbic 디렉토리에 있으며, 여기서 *instance_owner*는 인스턴스 소유자의 사용자 ID입니다. OS/2나 Windows 서버에서 파일은 *destination*\instance*instance_name*\qbic 디렉토리에 위치하고, 여기서 *destination*은 Image Extender가 설치된 디렉토리이며 *instance_nam*는 Extender 인스턴스 이름입니다.

QBIC 카탈로그 열기

카탈로그를 변경하는 후속 조치를 수행하려면 QBIC 카탈로그를 열어야 합니다. 예를 들어, 특성을 카탈로그에 추가하려면 QBIC 카탈로그를 열어야 합니다.

QBIC 카탈로그를 열려면, QbOpenCatalog API 호출이나 OPEN QBIC CATALOG 명령을 사용하십시오. QBIC 카탈로그를 열 때, 다음을 수행합니다.

- 카탈로그용 사용자 테이블과 이미지 컬럼을 이름 지정합니다.
- 카탈로그를 열고자 하는 모드를 지정합니다(이것은 OPEN QBIC CATALOG 명령 사용시 암시됩니다). 내용으로 이미지 탐색과 같이 카탈로그를 읽는 조작용으로 카탈로그를 열 수 있습니다. 또는 특성 추가와 같이 그것을 갱신하는 조작용 카탈로그를 열 수 있습니다. 읽기 조작용으로 카탈로그를 열려면 사용자 테이블에 대한 SELECT 권한을 가져야 합니다. 갱신 조작용으로 카탈로그를 열려면 사용자 테이블에 대한 UPDATE 권한을 가져야 합니다.

카탈로그가 이미 열려 있다면? 카탈로그가 다른 세션에서 갱신용으로 열려 있다면, 갱신 조작용 카탈로그를 열 수 없습니다. QBIC 카탈로그를 열 때, Image Extender는 현재 세션에서 이미 열려 있는 모든 QBIC 카탈로그를 닫습니다.

API 사용: QbCreateCatalog API 사용시, 카탈로그를 열려는 모드를 명시적으로 지정합니다. 다음을 지정하십시오.

- 그것에서 읽는 조작용으로 카탈로그를 여는 API 매개변수 qbiRead.
- 그것을 갱신하는 조작용으로 카탈로그를 여는 API 매개변수 qbiUpdate.

QbiRead 및 QbiUpdate는 QBIC, dmbqbapi.h용 Include(헤더) 파일에 정의된 상수입니다.

또한, 카탈로그 핸들을 지정해야 합니다. 카탈로그 핸들은 QbCatalogHandle의 QBIC 지정 데이터 유형을 갖습니다. 이 데이터 유형은 dmbqbapi.h에서도 정의됩니다. Image Extender는 카탈로그 핸들값을 API에서 출력으로 리턴합니다.

예를 들어, 다음의 API 호출은 카탈로그에서 읽는 조작용으로 QBIC 카탈로그를 엽니다.

```
SQLINTEGER mode;
QbCatalogHandle  *CatHdl;

mode=qbiRead;                                   /* open catalog for */
                                                /* read operations */
rc=QbOpenCatalog(
                 "employee",                    /* user table */
                 "picture",                     /* image column */
                 mode,                          /* open catalog mode */
                 &CatHdl);                      /* catalog handle */
```

명령행 사용: OPEN QBIC CATALOG 명령 수행시, Image Extender는 갱신 조작용으로 카탈로그를 열려고 합니다. 카탈로그가 현재 다른 세션에 갱신용으로 열려 있다면, Image Extender는 읽기 조작용으로 카탈로그를 엽니다.

예를 들어, 다음 명령은 QBIC 카탈로그를 엽니다. Image Extender는 갱신 조작용으로 그것을 열려고 합니다.

```
OPEN QBIC CATALOG employee picture
```

QBIC 관련 활동 종료시 카탈로그를 닫으십시오. QBIC 카탈로그를 열 때, Image Extender는 메모리와 같이 그것에 자원을 할당합니다. QBIC 관련 활동 종료시 카탈로그를 닫으십시오. 이것은 할당된 자원을 사용 가능하게 합니다.

자동 카탈로그화 설정 변경

자동 카탈로그화에서 수동 카탈로그화 또는 수동에서 자동으로 변경하려면, QbSetAutoCatalog API나 SET QBIC AUTOCATALOG 명령을 사용하십시오. QBIC 카탈로그는 카탈로그화 설정을 변경하기 전에 갱신용으로 열려 있어야 합니다.

변경은 소급되지 않습니다. 자동 카탈로그화 설정의 변경시, 이것은 변경 후 사용자 테이블 컬럼에 추가된 이미지에만 적용됩니다. 사용자 테이블 컬럼에 이미 저장된 이미지는 영향을 받지 않습니다. 예를 들어, 수동 카탈로그화에서 자동 카탈로그화로 설정 변경을 한다면, 변경 후 사용자 테이블 컬럼에 추가된 이미지만이 자동으로 카탈로그화될 것입니다. 테이블 컬럼에 이미지를 미리 카탈로그화하려면, 수동으로 카탈로그화해야 합니다. (이미지의 수동 카탈로그화 정보는 155 페이지의 『수동으로 이미지 카탈로그화』를 참조하십시오.)

API 사용: QbSetAutoCatalog API 사용시, QBIC 카탈로그 핸들을 지정하십시오(핸들은 QbOpenCatalog API로 카탈로그를 열 때 리턴됩니다). 또한 자동 카탈로그화용으로는 자동 카탈로그화 값 1을, 수동 카탈로그화용으로는 값 0을 지정하십시오.

다음의 예시에서, 수동 카탈로그화는 직원 테이블의 사진 컬럼에 있는 이미지와 연관된 QBIC 카탈로그용으로 지정됩니다. QBIC 카탈로그가 그것을 갱신하는 조작에 대해 처음 열림을 주목하십시오.

```
SQLINTEGER mode;
SQLINTEGER autoCatalog=0;                          /* manual cataloging */

QbCatalogHandle  *CatHdl;

mode=qbiUpdate;                                    /* open catalog for */
                                                   /* update */
/* Open a QBIC catalog */
rc=QbOpenCatalog(
                  "employee",                      /* user table */
                  "picture",                       /* image column */
                  mode,                            /* open catalog mode */
                  &CatHdl);                        /* catalog handle */
```

```
/* Change the auto catalog setting */
rc=QbSetAutoCatalog(
                  CatHdl,                          /* catalog handle */
                  autoCatalog);                    /* auto catalog flag */
```

명령행 사용: SET QBIC AUTOCATALOG 명령 발행시, ON을 지정하여 자동 카탈로그화를 나타냅니다. OFF를 지정하면 수동 카탈로그화를 나타냅니다. 명령은 현재 열려 있는 카탈로그에 대해 작동합니다.

예를 들어 다음의 명령은 현재 열린 QBIC 카탈로그에 대해 자동 카탈로그화 해제를 설정합니다.

```
SET QBIC AUTOCATALOG off
```

QBIC 카탈로그에 특성 추가

QBIC 카탈로그에 특성을 추가하려면, QbAddFeature API나 ADD QBIC FEATURE 명령을 사용하십시오. 이미지를 카탈로그화할 수 있기 전에 최소한 하나의 특성을 QBIC 카탈로그에 추가해야 합니다. QBIC 카탈로그는 특성 추가 전에 갱신용으로 열려 있어야 합니다.

카탈로그에 특성 추가시, 추가하려는 특성 이름을 지정하십시오(특성 이름은 표8에 나열되어 있습니다).

표 8. QBIC 특성 이름

특성 이름	설명
QbColorFeatureClass	평균 색상
QbColorHistogramFeatureClass	히스토그램 색상
QbDrawFeatureClass	위치 색상
QbTextureFeatureClass	텍스처

이미지를 다시 카탈로그화해야 할 수도 있습니다. QBIC 카탈로그에 특성을 추가하는 경우, Image Extender는 자동 카탈로그화가 설정된 경우에도 이미 카탈로그화된 이미지의 새로운 특성에 관한 데이터를 자동으로 저장하지 않습니다. 이미 카탈로그화된 이미지용 새 특성에 대한 데이터를 포함하려면, 이미지를 다시 카탈로그화해야 합니다(158 페이지의 『이미지 재카탈로그화』를 참조하십시오).

API 사용: QbAddFeature API 사용시, 특성 이름 외에 QBIC 카탈로그 핸들을 지정해야 합니다. 특성 이름 길이에 대한 상수 qbiMaxFeatureName의 사용을 주의하십시오. 상수는 QBIC에 대한 Include(헤더) 파일에 dmbqbapi.h (값 50)로 정의됩니다.

다음의 예시에서, QbAddFeature API는 히스토그램 색상 특성을 QBIC 카탈로그에 추가하는 데 사용됩니다.

```
char  featureName[qbiMaxFeatureName];

QbCatalogHandle   CatHdl;

strcpy(featureName,"QbColorHistogramFeatureClass");

rc=QbAddFeature(
                CatHdl,                              /* catalog handle */
                featureName);                        /* feature name */
```

명령행 사용: ADD QBIC FEATURE 명령은 현재 열려 있는 카탈로그에 대해 작동합니다. 다음의 예시에서, 명령은 현재 열려 있는 QBIC 카탈로그에 위치 색상 특성을 추가하는 데 사용됩니다.

```
ADD QBIC FEATURE QbDrawFeatureClass
```

QBIC 카탈로그에서 특성 제거

QBIC 카탈로그에서 특성을 제거하려면 QbRemoveFeature API나 REMOVE QBIC FEATURE 명령을 사용하십시오. Image Extender는 특성용 카탈로그 테이블를 삭제합니다. 그 결과, 특성용 데이터는 이미지 카탈로그화시 저장되지 않습니다. QBIC 카탈로그는 특성 제거 전에 갱신용으로 열려 있어야 합니다.

카탈로그에서 특성 제거시 제거하려는 특성 이름을 지정하십시오.

API 사용: QbRemoveFeature API 사용시, 특성 이름 외에 QBIC 카탈로그 핸들을 지정해야 합니다.

다음의 예시에서, QbRemoveFeature API는 히스토그램 색상 특성을 QBIC 카탈로그에서 제거하는 데 사용됩니다.

```
char  featureName[qbiMaxFeatureName];

QbCatalogHandle   CatHdl;
```

```
strcpy(featureName,"QbColorHistogramFeatureClass");

rc=QbRemoveFeature(
                CatHdl,                                 /* catalog handle */
                featureName);                           /* feature name */
```

명령행 사용: REMOVE QBIC FEATURE 명령은 현재 열려 있는 카탈로그에 대해 작동합니다. 다음의 예시에서, 명령은 현재 열려 있는 QBIC 카탈로그에서 위치 색상 특성을 제거하는 데 사용됩니다.

```
REMOVE QBIC FEATURE QbDrawFeatureClass
```

QBIC 카탈로그에 대한 정보 검색

다음의 QBIC 카탈로그에 대한 정보를 검색할 수 있습니다.

- 카탈로그와 연관된 사용자 테이블과 이미지 컬럼 이름.
- 데이터가 카탈로그에 저장된 특성 수와 특성 이름.
- 사용자 테이블에 저장시 Image Extender가 자동으로 이미지를 카탈로그화하는지의 여부.

사용자 테이블과 컬럼 이름, 특성 수 및 자동 카탈로그화 설정을 검색하려면 QbGetCatalogInfo API를 사용하십시오. 특성 이름을 검색하려면 QbListFeatures API를 사용하십시오. 정보를 모두 검색하려면, GET QBIC CATALOG INFO 명령을 사용하십시오.

QBIC 카탈로그는 정보 검색 전에 열려 있어야 합니다.

API 사용: QbGetCatalogInfo API 사용시, QBIC 카탈로그 핸들을 지정해야 합니다. 또한 Image Extender가 카탈로그 정보를 리턴하는 구조를 가리켜야 합니다. 카탈로그 정보 구조는 다음과 같이 QBIC에 대한 Include(헤더) 파일에 dmbqapi.h로 정의됩니다.

```
typedef struct{
          char       tableName[qbiMaxTableName+1]    /* user table */
          char       columnName[qbiMaxColumnName+1]  /* image column */
          SQLINTEGER featureCount;                   /* number of features */
          SQLINTEGER autoCatalog;                    /* auto catalog flag */
} QbCatalogInfo;
```

QbListFeatures API 호출 발행시, 리턴된 특성 이름을 보유하는 버퍼를 할당해야 합니다. 공백 문자는 버퍼에 저장된 특성 이름을 분리합니다. 또한 카탈로그 핸들

과 리턴된 특성 이름에 대한 버퍼 *크기*를 지정해야 합니다. 필요한 버퍼의 *크기*를 예측하려면, QbGetCatalogInfo API에 의해 리턴된 특성 계수를 사용할 수 있으며, 계수를 가장 긴 특성 이름의 길이로 곱할 수 있습니다. 상수 qbiMaxFeatureName를 가장 긴 특성 이름으로 사용할 수 있습니다.

다음 예시의 API 호출은 QBIC 카탈로그에 대한 정보를 검색합니다. QbGetCatalogInfo API가 리턴한 특성 계수와 qbiMaxFeature 이름 상수를 사용하여 QbListFeatures API에 대한 버퍼 *크기*를 계산하는 방법에 유의하십시오.

```
long  bufSize;
long  count;
char  *featureNames;

QbCatalogHandle  CatHdl;
QbCatalogInfo    catInfo;

/* Get user table name, image column name, feature count, */
/* and auto catalog setting */

rc=QbGetCatalogInfo(
                CatHdl,                                 /* catalog handle */
                &catInfo);                              /* catalog info. structure */

/* List feature names */

bufSize=catInfo.featureCount*qbiMaxFeatureName;
featureNames=malloc(bufSize);

rc=QbListFeatures(
                CatHdl,                         /* catalog handle */
                bufSize                         /* size of buffer */
                count,                          /* feature count */
                featureNames);                  /* buffer for feature names */
```

명령행 사용: GET QBIC CATALOG 명령은 현재 열려 있는 카탈로그에 대해 작동합니다. 다음의 예시에서, 명령은 현재 열려 있는 QBIC 카탈로그에 대한 정보를 검색하는 데 사용됩니다.

```
GET QBIC CATALOG INFO
```

수동으로 이미지 카탈로그화

카탈로그 작성시, Image Extender가 사용자 테이블에 이미지를 저장할 때 이미지 카탈로그화를 원하는지 표시합니다. 이미지가 자동으로 카탈로그화되지 않는다면, 사용자 테이블에 저장된 후 수동으로 카탈로그화됩니다. 수동으로 단일 이미지나 이미지의 전체 컬럼을 카탈로그화할 수 있습니다.

수동으로 단일 이미지 카탈로그화

수동으로 단일 이미지를 카탈로그화하려면 QbCatalogImage API를 사용하십시오. 명령행에서 개별 이미지를 식별할 아무런 방법이 없기 때문에, 명령으로 이미지를 카탈로그화 할 수 없습니다. API 사용시, 카탈로그 핸들과 이미지 핸들을 지정하십시오(사용자 테이블에서 이미지 핸들을 검색할 수 있습니다). QBIC 카탈로그는 수동으로 이미지를 카탈로그화하기 전에 열려 있어야 합니다.

예를 들어, 다음의 명령문은 사용자 테이블에서 이미지 핸들을 검색한 후, 이미지를 카탈로그화합니다.

```
/* Retrieve the image handle */

EXEC SQL BEGIN DECLARE SECTION;
char Img_hdl[251];
EXEC SQL END DECLARE SECTION;

QbCatalogHandle   CatHdl;

EXEC SQL SELECT PICTURE INTO :Img_hdl
  FROM EMPLOYEE
  WHERE NAME='Anita Jones';

/* Catalog the image*/

rc=QbCatalogImage(
                CatHdl,                                 /* catalog handle */
                Img_hdl);                               /* image handle */
```

수동으로 이미지 컬럼 카탈로그화

수동으로 이미지 컬럼을 카탈로그화하려면, QbCatalogColumn API나 CATALOG QBIC COLUMN 명령을 사용하십시오. Image Extender는 컬럼이 마지막으로 카탈로그화된 후 새로 삽입되거나 갱신되거나 또는 삭제된 컬럼에 있는 이미지만을 카탈로그화하며, 카탈로그에 있는 모든 특성의 이미지를 카탈로그화합니다. QBIC 카탈로그는 이미지 컬럼을 수동으로 카탈로그화하기 전에, 갱신을 위해 열려 있어야 합니다.

API 사용: QbCatalogColumn API 사용시, 카탈로그 핸들을 지정하십시오. Image Extender는 지정된 카탈로그와 연관된 사용자 테이블 컬럼에 있는 이미지를 사용합니다.

예를 들어, 다음의 API 호출을 지정된 카탈로그와 연관된 사용자 테이블 컬럼에 있는 카탈로그화되지 않은 이미지를 카탈로그화합니다. 이미지는 카탈로그의 모든 특성용으로 카탈로그화됩니다.

```
QbCatalogHandle   CatHdl;

rc=QbCatalogColumn(
                CatHdl);                              /* catalog handle */
```

명령행 사용: 이미지 컬럼을 수동으로 카탈로그화하려면, CATALOG QBIC COLUMN 명령을 사용하십시오. 이미지를 다시 카탈로그화하기 위해 이 명령을 사용할 수도 있습니다(158 페이지의 『이미지 재카탈로그화』를 참조하십시오). 매개변수 FOR와 NEW를 지정하십시오(FOR와 NEW는 기본 매개변수입니다).

다음 예시에서, 명령은 현재 열려 있는 카탈로그와 연관된 테이블 컬럼에 있는 카탈로그화되지 않은 이미지를 카탈로그화하는 데 사용됩니다. 이미지는 카탈로그의 모든 특성용으로 카탈로그화됩니다.

```
CATALOG QBIC COLUMN FOR NEW
```

이미지 카탈로그 해제

이미지 카탈로그 해제는 QBIC 카탈로그에서 이미지용 항목을 제거함을 의미합니다. 이미지를 카탈로그화 해제하려면 QbUncatalogImage API를 사용하십시오. 명령행에서 개별 이미지를 식별할 아무런 방법이 없기 때문에, 명령으로 이미지를 카탈로그화 해제할 수 없습니다. API 사용시, 카탈로그 핸들과 이미지 핸들을 지정하십시오(사용자 테이블에서 이미지 핸들을 검색할 수 있습니다). QBIC 카탈로그는 이미지를 카탈로그화 해제하기 전에, 갱신을 위해 열려 있어야 합니다.

예를 들어, 다음의 명령문은 사용자 테이블에서 이미지 핸들을 검색한 후 이미지를 카탈로그화 해제합니다.

```
/* Retrieve the image handle */

EXEC SQL BEGIN DECLARE SECTION;
char Img_hdl[251];
EXEC SQL END DECLARE SECTION;

QbCatalogHandle   CatHdl;

EXEC SQL SELECT PICTURE INTO :Img_hdl
  FROM EMPLOYEE
  WHERE NAME='Anita Jones';
```

```
/* Uncatalog the image */

rc=QbUncatalogImage(
                CatHdl,                                 /* catalog handle */
                Img_hdl);                               /* image handle */
```

이미지 재카탈로그화

이미지 카탈로그화시, Image Extender는 QBIC 카탈로그로 식별된 이미지 특성을 분석하고, 카탈로그에 그 특성을 저장합니다. QBIC 카탈로그에 특성 추가시, Image Extender는 미리 카탈로그된 이미지에 대한 새로운 특성을 자동으로 분석합니다. 새 특성값을 카탈로그에 추가하려면, 이미지를 모두 다시 카탈로그화해야 합니다.

QBIC 카탈로그에 이미지를 다시 카탈로그화하려면, QbReCatalogColumn API나 CATALOG QBIC COLUMN 명령을 사용하십시오. Image Extender는 현재 카탈로그에 있는 모든 특성 데이터를 제거합니다. 그런 다음 새로운 특성을 포함한, 모든 특성에 대해 이미지를 분석하고, 이미지를 카탈로그화합니다. QBIC 카탈로그는 이미지를 다시 카탈로그화하기 전에 열려 있어야 합니다.

API 사용: QbReCatalogColumn API 사용시, 카탈로그 핸들을 지정하십시오.

다음의 예시에서, QBIC의 카탈로그에 있는 이미지가 재분석됩니다.

```
QbCatalogHandle   CatHdl;

rc=QbReCatalogColumn(
                CatHdl);                                /* catalog handle */
```

명령행 사용: 이미지를 다시 카탈로그화하려면, CATALOG QBIC COLUMN 명령을 사용하십시오. 명령은 열려 있는 카탈로그에 대해 작동합니다. 또한 명령을 사용하여 이미지를 수동으로 카탈로그화할 수도 있습니다(155 페이지의 『수동으로 이미지 카탈로그화』 참조).

명령 수행시 매개변수 FOR와 ALL을 지정하십시오. 이것은 사용자가 Image Extender에서 이미지 모두를 다시 카탈로그하기를 원함을 의미합니다.

다음의 예시에서, 지금 열려 있는 QBIC 카탈로그에 카탈로그화된 이미지가 다시 카탈로그화됩니다.

```
CATALOG QBIC COLUMN FOR ALL
```

QBIC 카탈로그 재분산(EEE 전용)

노드 그룹에 노드를 추가하거나 노드 그룹에서 노드를 제거할 때는 DMBRedistribute API 또는 REDISTRIBUTE NODEGROUP 명령을 사용하여 QBIC 특성 데이터를 재분산하십시오. 이 명령은 대응 사용자 데이터와 동일한 노드에 QBIC 특성 데이터를 놓습니다.

재분산 프로세스가 오류를 리턴하면, 명령 응답에서 제공하는 지시사항에 따라 CONTINUE 매개변수가 있는 명령이나 CONTINUE 매개변수가 없는 명령을 재수행할 수 있습니다. 이 옵션은 시스템에 처음부터 시작하지 말고 정지된 부분에서 계속하도록 지시합니다. DB2의 REDISTRIBUTE 명령 수행 후 처음으로 REDISTRIBUTE NODEGROUP 명령을 수행할 때에는 CONTINUE 매개변수를 사용해서는 안됩니다.

데이터 무결성을 유지하려면, 한 번에 한 개의 노드 그룹을 재분산하십시오. 하나의 노드 그룹이 재분산을 완료한 후에 다른 노드 그룹의 재분산을 시작하십시오.

API 사용: 다음 예시는 groupone 노드 그룹에서 QBIC 특성 데이터를 재분산하는 방법을 보여줍니다.

```
#include <dmbrdst.h>

rc = DMBRedistribute(groupone,"continue");
```

명령행 사용: 다음 예시는 REDISTRIBUTE NODEGROUP 명령에 CONTINUE 매개변수를 지정하여 my_nodegroup 노드의 데이터를 재분산하는 방법을 보여줍니다.

```
redistribute nodegroup my_nodegroup continue
```

QBIC 카탈로그 닫기

QBIC 카탈로그를 닫으려면, QbCloseCatalog API나 CLOSE QBIC CATALOG 명령을 사용하십시오. 이것을 닫기 전에, 카탈로그는 열려 있어야 합니다.

API 사용: QbCloseCatalog API 호출 발행시 카탈로그 핸들을 지정하십시오. 예를 들어:

```
QbCatalogHandle   CatHdl;

rc=QbCloseCatalog(
                  CatHdl);                              /* catalog handle */
```

명령행 사용: CLOSE QBIC CATALOG 명령은 현재 열려 있는 카탈로그에 대해 작동합니다. 다음의 예시에서, 이 명령은 현재 열려 있는 QBIC 카탈로그를 닫는 데 사용됩니다.

```
CLOSE QBIC CATALOG
```

QBIC 카탈로그 삭제

QBIC 카탈로그를 삭제하면 카탈로그 테이블의 특성 데이터 모두를 삭제합니다. 그 결과, 관련 이미지는 더 이상 내용별로 조회할 수 없습니다. QBIC 카탈로그를 삭제하려면, 카탈로그에 연관된 테이블에 대한 ALTER나 CONTROL 권한을 가져야 합니다. 이것을 삭제하기 전에 카탈로그가 열려 있어야 합니다.

QBIC 카탈로그를 삭제하려면, QbDeleteCatalog API나 DELETE QBIC CATALOG 명령을 사용하십시오. QBIC 카탈로그 삭제시, 사용자 테이블 및 카탈로그와 연관된 컬럼을 이름 지정하십시오.

API 사용: 다음의 예시에서, QbDeleteCatalog API는 QBIC 카탈로그를 삭제하는 데 사용됩니다.

```
rc=QbDeleteCatalog(
                  "employee",                           /* user table */
                  "picture");                           /* image column */
```

명령행 사용: DELETE QBIC CATALOG 명령은 현재 열려 있는 카탈로그에 대해 작동합니다. 다음의 예시에서, 명령은 현재 열려 있는 QBIC 카탈로그를 삭제하는 데 사용됩니다.

```
DELETE QBIC CATALOG employee picture
```

QBIC 카탈로그 샘플 프로그램

162 페이지의 그림26은 QBIC 카탈로그를 작성하는 C로 코딩된 프로그램의 일부를 보여줍니다. 프로그램은 또한 이미지 컬럼을 QBIC 카탈로그로 카탈로그화합니다. SAMPLES 서브디렉토리에 있는 QBCATDMO.C 파일에서 완전한 프로그램을 찾을 수 있습니다. 완전한 프로그램을 수행하기 전에 ENABLE 및 POPULATE

샘플 프로그램을 수행해야 합니다(SAMPLES 서브디렉토리에도 있음). 샘플 프로그램에 대한 자세한 정보는 643 페이지의 『부록B. 샘플 프로그램 및 미디어 파일』을 참조하십시오.

프로그램에서 다음과 같은 사항을 주목하십시오.

1. dmbqbapi 헤더 파일 포함.
2. 데이터베이스에 연결.
3. 카탈로그 작성. 카탈로그는 자동 카탈로그화가 작동 중지된 상태로 작성됩니다.
4. 갱신용으로 카탈로그 열기.
5. 평균 색상 특성을 카탈로그에 추가.
6. 이미지 컬럼을 카탈로그화.
7. 카탈로그 닫기.

```
#include <sql.h>
#include <sqlcli.h>
#include <sqlcli1.h>
#include <dmbqbqpi.h> 1
#include <stdio.h>

/*****************************************************************************/
/*  Define the function prototypes                                           */
/*****************************************************************************/

void printError(SQLHSTMT hstmt);
void createCatalog();
void openCatalog();
void closeCatalog();
void addFeature();
void catalogImageColumn();

QbCatalogHandle cHdl = 0;

static SQLHENV henv;
static SQLHDBC hdbc;
static SQLHSTMT hstmt;
static SQLRETURN rc;
char tableName[] = "sobay_catalog";
char columnName[] = "covers";

SQLCHAR uid[18+1];
SQLCHAR pwd[30+1];
SQLCHAR dbnName[SQL_MAX_DSN_LENGTH+1];

void main ()
{
/*---- prompt for database name, userid, and password ----*/
  printf("Enter database name:\n");
  gets((char *) dbName);
  printf("Enter userid:\n");
  gets((char *) pwd);
/* set up the SQL CLI environment */
 SQlAllocEnv(&henv);
 SQLAllocConnect(henv, &hdbc);
 rc = SQLConnect(hdbc,dbname,SQL_NTS,uid,SQL_NTS,pwd,SQL_NTS); 2
 if (rc != SQL_SUCCESS)
 {
  printError(SQL_NULL_HSTMT);
  exit(1);
 }
```

그림 26. QBIC 카탈로그 샘플 프로그램 (1/4)

```
 createCatalog();
   openCatalog();
   addFeature();
 getCatalogInfo();
 listFeatures();
 catalogImageColumn();
 closeCatalog();

 SQLDisconnect(hdbc);
 SQLFreeConnect(hdbc);
 SQLFreeEnv(henv);
}
/*********************************************************************/
void createCatalog()
{
 SQLINTEGER autoCatalog = 0;
 SQLINTEGER retLen;
   SQLINTEGER errCode = 0;
   char errMsg[500];

   QbCreateCatalog( 3
         (char *) tableName,
         (char *) columnName,
         autoCatalog,
         0
        );

   DBiGetError(&errCode, errMsg);
   if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
}
/*********************************************************************/
void openCatalog()
{
   SQLINTEGER errCode = 0;
   char errMsg[500];
 SQLINTEGER mode = qbiUpdate;
```

그림 26. QBIC 카탈로그 샘플 프로그램 (2/4)

```
   QbOpenCatalog( 4
         (char *) tableName,
         (char *) columnName,
         mode,
         &cHdl
        );

   DBiGetError(&errCode, errMsg);
   if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
}
/*********************************************************************/
void addFeature()
{
 SQLINTEGER errCode=0;
 char errMsg[5
 if(cHdl) /* if we have an open catalog, else do nothing */
 {
   char featureName*lbrk.] = "QbColorFeatureClass"; 5
     QbaddFeature(
         cHdl,
         featureName
        );

     DBiGetError(&errCode, errMsg);
     if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);
 }
 else
 {
  exit(1);
 }
}
/*********************************************************************/
void catalogImageColumn()
{
   SQLINTEGER errCode = 0;
   char errMsg[500];
```

그림 26. QBIC 카탈로그 샘플 프로그램 (3/4)

```
 if(cHdl) /* if we have an open catalog, else do nothing */
 {
  SQLRETURN rc;
    QbCatalogColumn( 6
         cHdl,
        );

    DBiGetError(&errCode, errMsg);
    if(errCode) printf("Error code is %d Error Message %s", errCode, errMsg);

 else
 {
  exit(1);
 }
}
/*********************************************************************/
void closeCatalog()
{
 if(cHdl) /* if we have an open catalog, else do nothing */
 {
    QbCloseCatalog( 7
         cHdl,
        );
 }
}
/*********************************************************************/
```

그림 26. QBIC 카탈로그 샘플 프로그램 (4/4)

조회 빌드

내용별 이미지 조회시, 조회용 입력과 카탈로그화된 이미지의 목표 세트를 식별합니다. 조회용 입력은 조회에 사용된 특성, 특성값 및 특성 가중치 이름(즉, 각각의 특성에 위치한 강조사항)을 지정합니다.

이 입력을 제공하는 방법은 두 가지입니다.

- 조회에 조회 문자열을 지정하십시오. 조회 문자열은 특성, 특성값 및 조회에 대한 가중치를 지정하는 문자열입니다.
- 조회 오브젝트를 작성하고 조회에서 참조하십시오. 조회 오브젝트는 특성과 특성 가중치를 지정합니다. 이것은 또한 각각의 특성용 데이터 소스를 식별합니다. 데이터 소스는 각각의 특성값을 제공합니다.

조회 문자열 지정

특성, 특성값 및 조회용 특성 가중치를 식별하려면, 조회 문자열을 사용할 수 있습니다. **조회 문자열**은 *feature_name value* 형식을 갖는 문자열입니다. 여기서 *feature_name*은 QBIC 특성 이름이고, *value*는 특성에 연관된 값입니다.

조회에 다양한 특성을 지정할 수 있습니다. 그런 다음 『특성 값』에 설명된 것처럼, 각각의 특성에 대해 특성 이름-값 쌍을 지정합니다. 각각의 쌍은 AND절에 의해 분리됩니다. 조회에서 복수 특성을 지정시, 168 페이지의 『특성 가중치』에 설명된 것처럼 하나 이상의 특성에 하나의 가중치를 지정할 수도 있습니다. 조회 문자열은 이때, *feature_name value weight* 형식이며, 여기서 *weight*는 특성에 대해 할당된 가중치입니다.

Image Extender는 조회 문자열을 사용하는 하나의 API(QbQueryStringSearch)와 두 개의 UDF(QbScoreFromStr 및 QbScoreTBFromStr)를 제공합니다. 조회 발행시, 적절한 API나 UDF를 사용하고 입력 매개변수로 조회 문자열을 지정합니다. (세부사항은 177 페이지의 『이미지 내용별로 조회 발행』을 참조하십시오.)

특성 값

조회에서 각각의 특성에 대한 조회 문자열에 특성 값을 지정하십시오.

DB2 명령에서 조회를 전달할 때는, 조회가 적절히 기능하도록 특정 파일 이름 지정 규칙에 따라야 합니다. 공백이나 닫는 각괄호(>)를 포함하는 파일은 큰 따옴표로 묶어야 합니다. 다른 파일 이름들은 선택적으로 큰 따옴표로 묶을 수 있습니다. 파일 이름을 따옴표로 묶을 경우, 각 따옴표 앞에 Escape 문자(\)가 있어야 합니다. 조회가 DB2 명령 내에서 전달되지 않는 경우에는 따옴표와 escape 문자가 없어도 됩니다.

다음 예에서, 조회 문자열은 DB2 명령 내에서 전달됩니다.

```
db2 "select image_id from table
(mmdbsys.QbScoreTBFromStr
('texture file=<server,patterns/ptrn07.gif>',
'fabric',
'swatch_img',
10))
as T1"
```

표9는 각각의 특성에 대해 지정할 수 있는 값을 나열합니다. 각 기능 이름 바로 아래에는 대신 사용할 수 있는 짧은 버전이 있습니다.

표 9. 조회 문자열에 지정될 수 있는 특성 값

특성 이름	값
averageColor, average 또는 QbColorFeatureClass	color=<*Rvalue*, *Gvalue*, *Bvalue*> 각각의 색상값은 이미지의 빨간색 값(*Rvalue*), 녹색 값 (*Gvalue*) 및 파란색 값(*Bvalue*)을 식별하는 0에서 255까지의 정수입니다. file=<*file_location*, *filename*> *file_location*은 서버 파일의 경우 `server`입니다. *filename*은 파일이 상주하는 시스템에 적절한 형식으로 지정된 완전한 파일 경로이거나, 상대 파일 이름입니다. DB2 Extender는 환경 변수를 사용하여 상대 파일 이름을 해결합니다(633 페이지의 『파일 이름을 해석하기 위해 환경 변수가 사용되는 법』 참조).
histogram, histogramcolor 또는 QbColorHistogramFeatureClass	histogram=<(*hist_value*, *Rvalue*, *Gvalue*, *Bvalue*>), ... 각각의 히스토그램 색상 값은 히스토그램(*hist_value*)에서 그 색상의 백분율(1에서 100까지)과 빨간색 값(*Rvalue*), 녹색 값 (*Gvalue*) 및 파란색 값(*Bvalue*)을 식별하는 절에 지정됩니다. file=<*file_location*, *filename*> *file_location*은 서버 파일의 경우 `server`입니다. *filename*은 파일이 상주하는 시스템에 적절한 형식으로 지정된 완전한 파일 경로이거나, 상대 파일 이름입니다. DB2 Extender는 환경 변수를 사용하여 상대 파일 이름을 해결합니다.
draw, positional 또는 QbDrawFeatureClass	file=<*file_location*, *filename*> handle=<*image_handle*> *file_location*은 서버 파일의 경우 `server`입니다. *filename*은 파일이 상주하는 시스템에 적절한 형식으로 지정된 완전한 파일 경로이거나, 상대 파일 이름입니다. DB2 Extender는 환경 변수를 사용하여 상대 파일 이름을 해결합니다.

표 9. 조회 문자열에 지정될 수 있는 특성 값 (계속)

특성 이름	값
texture 또는 QbTextureFeatureClass	file=<*file_location*, *filename*> handle=<*image_handle*> *file_location*은 서버 파일의 경우 server입니다. *filename*은 파일이 상주하는 시스템에 적절한 형식으로 지정된 완전한 파일 경로이거나, 상대 파일 이름입니다. DB2 Extender는 환경 변수를 사용하여 상대 파일 이름을 해결합니다.

특성 가중치

조회 문자열에 복수 특성을 지정한다면, 하나 이상의 특성에 대해 가중치를 지정할 수도 있습니다. 특성 가중치는 유사 스코어를 계산하고, 이미지 내용별 조회 결과를 리턴할 때 Image Extender가 특성에 배치한 강조사항을 표시합니다. 특성 가중치가 높아질수록 조회에서 특성에 강조사항이 더욱 커집니다. 가중치는 0.0 이상의 실수(예를 들어, 2.5 또는 10.0)입니다. 조회 문자열에 가중치를 할당하지 않는 경우, Image Extender는 특성의 기본 가중치를 사용합니다. 해당 특성이 조회 문자열에 지정된 유일한 특성인 경우 가중치를 할당하는 것은 의미가 없습니다(특성은 조회에서 항상 완전 가중치를 가질 것입니다).

특성의 가중치는 조회에 지정된 다른 특성에 상대적입니다. 예를 들어, 조회 문자열에 평균 색상과 텍스처 특성을 그리고 평균 색상에 대해 가중치 2.0을 지정한다고 하면 Image Extender는 텍스처 값의 두 배 강조된 평균 색상 값을 제공합니다.

예시

다음 조회 문자열은 평균 빨간색을 지정합니다.

```
averageColor color=<255, 0, 0>
```

다음 조회 문자열은 10% 빨간색, 50% 녹색 및 40% 파란색으로 구성된 히스토그램을 지정합니다.

```
histogram histogram=<(10, 255, 0, 0), (50, 0, 255, 0),
                                      (40, 0, 0, 255)>
```

다음 조회 문자열은 평균 색상 값과 텍스처 값을 지정합니다. 텍스처 값은 서버 파일에 있는 이미지에 의해 제공됩니다. 텍스처 가중치는 평균 색상 값의 두 배입니다.

```
averageColor color=<30, 200, 25> and
  texture file=<server, "\patterns\pattern7.gif"> weight=2.0
```

조회 오브젝트 사용

특성, 특성 값 및 조회에 대한 특성 가중치를 식별하기 위해 조회 오브젝트를 사용할 수 있습니다. 조회 오브젝트를 작성하고 그것에 특성을 추가합니다. 그 때, 데이터 소스는 각각의 특성에 대한 데이터 소스를 지정합니다. 데이터 소스는 특성 값을 제공합니다. 예를 들어, 데이터 소스는 파일에 있는 이미지일 수 있습니다. 평균 색상이 적정 특성인 경우, 이미지의 평균 색상이 조회 오브젝트와 연관됩니다. 조회 오브젝트에 복수 특성을 추가한다면, 하나의 특성에 하나의 가중치를 할당할 수 있습니다.

Image Extender는 조회 오브젝트를 사용하는 세 개의 API(QbQuerySearch, QbQueryStringSearch 및 QbQueryNameSearch)와 두 개의 UDF(QbScoreFromName 및 QbScoreTBFromName)를 제공합니다. 조회 발행시, 적절한 API나 UDF를 사용하고 입력 매개변수로 조회 문자열을 지정합니다.(세부사항은 177 페이지의 『이미지 내용별로 조회 발행』을 참조하십시오.)

조회 오브젝트 작성

조회 오브젝트를 작성하려면 QbQueryCreate API를 사용하십시오. 응답으로, Image Extender는 조회 오브젝트용 핸들을 리턴합니다. 핸들은 QBIC에 대한 Include(헤더) 파일에 dmbqbapi.h로 정의된 QbQueryHandle의 QBIC 지정 데이터 유형을 갖습니다.

API 사용시, 조회 오브젝트 핸들을 가리켜야 합니다. 특성 추가와 같이 조회 오브젝트에 다른 조작을 수행하는 API에 핸들도 지정해야 합니다.

예를 들어, 다음의 API 호출은 조회 오브젝트를 작성합니다.

```
QbQueryHandle  qHandle;

rc=QbQueryCreate(
             &qHandle);                          /* query object handle */
```

특성을 조회 오브젝트에 추가

조회 오브젝트에 특성을 추가하여 Image Extender가 조회하기를 원하는 이미지 특성을 식별합니다.

조회 오브젝트에 특성을 추가하려면, QbQueryAddFeature API를 사용하십시오. API 사용시, 조회 오브젝트 핸들을 지정하십시오. 또한, 특성을 이름 지정합니다. API에 하나의 특성만을 지정할 수 있습니다. 조회 오브젝트에 추가하려는 각각의 특성에 대해 분리 API 호출을 발행해야 합니다.

다음의 예시에서, QbQueryAddFeature API는 평균 색상 특성을 조회 오브젝트에 추가하는 데 사용됩니다.

```
char  featureName[qbiMaxFeatureName];
QbQueryHandle  qHandle;

rc=QbQueryAddFeature(
                qHandle,                          /* query object handle */
                "QbColorFeatureClass");           /* feature name */
```

조회 오브젝트에 특성용 데이터 소스 지정

조회 오브젝트에 특성용 데이터 소스를 지정하려면, QbQuerySetFeatureData API를 사용하십시오. 데이터 소스는 다음과 같습니다.

- 사용자 테이블 컬럼에 카탈로그화되거나 카탈로그화 해제된 이미지
- 클라이언트 워크스테이션의 이미지 파일
- 클라이언트 워크스테이션의 버퍼에 있는 이미지

또한, 평균 색상이나 히스토그램 색상 특성에 대해 데이터를 명시하여 지정할 수 있습니다. 예를 들어, 평균 색상의 빨간색, 녹색 및 파란색 값을 지정할 수 있습니다.

API 사용시:

- 조회 오브젝트 핸들을 지정하십시오.
- 특성 이름을 지정하십시오.
- QbImageSource 구조를 지시하십시오(세부사항은 171페이지를 참조하십시오).

데이터 소스 구조 사용: 다양한 구조가 조회 오브젝트에 대한 데이터 소스 정보를 제공하는 데 사용됩니다. 구조는 다음과 같습니다.

- QbImageSource
- QbColor
- QbHistogramColor

QbImageSource: QbImageSource 구조는 조회 오브젝트에 특성용 소스 유형을 식별합니다. 구조는 다음과 같이 QBIC에 대한 Include(헤더) 파일에서 dmbqbapi.h로 정의됩니다.

```
typedef struct{
          SQLINTEGER    type;
          union {
                char              imageHandle[MMDB_BASE_HANDLE_LEN+1];
                QbImageFile       clientFile;
                QbImageBuffer     buffer;
                QbSampleSource    reserved;
                QbColor           averageColor;
                QbHistogramColor  histogramColor[qbiHistogramCount];
          };
} QbImageSource;
```

QbImageSource 구조의 유형 필드는 소스 유형을 표시합니다. 다음과 같이 필드에 값을 설정할 수 있습니다.

값	의미
qbiSource_ImageHandle	소스는 사용자 테이블 컬럼에 있습니다.
qbiSource_ClientFile	소스가 클라이언트 워크스테이션 파일에 있습니다.
qbiSource_Buffer	소스가 클라이언트 워크스테이션 버퍼에 있습니다.
qbiSource_ServerFile	소스는 서버 파일에 있습니다.
qbiSource_AverageColor	소스는 평균 색상 스펙입니다.
qbiSource_HistogramColor	소스는 히스토그램 색상 스펙입니다.

이러한 설정은 적절한 특성용으로만 유효합니다. 예를 들어, qbiSource_AverageColor는 평균 색상 특성용으로만 유효합니다.

유형 필드를 qbiSource_ServerFile로 설정할 경우, 서버에 있는 파일의 이름과 유형에 대해 clientFile을 사용하십시오.

소스 유형에 따라, Image Extender는 또한 지정한 다른 정보도 조사합니다. 이것은 172 페이지의 표10에 나타납니다.

표 10. QbImageSource에서 Image Extender의 조사 내용

소스	Image Extender의 조사 내용	지정 장소
사용자 테이블	이미지 핸들	QbImageSource의 이미지 핸들 필드
파일	파일 이름 파일 형식	QbImageSource의 clientFile 필드
버퍼	파일 이름	QbImageBuffer (이 구조 사용에 대한 세부사항은 172페이지를 참조하십시오.)
평균 색상 스펙	빨간색, 녹색 및 파란색 값	QbColor 구조(이 구조 사용에 대한 세부사항은 172페이지를 참조하십시오.)
히스토그램 색상 스펙	색상 값과 백분율	QbHistogramColor 구조(이 구조 사용에 대한 세부사항은 172페이지를 참조하십시오.)

QbImageBuffer: 데이터 소스가 버퍼에 있는 경우, 이미지 형식, 길이 및 내용을 지정하려면 QbImageBuffer 구조를 사용하십시오. 구조는 다음과 같이 QBIC에 대한 Include(헤더) 파일에서 dmbqbapi.h로 정의됩니다.

```
typedef struct{
          char          format[qbiImageFormatLength+1];
          SQLINTEGER    length;
          char*         image;
} QbImageBuffer;
```

QbColor: 데이터 소스가 평균 색상 스펙에 있는 경우, 평균 색상의 빨간색, 녹색, 파란색 값을 지정하려면 QbColor 구조를 사용하십시오. 구조는 다음과 같이 QBIC에 대한 Include(헤더) 파일에서 dmbqbapi.h로 정의됩니다.

```
typedef struct{
          SQLUSMALLINT    red;          /*0 off - 65535 (fully on) */
          SQLUSMALLINT    green;        /*0 off - 65535 (fully on) */
          SQLUSMALLINT    blue;         /*0 off - 65535 (fully on) */
} QbColor;
```

평균값 계산에 인수 분해된 빨간색, 녹색, 파란색 픽셀의 양을 표시하려면, QbColor에 값을 설정하십시오. 값의 범위는 0에서 65535까지입니다. 값 0은 항목 무시를 의미합니다.

QbHistogramColor: 히스토그램 색상 스펙 각각의 색 구성요소를 지정하려면, QbHistogramColor 구조를 사용하십시오. 히스토그램 색상에 대한 완전 스펙은 QbHistogramColor 구조 배열로 되어 있습니다. 각각의 구조는 색상 값과 백분율

을 포함합니다. 색상 값은 빨간색, 녹색 및 파란색 픽셀 값으로 구성됩니다. 백분율은 목표 이미지에 필요한 해당 색의 비율을 지정합니다.

구조는 다음과 같이 QBIC에 대한 Include(헤더) 파일에서 dmbqbapi.h로 정의됩니다.

```
typedef struct{
          QbColor         color;
          SQLUSMALLINT    percentage;   /*0 - 100 */
} QbHistogramColor;
```

색상에 대한 빨간색, 녹색 및 파란색 픽셀의 총합을 표시하려면, 값을 QbColor에 설정하십시오. 값의 범위는 0에서 65535까지입니다. 목표 이미지에 필요한 지정된 색상의 백분율을 표시하려면 비율을 설정하십시오. 값의 범위는 1에서 100까지입니다. 히스토그램에 색상 구성요소에 대한 비율의 총합은 100 이하여야 합니다.

예시: 다음 예시에 있는 API는 조회 오브젝트에 히스토그램 색상 특성에 대한 데이터 소스를 지정합니다. 데이터 소스는 클라이언트 워크스테이션에 있는 파일입니다.

```
char           featureName[qbiMaxFeatureName];
QbQueryHandle  qHandle;
QbImageSource  imgSource;

imgSource.type=qbiSource_ClientFile;
strcpy(imgSource.clientFile.fileName,"/tmp/image.gif");
strcpy(imgSource.clientFile.format,"GIF");

rc=QbQuerySetFeatureData(
                qHandle,                                /* query object handle */
                "QbColorHistogramFeatureClass",         /* feature name */
                &imgSource);                            /* feature data source */
```

다음의 예시에서, 데이터 소스는 빨간색 평균 색상 스펙입니다.

```
char                 featureName[qbiMaxFeatureName];
QbColor              avgColor;
QbImageSource        imgSource;

imgSource.type=qbSource_AverageColor;
avgColor.red=255;
avgColor.green=0;
avgColor.blue=0;
strcpy(featureName,"QbColorFeatureClass");

rc=QbQuerySetFeatureData(
                qHandle,                                /* query object handle */
                featureName,                            /* feature name */
                &imgSource);                            /* feature data source */
```

조회 오브젝트에서 특성 가중치 설정

조회 오브젝트에 하나 이상의 특성을 추가했다면, 하나 이상의 특성이 조회에 제공되는 가중치를 지정할 수 있습니다. 특성 가중치를 지정하려면, QbQuerySetFeatureWeight API를 사용하십시오. 특성 가중치는 유사 스코어를 계산하고 이미지 내용별 조회 결과를 리턴할 때 Image Extender가 특성에 배치한 강조사항을 표시합니다. 특성용으로 지정한 가중치가 높아질수록 조회 오브젝트에 있는 특성에 강조사항이 더욱 커집니다.

QbQuerySetFeatureWeight를 발행하는 각각의 시간에 오직 하나의 특성에 대한 가중치를 지정할 수 있더라도, 조회 오브젝트에서 하나 이상의 특성에 대해 하나의 가중치를 지정할 수 있습니다. 가중치를 조회 오브젝트의 특성에 할당하지 않는 경우, Image Extender가 특성의 기본 가중치를 사용합니다. 한 특성에 하나의 가중치를 할당하는 것은 해당 특성이 조회 오브젝트에 있는 유일한 특성임을 의미하지는 않습니다.(그러한 특성은 조회 오브젝트에서 항상 완전 가중치를 가질 것입니다.)

API 사용시:

- 조회 오브젝트 핸들을 지정하십시오.
- 특성 이름을 지정하십시오.
- 특성 가중치를 지정하십시오. 가중치는 0 이상의 실수로 설정할 수 있습니다. 예를 들어, 2.5나 10.0과 같이 지정한 값이 높아질수록 특성에 강조사항이 더욱 커집니다. 설정은 조회 오브젝트의 특성에 대해 이전에 설정된 모든 가중치를 변경합니다.

다음의 예시에서, 조회 오브젝트는 평균 색상 특성과 최소한 하나 이상의 다른 특성을 포함합니다. QbQuerySetFeatureWeight API는 조회 오브젝트에 평균 색상 특성에 대한 가중치를 지정하는 데 사용됩니다.

```
char                 featureName[qbiMaxFeatureName];
double               weight;
QbQueryObjectHandle  qoHandle;

strcpy(featureName,"QbColorFeatureClass");
weight=5.00;

rc=QbQuerySetFeatureWeight(
                qoHandle,                          /* query object handle */
                featureName,                       /* feature name */
                &weight);                          /* feature weight */
```

조회 문자열 저장 및 재사용

조회 오브젝트는 사용자가 그것을 저장하지 않는다면 일시적인 것입니다. 단일 데이터베이스 연결 동안에만 존재합니다. 조회에서 조회 문자열을 저장하여 프로그램에서, 또는 현재 데이터베이스 연결이 제거된 후에도 프로그램 호출에서 다시 사용할 수 있습니다.

Image Extender는 조회 오브젝트에서 조회 문자열을 리턴하는 QbQueryGetString API를 제공합니다. 사용자는 다른 이미지 내용별 조회에서 그 조회 문자열을 QbQueryStringSearch API나 QbScoreFromStr 및 QbScoreTBFromStr UDF에 대한 입력으로 사용할 수 있습니다(177 페이지의 『이미지 내용별로 조회 발행』 참조).

조회 문자열은 다음을 사용하여 조회를 빌드할 때 빌드됩니다.

- QbQueryCreate
- QbQueryAddFeature
- QbQuerySetFeatureData
- QbQuerySetFeatureWeight
- QbQueryRemoveFeature

조회를 빌드하고 나면, QbQueryGetString을 호출하여 문자열을 확보할 수 있습니다. 해당 프로그램 내의 호출에서 이 조회 문자열을 사용하거나, 그 뒤의 응용프로그램 호출이나 다른 데이터베이스 연결에서 사용하기 위해 이를 파일에 저장할 수 있습니다. QbQueryGetString에 의해 리턴된 조회 문자열을 사용하는 것을 완료하였으면, 명시적으로 그 공간을 사용할 수 있도록 비워야 합니다.

다음 예시에서, QbQueryGetString은 조회 오브젝트에서 조회 문자열을 검색하기 위해 사용됩니다.

```
SQLRETURN rc;
char* qryString;
QbQueryHandle  qHandle;

.....           /* Here you create and use the query */

rc = QbQueryGetString(qHandle, &qryString);
if ( rc == 0) {
```

```
    ...         /* Use the query string as input here  */
    free((void *)qryString);
    qryString=(char *)0;
}
```

제한사항:: 클라이언트 파일을 사용하여 특성에 대한 데이터 소스를 지정할 경우, 조회 문자열은 특성 데이터를 반영하지 않습니다.

조회 오브젝트에 대한 정보 검색

조회 오브젝트에 추가된 특성 종류를 결정할 수 있습니다. 특성의 현재 가중치를 판별할 수도 있습니다.

API	검색
QbQueryGetFeatureCount	조회 오브젝트에 있는 특성 수
QbQueryListFeatures	조회 오브젝트에 있는 특성 이름

QbQueryGetFeatureCount API 발행시, 조회 오브젝트 핸들을 지정하십시오. 계수기도 지시해야 합니다. Image Extender는 계수기에 특성 계수를 리턴합니다.

다음의 예시에서, QbQueryGetFeatureCount API는 조회 오브젝트의 특성 수를 결정하는 데 사용됩니다.

```
SQLINTEGER     count;
QbQueryHandle  qHandle;

rc=QbQueryGetFeatureCount(
                qHandle,                          /* query object handle */
                &count);                          /* feature count */
```

QbQueryListFeatures API 호출 발행시, 리턴된 특성 이름을 보유하는 버퍼를 할당해야 합니다. 카탈로그 핸들과 리턴되는 특성 이름에 대한 버퍼 *크기*도 지정해야 합니다.

다음의 예시에서, QbQueryListFeatures API는 조회 오브젝트에 있는 각각의 특성명을 검색하는 데 사용됩니다.

```
SQLINTEGER     retCount,bufSize;
char*          featureName;
QbQueryHandle  qHandle;

bufSize=qbiMaxFeatureName;
featureName=(char*)malloc(bufSize);

rc=QbQueryListFeatures(
                qHandle,                          /* query object handle */
```

```
                bufSize                          /* size of buffer */
                &retCount,                       /* feature count */
                featureName);                    /* buffer for feature names */
```

조회 오브젝트에서 특성 삭제

조회 오브젝트에서 QbQueryRemoveFeature API를 사용하여 특성을 삭제하십시오. API 사용시, 조회 오브젝트 핸들을 지정하고 특성을 이름 지정하십시오.

다음의 예시에서, QbQueryRemoveFeature API는 조회 오브젝트에서 히스토그램 색상 특성을 삭제하는 데 사용됩니다.

```
char           featureName[qbiMaxFeatureName];
QbQueryHandle  qHandle;

strcpy(featureName,"QbColorHistogramFeatureClass");

rc=QbQueryRemoveFeature(
                qHandle,                          /* query object handle */
                featureName);                     /* feature name */
```

조회 오브젝트 삭제

이름 없는 조회 오브젝트를 QbQueryDelete API로 삭제하십시오. Image Extender는 현재 연결된 데이터베이스에서 조회를 삭제합니다.

QbQueryDelete API 사용시, 조회 오브젝트 핸들을 지정하십시오.

다음의 예시에서, QbQueryDelete API는 조회 오브젝트를 삭제하는 데 사용됩니다.

```
QbQueryHandle  qHandle;

rc=QbQueryDelete(
           qHandle);                              /* query object handle */
```

이름이 지정된 조회를 사용하였으면, QbQueryNameDelete API를 사용하여 조회 오브젝트를 삭제하십시오.

이미지 내용별로 조회 발행

이미지를 카탈로그화한 후, 내용별로 하나 이상의 이미지를 조회할 수 있습니다. 내용별로 이미지 조회시, 조회용 입력과 카탈로그화된 이미지의 목표 세트를 식별합니다. 조회 문자열(166 페이지의 『조회 문자열 지정』 참조)이나 조회 오브젝트(169 페이지의 『조회 오브젝트 사용』 참조)에서 입력을 지정할 수 있습니다.

조회 문자열을 사용한다면, DB2 명령행이나 프로그램 내에서 조회를 제출할 수 있습니다. 조회 오브젝트를 사용한다면, 프로그램 내에서 해당되는 핸들을 참조하여 조회를 제출할 수 있습니다.

Image Extender는 조회에 지정된 특성값을 목표 이미지에 있는 값과 비교하고, 각각의 이미지에 대한 스코어를 계산합니다. 스코어는 목표 이미지 특성값과 조회에 지정된 특성값의 유사성을 나타냅니다.

조회값과 가장 유사한 특성값의 이미지를 검색할 수 있습니다. 또한, 카탈로그화된 단일 이미지를 조회하여 스코어를 확보하거나, 테이블 컬럼에 있는 모든 카탈로그화된 이미지에 대한 스코어를 확보할 수도 있습니다.

이미지 조회

Image Extender는 테이블 컬럼에 카탈로그화된 이미지를 조회하는 세 개의 API를 제공합니다. API는 입력시 API가 조회 문자열이나 조회 오브젝트를 요청하는지에 따라서만 달라집니다.

API	입력
QbQueryStringSearch	조회 문자열
QbQuerySearch	조회 오브젝트 핸들
QbQueryNameSearch	조회 오브젝트 이름

세 개의 API 모두에서 다음을 수행합니다.

- 탐색할 이미지가 있는 테이블과 컬럼을 이름 지정합니다. 이미지는 QBIC 카탈로그에서 카탈로그화되어야 합니다.
- 리턴 결과의 최대 수를 지정합니다.
- 조회 범위를 지정하는 구조를 가리킵니다. 0, 널(NULL) 값 또는 빈 문자열에 포인터를 설정하십시오. 이것은 테이블 컬럼에 카탈로그화된 이미지 모두가 탐색되도록 지정합니다.
- 결과가 배열에 저장되는지를 표시하는 상수 qbiArray를 지정합니다. qbiArray 상수는 QBIC에 대한 Include(헤더) 파일에 dmbqbapi.h로 정의됩니다.

또한 사용자는 탐색 결과가 있는 출력 구조 배열을 가리킵니다. 응답으로, Image Extender는 조회 특성값과 가장 유사한 특성값의 목표 이미지 핸들을 이러한 구

조에 리턴합니다. 또한, 이미지 특성값의 조회값에 대한 유사 정도를 표시하는 각각의 이미지에 대한 스코어를 리턴합니다. 구조는 다음과 같이 QBIC에 대한 Include(헤더) 파일에서 dmbqbapi.h로 정의됩니다.

```
typedef struct{
          char       imageHandle[MMDB_BASE_HANDLE_LEN+1];
          SQLDOUBLE  SCORE
} QbResult;
```

지정한 결과의 최대 수를 보유하기에 충분히 큰 배열을 할당하고 API에서 배열을 지시해야 합니다. 계수기도 지시해야 합니다. Image Extender는 그것이 리턴한 결과 수에 계수값을 설정합니다.

다음의 예시에서, QbQueryStringSearch API는 테이블 컬럼에 카탈로그화된 이미지를 내용별로 조회하는 데 사용됩니다. 조회 범위를 가리키는 포인터가 0 값에 설정됨을 유의하십시오.

```
QbResult       returns[MaxQueryReturns];
SQLINTEGER     maxResults=qbiMaxQueryReturns;
SQLINTEGER     count;
QbQueryHandle  qHandle;
QbResult       results[qbiMaxQueryReturns];

rc=QbQueryStringSearch(
          "QbColorFeatureClass color=<255, 0, 0>" /*query string */
          "employee",                          /* user table */
          "picture",                           /* image column */
          maxResults,                          /* maximum number of results */
          0,                                   /* query scope pointer * /
          qbiArray,                            /* store results in an array */
          &count,                              /* count of returned images */
          results);                            /* array of returned results */
```

QbQuerySearch API를 사용하는 요청은 다음과 같습니다. 조회 오브젝트 핸들이 입력으로 지정됨을 유의하십시오.

```
QbResult       returns[MaxQueryReturns];
SQLINTEGER     maxResults=qbiMaxQueryReturns;
SQLINTEGER     count;
QbQueryHandle  qHandle;
QbResult       results[qbiMaxQueryReturns];

rc=QbQuerySearch(
                qHandle,                       / query object handle */
               "employee",                     /* user table */
               "picture",                      /* image column */
               maxResults,                     /* maximum number of results */
               0,                              /* query scope pointer * /
```

```
qbiArray,                       /* store results in an array */
&count,                         /* count of returned images */
results);                       /* array of returned results */
```

이미지 스코어 검색

Image Extender는 테이블 컬럼에 있는 카탈로그화된 이미지의 스코어를 검색하기 위해 SQL문에서 사용할 수 있는 4개의 UDF를 제공합니다. 스코어는 0.0에서 무한에 가까운 매우 큰 숫자까지의 배정밀도 부동 소수점 값입니다. 스코어가 낮을수록 이미지의 특성값은 조회에 지정된 특성값과 더욱 일치합니다. 스코어 0.0은 이미지가 정확히 일치함을 의미합니다.

UDF는 다음과 같습니다.

- QbScorefromStr
- QbScoreTBfromStr
- QbScoreFromName
- QbScoreTBFromName

권장사항: 카탈로그화된 단일 이미지의 스코어를 확보하려면 QbScoreFromStr UDF를 사용하십시오. 테이블 컬럼에서 여러 개의 카탈로그화된 이미지의 스코어를 확보하려면 QbScoreTBFromStr UDF를 사용하십시오.

단일 이미지의 스코어 검색

테이블 컬럼에서 카탈로그화된 단일 이미지의 스코어를 확보하려면 QbScoreFromStr UDF를 사용하십시오. QbScoreFromStr UDF에 대한 입력으로서 조회 문자열을 지정하십시오. QbScoreFromName UDF를 사용할 경우, QbScoreFromName UDF에 대한 입력으로 조회 오브젝트의 이름을 지정하십시오. 두 개의 UDF에서, 목표 이미지가 있는 테이블 컬럼 이름을 지정하기도 합니다.

다음 조회에서, QbScoreFromStr UDF는 테이블 컬럼에서 평균 색상 스코어가 빨간색에 가까운 카탈로그화된 이미지를 찾는 데 사용됩니다.

```
SELECT name, description
decimal (QbScoreFromStr(swatch_img,
            'QbColorFeatureClass color=<255, 0, 0>'),   /* query string *
                  10, 5) AS score
FROM fabric                                             /* table column */
ORDER BY score
```

여러 이미지의 스코어 검색

테이블 컬럼에서 여러 개의 카탈로그화된 이미지의 스코어를 확보하려면 QbScoreTBFromStr UDF를 사용하십시오. 이름이 지정된 조회가 있으면, QbScoreTBFromName UDF를 사용할 수 있습니다. 양쪽 UDF는 이미지 핸들과 스코어의 2컬럼 테이블을 리턴합니다. 테이블에 있는 행은 스코어에 대한 내림차순입니다. 결과 테이블의 핸들 컬럼 이름은 IMAGE_ID이고, 스코어 컬럼 이름은 SCORE입니다.

QBScoreTBFromStr UDF에 대한 입력으로 조회 문자열을 지정하십시오. QbScoreTBFromName UDF에 입력시, 조회 오브젝트 이름을 지정하십시오. 두 개의 UDF에서, 목표 이미지가 있는 테이블과 컬럼 이름을 지정하기도 합니다. 결과 테이블에 리턴하는 행의 최대수를 지정할 수도 있습니다. 결과의 최대 수를 지정하지 않으면, UDF는 목표 테이블 컬럼에 각각의 카탈로그화된 이미지를 리턴할 것입니다.

다음 조회에서, QbScoreTBFromStr UDF는 테이블 컬럼에서 텍스처가 서버 파일에 있는 이미지의 텍스처와 가장 가까운 10개의 카탈로그화된 이미지를 찾는 데 사용됩니다.

```
SELECT name, description
FROM fabric
WHERE CAST (swatch_img as varchar(250))IN
 SELECT CAST (image_id as varchar(25)) FROM TABLE
 (QbScoreTBFromStr
 (QbTextureFeatureClass file=<server,"patterns/ptrn07.gif">'   /*query string */

      'fabric',                                                /* table */
      'swatch_img',                                            /* table column */
      10))                                          /* maximum number of results */
AS T1));
```

QBIC 조회 샘플 프로그램

183 페이지의 그림27은 QBIC 조회를 빌드하고 실행하는 C로 코딩된 프로그램의 일부분을 보여줍니다. 그림에 있는 코드는 평균 색상에 의한 이미지 조회입니다. 이것은 색상이나 이미지 파일 이름을 입력하도록 프롬프트합니다. 사용자는 또한 후속 조회용 예시 이미지로 조회에서 리턴하는 이미지를 사용할 수도 있습니다. 그런 다음 프로그램은 이미지 컬럼을 조회하기 위해 평균 색상으로서 이름 지정된 색상이나 이미지 색상을 사용합니다.

SAMPLES 서브디렉토리에 있는 QBICDEMO.C 파일에서 완전한 프로그램을 찾을 수 있습니다. 완전한 프로그램은 평균 색상뿐 아니라 히스토그램이나 위치 색상으로 이미지를 조회하는 데 사용될 수 있습니다. 완전한 프로그램을 수행하려면, ENABLE, POPULATE 및 QBCATDMO 샘플 프로그램을 수행해야 합니다 (SAMPLES 서브디렉토리에도 있음). 샘플 프로그램에 대한 자세한 정보는 643 페이지의 『부록B. 샘플 프로그램 및 미디어 파일』을 참조하십시오.

183 페이지의 그림27에서 다음 사항에 유의하십시오.

1 dmbqbapi 헤더 파일 포함.

2 데이터베이스 정보용 사용자를 프롬프트.

3 데이터베이스에 연결.

4 조회 오브젝트 작성.

5 조회 오브젝트에 특성을 추가.

6 입력 유형에 대해 사용자를 프롬프트(색상 이름, 이미지 파일 또는 이미 검색된 이미지).

7 특성용 데이터 소스 지정. 데이터 소스는 평균 색상에 대한 명시적인 스펙입니다.

8 조회 발행. Image Extender는 이미지의 전체 컬럼을 탐색합니다. 이것은 리턴된 이미지의 최대 수로 10을 지정하기도 합니다.

9 리턴된 이미지 설정에 다음 이미지를 표시. 이미지 표시에 대한 자세한 정보는 144 페이지의 『전체 크기 이미지 또는 비디오 프레임 표시』를 참조하십시오.

10 조회 오브젝트 삭제.

SAMPLES 서브디렉토리는 QBIC 조회를 빌드하고 사용하는 방법을 예증하는 또 다른 프로그램을 포함합니다. 프로그램 `QbicQry.java`는 QBIC 조회에 대한 검색 기준을 그래픽으로 지정하는 방법을 보여줍니다. 예를 들어, 프로그램은 평균 색상을 선택할 수 있도록 색상 선택기를 제공합니다. 프로그램은 선택사항을 조회 문자열로 변환시킵니다.

```
#include <sql.h>
#include <sqlcli.h>
#include <sqlcli1.h>
#include <dmbqbqpi.h> 1
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <color.h>
#include <ctype.h>

#define   MaxQueryReturns        10

#define   MaxDatabaseNameLength  SQL_SH_IDENT
#define   MaxUserIdLength        SQL_SH_IDENT
#define   MaxPasswordLength      SQL_SH_IDENT
#define   MaxTableNameLength     SQL_LG_IDENT
#define   MaxColumnNameLength    SQL_LG_IDENT

static  char       databaseName[MaxDatabaseNameLength+1];
static  char       userid[MaxUserIdLength+1];
static  char       password[MaxPasswordLength+1];

static  char       tableName[MaxTableNameLength+1];
static  char       columnName[MaxColumnNameLength+1];

static  char       line[4000];

static  QbResult   results[MaxQueryReturns];
static  long       currentImage   = -1;
static  long       imageCount  = 0;

static  char*      tableAndColumn;

static  QbQueryHandle   averageHandle = 0;
static  QbQueryHandle   histogramHandle = 0;
static  QbQueryHandle   drawHandle = 0;
static  QbQueryHandle   lastHandle = 0;

static  SQLHENV    henv;
static  SQLHDBC    hdbc;
static  SQLHSTMT   hstmt;
static  SQLRETURN  rc;

static  char*    listQueries =
        "SELECT NAME,DESCRIPTION FROM MMDBSYS.QBICQUERIES ORDER BY NAME";

static  char*    menu[] = {
```

그림 27. QBIC 조회 샘플 프로그램 (1/6)

```
/*
 12345678901234567890123456789012345678901234567890123456789012345678901234567890 */
"",
"+-----------------------------------------------------------------------------+",
"| AVERAGE COLOR colorname                                                     |",
"| AVERAGE FILE filename format                                                |",
"| AVERAGE LAST                                                                |",
"| Press Enter to display the next image in the series                         |",
"+-----------------------------------------------------------------------------+",
"",
0
};

static  char*       help[] = {
"",
"AVERAGE     Execute an average color query",
"  COLOR       Specifies the color to query for",
"  FILE        Specifies the file to compute the average color from",
"  LAST        Specifies the last displayed image be used to compute the color",
"  NEXT        Displays the next image from the current query or nothing if",
"                all of the image have been displayed."
"",
">>pause<<",
0
};
/*****************************************************************************/
/* doNext()                                                                  */
/*****************************************************************************/
static void doNext(void)
{
        int ret;
   if (currentImage < imageCount)
      currentImage++;
   if (currentImage < imageCount)
      ret = DBiBrowse("/usr/local/bin/xv %s", MMDB_PLAY_HANDLE,
            results[currentImage].imageHandle, MMDB_PLAY_NO_WAIT);  9
}
```

그림 27. QBIC 조회 샘플 프로그램 (2/6)

```
/*****************************************************************************/
/* doAverage()                                                               */
/*****************************************************************************/
static void doAverage(void)
{
   QbQueryHandle qohandle = 0; QbImageSource is; char* type;
   char* arg1; char* arg2;

   type = nextWord(0);
   if (abbrev(type, "color", 1)) {
      is.type = qbiSource_AverageColor;
      arg1 = nextWord(0);
      if (arg1 == 0) {
         printf("AVERAGE COLOR command requires a colorname argument.\n");
         return;
      }
      if (getColor(arg1, &is.averageColor) == 0) {
         printf("The colorname entered was not recognized.\n");
         return;
      }
   }
   else if (abbrev(type, "file", 1)) {
      is.type = qbiSource_ClientFile;
      arg1 = nextWord(0);
      if (arg1 == 0) {
         printf("AVERAGE FILE command requires a filename argument.\n");
         return;
      }
      arg2 = nextWord(0);
      if (arg2 == 0) {
         printf("AVERAGE FILE command requires a file format argument.\n");
         return;
      }
      strcpy(is.clientFile.fileName, arg1);
      strcpy(is.clientFile.format, arg2);
   }
   else if (abbrev(type, "last", 1)) {
      is.type = qbiSource_ImageHandle;
      if (0 <= currentImage &&; currentImage < imageCount)
         strcpy(is.imageHandle, results[currentImage]imageHandle);
      else {
         printf("No last image for AVERAGE LAST command\n");
         return;
      }
   }
```

그림 27. QBIC 조회 샘플 프로그램 (3/6)

```
   else {
      printf("AVERAGE command only supports COLOR, FILE, and LAST types.\n");
      return;
   }

   _QbQuerySetFeatureData(averageHandle, "QbColorFeatureClass", &is); 7
   _QbQuerySearch(averageHandle, tableAndColumn, "IMAGE",
         MaxQueryReturns, 0, 0, &imageCount, results); 8
   lastHandle = averageHandle;

   currentImage = -1;
}
/*****************************************************************************/
/* commandLoop()                                                             */
/*****************************************************************************/
void commandLoop(void)
{
   int  done = 0;

   while (!done) { 6
      displayText(menu);
      printf("%d", currentImage + 1);
      if (0 <= currentImage && currentImage < imageCount)
            printf("  %8.6f", results[currentImage].score);
      printf("> ");
      gets(line);
      done = processCommand(line);
   }
}
```

그림 27. QBIC 조회 샘플 프로그램 (4/6)

```
/*****************************************************************************/
/* main()                                                                    */
/*****************************************************************************/
void main(void)
{
   char*                inst;
   int          i;

   printf("\n\n");
   printf("Please enter: database_name [user_id] [password] "\n"); 2
   gets(line);
   if (copyWord(line, databaseName, sizeof(databaseName)) == 0)
      exit(0);
   copyWord(0, userid, sizeof(userid));
   copyWord(0, password, sizeof(password));
   printf("\n");

   if (SQLAllocEnv(&henv) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLAllocConnect(henv, &hdbc) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLConnect(hdbc, 3
                  (SQLCHAR*)databaseName,
                  SQL_NTS,
                  (SQLCHAR*)userid,
                  SQL_NTS,
                  (SQLCHAR*)password,
                  SQL_NTS) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);

   printf("Initializing . . .\n");
```

그림 27. QBIC 조회 샘플 프로그램 (5/6)

```
   inst = getenv("DB2INSTANCE");
   if (inst != 0 &&; strcmp(inst, "keeseyt") == 0)
      tableAndColumn = "KEESEY.TEST";
   else
      tableAndColumn = "QBICDEMO.TEST";

   _QbQueryCreate(&averageHandle); 4
   _QbQueryAddFeature(averageHandle, "QbColorFeatureClass");
   _QbQueryCreate(&histogramHandle);
   _QbQueryAddFeature(histogramHandle, "QbColorHistogramFeatureClass");
   _QbQueryCreate(&drawHandle);
   _QbQueryAddFeature(drawHandle, "QbDrawFeatureClass"); 5

   commandLoop();

   _QbQueryDelete(drawHandle);
   _QbQueryDelete(histogramHandle); 10
   _QbQueryDelete(averageHandle);

   if (SQLDisconnect(hdbc) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLFreeConnect(hdbc) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
   if (SQLFreeEnv(henv) != SQL_SUCCESS)
      sqlError(SQL_NULL_HSTMT);
}
```

그림 27. QBIC 조회 샘플 프로그램 (6/6)

제14장 비디오 장면 변경 검출

이 장에서는 DB2 Video Extender API로 비디오 클립에서 장면 변경의 검출 방법을 설명합니다. 이들 API는 Windows 3.1을 제외한 모든 DB2 Video Extender 플랫폼에서 사용 가능합니다. 비디오 장면 변경 검출은 MPEG-1 형식의 비디오 클립에 대해서만 지원됩니다.

비디오 장면 변경이란?

다음의 TV 방송국을 위해 프로그램을 비디오 테잎으로 녹화하는 TV 스튜디오를 상상해 보십시오. 최근 스튜디오는 Video Extender를 사용하여 DB2 데이터베이스에 비디오 테잎 클립을 저장하기 시작했습니다. 이것은 스튜디오 직원에게 프로그램 클립을 보는 기회뿐 아니라, 자신의 프로그램에 대한 전통적 정보 유형을 조회할 기회를 제공합니다.

스튜디오 직원들은 비디오 클립의 미리보기 옵션을 좋아할 것입니다. 그들은 스토리보드라 불리는 시각적 요약을 볼 수 있는 기능을 원합니다. 스토리보드의 예시는 190 페이지의 그림28에 나타납니다. 스토리보드 보기는 스튜디오 직원들이 비디오 전체를 볼 필요가 없다면, 비디오의 요점만 볼 수 있도록 돕습니다. 이것은 그들의 필요(예를 들어, 이것이 다운로딩과 보기 어느 쪽을 원하는지)에 맞는 비디오인지를 결정하도록 도울 수도 있습니다. 이러한 요구사항은 스튜디오에서 아주 중요합니다. 전체 비디오 대신에 스토리보드 보기는 다운로드와 보기 시간을 극적으로 줄일 수 있습니다. 이러한 방법의 비디오 장면 변경 검출 기능 사용에 대한 자세한 정보는 208 페이지의 『비디오에 있는 모든 컷에 대한 정보 저장』을 참조하십시오.

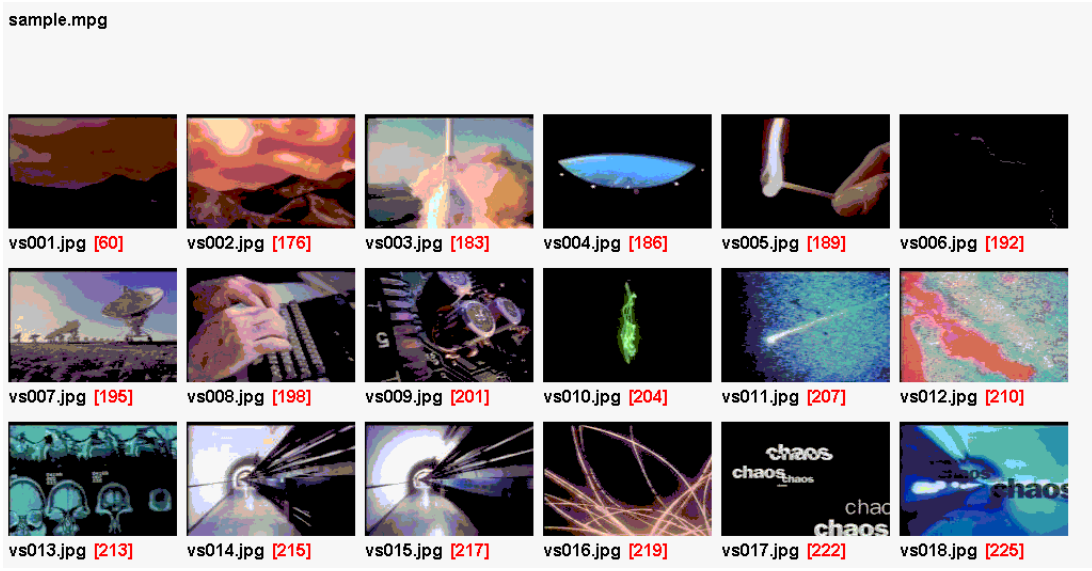


그림 28. 비디오 스토리보드. 대표 프레임은 내용과 비디오의 흐름을 요약합니다.

스튜디오는 자신의 스토리보드용 대표 프레임을 캡처하는 Video Extender의 비디오 장면 변경 검출 기능의 사용을 계획중입니다.

비디오 장면 변경은 두 개의 연속된 프레임 사이의 중요한 차이점이 있는 비디오 클립에 있는 포인트입니다. 이것은 예를 들어, 비디오를 레코드하는 카메라가 비디오의 뷰 포인트를 변경할 때 발생합니다. 두 장면 변경 사이에 프레임은 **컷**을 구성합니다.

Video Extender의 장면 변경[6] 검출시, 이것은 연관된 컷에 대한 데이터를 녹화합니다. 데이터는 컷을 시작하는 프레임 번호, 컷을 종료하는 프레임 번호 및 컷 내의 대표 프레임 번호를 포함합니다. 컷 데이터에는 대표 프레임의 픽셀 내용도 있습니다.

6. 비디오 장면 변경 검출 코드에는 University of California at Berkeley MPEG 해독기와 Boston University Multimedia Communication Laboratory의 수정사항이 들어 있습니다.

장면 변경의 찾기와 이용

Video Extender는 컷이나 비디오 클립에 있는 프레임을 찾는 데 이용할 수 있는 응용프로그램 프로그래밍 인터페이스 세트를 제공합니다. 컷이나 프레임을 찾은 후, 컷의 시작 및 끝 프레임 번호나 프레임의 픽셀 내용 등에 관한 데이터를 액세스할 수 있습니다. 그런 다음, 이러한 정보의 이후 처리를 위해 프로그램으로 전달할 수 있습니다. 예를 들어, 프레임 내용을 그것을 표시할 수 있는 프로그램으로 전달할 수 있습니다.

Video Extender는 **컷 카탈로그**에 있는 컷 데이터를 저장하기 위해 API도 제공합니다. 컷 카탈로그는 데이터베이스나 파일에 있을 수 있습니다. 파일의 컷 카탈로그를 액세스하거나 데이터베이스에 있는 컷 카탈로그의 읽기 전용 뷰에 액세스할 수 있습니다.

컷 카탈로그 파일에는 다음의 데이터에 대한 필드가 있습니다.

- 컷 카탈로그 이름
- Video Extender가 컷을 검출하는 방법을 제어하는 값(예를 들면, 컷에서 프레임의 최소 수)
- 컷용 대표 프레임으로 저장될 프레임 수와 프레임 종류를 제어하는 값
- 컷 번호
- 시작 프레임 번호
- 끝 프레임 번호
- 대표 프레임 번호
- 대표 프레임의 내용이 있는 파일 이름

데이터베이스에 있는 컷 카탈로그 뷰에는 다음 데이터에 대한 컬럼이 있습니다.

- 컷 핸들
- 비디오 테이블 이름
- 비디오 컬럼 이름
- 비디오 핸들
- 비디오 파일 이름
- 시작 프레임 번호

- 끝 프레임 번호
- 대표 프레임 번호
- 대표 프레임 데이터
- 주석

컷 카탈로그가 데이터베이스에 있는 경우, 컷 카탈로그 파일의 데이터를 액세스할 수 있거나 데이터를 조회할 수 있습니다. 대표 프레임 정보는 스토리보드 표시에서 특히 유용합니다. 또한 컷 카탈로그가 데이터베이스에 있다면, 다른 테이블에 관련된 컷 데이터를 조인할 수 있습니다. 예를 들어, 비디오 스튜디오는 데이터베이스에 컷 카탈로그를 작성할 수 있습니다. 그런 다음, 그들은 클립에 대한 정보뿐 아니라 비디오 클립이 있는 테이블로 카탈로그 데이터를 조인할 수 있습니다. 이러한 방법으로, 그들은 클립 내의 컷을 식별할 뿐 아니라, 클립과 클립에 대한 업무 정보를 검색하는 단일 조회를 사용할 수 있을 것입니다.

컷 검출 데이터 구조

컷 검출에 관련된 데이터는 컷 검출 헤더 파일, dmbshot.h에 있는 구조로 저장됩니다. 많은 컷 검출 API는 하나 이상의 이러한 구조를 지시할 것을 요청합니다. 이 구조의 일부분은 입력으로 Video Extender가 사용하는 데이터를 포함하는 데 사용됩니다. 예를 들어, 컷 제어 구조에는 컷 검출을 제어하는 정보가 있습니다. 대부분의 구조는 그것이 비디오에서 검색하는 데이터를 저장하기 위해 Video Extender에 의해 사용됩니다. 예를 들어, 비디오 프레임 데이터 구조에는 프레임의 픽셀 내용이 있습니다.

컷 검출용으로 사용되는 구조는 DBvIOType, DBvShotControl, DBvShotType, DBvFrameData 및 DBvStoryboardCtrl입니다.

DBvIOType

DBvIOType 데이터 구조에는 그것의 형식, 차원, 프레임 수와 같은 비디오에 대한 데이터가 있습니다. 데이터 구조는 다음과 같이 정의됩니다.

```
typedef struct {

   FILE *hFile;                    /* file handle for the video */
   char vhandle[255];              /* video handle (if from database)
   char vtable[255];               /* video table name (if from database) */
   char vcolumn[255];              /* video column name (if from database) */
   char vFile[255];                /* name of video file */
   char idxFile[255];              /* name of index file */
```

```
   char isIdx;                  /* 1 if the index exists, 0 otherwise */
   char isInDb;                 /* 1 if from DB, 0 if from file */
   int format;                  /* Format of the video */
   unsigned long dx, dy;        /* Dimensions of the video */
   unsigned long totalFrames;   /* TotalFrames in the video */
   unsigned long markFrame;     /* used by shot detection */
   unsigned long currentFrame;  /* The current video frame */
   DBvFrameData fd;             /* Frame data for current frame */
   DBvDCFrameData fdDc;         /* Frame data for DC images */
   unsigned char BGRValid;      /* reserved */
   unsigned short usDeviceID;   /* reserved */
   unsigned long hwnd;          /* reserved */
   int videoReset;              /* Flag if video is opened or seeked */
   int firstshot;               /* Used internally to indicate the first call */
   void *reserved               /* reserved */

} DBvIOType;
```

DBvShotControl

DBvShotControl 데이터 구조에는 검출 방법과 같은 컷 검출을 제어하는 데 사용되는 정보가 있습니다. 데이터 구조는 다음과 같이 정의됩니다.

```
typedef struct {

  unsigned long reserved;
  unsigned long method;           /* detection method */

    #define DETECT_CORRELATION  0x00000001
    #define DETECT_HISTOGRAM    0x00000002
    #define DETECT_CORRHIST     0x00000003
    #define DETECT_CORRHISTDISS 0x00000004


  int normalCorrValue;            /* Correlation threshold */
  int sceneCutSkipXY;             /* reserved */
  int CorrHistThresh;             /* Histogram threshold */
  int DissThresh;                 /* Dissolve threshold */
  int DissCacheSize;              /* Dissolve cache size */
  int DissNumCaches;              /* Dissolve cache number */
  int minShotSize;                /* Minimum frames in a shot */

} DBvShotControl;
```

표11에서는 DBvShotControl에 있는 각각의 필드와 그것이 허용하는 기본 설정값을 설명합니다. 이러한 필드를 기본값으로 초기화하려면, 197 페이지의 『컷 검출 데이터 구조에서 값 초기화』에 설명된 것처럼 DBvInitShotControl API를 사용하십시오.

DBvShotControl 설정은 비디오 유형에 따라 다릅니다. 계수화한 비디오에서 장면 변경은 내용과 형식에 따라 상당히 다릅니다. 또한 장면 변경 알고리즘의 정확도는 비디오에 따라 다릅니다. 전체 프레임 모양에서 아주 다르게 정의된 장면 변경은 더욱 미묘한 변경 유형 또는 전체 색상 내용이 동일한 변경보다 더 정확히 검출됩니다. 기본 DBvShotControl 필드 설정이 응용프로그램 대부분에 대해 수행이 원활할지라도, 검출하지 않거나 잘못 검출하는 경우를 줄이기 위해 이러한 설정을 조정해야 합니다.

표 11. DBvShotControl 필드

필드	의미
method	Video Extender가 장면 변경을 검출하는 데 사용하는 방법을 나타냅니다. 다음 방법 중 하나를 선택할 수 있습니다. DETECT_CORRELATION. 두 개의 연속된 프레임에서 픽셀 비교. 그 차이가 상관 임계값을 초과한다면, 장면 변경을 검출합니다. DETECT_HISTOGRAM. 두 개의 연속된 프레임의 히스토그램 값 비교. 히스토그램 값은 프레임에서 색상 분포를 측정. 그 차이가 히스토그램 임계값을 초과한다면, 장면 변경을 검출합니다. DETECT_CORRHIST. 가능한 장면 변경을 식별하기 위해 상관 방법을 사용합니다. 그때 가능한 장면 변경으로 표시된 프레임에 대한 히스토그램 방법을 사용하십시오. 히스토그램 임계값을 초과한다면, 장면 변경을 검출합니다. DETECT_CORRHISTDISS. DETECT_CORRHIST용과 동일하나, 해제용 추가 프레임을 검토합니다. 기본 방법은 DETECT_CORRHIST입니다.
normalCorrValue	상관 임계값을 지정하는 0에서 100까지의 정수 값. 이것은 두 개의 프레임에 있는 픽셀 간의 최소 상관 계수 값을 제공합니다. 값 0은 다음 프레임에 대한 장면 변경을 항상 검출함을 의미합니다. 값 100은 모든 픽셀이 하나의 프레임에서 다음 프레임으로 변경시에만 장면 변경을 검출함을 의미합니다. 기본값은 60입니다.
sceneCutSkipXY	예약됨.
CorrHistThresh	히스토그램 임계값을 지정하는 0에서 100까지의 정수 값. 이것은 연속하는 프레임의 히스토그램 값 사이의 차를 측정합니다. 값 0은 히스토그램 값이 한 프레임과 다음 프레임에서 아주 다른 경우에만 장면 변경을 검출함을 의미합니다. 값 100은 다음 프레임에 대한 장면 변경을 항상 검출함을 의미합니다. 기본값은 10입니다.
DissThresh	시험 임계값 해제를 지정하는 0에서 100까지의 정수 값. 이것은 해제 검출을 위해 시험 해제를 전달해야 하는 프레임에서 픽셀의 백분율을 측정합니다. 값 0은 프레임에 대한 해제를 항상 검출함을 의미합니다. 값 100은 프레임에 있는 모든 픽셀이 시험 해제를 전달하는 경우에만 해제를 검출함을 의미합니다. 기본값은 15입니다.

표 11. DBvShotControl 필드 (계속)

필드	의미
DissCacheSize	시험 해제의 경사 부분에 사용된 프레임 수를 지정하는 정수 값. 기본값은 4입니다.
DissNumCaches	시험 해제의 일치 부분에 사용되는 프레임 수를 지정하는 정수 값. 기본값은 7입니다.
minShotSize	컷용 프레임의 최소 수를 지정하는 정수 값. 컷이 검출되려면, 적어도 최소값과 동일한 수의 프레임이 있어야 합니다. 기본값은 5입니다.

DBvShotType

DBvShotType 데이터 구조에는 시작 프레임 번호, 끝 프레임 번호 및 대표 프레임 번호와 대표 프레임과 같은 컷에 대한 정보와 대표 프레임의 픽셀 내용에 대한 포인터가 있습니다. 데이터 구조는 다음과 같이 정의됩니다.

```
typedef struct {

  unsigned long startFrame;       /* starting frame number */
  unsigned long endFrame;         /* ending frame number */
  unsigned long repFrame;         /* representative frame number */
  DBvFrameData fd;                /* data for representative shot */
  unsigned long dx;               /* frame data width in pixels */
  unsigned long dy;               /* frame data height in pixels */
  char *comment;                  /* shot remark */

} DBvShotType;
```

DBvFrameData

DBvFrameData 데이터 구조에는 프레임의 픽셀 내용이 있습니다. 데이터 구조는 다음과 같이 정의됩니다.

```
typedef struct                    /* video frame data */
{
  /* MPEG 1 pixels */
  unsigned char *luminance;       /* Luminance pixel plane (black and white) */
  unsigned char *Cr;              /* Cr pixel plane */
  unsigned char *Cb;              /* Cb pixel plane */
  unsigned char *reserved;

  } DBvFrameData;
```

DBvStoryboardCtrl

DBvStoryboardCtrl 데이터 구조에는 비디오 카탈로그에 저장된 컷에 대한 대표 프레임의 종류와 수를 제어하는 값이 있습니다. 이 값들의 이용 방법에 대한 설명은 210 페이지의 『스토리보드 빌드』를 참조하십시오. 데이터 구조는 다음과 같이 정의됩니다.

```
typedef struct {

  int thresh1;                    /* threshold for small to medium scenes */
  int thresh2;                    /* threshold for medium to large scenes */
  int delta;                      /* offset used for representative frames */

} DBvStoryboardCtrl;
```

표12는 DBvStoryboardCtrl에 있는 각각의 필드와 그의 기본 설정값을 설명합니다. 이 필드들을 기본값으로 초기화하려면, 197 페이지의 『컷 검출 데이터 구조에서 값 초기화』에 설명된 것처럼 DBvInitStoryboardCtrl API를 사용하십시오.

DBvStoryboardCtrl 설정은 비디오 유형에 따라 다릅니다: 스토리보드에 선택적인 프레임 종류와 수는 비디오 유형에 따라 다릅니다. 기본 DBvStoryboardCtrl 필드 설정이 다양한 비디오 유형에 대해 잘 작동되더라도, 비디오의 시험 부분집합에 이러한 설정을 사용해 볼 수 있습니다. 그런 다음 비디오의 더 큰 집합용 스토리보드를 빌드하기 전에 적절한 값으로 설정을 조정할 수 있습니다.

표 12. DBvStoryboardCtrl 필드

필드	의미
thresh1	짧은 컷의 임계 값을 식별합니다 . thresh1보다 작은 수의 프레임을 갖는 컷이 짧은 컷입니다. 카탈로그화된다면, 짧은 컷에 대한 정보는 하나의 대표 프레임(중간 프레임)을 포함할 것입니다. 기본값은 90입니다. thresh1값이 -1로 설정된다면, 컷은 짧은 컷으로 간주될 것입니다(실제 길이와 상관없이).
thresh2	중대형 컷의 임계 값을 식별합니다. thresh1 값 이상 thresh2 값 이하의 프레임을 갖는 컷은 중간 컷으로 간주됩니다. 카탈로그화된다면, 중간 컷에 대한 정보는 두 개의 대표 프레임을 포함할 것입니다. 대표 프레임 위치는 델타 필드 값으로 제어됩니다. thresh2 값 초과인 컷은 긴 컷으로 간주됩니다. 카탈로그화된다면, 긴 컷에 대한 정보는 세개의 대표 프레임을 포함할 것입니다. 처음과 마지막 대표 프레임 위치는 델타 필드 값으로 제어됩니다. 두 번째 프레임은 중간 프레임입니다. 기본값은 150입니다. thresh2 값이 -1로 설정된다면, 컷은 짧은 컷으로 간주될 것입니다(실제 길이와 상관없이).
delta	대표 프레임으로 사용되는 옵셋을 확인합니다. 중간 컷과 긴 컷에 대해, 처음 대표 프레임은 컷의 처음에서 델타의 프레임 수까지의 옵셋입니다. 마지막 대표 프레임은 컷의 마지막에서 델타의 프레임 수까지의 옵셋입니다. 기본값은 5입니다.

컷 검출 데이터 구조에서 값 초기화

DBvShotControl 데이터 구조에서 값은 컷 검출을 제어합니다. DBvStoryboardCtrl 데이터 구조에 값은 스토리보드의 빌드를 제어합니다. 이러한 데이터 구조에 대한 필드용 값을 명확히 지정할 수 있습니다. 또한, 이 구조에 있는 값을 기본값으로 초기화할 수 있습니다. DBvStoryboardCtrl 데이터 구조에 있는 기본값은 194 페이지의 표11을 참조하십시오. DBvStoryboardCtrl 데이터 구조에 있는 기본값은 196 페이지의 표12를 참조하십시오.

DBvShotControl 데이터 구조에 값을 초기화하려면 DBvInitShotControl API를 사용하십시오. API 사용시, 컷 제어 구조를 지정해야 합니다. 예를 들어 다음의 명령문은 DBvShotControl 구조에 있는 필드를 기본값으로 초기화합니다.

```
DBvShotControl    shotCtrl;

rc=DBvInitShotControl(
             shotCtrl);          /* pointer to shot control structure */
```

DBvInitStoryboardCtrl API를 사용하여 DBvStoryboardCtrl 구조에 있는 값을 초기화하십시오. API 사용시, 스토리보드 제어 구조를 지정해야 합니다. 예를 들어 다음의 명령문은 DBvStoryboardCtrl 구조에 있는 필드를 기본값으로 초기화합니다.

```
DBvStoryboardCtrl    sbCtrl;

rc=DBvInitStoryboardCtrl(
             sbCtrl);          /* pointer to storyboard control structure */
```

컷이나 프레임 확보

비디오에서 컷이나 프레임을 확보하기 위해 Video Extender를 사용할 수 있습니다. 컷이나 프레임을 확보하기 전에, 컷 검출용 비디오를 열어야 합니다. Video Extender는 프레임과 컷을 액세스하기 위해 색인을 이용합니다. 컷이나 프레임을 확보하기 전에 비디오용 색인을 작성해야 합니다.

비디오가 열리고 색인이 작성된 후, 비디오의 다음 컷이나 프레임을 확보하거나 프레임 번호로 특정 프레임을 확보할 수 있습니다. Video Extender는 MPEG-1 형식으로 비디오 클립을 처리할 수 있습니다. 검색 프레임을 RGB 형식을 요청하는 프로그램으로 사용하려는 경우, Video Extender API를 사용하여 프레임을 해당 형식으로 변환할 수 있습니다.

컷 검출용 비디오 열기

DBvOpenFile API를 사용하여 파일에 저장된 비디오를 여십시오. 파일은 클라이언트에서 반드시 액세스 가능해야 합니다. DBvOpenHandle API를 사용하여 데이터베이스 테이블에 저장된 비디오를 여십시오. 응용프로그램은 데이터베이스에 먼저 연결해야 합니다. 비디오가 데이터베이스 테이블에 저장된다면, Video Extender는 비디오를 임시 파일로 복사합니다. 임시 파일은 클라이언트 환경 변수 DB2VIDEOTEMP에 지정된 디렉토리에 위치됩니다. 비디오 열기는 컷 검출용 임시 파일을 초기화합니다. Video Extender는 비디오 시작 포인터, 즉 프레임 0을 설정합니다.

양쪽의 API를 사용할 때, 비디오 데이터 구조(DBvIOType)에 대한 포인터를 포함하는 데 사용되는 영역을 지시해야 합니다. Video Extender는 이 구조를 API 호출에 대한 응답으로 할당하고 비디오에 대한 정보를 저장하는 데 사용합니다. 구조는 또한 현재 프레임의 픽셀 내용이 있는 프레임 데이터 구조(DBvFrameData)를 가리킵니다. 이 구조에 대한 자세한 설명은 192 페이지의 『컷 검출 데이터 구조』를 참조하십시오. DBvOpenFile API에 대해 비디오 파일 이름도 지정해야 합니다. DBvOpenHandle API에 대해 비디오 핸들도 지정해야 합니다.

예를 들어, 다음의 명령문은 파일에 저장된 컷 검출용 비디오를 엽니다.

```
DBvIOType    *videoptr;

rc=DBvOpenFile (
       &videoptr,                          /* pointer to video structure pointer */
       "/employee/video/rsmith.mpg");      /* video file */
```

다음 명령문은 데이터베이스 테이블에 저장된 컷 검출용 비디오를 엽니다.

```
EXEC SQL BEGIN DECLARE SECTION;
char Vid_hdl[251];
EXEC SQL END DECLARE SECTION;

DBvIOType    *videoptr;

EXEC SQL SELECT VIDEO INTO :Vid_hdl
FROM EMPLOYEE
WHERE NAME="Anita Jones";

rc=DBvOpenHandle(
       &videoptr,                          /* pointer to video structure pointer */
       vid_hdl);                           /* video handle*/
```

비디오 색인

Video Extender는 비디오에 있는 프레임과 컷을 액세스하기 위해 색인을 이용합니다. 비디오에서 컷이나 프레임을 확보하기 전에 비디오용 색인을 작성해야 합니다(MPEG 형식은 프레임과 컷용 색인을 제공하지 않습니다.). 색인은 MPEG-1 비디오를 포함하는 비트 스트림에 프레임 수를 맵핑합니다.

DBvCreateIndexFromVideo API 또는 DBvCreateIndex API를 사용하여 비디오의 색인을 작성할 수 있습니다. 단, DBvOpenFile API 또는 DBvOpenHandle API를 사용하여 컷 검출용 비디오를 연 경우에는 명시적으로 색인을 작성하지 않아도 됩니다. Video Extender가 자동으로 색인을 작성합니다 (비디오를 여는 방법에 대해서는 198 페이지의 『컷 검출용 비디오 열기』를 참조하십시오.).

색인이 작성되면(명시적으로나 자동으로), DB2 Video Extender는 비디오 파일과 동일한 경로에 색인을 저장하려고 합니다. DB2 Video Extender는 먼저 fname.ext.idx로 색인 파일을 저장하려고 하며, 여기서 fname은 비디오 파일의 이름이고 ext는 비디오 파일의 확장자입니다. 이러한 시도에 실패하면, Video Extender는 비디오 파일과 동일한 디렉토리에 fname.idx로 파일을 저장하려고 합니다. 이것이 실패하면, 국지 디렉토리에 먼저 fname.ext.idx로 색인 파일을 저장하려고 한 후 fname.idx로 색인 파일을 저장하려고 합니다.

파일을 열 때, Video Extender는 다음의 순서로 색인 파일을 찾습니다.

1. 쓰기 가능 버전의 색인 파일을 찾은 후 읽기 전용 버전을 찾습니다.
2. 비디오 파일과 동일한 경로에 있는 색인 파일을 찾은 후 현재 디렉토리에 있는 색인 파일을 찾습니다.
3. 이름이 fname.ext.idx인 색인을 찾은 후 이름이 fname.idx인 색인을 찾습니다. 여기서 fname은 비디오 파일의 이름이고, ext는 비디오 파일의 확장자입니다.

예를 들어, 이름이 myvideo.mpg인 비디오 파일에 대한 색인이 작성된 경우, Video Extender는 먼저 이 비디오 파일과 동일한 경로에 있는 이름이 myvideo.mpg.idx인 쓰기 가능 색인을 찾습니다.

DBvCreateIndexFromVideo API 사용시, DBvIOType 데이터 구조를 지정하십시오. Video Extender는 구조에 색인 파일 이름을 저장합니다. 이 구조에 대한 설

명은 192 페이지의 『컷 검출 데이터 구조』를 참조하십시오. 예를 들어, 다음의 명령문은 컷 검출용으로 미리 열린 비디오용 색인을 작성합니다.

```
DBvIOType    *video;

rc=DBvCreateIndexFromVideo(
       video);                  /* pointer to video structure */
```

DBvCreateIndex API 사용시, 비디오 파일 이름을 지정하십시오. Video Extender는 파일(비디오가 있는 동일 디렉토리에 있는)에 색인을 저장합니다. 예를 들어, 다음의 명령문은 비디오 파일용 색인을 작성합니다.(파일은 컷 검출용으로 미리 열리지 않았습니다.)

```
rc=DBvCreateIndex(
       "/employee/video/rsmith.mpg");    /* video file */
```

색인이 비디오용으로 존재하는지 판별할 수도 있습니다. DBvIsIndex API를 사용하여 색인을 점검하십시오. API는 색인이 비디오용으로 존재하지 않는다면 상태 변수값을 0으로, 색인이 비디오용으로 존재한다면 1로 설정합니다. 예를 들어, 다음의 명령문은 비디오 파일용 색인의 존재를 점검합니다.

```
short  *status

rc=DBvIsIndex(
       "/employee/video/rsmith.mpg",     /* video file */
       &status);                         /* status indicator */
```

비디오 색인을 백업하십시오. 비디오 색인을 복구해야 하는 경우 비디오 색인 파일을 백업하십시오. 파일은 Video Extender가 설치된 디렉토리에 위치합니다.

프레임 확보

비디오에 있는 현재 프레임을 확보할 수 있습니다. 현재 프레임을 특정 프레임 번호가 되도록 설정하기도 합니다. DBvGetFrame API를 사용하여 비디오에 있는 현재 프레임을 확보하십시오. DBvSetFrameNumber API를 사용하여 현재 프레임을 특정 프레임 번호로 설정하십시오.

DBvGetFrame API를 사용하여 비디오 구조를 설정하십시오. 예를 들어, 다음의 명령문은 비디오에서 현재 프레임을 확보합니다.

```
DBvIOType    *video;

rc=DBvGetFrame(
       video);                  /* pointer to video structure */
```

DBvSetFrameNumber API 사용시 비디오 구조와 현재 프레임으로 설정하려는 프레임 번호를 지정하십시오. 예를 들어, 다음의 명령문은 프레임 번호 85로 현재 프레임을 설정하고 그 프레임을 확보합니다.

```
DBvIOType    *video;

rc=DBvSetFrameNumber(
       video,                   /* pointer to video structure */
       85);                     /* frame number */

rc=DBvGetFrame(
       video);                  /* pointer to video structure */
```

출력에서 DBvSetFrameNumber API는 DBvIOType 구조에 currentFrame 필드를 재설정합니다. DBvGetFrame API는 프레임의 픽셀 내용을 DBvFrameData 구조에 위치시킵니다. 이러한 구조에 대한 설명은 192 페이지의 『컷 검출 데이터 구조』를 참조하십시오.

컷 확보

비디오에서 다음 컷을 확보하려면 DBvDetectShot API를 사용하십시오. DBvDetectShot API 사용시 다음 데이터 구조를 가리켜야 합니다.

- 비디오(DBvIOType)
- 컷 제어(DBvShotControl)
- 컷 유형(DBvShotType)

또한 탐색용 시작 프레임을 가리켜야 합니다. Video Extender는 비디오의 그 지점에서 다음 컷에 대해 탐색할 것입니다.

API의 결과로, Video Extender는 shotDetected 플래그를 설정하고 다음 컷의 시작 프레임과 프레임 데이터를 가리킵니다. shotDetected 플래그가 1로 설정된다면 컷이 검출됩니다. 이러한 경우, Video Extender는

- DBvIOType에 있는 currentFrame 필드를 다음 컷의 시작 프레임으로 설정합니다.

- DBvIOType에 있는 fd 필드에 다음 컷의 시작 프레임에 대한 데이터를 위치시킵니다.
- 시작 프레임 번호, 끝 프레임 번호, 대표 프레임 번호, 대표 프레임 데이터 및 다음 컷의 주석 등을 포함하도록 DBvShotType을 설정합니다.

shotDetected 플래그가 0으로 설정된다면 컷은 검출되지 않습니다. 이 경우, Video Extender는 비디오 끝에 도달하였음을 나타내는 코드를 리턴합니다.

이러한 구조에 대한 설명은 192 페이지의 『컷 검출 데이터 구조』를 참조하십시오.

예를 들어, 다음의 명령문은 비디오의 다음 컷을 요청합니다.

```
DBvIOType    *video;
long start_frame = 1;
char shotDetected = 0;
DBvShotControl    shotCtrl;
DBvShotType    shot;

shotCtrl→method=DETECT_CORRHIST
shotCtrl→normalCorrValue=60;
shotCtrl→sceneCutSkipXY=1;
shotCtrl→CorrHistThresh=10;
shotCtrl→DissThresh=10;
shotCtrl→DissCacheSize=4;
shotCtrl→DissNumCaches=7;
shotCtrl→minShotSize=0;

rc=DBvDetectShot(
       video,              /* pointer to video structure */
       start_frame,        /* starting frame for search */
       &shotDetected,      /* shot detected flag */
                           /* 1=detected, 0=not detected */
       shotCtrl,           /* pointer to shot control structure */
       &shot);             /* pointer to shot type structure */
```

검색된 프레임의 형식 변환

MPEG-1 프레임 내용은 YUV 형식(프레임의 발광 픽셀 플레인, Cr 픽셀 플레인 및 Cb 픽셀 플레인 정보를 포함하는 형식)입니다. 비디오 프레임을 편집하려 한다면, YUV에서 RGB 형식으로 프레임을 변환하는 것이 간편하다는 것을 알 수 있을 것입니다. Video Extender는 YUV 형식에서 24 비트 RGB 형식으로 검색된 MPEG-1 프레임을 변환하는 DBvFrameDatato24BitRGB API를 제공합니다. API를 사용하려면, 우선 목표 버퍼를 할당해야 합니다.

API 실행시, 목표 버퍼와 변환하려는 프레임 데이터를 가리켜야 합니다. 프레임 폭과 높이도 지정해야 합니다.(프레임용 DBvIOType 구조에서 프레임 데이터, 높이 및 폭을 확보할 수 있습니다.) 예를 들어, 다음의 명령문은 MPEG-1 프레임을 24비트 RGB 형식으로 변환합니다.

```
char RGB[18000];
DBvIOType      *video;
DBvFrameData   fd;

rc=DBvGetNextFrame(
       video);                  /* pointer to video structure */

fd=video.fd
dx=video.dx
dy=video.dy

rc=DBvFrameDataTo24BitRGB (
       RGB,                      /* pointer to target buffer */
       &fd,                      /* pointer to frame data */
       dx,                       /* frame width */
       dy);                      /* frame height */
```

비디오 파일 닫기

컷 검출에 대해 열려 있는 비디오 파일을 닫으려면, DBvClose API를 사용하십시오. API 사용시, 파일용 비디오 구조에 포인터를 지정합니다.

예를 들어, 다음의 명령문은 컷 검출에 대해 열려 있는 비디오 파일을 닫습니다.

```
DBvIOType    *video;

rc=DBvClose (video);
```

검색된 프레임 표시

검색된 MPEG-1 프레임 내용은 YUV 형식입니다. 이것은 대부분의 이미지 표시 프로그램에 의해 표시되는 형식이 아닙니다. 검색된 비디오 프레임을 표시하려면, BMP 형식과 같이 이미지 표시 프로그램이 이해할 수 있는 형식으로 프레임을 두어야 합니다. 예를 들어, MPEG-1 프레임을 표시하려면,

1. 검색된 MPEG-1 프레임을 YUV 형식에서 24비트 RGB 형식으로 변환하는 DBvFrameDatato24BitRGB API를 사용하십시오. DBvFrameDatato24BitRGB API 사용 정보는 202 페이지의 『검색된 프레임의 형식 변환』을 참조하십시오.

2. 변환된 프레임에 적절한 헤더를 추가하십시오. 예를 들어, BMP 형식은 이미지 폭과 높이와 같은 정보를 포함하는 헤더가 필요합니다.
3. 프레임 내용을 헤더와 함께 파일에 복사하십시오.
4. 파일을 표시하려면, DBiBrowse API를 사용하십시오. DBiBrowse API 정보는 139 페이지의 『표시 또는 재생 API 사용』을 참조하십시오.

컷 카탈로그화

컷 카탈로그에 컷에 대한 정보를 저장할 수 있습니다. Video Extender는 다음을 수행하는 API를 제공합니다.

- 데이터베이스에서 컷 카탈로그의 작성과 관리. 다음과 같은 경우 API를 사용할 수 있습니다.
  - 데이터베이스에 컷 카탈로그 작성.
  - 단일 컷에 대한 정보를 컷 카탈로그에 저장.
  - 비디오에 있는 모든 컷에 대한 정보를 컷 카탈로그에 저장.
  - 컷 카탈로그의 컷에 대해 저장된 정보 변경.
  - 컷 카탈로그에 컷 정보 병합.
  - 컷 카탈로그에서 컷 정보 삭제.
  - 데이터베이스에서 컷 카탈로그 삭제.
- 카탈로그 파일의 작성과 비디오에 있는 모든 컷에 대한 정보 저장. API는 카탈로그 파일을 작성하고 컷 데이터로 그것을 채우기 위해 제공됩니다. 컷 카탈로그 파일의 데이터를 액세스하고 조작할 수 있으나, 그것을 수행하기 위해 어떠한 API도 제공되지 않습니다.

카탈로그화된 컷은 스토리보드용 입력을 제공합니다. 컷 카탈로그(카탈로그는 데이터베이스나 파일에 있음)에 컷 정보를 저장한 후, 컷 관련 응용프로그램에 이 정보를 사용할 수 있습니다. 예를 들어, 비디오에 있는 모든 컷용 대표 프레임을 확보하고 이것을 스토리보드에 표시하는 응용프로그램을 작성할 수 있습니다.

데이터베이스용으로 컷 카탈로그를 작성하기만 하면 됩니다. 카탈로그가 데이터베이스에 있기를 원하는 경우에만, 컷 카탈로그를 작성해야 합니다. Video Extender는 비디오의 컷에 대한 데이터를 저장하고 파일로의 출력 의사를 나타낼 때, 컷 카탈로그 파일을 자동으로 작성합니다.

카탈로그 작성 전(데이터베이스 전용)

데이터베이스에 카탈로그를 작성하고 사용하기 전에 다음을 수행해야 합니다.

- SQLConnect 호출을 실행하십시오. 테이블 집합으로 구성된 DB2 데이터베이스의 컷 카탈로그. 데이터베이스에서 컷 카탈로그를 작성하거나 그것을 조작하기 전에, SQLConnect 호출로 데이터베이스에 연결해야 합니다.(SQLConnect는 DB2 콜 레벨 인터페이스 호출입니다.) 호출은 컷 카탈로그를 관리하는 API에 지정되어야 하는 연결 핸들을 리턴합니다.
- 이미지 데이터에 대해 데이터베이스 사용을 가능하게 하십시오. 데이터베이스에서 컷 카탈로그를 작성하기 전에 DB2Image 데이터 유형에 대해 데이터베이스 사용을 가능하게 해야 합니다. 이것이 컷 카탈로그에 저장하는 다른 정보 외에, Video Extender는 각각의 카탈로그화된 컷에 대한 대표 프레임을 저장합니다. 대표 프레임의 데이터 유형은 DB2Image입니다.

컷 카탈로그 작성(데이터베이스 전용)

데이터베이스에서 컷 카탈로그를 작성하려면 DBvCreateShotCatalog API를 사용하십시오. (Video Extender는 컷에 대한 데이터 저장과 파일 출력을 원할 때, 컷 카탈로그 파일을 자동으로 작성합니다.) 카탈로그는 컷 관련 정보를 저장하는 테이블로 구성됩니다. SQL을 사용하여 테이블 뷰를 조회할 수 있습니다. 표13은 뷰의 컬럼을 나타냅니다.

표 13. 컷 카탈로그 뷰에 있는 컬럼

컬럼 이름	데이터 유형	설명
SHOTHANDLE	CHAR(36)	컷 핸들
VIDEOHANDLE	VARCHAR(254)	비디오 핸들. 이 컬럼은 비디오가 DBvOpenHandle API로 열려 있는 경우에만 값을 포함합니다.

표 13. 컷 카탈로그 뷰에 있는 컬럼 (계속)

컬럼 이름	데이터 유형	설명
VIDEOTABLE	VARCHAR(254)	비디오가 있는 테이블. 이 컬럼은 비디오가 DBvOpenHandle API로 열려 있는 경우에만 값을 포함합니다.
VIDEOCOLUMN	VARCHAR(254)	비디오가 있는 테이블 컬럼. 이 컬럼은 비디오가 DBvOpenHandle API로 열려 있는 경우에만 값을 포함합니다.
VIDEOFILE	VARCHAR(254)	비디오 파일 이름. 이 컬럼은 비디오가 DBvOpenFile API로 열려 있는 경우에만 값을 포함합니다.
STARTFRAME	INTEGER	시작 프레임 번호
ENDFRAME	INTEGER	끝 프레임 번호
REPFRAME	INTEGER	대표 프레임 번호
REPFRAMEDATA	DB2IMAGE	대표 프레임 데이터
COMMENTS	LONG VARCHAR	주석

데이터베이스에서 작성한 카탈로그의 컷 카탈로그 수에 있어, 그리고 각 컷 카탈로그에 정보를 저장한 컷 종류에 대해 융통성을 갖습니다. 많은 비디오용 컷 정보를 저장하는 하나의 카탈로그를 작성하거나, 분리 카탈로그에 저장된 각각의 비디오용 컷 정보를 갖거나 여러 카탈로그에 비디오 내의 여러 컷에 대한 정보를 저장할 수 있습니다.

API 사용시, 카탈로그 이름을 지정해야 합니다. 16 문자를 넘는 이름은 절단됩니다. 또한 데이터베이스에 대한 SQLConnect 호출에서 리턴한 데이터베이스 연결 핸들을 지정해야 합니다. 예를 들어, 다음의 명령문은 hotshots이라는 컷 카탈로그를 작성합니다.

```
SQLHDBC  hdbc;

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvCreateShotCatalog(
                "hotshots",          /* shot catalog name */
                hdbc);               /* database connection handle */
```

컷 카탈로그 뷰는 이름 지정된 MMDBSYS.SVcatname이고, 여기서 catname는 컷 카탈로그 이름입니다. 예를 들어, hotshots이라는 카탈로그의 뷰는 MMDBSYS.SVHOTSHOTS입니다.

단일 컷에 대한 정보 저장(데이터베이스)

컷 카탈로그에 단일 컷 정보를 저장하려면, DBvInsertShot API를 사용하십시오. 컷 카탈로그가 데이터베이스에 있는 경우에만, 비디오에 단일 컷 정보를 저장할 수 있습니다. 카탈로그에 저장된 정보는 다음을 포함합니다.

- 컷 핸들
- 비디오 테이블 이름(테이블에 저장된 비디오 클립용)
- 비디오 컬럼 이름(테이블에 저장된 비디오 클립용)
- 비디오 핸들(테이블에 저장된 비디오 클립용)
- 비디오 파일 이름(파일에 저장된 비디오 클립용)
- 시작 프레임 번호
- 끝 프레임 번호
- 대표 프레임 번호
- 대표 프레임 데이터

그러나 주석은 컷에 대해 저장되지 않습니다. 컷에 대해 저장된 정보에 주석을 추가하는 방법에 관한 설명은 212 페이지의 『컷에 대한 주석 지정(데이터베이스 전용)』을 참조하십시오.

DBvInsertShot API 사용시, 컷 카탈로그 이름과 컷에 포인터를 지정해야 합니다. 컷에 포인터를 설정하는 한 가지 방법은 201 페이지의 『컷 확보』에 설명된 것처럼 다음 컷을 확보하는 것입니다. 또한 데이터베이스에 대한 SQLConnect 호출에서 리턴하는 데이터베이스 연결 핸들을 지정해야 합니다. 예를 들어, 다음의 명령문은 프레임 1 다음 컷을 확보하여 hotshots라는 컷 카탈로그에 컷에 관한 정보를 저장합니다.

```
SQLHDBC  hdbc;
SQLHENV  henv;
DBvIOType    *video;
long start_frame = 1;
char shotDetected = 0;
DBvShotControl    shotCtrl;
```

```
DBvShotType    shot;

shotCtrl→method=DETECT_CORRHIST
shotCtrl→normalCorrValue=60;
shotCtrl→sceneCutSkipXY=1;
shotCtrl→CorrHistThresh=10;
shotCtrl→DissThresh=10;
shotCtrl→DissCacheSize=4;
shotCtrl→DissNumCaches=7;
shotCtrl→minShotSize=0;

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvDetectShot(
       video,
       start_frame,
       &shotDetected,
       &shotCtrl,
       &shot)

rc=DBvInsertShot (
       "hotshots",               /*shot catalog name*/
       shot,                     /*pointer to shot*/
       hdbc);                    /*database connection handle*/
```

비디오에 있는 모든 컷에 대한 정보 저장

비디오에 있는 모든 컷 정보를 컷 카탈로그 정보에 저장하려면, DBvBuildStoryboardTable API나 DBvBuildStoryboardFile API를 사용하십시오. DBvBuildStoryboardTable API는 데이터베이스에 상주하는 컷 카탈로그에 정보를 저장합니다. DBvBuildStoryboardFile API는 컷 카탈로그 파일을 작성하고 파일에 컷 정보를 저장합니다.

양쪽 API에 대해, 소스 비디오는 데이터베이스나 파일에 있을 수 있습니다.

API 사용시, 다음을 수행해야 합니다.

- 컷 카탈로그 이름을 지정하십시오.
- 비디오 구조를 가리키십시오.
- DBvShotControl 데이터 구조를 가리키십시오.

- DBvStoryboardCtrl 데이터 구조를 가리키십시오. 이 데이터 구조에 있는 값은 컷 카탈로그에 대표 프레임으로서 저장될 비디오 프레임 수와 종류를 제어합니다. 이 값의 설정 정보는 210 페이지의 『스토리보드 빌드』를 참조하십시오.

DBvBuildStoryboardTable API 전용의 경우, 데이터베이스에 대한 SQLConnect 호출이 리턴하는 데이터베이스 연결 핸들을 지정해야 합니다.

예를 들어, 다음의 명령문은 컷 카탈로그에 비디오의 모든 컷에 대한 정보를 저장합니다. 컷 카탈로그는 데이터베이스에 있습니다.

```
SQLHDBC  hdbc;
SQLHENV  henv;
DBvIOType    *video;
DBvShotControl    shotCtrl;
DBvStoryBoardCtrl    sbCtrl;

sbCtrl→thresh1=50
sbCtrl→thresh2=500;
sbCtrl→delta=20;

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvBuildStoryboardTable (
       "hotshots",               /*shot catalog name*/
       video,                    /*pointer to video structure*/
       shotCtrl,                 /*pointer to shot control structure*/
       sbctrl,                   /*pointer to storyboard control structure*/
       hdbc);                    /*database connection handle*/
```

다음 명령문은 컷 카탈로그 파일을 작성하고 파일에 비디오에 있는 모든 컷 파일에 대한 정보를 저장합니다.

```
DBvIOType    *video;
DBvShotControl    shotCtrl;
DBvStoryBoardCtrl    sbCtrl;

sbCtrl→thresh1=50
sbCtrl→thresh2=500;
sbCtrl→delta=20;

rc=DBvBuildStoryboardFile (
       "hotshots",               /*shot catalog file name*/
       video,                    /*pointer to video structure*/
       shotCtrl,                 /*pointer to shot control structure*/
       sbctrl);                  /*pointer to storyboard control structure*/
```

스토리보드 빌드

그 이름이 암시하듯이, DBvBuildStoryboardTable API와 DBvBuildStoryboardFile API는 스토리보드에 사용되는 정보 저장에 특히 유용합니다. **스토리보드**는 비디오의 시각적 요약입니다. 컷 카탈로그에 비디오용으로 저장된 대표 프레임을 표시하여 스토리보드를 작성할 수 있습니다.

DBvBuildStoryboardTable API와 DBvBuildStoryboardFile API는 하나 이상의 컷용 대표 프레임을 저장합니다. DBvStoryboardCtrl 구조에 지정하는 값이 컷에 저장되는 대표 프레임의 수와 사용할 프레임을 제어합니다. DBvStoryboardCtrl 구조 정의에 대해서는 192 페이지의 『컷 검출 데이터 구조』를 참조하십시오. 그림 29는 DBvStoryboardCtrl 필드에서의 값 사용법을 설명합니다.

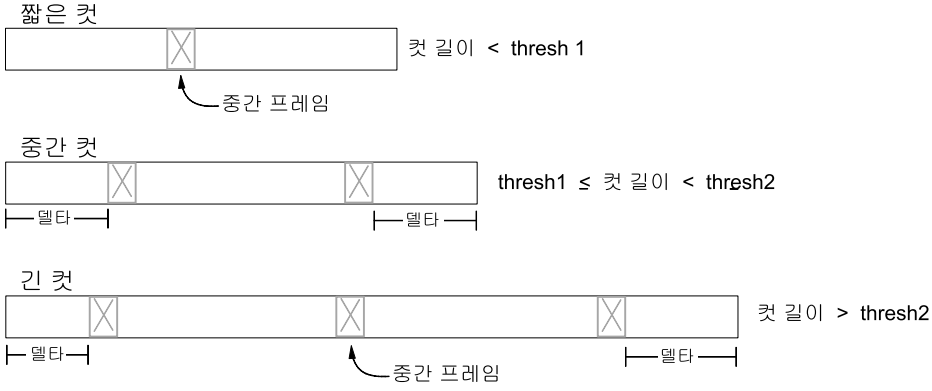


그림 29. DBvStoryboardCtrl 구조에서의 값 사용법

그림29에서는 다음을 설명합니다.

- 하나의 대표 프레임만이 짧은 컷용으로 저장됩니다. 짧은 컷의 프레임 수는 DBvStoryboardCtrl 데이터 구조에 있는 thresh1 미만입니다. 대표 프레임은 컷의 중간 프레임입니다.
- 두 개의 대표 프레임은 중간 컷용으로 저장됩니다. 중간 컷의 프레임 수는 DBvStoryboardCtrl 데이터 구조에서 thresh1 값 이상이고, thresh2 이하입니다. DBvStoryboardCtrl 데이터 구조에 있는 델타 값은 비디오 시작에서 처음 대표 프레임까지 프레임 수를 판별합니다. 델타 값은 또한, 비디오 끝에서 두 번째 대표 프레임까지의 프레임 수를 판별합니다.

- 세 개의 대표 프레임은 긴 컷용으로 저장됩니다. 긴 컷의 프레임 수는 DBvStoryboardCtrl 데이터 구조에 있는 thresh2 값을 넘습니다. DBvStoryBoardCtrl 데이터 구조에 있는 델타값은 비디오 시작에서 처음 대표 프레임까지 프레임 수를 판별합니다. 두 번째 대표 프레임은 비디오의 중간 프레임입니다. 비디오 끝에서 세 번째 대표 프레임까지의 거리는 델타값에 의해 판별됩니다.

임의의 컷은 thresh1이나 thresh2 값이 -1로 설정되는 경우, 짧은 컷으로 처리될 수 있습니다. 이런 경우, 하나의 대표 프레임 즉 중간 프레임만이 컷 카탈로그에 컷용으로 저장될 것입니다.

DBvStoryboardCtrl 데이터 구조에 있는 값 이외에 DBvShotControl 데이터 구조의 필드는 어떤 대표 프레임이 스토리보드에 후속 표시를 위해 저장되는지에 영향을 줍니다. 예를 들어, DBvShotControl 데이터 구조에 있는 CorrHistThresh, normalcorrValue 및 minShotSize 필드는 컷 검출용 임계값을 지정하여 비디오 스토리보드에 표시될 프레임 종류에 영향을 줍니다. 스토리보드 사용 컷 정보를 저장하는 DBvBuildStoryBoardTable API 및 DBvBuildStoryBoardFile API의 사용시, 사용자는 우선 DBvStoryBoardCtrl과 DBvShotControl 데이터 구조에 대한 초기 설정을 사용하여 시범 수행을 할 지 모릅니다. 이 때, 이 데이터 구조의 다양한 필드에서 값을 변경하여 결과를 조정할 수 있습니다.

스토리보드 표시

스토리보드를 표시하는 프로그램을 작성할 수 있습니다. 비디오용 컷 카탈로그에 저장된 대표 프레임을 액세스하여 이를 수행합니다. DBvBuildStoryboardFile API가 비디오용 컷을 저장하는 데 사용되었다면, 컷 카탈로그 파일은 대표 프레임용 GIF 파일을 가리킵니다. 브라우저나 표시 프로그램을 사용하여 이들 GIF 파일을 표시할 수 있습니다.

DBvBuildStorybBoardTable API가 비디오용 컷을 저장하는 데 쓰였다면, 컷 카탈로그(데이터베이스에 저장된)에는 대표 프레임에 대한 데이터가 있습니다. 컷 카탈로그 뷰에서 대표 프레임 데이터를 액세스할 수 있습니다. (뷰에 대한 설명은 205 페이지의 표13을 참조하십시오.) 대표 프레임 데이터는 YUV 형식입니다. 이것은 대부분의 이미지 표시 프로그램에 의해 표시되는 형식이 아닙니다. 대표 프레임을 표시하려면, 203 페이지의 『검색된 프레임 표시』에 설명된 것처럼 DBvFrameDatato24BitRGB API를 사용하여 프레임 데이터를 변환할 수 있습니다. 그런 다음, 적절한 브라우저와 표시 프로그램을 사용하여 대표 프레임을 표시할 수 있습니다.

스토리보드 샘플 프로그램

SAMPLES 서브디렉토리에는 비디오용 스토리보드의 빌드과 표시를 예증하는 두 개의 샘플 프로그램이 있습니다. makesf.exe 파일에 있는 하나의 샘플 프로그램은 DBvBuildStoryBoardFile API를 사용하여 컷 카탈로그 파일을 작성하고 그 파일에 컷 데이터를 저장합니다. makehtml.exe에 있는 다른 샘플 프로그램은 컷 카탈로그에 액세스하여 웹 브라우저에서 표시할 수 있는 HTML 페이지를 작성합니다.

컷에 대한 주석 지정(데이터베이스 전용)

컷 카탈로그에 컷에 대한 다른 정보로 저장되는 주석을 지정할 수 있습니다. 주석을 지정하려면, DBvSetShotComment API를 사용하십시오.

API 사용시, 주석을 저장하는 컷 카탈로그의 이름, 주석이 추가되는 컷 핸들 및 주석을 지정해야 합니다. 또한 데이터베이스에 대한 SQLConnect 호출에서 리턴하는 데이터베이스 연결 핸들을 지정해야 합니다. 예를 들어, 다음 명령문은 hotsots라는 컷 카탈로그에 컷(프레임 번호 85로 시작하는)에 대한 주석을 추가합니다.

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle[37];

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

EXEC SQL SELECT SHOTHANDLE INTO :shothandle
  FROM MMDBSYS.SVHOTSHOTS
  WHERE STARTFRAME=85;

rc=DBvSetShotComment (
        "hotshots",                     /*shot catalog name*/
        shothandle,                     /*shot handle*/
        "shot of beach at sunset",      /*comment*/
        hdbc);                          /*database connection handle*/
```

컷에 대해 저장되는 정보 변경(데이터베이스 전용)

컷 카탈로그에 컷에 대해 저장된 정보를 변경할 수 있습니다. DBvUpdateShot API를 사용하여 수행할 수 있습니다. DBvShotType 구조에 대체 정보를 두십시오. 설사 정보가 변경되지 않더라도 남아있는 필드에 대한 정보도 지정해야 합니다. DBvUpdateShot API 사용시, 카탈로그 이름과 DBvShotType 구조로의 포인터

를 지정하십시오. 또한 데이터베이스에 대한 SQLConnect 호출에서 리턴하는 데이터베이스 연결 핸들을 지정해야 합니다.

컷에 대한 정보 변경시, 정보와 함께 저장된 주석을 변경할 수 있는 옵션을 갖습니다. 주석을 변경하려면, 그것을 DBvShotType 구조에 지정하십시오. 이전 내용을 유지하려 한다면, DBvShotType 구조에 널(Null) 값을 지정하십시오.

예를 들어, 다음 명령문은 hotshots라는 카탈로그에 컷용으로 저장된 정보를 변경합니다. 컷은 프레임 번호 85에서 시작됩니다.

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle[37];
DBvShotType    shot;
DBvFrameData  fd110;

/* get shot handle */

EXEC SQL SELECT SHOTHANDLE INTO :shothandle
  FROM MMDBSYS.SVHOTSHOTS
  WHERE STARTFRAME=85;

/* change shot attribute */

shot.startFrame=110;
shot.endFrame=200;
shot.repframe=110;
shot.fd=fd110;
shot.comment=NULL;

/* update shot information */

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvUpdateShot (
        "hotshots",                    /*shot catalog name*/
        shot,                          /*shot information*/
        hdbc);                         /*database connection handle*/
```

컷 카탈로그에 컷 정보 병합(데이터베이스 전용)

컷 카탈로그에 두 개의 컷에 대해 저장된 정보를 병합할 수 있습니다. 컷 정보 병합시, 처음 컷과 두 번째 컷을 식별하여 병합 순서를 표시합니다. 처음 컷의 시작

프레임 번호는 병합된 컷의 시작 프레임 번호로 저장됩니다. 처음과 두 번째 컷 간의 최대 프레임 번호는 병합된 컷의 끝 프레임 번호로 저장됩니다. 병합은 병합된 컷에 대한 정보로 처음 컷에 대해 저장된 정보를 대체합니다. 두 번째 컷에 대해 저장된 정보는 컷 카탈로그에서 삭제됩니다.

컷 카탈로그에 두 개의 컷에 대한 정보를 병합하려면, DBvMergeShots API를 사용하십시오. API 사용시, 병합되는 처음과 두 번째 컷 핸들이 따르는 컷 카탈로그 이름을 지정하십시오. 또한 데이터베이스에 대한 SQLConnect 호출에서 리턴하는 데이터베이스 연결 핸들을 지정해야 합니다. 예를 들어, 다음 명령문은 카탈로그 이름 hotshots에 두 개의 컷에 대해 저장된 정보를 병합합니다. 처음 컷은 프레임 85에서 시작하고 두 번째 컷은 프레임 210에서 시작합니다.

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle1[37];
char shothandle2[37];

EXEC SQL SELECT SHOTHANDLE INTO :shothandle1
  FROM MMDBSYS.SVHOTSHOTS1
  WHERE STARTFRAME=85;

EXEC SQL SELECT SHOTHANDLE INTO :shothandle2
  FROM MMDBSYS.SVHOTSHOTS2
  WHERE STARTFRAME=210;

SQLAllocConnect(henv,&hdbc)

rc = SQLConnect(hdbc,"hotshots",SQL_NTS,id,SQL_NTS,password,SQL_NTS);

rc=DBvMergeShots (
        "hotshots",                    /*shot catalog name*/
        shothandle1,                   /*shot handle for first shot*/
        shothandle2,                   /*shot handle for second shot*/
        hdbc);                         /*database connection handle*/
```

컷 카탈로그에서 컷에 대한 정보 삭제(데이터베이스 전용)

컷 카탈로그에서 컷에 대한 정보를 삭제하려면 DBvDeleteShot API를 사용하십시오. API 사용시, 컷 핸들이 따르는 컷 카탈로그 이름을 지정하십시오. 또한 데이터베이스에 대한 SQLConnect 호출에서 리턴하는 데이터베이스 연결 핸들을 지정해야 합니다. 예를 들어, 다음의 명령문은 hotshots라는 컷 카탈로그에서 컷(프레임 번호 85에서 시작하는)에 관한 정보를 삭제합니다.

```
SQLHDBC  hdbc;
SQLHENV  henv;
char shothandle[37];

EXEC SQL SELECT shothandle INTO :shothandle
  FROM mmdbsys.svhotshots
  WHERE startframe=85;

rc=DBvDeleteShot (
        "hotshots",                    /*shot catalog name*/
        shothandle,                    /*shot handle*/
        hdbc);                         /*database connection handle*/
```

컷 카탈로그 삭제(데이터베이스 전용)

컷 카탈로그를 삭제하려면 DBvDeleteShotCatalog API를 사용하십시오. API 사용시, 삭제되는 컷 카탈로그 이름과 데이터베이스에 대한 SQLConnect 호출이 리턴한 데이터베이스 연결 핸들을 지정하십시오. 예를 들어, 다음의 명령문은 hotshots라는 컷 카탈로그를 삭제합니다.

```
SQLHDBC  hdbc;
SQLHENV  henv;

rc=DBvDeleteShotCatalog (
        "hotshots",             /*shot catalog name*/
        hdbc);                  /*database connection handle*/
```

제4부 참조 정보

제15장 사용자 정의 유형 및 사용자 정의 함수

이 장에서는 DB2 Extender에서 작성되는 UDT와 UDF에 대한 참조 정보를 제공합니다.

스키마

Extender는 UDT 및 UDF를 포함한, 모든 오브젝트 관계형 오브젝트에 대해 MMDBSYS 스키마를 사용합니다.

사용자 정의 유형

표14는 DB2 Extender에서 작성되는 사용자 정의 유형을 나열하고 설명한 것입니다. 각각의 UDT에 대한 DB2 소스 데이터 유형도 나열되어 있습니다.

표 14. DB2 Extender에서 작성되는 사용자 정의 유형

UDT	소스 데이터 유형	설명
DB2IMAGE	VARCHAR(250)	이미지 핸들. 이미지 오브젝트의 액세스에 필요한 정보가 있는 가변 길이의 문자열. 이미지 핸들은 Image Extender에 대해 사용이 가능한 사용자 테이블 컬럼에 저장됩니다.
DB2AUDIO	VARCHAR(250)	오디오 핸들. 오디오 오브젝트의 액세스에 필요한 정보가 있는 가변 길이의 문자열. 오디오 핸들은 Audio Extender에 대해 사용이 가능한 사용자 테이블 컬럼에 저장됩니다.
DB2VIDEO	VARCHAR(250)	비디오 핸들. 비디오 오브젝트의 액세스에 필요한 정보가 있는 가변 길이의 문자열. 비디오 핸들은 Video Extender에 대해 사용이 가능한 사용자 테이블 컬럼에 저장됩니다.

사용자 정의 함수

이 절에서는 DB2 Extender에 대한 참조 정보를 제공합니다. UDF는 알파벳 순서로 나열됩니다.

다음 정보는 각각의 UDF에 대해 제공됩니다.

- UDF를 제공하는 Extender
- 간단한 설명
- UDF용 Include(헤더) 파일
- UDF의 SQL 구문
- UDF 매개변수의 데이터 유형을 포함하는 설명
- 데이터 유형을 포함한, UDF가 리턴하는 값
- 사용 예

표15는 UDF를 나열하고 각각의 UDF를 제공하는 Extender를 식별합니다. 표에서는 각 UDF에 대한 상세한 정보를 찾을 수 있는 위치도 나와 있습니다. 표에 있는 UDF는 Embedded SQL이나 DB2 CLI 호출에서 코딩이 가능합니다.

표 15. DB2 Extender UDF

UDF	설명	Image	Audio	Video	참조 페이지
AlignValue	WAVE 오디오나 비디오의 오디오 트랙에서 샘플당 바이트 수를 리턴합니다.		x	x	224
AspectRatio	MPEG1 및 MPEG2 비디오 첫번째 트랙의 영상비를 리턴합니다.			x	226
BitsPerSample	WAVE나 오디오에 있는 AIFF 오디오 또는 비디오의 오디오 트랙에서 각각의 샘플을 표현하되는 데 사용하는 데이터의 비트 수를 리턴합니다.		x	x	227
BytesPerSec	WAVE 오디오에 대해 초당 평균 바이트 수로 데이터 전송율을 리턴합니다.		x		228
Comment	이미지나 오디오 또는 비디오와 함께 저장된 주석을 리턴하거나 갱신합니다.	x	x	x	229

표 15. DB2 Extender UDF (계속)

UDF	설명	Image	Audio	Video	참조 페이지
CompressType	MPEG-1과 같은 비디오의 압축 형식을 리턴합니다.			x	231
Content	이미지나 오디오 또는 비디오의 내용을 데이터베이스에서 검색하거나 갱신합니다.	x	x	x	232
DB2Audio	데이터베이스 테이블에 오디오 내용을 저장합니다.		x		238
DB2Image	데이터베이스 테이블에 이미지 내용을 저장합니다.	x			242
DB2Video	데이터베이스 테이블에 비디오 내용을 저장합니다.			x	248
Duration	WAVE나 AIFF 오디오 또는 비디오의 존속 기간(즉, 초당 재생 시간)을 리턴합니다.		x	x	253
Filename	이미지나 오디오 또는 비디오 내용이 있는 서버 파일 이름을 리턴합니다.	x	x	x	254
FindInstrument	MIDI 오디오에 있는 지정 악기의 처음 발생 트랙 번호를 리턴합니다.		x		255
FindTrackName	MIDI 오디오에서 지정 이름 지정된 트랙 번호를 리턴합니다.		x		256
Format	이미지나 오디오 또는 비디오 형식을 리턴합니다.	x	x	x	257
FrameRate	초당 프레임으로 비디오 처리량을 리턴합니다.			x	258
GetInstruments	MIDI 오디오에 있는 악기 이름 모두를 리턴합니다.		x		259
GetTrackNames	MIDI 오디오에 있는 트랙 이름 모두를 리턴합니다.		x		260
Height	픽셀 단위로 이미지나 비디오 프레임의 높이를 리턴합니다.	x		x	261
Importer	데이터베이스 테이블에 이미지, 오디오 또는 비디오를 저장한 사람의 사용자 ID를 리턴합니다.	x	x	x	262

표 15. DB2 Extender UDF (계속)

UDF	설명	Image	Audio	Video	참조 페이지
ImportTime	데이터베이스 테이블에 이미지나 오디오 또는 비디오 저장 시간을 나타내는 시간소인을 리턴합니다.	x	x	x	263
MaxBytesPerSec	초당 바이트 수로 비디오의 최대 처리량을 리턴합니다.			x	264
NumAudioTracks	비디오나 MIDI 오디오에서 오디오 트랙의 수를 리턴합니다.		x	x	265
NumChannels	WAVE나 AIFF 오디오 또는 비디오에서 레코드된 오디오의 채널 수를 리턴합니다.		x	x	266
NumColors	이미지에 있는 색상 수를 리턴합니다.	x			267
NumFrames	비디오에 있는 프레임 수를 리턴합니다.			x	268
NumVideoTracks	비디오에 있는 비디오 트랙 수를 리턴합니다.			x	269
QbScoreFromName	(이름 지정된 조회 오브젝트를 사용하여) 이미지 스코어를 리턴합니다. (QbScore를 대체하십시오.)	x			270
QbScoreFromStr	(조회 문자열을 사용하여) 이미지 스코어를 리턴합니다.	x			272
QbScoreTBFromName	(이름 지정된 조회 오브젝트를 사용하여) 이미지 컬럼에서 스코어 테이블을 리턴합니다.	x			274
QbScoreTBFromStr	(조회 문자열을 사용하여) 이미지 컬럼에서 스코어 테이블을 리턴합니다.	x			276
Replace	데이터베이스에 저장된 이미지나 오디오 또는 비디오의 내용을 갱신하고 그것의 주석을 갱신합니다.	x	x	x	278
SamplingRate	WAVE나 AIFF, 또는 비디오에 있는 오디오 트랙의 샘플링을 초당 샘플 수로 리턴합니다.		x	x	282
Size	이미지나 오디오 또는 비디오 크기(바이트 단위)를 리턴합니다.	x	x	x	283

표 15. DB2 Extender UDF (계속)

UDF	설명	Image	Audio	Video	참조 페이지
Thumbnail	데이터베이스에 저장된 이미지나 비디오 프레임의 썸네일 크기 버전을 리턴하거나 갱신합니다.	x		x	284
TicksPerQNote	1/4 노트당 틱 수로 레코드된 MIDI 오디오의 시계 속도를 리턴합니다.		x		286
TicksPerSec	초당 틱 수로 레코드된 MIDI 오디오의 시계 속도를 리턴합니다.		x		287
Updater	데이터베이스 테이블에 이미지나 오디오 또는 비디오를 마지막으로 갱신한 사람의 사용자 ID를 리턴합니다.	x	x	x	288
UpdateTime	데이터베이스 테이블에 이미지나 오디오 또는 비디오가 마지막으로 갱신된 시간을 표시하는 시간소인을 나타냅니다.	x	x	x	289
Width	이미지나 비디오 프레임의 픽셀 단위 폭을 리턴합니다.	x		x	290

AlignValue

Image	Audio	Video
	X	X

WAVE 오디오나 비디오의 오디오 트랙에서 샘플당 바이트 수를 리턴합니다. WAVE는 샘플당 하나의 바이트(『바이트 맞춤』으로 언급된 8비트 모노)나 샘플당 두 개의 바이트(『단어 맞춤』으로 언급된 8비트 스테레오) 또는 샘플당 네 개의 바이트(『두 단어 맞춤』으로 언급된 16비트 스테레오)를 사용하여 그것의 데이터를 저장할 수 있습니다.

Include 파일

오디오 dmbaudio.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2AUDIO 또는 DB2VIDEO)
오디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

WAVE 오디오나 비디오에 있는 오디오 트랙의 샘플 값당 바이트 수(SMALLINT). 가능한 값은 다음과 같습니다.

1 바이트 맞춤

2 단어 맞춤

4 두 단어 맞춤

널(Null) 값 기타 형식의 오디오

예

단어 맞춤된 employee 테이블의 sound 컬럼에 저장된 모든 오디오의 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
       INTO :hvAud_fname
       FROM EMPLOYEE
       WHERE ALIGNVALUE(SOUND) = 2;
```

Anita Jones에 대한 employee 테이블의 video 컬럼에 저장된 비디오에 있는 오디오 트랙의 샘플 값당 바이트 수를 찾으십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
       short hvAlign_val;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT ALIGNVALUE(VIDEO)
       INTO :hvAlign_val
       FROM EMPLOYEE
       WHERE NAME='Anita Jones';
```

AspectRatio

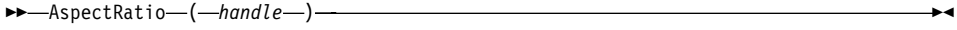
Image	Audio	Video
		X

MPEG 비디오의 첫번째 트랙의 영상비를 리턴합니다.

Include 파일

dmbvideo.h

구문



매개변수(데이터 유형)

handle(DB2VIDEO)

비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

MPEG 비디오의 첫번째 트랙의 영상비나 기타 형식의 비디오에 대해 널(NULL) 값(SMALLINT)

예

employee 테이블의 video 컬럼에 Robert Smith에 대해 저장된 비디오 영상비를 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvAsp_ratio;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT ASPECTRATIO(VIDEO)
      INTO :hvAsp_ratio
      FROM EMPLOYEE
      WHERE NAME='Robert Smith';
```

BitsPerSample

Image	Audio	Video
	X	X

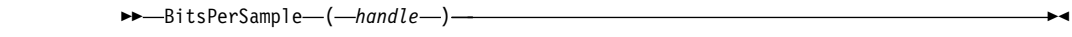
WAVE나 오디오에 있는 AIFF 오디오 또는 비디오의 오디오 트랙에서 각각의 샘플에 표현하는 데 사용되는 데이터의 비트 수를 리턴합니다.

Include 파일

오디오 dmbaudio.h

비디오 dmbvideo.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO 또는 DB2VIDEO)
비디오나 비디오 핸들이 있는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오나 WAVE 또는 AIFF 오디오의 각각의 샘플을 표현하는 데 사용되는 데이터의 비트 수(SMALLINT). 기타 형식의 오디오에 대해 널(NULL) 값을 리턴합니다.

예

샘플당 8비트인 employee 테이블의 sound 컬럼에 저장된 모든 WAVE 오디오의 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE FORMAT(SOUND)='WAVE'
      AND BITSPERSAMPLE(SOUND) = 8;
```

BytesPerSec

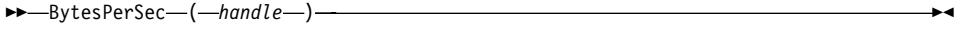
Image	Audio	Video
	X	

WAVE 오디오에 대해 초당 평균 바이트 수로 데이터 전송율을 리턴합니다.

Include 파일

dmbaudio.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO)

오디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

데이터 전송율(INTEGER). 기타 형식의 오디오에 대해 널(NULL) 값을 리턴합니다.

예

초당 평균 바이트 수로 전송율이 44100 미만인 employee 테이블의 sound 컬럼에 저장된 모든 오디오의 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE BYTESPERSEC(SOUND) < 44100;
```

Comment

Image	Audio	Video
X	X	X

이미지, 오디오 또는 비디오와 함께 저장된 주석을 리턴하거나 검색합니다.

Include 파일

이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

구문

주석 검색

구문

주석 갱신

매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)

이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

new_comment(LONG VARCHAR)

갱신할 새 주석. 널(NULL) 값 또는 빈 문자열은 기존 주석을 삭제합니다.

리턴 값(데이터 유형)

갱신의 경우 이미지나 오디오 또는 비디오의 핸들(DB2MAGE, DB2AUDIO 또는 DB2VIDEO). 검색의 경우 주석(LONG VARCHAR).

예

관련 주석에 『confidential』 단어가 있는 employee의 picture 컬럼에서 모든 이미지의 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[255;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE COMMENT(PICTURE)
        LIKE '%confidential%';
```

employee 테이블의 video 컬럼에 있는 Anita Jones의 비디오 클립과 연관된 주석을 갱신하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
struct{
      short len;
      char data[4000];
      }hvRemarks;
EXEC SQL END DECLARE SECTION;

/* Get the old comment */

EXEC SQL SELECT COMMENT(VIDEO)
     INTO :hvRemarks
     FROM EMPLOYEE
     WHERE NAME =  'Anita Jones';

/* Update the comment */

hvRemarks.data[hvRemarks.len]='\0';
strcat (hvRemarks.data, "Updated video");
hvRemarks.len=strlen(hvRemarks.data);

EXEC SQL UPDATE EMPLOYEE
     SET VIDEO=COMMENT(VIDEO, :hvRemarks)
     WHERE NAME =  'Anita Jones';
```

CompressType

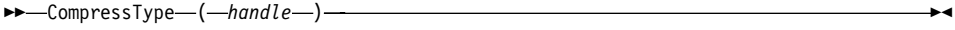
Image	Audio	Video
		X

MPEG-1과 같은 비디오의 압축 형식을 리턴합니다.

Include 파일

dmbvideo.h

구문



매개변수(데이터 유형)

handle(DB2VIDEO)

비디오 핸들이 있는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오의 압축 형식(VARCHAR(8))

예

압축 형식이 MPEG-1인 employee 테이블의 video 컬럼에 저장된 모든 비디오의 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE COMPRESSTYPE(VIDEO) =
'MPEG1';
```

Content

Image	Audio	Video
X	X	X

이미지나 오디오 또는 비디오의 내용을 데이터베이스에서 검색하거나 갱신합니다. 내용을 클라이언트 버퍼, 클라이언트 파일 또는 서버 파일로 검색할 수 있습니다.

Include 파일

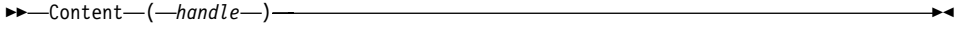
이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

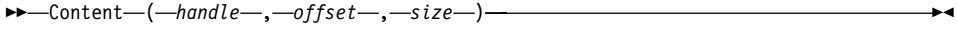
구문

내용을 **버퍼나 클라이언트 파일로 검색**



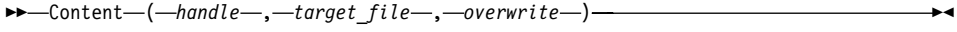
구문

내용의 세그먼트를 버퍼나 클라이언트 파일로 검색



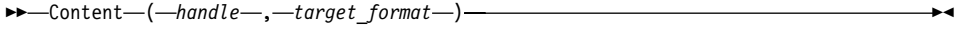
구문

내용을 서버 파일로 검색



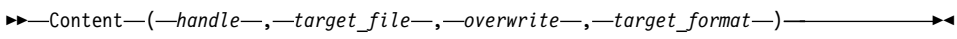
구문

형식 변환을 사용하여 내용을 버퍼나 클라이언트 파일로 검색(이미지 전용)



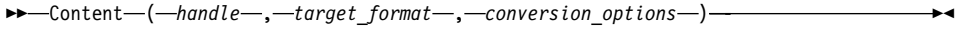
구문

형식 변환을 사용하여 내용을 서버 파일로 검색(이미지 전용)



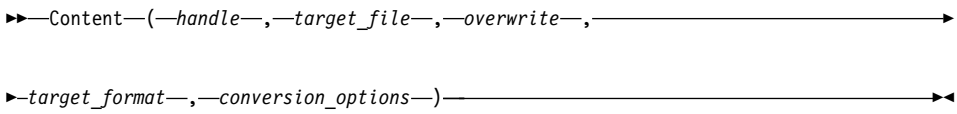
구문

형식 변환 및 추가 변경을 사용하여 내용을 버퍼 또는 클라이언트 파일로 검색(이미지 전용)



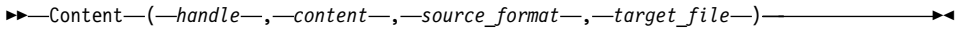
구문

형식 변환 및 추가 변경을 사용하여 서버 파일로 검색(이미지 전용)



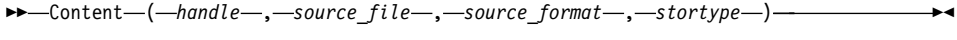
구문

버퍼나 클라이언트 파일로부터 내용 갱신



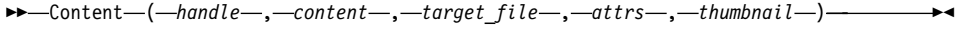
구문

서버 파일로부터 내용 갱신



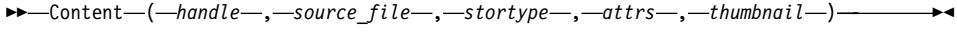
구문

버퍼 또는 클라이언트 파일로부터 사용자 제공 속성으로 내용 갱신



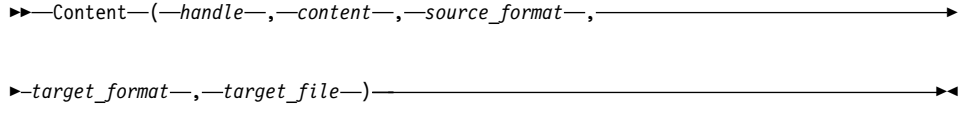
구문

서버 파일로부터 사용자 제공 속성으로 내용 갱신



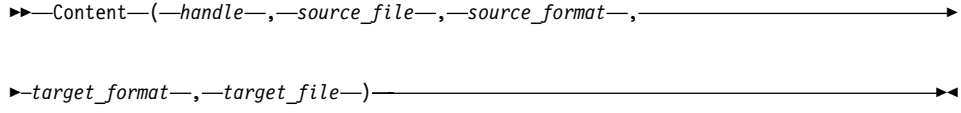
구문

형식 변환을 사용하여 버퍼나 클라이언트 파일로부터 내용 갱신(이미지 전용)



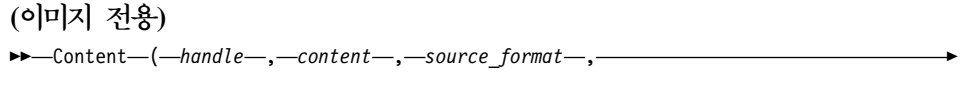
구문

형식 변환을 사용하여 서버 파일로부터 내용을 갱신(이미지 전용)



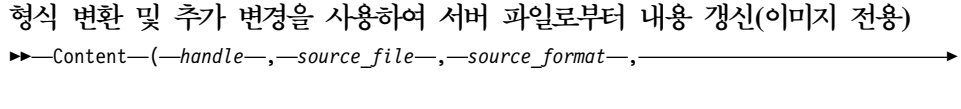
구문

형식 변환 및 추가 변경을 사용하여 버퍼 또는 클라이언트 파일로부터 내용 갱신 (이미지 전용)



구문

형식 변환 및 추가 변경을 사용하여 서버 파일로부터 내용 갱신(이미지 전용)



매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)

이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

offset(INTEGER)

검색될 이미지나 오디오 또는 비디오의 시작 옵셋(원점1).

size(INTEGER)

검색될 이미지나 오디오 또는 비디오의 바이트 수.

source_file (LONG VARCHAR)

이미지나 오디오 또는 비디오 갱신용 내용이 있는 파일 이름.

target_file (LONG VARCHAR)

검색의 경우, 이미지나 오디오 또는 비디오를 검색하는 파일 이름. 갱신의 경우, 갱신될 이미지나 오디오 또는 비디오가 있는 파일 이름.

stortype(INTEGER)

갱신된 이미지나 오디오 또는 비디오가 저장될 장소를 표시하는 변수. 상수 MMDB_STORAGE_TYPE_INTERNAL(값=1)는 갱신되는 오브젝트가 BLOB과 같은 데이터베이스에 BLOB으로 저장될 것임을 나타냅니다. 상수 MMDB_STORAGE_TYPE_EXTERNAL(값=0)는 갱신되는 오브젝트가 서버 파일에 저장될 것임을 나타냅니다.

overwrite(INTEGER)

목표 파일이 존재한다면 그것을 겹쳐쓸 것인지 표시하는 값. 값은 0 또는 1이 될 수 있습니다. 0 값은 목표 파일이 겹쳐쓰이지 않는 것을 의미합니다.(사실상 검색되지 않습니다.) 값 1은 목표 파일이 이미 존재한다면 그것이 겹쳐쓰일 것을 의미합니다.

target_format(VARCHAR(8))

검색 또는 갱신 후의 이미지 형식. 소스 이미지 형식은 적절하게 변환될 것입니다. target_file이 source_file과 동일한 경우, 서버 파일로의 이미지 검색에 대해 목표 형식은 소스 형식과 같아야 합니다. MPG1 형식에 대해 MPG1, mpg1, MPEG1 또는 mpeg1을 지정할 수 있습니다. MPG2 형식에 대해 MPG2, mpg2, MPEG2 또는 mpeg2를 지정할 수 있습니다.

conversion_options(VARCHAR(100))

그것의 검색 또는 갱신시 이미지에 적용되는, 회전이나 압축과 같은 변경사항을 지정합니다. 지원되는 변환 옵션에 대해서는 99 페이지의 표6을 참조하십시오.

content(BLOB(2G) AS LOCATOR)

이미지나 오디오 또는 비디오 갱신용 내용이 있는 호스트 변수. 호스트 변

수는 BLOB, BLOB_FILE 또는 BLOB_LOCATOR 유형이 가능합니다. DB2는 내용의 데이터 유형을 BLOB_LOCATOR로 진행시키고 Content UDF로 위치 지정자를 전달합니다.

source_format(VARCHAR(8))

이미지나 오디오 또는 비디오 갱신용 소스 형식. 널(NULL) 값이나 빈 문자열, 또는 이미지 전용으로 문자열 ASIS가 지정될 수 있습니다. 이 세 가지 경우에 Extender는 형식을 자동으로 결정하려고 할 것입니다. MPG1 형식에 대해 MPG1, mpg1, MPEG1 또는 mpeg1을 지정할 수 있습니다. MPG2 형식에 대해 MPG2, mpg2, MPEG2 또는 mpeg2를 지정할 수 있습니다.

attrs(LONG VARCHAR FOR BIT)

이미지, 오디오 또는 비디오의 속성

thumbnail(LONG VARCHAR FOR BIT DATA)

이미지나 비디오 프레임의 썸네일(이미지와 비디오 전용)

리턴 값(데이터 유형)

버퍼로 검색된 경우, 검색된 이미지, 오디오 또는 비디오의 내용(BLOB(2G) AS LOCATOR). 파일로 검색한 경우, VARCHAR(254).

갱신의 경우, 갱신되는 이미지나 오디오 또는 비디오의 핸들(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO).

예

employee 테이블의 picture 컬럼에 Anita Jones용으로 저장된 이미지를 서버 파일로 검색합니다.

```
struct{
      short len;
      char data[250];
      }hvImg_fname;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT (PICTURE,
       '/employee/images/ajones.bmp',1)
      INTO :hvImg_fname
      FROM EMPLOYEE
      WHERE NAME='Anita Jones';
```

employee 테이블의 sound 컬럼에 Robert Smith를 위해 저장된 1-MB 오디오 클립을 클라이언트 버퍼로 검색하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS BLOB_LOCATOR audio_loc;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT CONTENT (SOUND, 1, 1000000)
      INTO :audio_loc
      FROM EMPLOYEE
      WHERE NAME='Robert Smith';
```

employee 테이블의 picture 컬럼에서 Anita Jones의 이미지를 갱신하십시오. 그것의 원래 크기의 50%로 이미지를 줄이고 BMP에서 GIF로 이미지 형식을 변경합니다.

```
EXEC SQL UPDATE EMPLOYEE
  SET picture = CONTENT(PICTURE,
       '/employee/newimg/ajones.bmp',
       'BMP',
       'GIF',
        '-s 0.5',
       '');
  WHERE NAME='Anita Jones';
```

DB2Audio

Image	Audio	Video
	X	

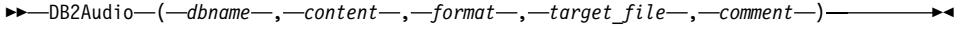
데이터베이스 테이블에 오디오 내용을 저장합니다. 오디오 소스는 클라이언트 버퍼, 클라이언트 파일 또는 서버 파일에 있을 수 있습니다. 이미지는 BLOB으로 데이터베이스 테이블이나, 서버 파일(데이터베이스 테이블이 참조하는)에 저장될 수 있습니다. 오디오 소스는 DB2 Audio Extender가 저장영역에 대한 그것의 속성을 표시하는 경우에는 지원되는 형식으로 있을 수 있고, 또는 속성이 UDF에 지정되어야 하는 경우에 지원되지 않은 형식으로 있을 수 있습니다.

Include 파일

dmbaudio.h

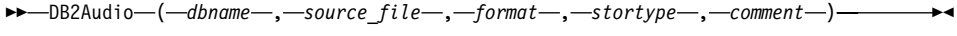
구문

버퍼나 클라이언트 파일로부터 내용 저장



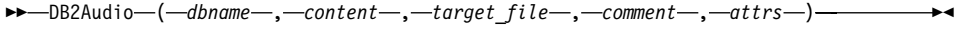
구문

서버 파일로부터 내용 저장



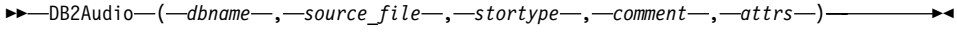
구문

버퍼 또는 클라이언트 파일로부터 사용자 제공 속성으로 내용 저장



구문

서버 파일로부터 사용자 제공 속성으로 내용 저장



매개변수(데이터 유형)

dbname(VARCHAR(18))

CURRENT SERVER 특수 레지스터가 표시하는 것과 같은 현재 연결된 데이터베이스 이름.

content(BLOB(2G) AS LOCATOR)

오디오 내용이 있는 호스트 변수. 호스트 변수는 BLOB, BLOB_FILE 또는 BLOB_LOCATOR 유형이 가능합니다. DB2는 내용의 데이터 유형을 BLOB-LOCATOR로 진행시키고 LOB 위치 지정자를 DB2Audio UDF에 전달합니다.

format(VARCHAR(8))

소스 오디오의 형식. 널(NULL)값이나 빈 문자열이 지정될 수 있으며, 이 경우 Audio Extender가 소스 형식을 자동으로 결정하려 할 것입니다. 오디오는 그것의 소스와 동일 형식으로 저장될 것입니다. 지원되는 오디오 형식에 대해서는 97 페이지의 표5를 참조하십시오.

target_file(LONG VARCHAR)

목표 서버 파일(서버 파일로 저장용) 이름 또는 널(NULL) 값이나 빈 문자열(BLOB으로 데이터베이스 테이블로 저장용). 목표 파일은 완전히 규정된 이름이 될 수 있습니다. 이름이 규정되지 않는다면, 서버에 있는 DB2AUDIOSTORE와 DB2MMSTORE 환경 변수가 파일을 위치 지정하는 데 사용됩니다.

source_file(LONG VARCHAR)

소스 서버 파일 이름. 소스 파일 이름은 완전히 규정된 이름이나 규정되지 않은 이름일 수 있습니다. 널(NULL)값이나 빈 문자열이 될 수는 없습니다. 이름이 규정되지 않는다면, 서버에 있는 DB2AUDIOPATH와 DB2MMPATH 환경 변수가 파일을 위치지정하는 데 사용됩니다.

stortype(INTEGER)

오디오가 저장될 장소를 표시하는 값. 상수 MMDB_STORAGE_TYPE_INTERNAL(값=1)은 오디오가 BLOB으로 데이테베이스에 저장될 것임을 나타냅니다. 상수 MMDB_STORAGE_TYPE_EXTERNAL(값=0)은 오디오 내용이 서버 파일(데이터베이스에서 지시됨)에 저장될 것임을 나타냅니다.

comment(LONG VARCHAR)

오디오와 함께 저장되는 주석

attrs(LONG VARCHAR FOR BIT DATA)

오디오 속성

리턴 값(데이터 유형)

오디오 핸들(DB2AUDIO)

예

Anita Jones의 오디오 클립이 있는 레코드를 employee 테이블에 삽입하십시오. 오디오 소스는 클라이언트 버퍼에 있습니다. BLOB으로 테이블에 오디오 클립을 저장하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
     SQL TYPE IS BLOB (5M) aud_seg;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2AUDIO(
         CURRENT SERVER,
         :aud_seg,
         'WAVE',
         CAST(NULL as LONG VARCHAR),

         'Anita''s voice'));
```

Robert Smith의 오디오 클립이 있는 레코드를 employee 테이블에 삽입하십시오. 오디오 소스는 서버 파일에 있습니다. employee 테이블 레코드는 파일을 지시할 것입니다.

```
EXEC SQL BEGIN DECLARE SECTION;
     long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType = MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '384779',
      'Robert Smith',
      DB2AUDIO(
```

```
        CURRENT SERVER,
        '/employee/sounds/rsmith.wav',
        'WAV',
        :hvStorageType,
        'Robert''s voice'));
```

Anita Jones의 오디오 클립이 있는 레코드를 employee 테이블에 삽입하십시오. BLOB으로 오디오 클립을 저장합니다. 서버 파일에 있는 소스 오디오 클립은 사용자 정의 형식으로 44.1KHz 샘플링율의 두 가지 레코드된 채널이 있습니다.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[600];
}hvAudattr;
EXEC SQL END DECLARE SECTION;

MMDBAudioAttrs          *paudiattr;

hvStorageType = MMDB_STORAGE_TYPE_INTERNAL;

paudioattr=(MMDBAudioAttrs *) hvAudattr.data;
strcpy(paudioAttr→cFormat,"cFormatA");
paudioAttr→ulSamplingRate=44100;
paudioAttr→usNumChannels=2;
hvAudattrs.len=sizeof(MMDBAudioAttrs);

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
               DB2AUDIO(
               CURRENT SERVER,
               '/employee/sounds/ajones.aud',
               :hvStorageType,
               'Anita"s voice',
               :hvAudattr)
       );
```

DB2Image

Image	Audio	Video
X		

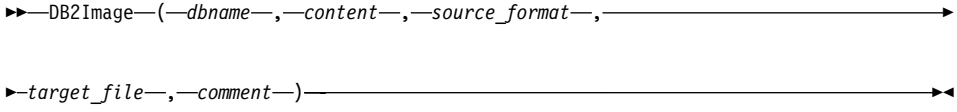
데이터베이스 테이블에 이미지 내용을 저장합니다. 이미지 소스는 클라이언트 버퍼, 클라이언트 파일 또는 서버 파일에 있을 수 있습니다. 이미지는 BLOB으로 데이터베이스 테이블이나, 서버 파일(데이터베이스 테이블이 참조하는)에 저장될 수 있습니다. 이미지 소스는 DB2 Image Extender가 저장영역에 대한 속성을 표시하는 경우에는 지원되는 형식으로 있을 수 있고, 또는 속성이 UDF에 지정되어야 하는 경우에 지원되지 않은 형식으로 있을 수 있습니다.

Include 파일

dmbimage.h

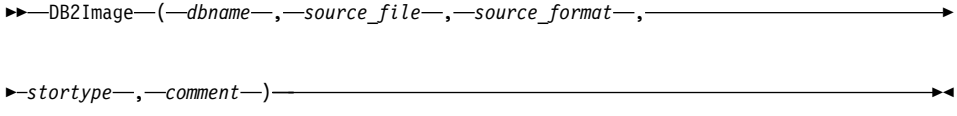
구문

버퍼나 클라이언트 파일로부터 내용 저장



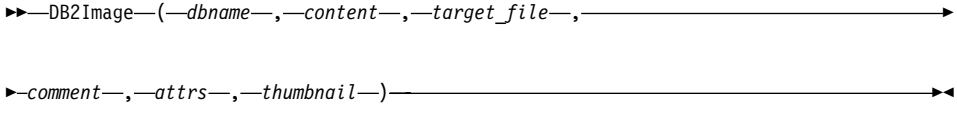
구문

서버 파일로부터 내용 저장



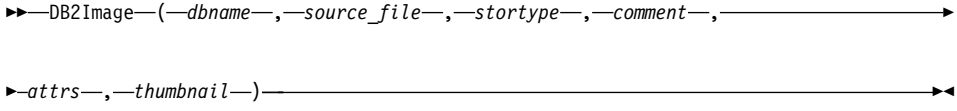
구문

버퍼 또는 클라이언트 파일로부터 사용자 제공 속성으로 내용 저장



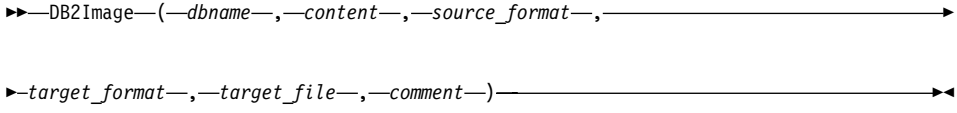
구문

서버 파일로부터 사용자 제공 속성으로 내용 저장



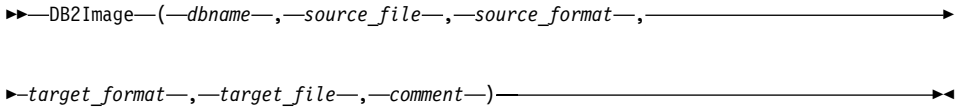
구문

형식 변환을 사용하여 버퍼 또는 클라이언트 파일로부터 내용 저장



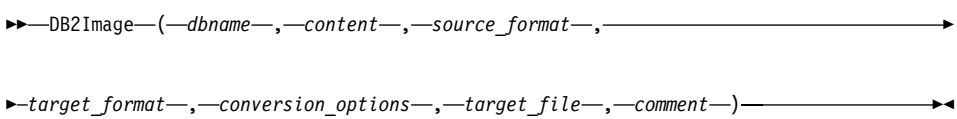
구문

형식 변환을 사용하여 서버 파일의 내용 저장



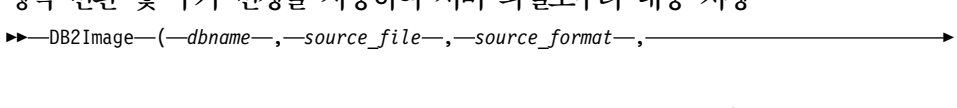
구문

형식 변환 및 추가 변경을 사용하여 버퍼 또는 클라이언트 파일로부터 내용 저장



구문

형식 변환 및 추가 변경을 사용하여 서버 파일로부터 내용 저장



매개변수(데이터 유형)

dbname(VARCHAR(18))

CURRENT SERVER 특수 레지스터가 표시하는 것과 같은 현재 연결된 데이터베이스 이름.

content(BLOB(2G) AS LOCATOR)

이미지 내용이 있는 호스트 변수. 호스트 변수는 BLOB, BLOB_FILE 또는 BLOB_LOCATOR 유형이 가능합니다. DB2는 내용의 데이터 유형을 BLOB_LOCATOR로 진행시키고 LOB 위치 지정자를 DB2Image UDF로 전달합니다.

source_format(VARCHAR(8))

소스 이미지 형식. 널(NULL)값이나 빈 문자열 또는 문자열 ASIS이 지정될 수 있으며, 어떤 경우에든 Audio Extender가 소스 형식을 자동으로 결정하려 할 것입니다. 이미지는 소스와 동일 형식으로 저장될 것입니다. 지원되는 형식에 대해서는 97 페이지의 표5를 참조하십시오.

target_format(VARCHAR(8))

저장 후의 이미지 형식. 소스 이미지 형식은 적절하게 변환될 것입니다.

target_file(LONG VARCHAR)

목표 서버 파일(서버 파일로 저장용) 이름 또는 널(NULL) 값이나 빈 문자열(BLOB으로 데이터베이스 테이블로 저장용). 목표 파일은 완전히 규정된 이름이 될 수 있습니다. 이름이 규정되지 않는다면, 서버에 있는 DB2IMAGESTORE와 DB2MMSTORE 환경 변수가 파일을 위치지정하는 데 사용됩니다. 이미지가 형식 변환을 사용하여 저장된다면, 목표 파일로의 경로가 DB2IMAGEPATH와 DB2MMPATH 환경 변수에 지정되어야 합니다.

source_file(LONG VARCHAR)

소스 서버 파일 이름. 소스 파일 이름은 완전히 규정된 이름이나 규정되지 않은 이름일 수 있습니다. 널(NULL)값이나 빈 문자열이 될 수는 없습니다. 이름이 규정되지 않는다면, 서버에 있는 DB2IMAGEPATH와 DB2MMPATH 환경 변수가 파일을 위치지정하는 데 사용됩니다.

stortype(INTEGER)

이미지가 저장될 장소를 표시하는 값. 상수

MMDB_STORAGE_TYPE_INTERNAL(값=1)은 이미지가 BLOB으로 데이터베이스에 저장될 것임을 나타냅니다. 상수 MMDB_STORAGE_TYPE_EXTERNAL(값=0)은 이미지 내용이 서버 파일(데이터베이스에서 지시됨)에 저장될 것임을 나타냅니다.

comment(LONG VARCHAR)

이미지와 함께 저장되는 주석

attrs(LONG VARCHAR FOR BIT DATA)

이미지 속성

thumbnail(LONG VARCHAR FOR BIT DATA)

이미지 썸네일

conversion_options(VARCHAR(100))

저장시 이미지에 적용되는, 회전이나 압축과 같은 변경사항을 지정합니다. 지원되는 변환 옵션에 대해서는 99 페이지의 표6을 참조하십시오.

리턴 값(데이터 유형)

이미지 핸들(DB2IMAGE)

예

Anita Jones의 이미지가 있는 레코드를 employee 테이블에 삽입하십시오. 이미지 소스는 클라이언트 버퍼에 있습니다. BLOB으로 테이블에 이미지 클립을 저장하십시오.

```
EXEC SQL BEGIN DECLARE SECTION
     SQL TYPE IS BLOB (2M) hvImg
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2IMAGE(
         CURRENT SERVER,
         :hvImg,
         'ASIS',
         CAST(NULL as LONG VARCHAR),

         'Anita''s picture'));
```

Robert Smith의 이미지가 있는 레코드를 employee 테이블에 삽입하십시오. 이미지 소스는 서버 파일에 있습니다. employee 테이블 레코드는 파일을 지시할 것입니다. 저장시 이미지 형식을 BMP에서 GIF로 변환하십시오. 또한 이미지는 110 픽셀 폭과 150 픽셀 높이로 잘라서 LZW 유형 압축을 사용하여 압축하십시오.

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '384779',
      'Robert Smith',
      DB2IMAGE(
         CURRENT SERVER,
         '/employee/pictures/rsmith.bmp',
         'BMP',
         'GIF',
         '-x 110 -y 150 -c 14',
         '',
         'Robert"s picture'));
```

Robert Smith의 이미지가 있는 레코드를 employee 테이블에 삽입하십시오. 서버 파일에 있는 소스 이미지는 사용자 정의 형식으로, 640 픽셀 높이와 480 픽셀의 폭을 갖습니다. 이미지를 BLOB으로 저장하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[400];
}hvImgattrs;
EXEC SQL END DECLARE SECTION;

DB2IMAGEATTRS    *pimgattr;

hvStorageType = MMDB_STORAGE_TYPE_INTERNAL;

pimgattr = (DB2IMAGEATTRS *) hvImgattrs.data;
strcpy(pimgattrt→cFormat,"FormatI");
pimgattr→width=640;
pimgattr→height=480;
hvImgattrs.len=sizeof(DB2IMAGEATTRS);

EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2IMAGE(
          CURRENT SERVER,
          '/employee/images/ajones.bmp',
          :hvStorageType,
```

```
   'Anita''s picture',
   :hvImgattrs,
   CAST(NULL as LONG VARCHAR))
);
```

DB2Video

Image	Audio	Video
		X

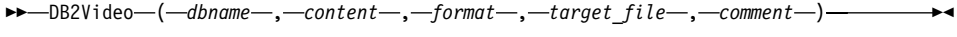
데이터베이스 테이블에 비디오 내용을 저장합니다. 비디오 소스는 클라이언트 버퍼, 클라이언트 파일 또는 서버 파일에 있을 수 있습니다. 비디오는 BLOB으로 데이터베이스 테이블이나, 서버 파일(데이터베이스 테이블이 참조하는)에 저장될 수 있습니다. 비디오 소스는 DB2 Video Extender가 저장영역에 대한 속성을 표시하는 경우에는 지원되는 형식으로 있을 수 있고, 또는 속성이 UDF에 지정되어야 하는 경우에 지원되지 않은 형식으로 있을 수 있습니다.

Include 파일

dmbvideo.h

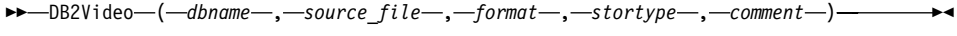
구문

버퍼나 클라이언트 파일로부터 내용 저장



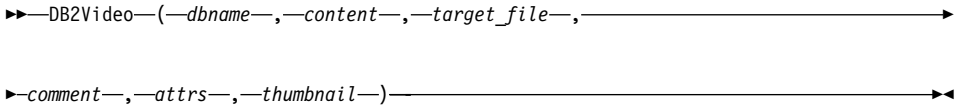
구문

서버 파일로부터 내용 저장



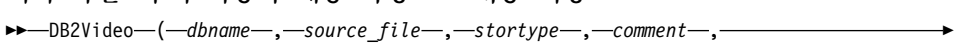
구문

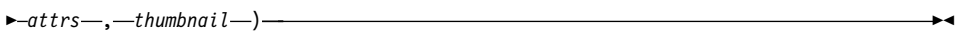
버퍼 또는 클라이언트 파일로부터 사용자 제공 속성으로 내용 저장



구문

서버 파일로부터 사용자 제공 속성으로 내용 저장





매개변수(데이터 유형)

dbname(VARCHAR(18))

CURRENT SERVER 특수 레지스터가 표시하는 것과 같은 현재 연결된 데이터베이스 이름.

content(BLOB(2G) AS LOCATOR)

비디오 내용이 있는 호스트 변수. 호스트 변수는 BLOB, BLOB_FILE 또는 BLOB_LOCATOR 데이터 유형이 될 수 있습니다. DB2는 내용을 BLOB_LOCATOR로 진행시키고 LOB 위치 지정자를 DB2Video UDF로 전달합니다. 내용이 클라이언트 버퍼에 있다면, 최소한 버퍼에는 완전한 비디오 헤더가 읽히는지 확인하는 내용의 처음 640 KB가 있어야 합니다.

format(VARCHAR(8))

소스 비디오 형식. 널(NULL) 값 또는 빈 문자열을 지정할 경우, Video Extender는 자동으로 소스 형식을 판별하려고 합니다. 비디오는 소스와 동일 형식으로 저장됩니다. 지원되는 비디오 형식에 대해서는 97 페이지의 표5를 참조하십시오. MPG1 형식에 대해 MPG1, mpg1, MPEG1 또는 mpeg1을 지정할 수 있습니다. MPG2 형식에 대해 MPG2, mpg2, MPEG2 또는 mpeg2를 지정할 수 있습니다.

target_file(LONG VARCHAR)

목표 서버 파일(서버 파일로 저장용) 이름 또는 널(NULL) 값이나 빈 문자열(BLOB으로 데이터베이스 테이블로 저장용). 서버 파일은 완전히 규정된 이름이어야 합니다. 이름이 규정되지 않는다면, 서버에 있는 DB2VIDEOSTORE와 DB2MMSTORE 환경 변수가 파일을 위치지정하는 데 사용됩니다.

source_file(LONG VARCHAR)

소스 서버 파일 이름. 이름은 완전히 규정된 이름이나 규정되지 않은 이름일 수 있습니다. 널(NULL) 값이나 빈 문자열이 될 수는 없습니다. 이름이 규정되지 않는다면, 서버에 있는 DB2VIDEOPATH와 DB2MMPATH 환경 변수가 파일을 위치지정하는 데 사용될 것입니다.

stortype(INTEGER)

비디오가 저장될 장소를 표시하는 값. 상수 MMDB_STORAGE_TYPE_INTERNAL(값=1)은 비디오가 BLOB으로 데이터베이스에 저장될 것검을 나타냅니다. 상수 MMDB_STORAGE_TYPE_EXTERNAL(값=0)은 비디오 내용이 서버 파일(데이터베이스에서 지시됨)에 저장될 것임을 나타냅니다.

comment(LONG VARCHAR)

비디오와 함께 저장되는 주석.

attrs(LONG VARCHAR FOR BIT DATA)

비디오 속성.

thumbnail(LONG VARCHAR FOR BIT DATA)

비디오를 나타내는 썸네일 이미지.

리턴 값(데이터 유형)

비디오 핸들(DB2VIDEO)

예

Anita Jones용 비디오 클립이 있는 레코드를 employee 테이블에 삽입하십시오. 비디오 소스는 클라이언트 버퍼에 있습니다. BLOB으로 테이블에 비디오 클립을 저장하십시오.

```
EXEC SQL BEGIN DECLARE SECTION
     SQL TYPE IS BLOB (8M) vid;
EXEC SQL END DECLARE SECTION;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '128557',
      'Anita Jones',
      DB2VIDEO(
         CURRENT SERVER,
         :vid,
         'MPEG1',
         CAST(NULL as LONG VARCHAR),
         'Anita''s video'));
```

Robert Smith의 비디오 클립이 있는 레코드를 employee 테이블에 삽입하십시오. 비디오 소스는 서버 파일에 있습니다. employee 테이블 레코드는 파일을 지시할 것입니다.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType = MMDB_STORAGE_TYPE_EXTERNAL;

EXEC SQL INSERT INTO EMPLOYEE VALUES(
      '384779',
      'Robert Smith',
      DB2VIDEO(
         CURRENT SERVER,
         '/employee/videos/rsmith.mpg',
         'MPEG1',
         :hvStorageType,
         'Robert''s video'));
```

비디오 클립이 있는 레코드를 데이터베이스 테이블에 삽입하십시오. 서버 파일에 있는 소스 비디오 클립은 사용자 정의 형식을 갖습니다. 서버 파일에 있는 비디오 내용을 유지하십시오.(데이터베이스 레코드는 파일을 지시할 것입니다.) 비디오를 나타내는 썸네일을 저장하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvStorageType;
struct {
       short len;
       char data[400];
}hvVidattrs;
struct {
       short len;
       char data[10000];
}hvThumbnail;
EXEC SQL END DECLARE SECTION;

MMDBVideoAttrs         *pvideoAttr;

hvStorageType = MMDB_STORAGE_TYPE_EXTERNAL;

pvideoAttr=(MMDBVideoAttrs *)hvVidattrs.data;
strcpy(pvideoAttr→cFormat,"Formatv");
pvideoAttr.len=sizeof(MMDBVideoAttrs);
:
:

/* Generate thumbnail and assign data   */
/* in video structure                   */
:
:
```

```
EXEC SQL INSERT INTO EMPLOYEE VALUES(
        '128557',
        'Anita Jones',
        DB2VIDEO(
          CURRENT SERVER,
          '/employee/videos/ajones.vid',
          :hvStorageType,
          'Anita''s video',
          :hvVidattrs,
          :hvThumbnail)
       );
```

Duration

Image	Audio	Video
	X	X

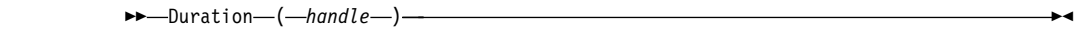
WAVE나 AIFF 오디오 또는 비디오의 존속 기간(즉, 초당 재생 시간)을 리턴합니다.

Include 파일

오디오 dmbaudio.h

비디오 dmbvideo.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO 또는 DB2VIDEO)
오디오나 비디오 핸들이 있는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오의 존속 기간(초 단위) 또는 WAVE, AIFF 또는 사용자 정의 형식 오디오의 존속 기간(초 단위)(INTEGER). 기타 형식의 오디오에 대해 널(NULL) 값을 리턴합니다.

예

employee 테이블의 video 컬럼에 저장된 모든 비디오의 존속 기간을 표시하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvDur_vid;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT DURATION(VIDEO)
     INTO :hvDur_vid
     FROM EMPLOYEE;
```

Filename

Image	Audio	Video
X	X	X

오브젝트 내용이 파일(데이터베이스 테이블에서 지시됨)에 저장된다면, 이미지나 오디오 또는 비디오 내용이 있는 서버 파일 이름을 리턴합니다. 데이터베이스 테이블에 이미지나 오디오 또는 비디오가 저장된다면, 널(NULL) 값이 리턴됩니다.

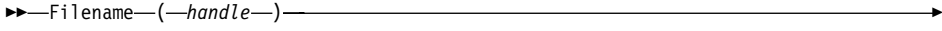
Include 파일

이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

구문



매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)
이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

오브젝트 내용이 서버 파일에 있는 경우, 서버 파일의 파일 이름(VARCHAR(250)). 오브젝트가 BLOB으로 저장된 경우에는 널(NULL) 값.

예

employee 테이블에 Robert Smith 항목에 대한 비디오의 파일 이름을 표시하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NAME='Robert Smith';
```

FindInstrument

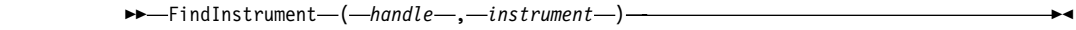
Image	Audio	Video
	X	

MIDI 오디오에 있는 지정 악기의 처음 발생 트랙 번호를 리턴합니다.

Include 파일

dmbaudio.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO)

오디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

instrument(VARCHAR(255))

탐색되는 악기 이름. Audio Extender는 제공된 이름과 정확히 일치하는 악기 이름을 찾을 것입니다.

리턴 값(데이터 유형)

지정 악기 이름의 처음 발생하는 트랙 번호(SMALLINT). 지정 이름의 악기가 발견되지 않는다면 값 -1이 리턴됩니다. 널(NULL)은 기타 형식의 오디오에 대해 리턴됩니다.

예

employee 테이블의 sound 컬럼에 저장된 Robert Smith의 MIDI 오디오 레코딩에서 PIANO의 처음 발생을 찾으십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvInstr;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FINDINSTRUMENT(SOUND, 'PIANO')
       INTO :hvInstr
       FROM EMPLOYEE
       WHERE NAME = 'Robert Smith';
```

FindTrackName

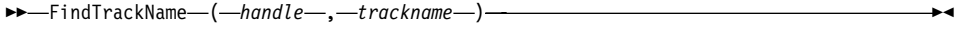
Image	Audio	Video
	X	

MIDI 오디오에서 지정 이름 지정된 트랙 번호를 리턴합니다.

Include 파일

dmbaudio.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO)

오디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

trackname(VARCHAR(255))

탐색되는 트랙 이름. Audio Extender는 제공된 이름과 정확히 일치하는 트랙 이름을 찾을 것입니다.

리턴 값(데이터 유형)

지정 악기 이름의 이름 지정된 트랙 번호(SMALLINT). 지정 트랙 이름이 발견되지 않는다면 값 -1이 리턴됩니다. 널(NULL) 값이 기타 형식의 오디오에 대해 리턴됩니다.

예

Robert Smith의 MIDI 오디오 레코딩에 WELCOME이라는 트랙이 있는지 판별하십시오. 오디오 레코딩은 employee 테이블의 sound 컬럼에 저장됩니다.

```
EXEC SQL BEGIN DECLARE SECTION;
      short hvTrack;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FINDTRACKNAME(SOUND,
           'WELCOME')
      INTO :hvTrack
      FROM EMPLOYEE
      WHERE NAME = 'Robert Smith';
```

Format

Image	Audio	Video
X	X	X

이미지나 오디오 또는 비디오 형식을 리턴합니다.

Include 파일

이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)

이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

이미지, 오디오 또는 비디오 형식(VARCHAR(8)). 지원되는 이미지, 오디오 및 비디오 형식에 대해서는 97 페이지의 표5를 참조하십시오.

예

employee 테이블의 picture 컬럼에 저장된 GIF 형식으로 이미지의 모든 직원 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
char hvName[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT NAME
     INTO :hvName
     FROM EMPLOYEE
     WHERE FORMAT(PICTURE)='GIF';
```

FrameRate

Image	Audio	Video
		X

초당 프레임으로 비디오 처리량을 리턴합니다.

Include 파일

dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2VIDEO)

비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오의 프레임율(SMALLINT). 처리량 비율이 변수라면, 널(NULL) 값을 리턴합니다.

예

Anita Jones에 대하여 employee 테이블의 video 컬럼에 저장된 비디오의 프레임 비율을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
short hvFm_rate;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FRAMERATE (VIDEO)
     FROM EMPLOYEE
      INTO :hvFm_rate
     WHERE NAME='Anita Jones';
```

GetInstruments

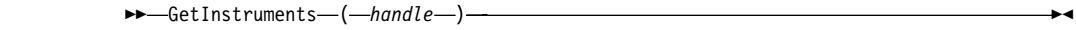
Image	Audio	Video
	X	

MIDI 오디오에 있는 악기 이름 모두를 리턴합니다.

Include 파일

dmbaudio.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO)

오디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

MIDI 오디오에 있는 악기 이름 모두(VARCHAR(1536)). 값은 트랙 번호순으로 리턴됩니다(예를 들어, PIANO; TRUMPET; BASS). 결과는 *n*이 MIDI 오디오의 트랙 번호일 때 *n* 필드로 나뉩니다. 트랙에 연관된 악기가 없으면 필드는 비게 됩니다. 널(NULL) 값이 MIDI 외에 다른 형식의 오디오에 대해 리턴됩니다.

예

Robert Smith의 MIDI 오디오 레코딩에서 모든 악기(즉, 트랙 번호와 악기 이름)를 찾으십시오. 오디오 레코딩은 employee 테이블의 sound 컬럼에 저장됩니다.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_Instr[1536];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT GETINSTRUMENTS(SOUND)
       INTO :hvAud_Instr
       FROM EMPLOYEE
       WHERE NAME = 'Robert Smith';
```

GetTrackNames

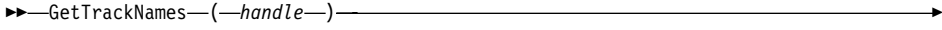
Image	Audio	Video
	X	

MIDI 오디오에 있는 트랙 이름 모두를 리턴합니다.

Include 파일

dmbaudio.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO)

오디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

MIDI 오디오에 있는 모든 트랙 이름(VARCHAR(1536)). 값은 트랙 번호순으로 리턴됩니다(예를 들어, PIANO TUNE; TRUMPET FANFARE). 결과는 *n*이 MIDI 오디오의 트랙 번호일 때 *n* 필드로 나뉩니다. 트랙에 이름이 없으면 필드는 비게 됩니다. 널(NULL) 값이 MIDI 외에 다른 형식의 오디오에 대해 리턴됩니다.

예

employee 테이블의 sound 컬럼에 저장된 Robert Smith의 MIDI 오디오 레코딩에서 모든 트랙 번호와 트랙 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvTracks[1536];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT GETTRACKNAMES(SOUND)
       INTO :hvTracks
       FROM EMPLOYEE
       WHERE NAME = 'Robert Smith';
```

Height

Image	Audio	Video
X		X

픽셀 단위로 이미지나 비디오 프레임의 높이를 리턴합니다.

Include 파일

이미지 dmbimage.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2IMAGE 또는 DB2VIDEO)

이미지나 비디오 핸들이 있는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

픽셀 단위의 높이(INTEGER)

예

employee 테이블의 picture 컬럼에서 500 픽셀보다 짧은 모든 이미지의 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE HEIGHT(PICTURE)<500;
```

Importer

Image	Audio	Video
X	X	X

데이터베이스 테이블에 이미지, 오디오 또는 비디오를 저장한 사람의 사용자 ID를 리턴합니다.

Include 파일

이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)
이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

반입자의 사용자 ID(CHAR(8))

예

사용자 ID rsmith로 employee 테이블의 sound 컬럼에 저장된 오디오용 파일 이름 모두를 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
     INTO :hvAud_fname
     FROM EMPLOYEE
     WHERE IMPORTER(SOUND)='rsmith';
```

ImportTime

Image	Audio	Video
X	X	X

데이터베이스 테이블에 이미지나 오디오 또는 비디오 저장 시간을 나타내는 시간 소인을 리턴합니다.

Include 파일

이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)

이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

이미지, 오디오 또는 비디오 저장시 시간소인

예

1년 이상이 경과한 employee 테이블의 picture 컬럼에 저장된 이미지용 파일 이름 모두를 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE(CURRENT TIMESTAMP -
        IMPORTTIME(PICTURE))>365;
```

MaxBytesPerSec

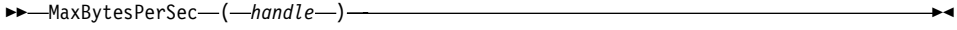
Image	Audio	Video
		X

초당 바이트 수로 비디오의 최대 처리량을 리턴합니다.

Include 파일

dmbvideo.h

구문



매개변수(데이터 유형)

handle(DB2VIDEO)

비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오 처리량(INTEGER). 처리량 비율이 변수라면, 널(NULL) 값을 리턴합니다.

예

Anita Jones에 대하여 employee 테이블의 video 컬럼에 저장된 최대 비디오 처리량을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvMax_BytesPS;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT MAXBYTESPERSEC(VIDEO)
     INTO :hvMax_BytesPS
     FROM EMPLOYEE
     WHERE NAME='Anita Jones';
```

NumAudioTracks

Image	Audio	Video
	X	X

비디오나 MIDI 오디오에서 오디오 트랙의 수를 리턴합니다.

Include 파일

오디오 dmbaudio.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2AUDIO 또는 DB2VIDEO)
비디오나 비디오 핸들이 있는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오나 MIDI 오디오에 있는 오디오 트랙의 수(SMALLINT). 기타 형식의 오디오에 대해 널(NULL) 값을 리턴합니다.

예

employee 테이블의 video 컬럼에서 오디오 트랙이 전혀 없는 임의의 비디오 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NUMAUDIOTRACKS(VIDEO) = 0;
```

NumChannels

Image	Audio	Video
	X	X

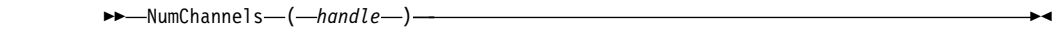
WAVE나 AIFF 오디오 또는 비디오에서 레코드된 오디오의 채널 수를 리턴합니다.

Include 파일

오디오 dmbaudio.h

비디오 dmbvideo.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO 또는 DB2VIDEO)
비디오나 비디오 핸들이 있는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오나 WAVE 또는 AIFF 오디오에 있는 레코드된 오디오의 채널 수 (SMALLINT). 기타 형식의 오디오에 대해 널(NULL) 값을 리턴합니다.

예

employee 테이블의 sound 컬럼에서 스테레오로 레코드(즉, 2 채널)된 오디오 파일 이름 모두를 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
     INTO :hvAud_fname
     FROM EMPLOYEE
     WHERE NUMCHANNELS(SOUND) = 2;
```

NumColors

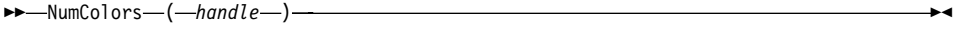
Image	Audio	Video
X		

이미지에 있는 색상 수를 리턴합니다.

Include 파일

dmbimage.h

구문



매개변수(데이터 유형)

handle(DB2IMAGE)

이미지 핸들이 있는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

이미지의 색상 수(INTEGER)

예

employee 테이블의 picture 컬럼에서 16 색상 미만의 색상을 갖는 이미지용 이미지 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE NUMCOLORS(PICTURE) < 16;
```

NumFrames

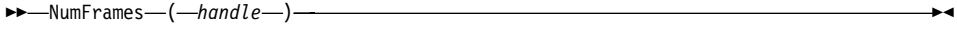
Image	Audio	Video
		X

비디오에 있는 프레임 수를 리턴합니다.

Include 파일

dmbvideo.h

구문



매개변수(데이터 유형)

handle(DB2VIDEO)

비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오의 프레임 수(INTERGER). 처리량 비율이 변수라면, 널(NULL) 값을 리턴합니다.

예

Robert Smith에 대하여 employee 테이블의 video 컬럼에 저장된 비디오의 프레임 수를 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
long hvNum_Frames;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT NUMFRAMES (VIDEO)
     INTO :hvNum_Frames
     FROM EMPLOYEE
     WHERE NAME='Robert Smith';
```

NumVideoTracks

Image	Audio	Video
		X

비디오에 있는 비디오 트랙 수를 리턴합니다.

Include 파일

dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2VIDEO)

비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오 트랙 수(SMALLINT)

예

employee 테이블의 video 컬럼에서 하나 이상의 비디오 트랙이 있는 비디오 모두의 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvVid_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (VIDEO)
     INTO :hvVid_fname
     FROM EMPLOYEE
     WHERE NUMVIDEOTRACKS(VIDEO) > 1;
```

QbScoreFromName

Image	Audio	Video
X		

조회 오브젝트의 특성에 대해 이미지 특성의 유사 정도를 표현하는 숫자인 이미지 스코어를 리턴합니다. 이미지 핸들이 속한 컬럼과 연관된 QBIC 카탈로그는 이미지의 스코어를 계산하는 데 사용됩니다. 스코어가 낮을수록 이미지 특성은 지정된 조회 오브젝트의 특성과 더욱 일치합니다. (QbScoreFromName는 QbScore를 대체하지만, QbScore는 여전히 수용됩니다.)

주:

1. **EEE 전용:** QbScoreFromName은 파티션된 데이터베이스 환경에서 지원되지 않습니다. QbQueryGetString API를 사용하여 조회 문자열을 확보한 후에 대신 QbScoreFromStr UDF를 사용하십시오.
2. QbScoreFromName은 파티션되지 않은 데이터베이스 환경의 경우, 나중 릴리스에서 거부될 것입니다. 조회를 다시 사용하려면, QbQueryGetString API를 사용하여 조회 문자열을 확보하고 응용프로그램에서 나중에 사용하기 위해 그 문자열을 저장해야 합니다.

Include 파일

없음

구문

구문

거부된 버전

매개변수(데이터 유형)

imgHandle(DB2Image)

이미지 핸들.

queryName(varchar(18))

조회 오브젝트 이름.

리턴 값(데이터 유형)

이미지 스코어(DOUBLE). 스코어의 범위는 0.0에서부터 무한에 가까운 매우 큰 수에 달할 수 있습니다. 스코어가 낮을수록 이미지의 특성값은 조회에 지정된 특성값과 더욱 일치합니다. 스코어 0.0은 이미지가 정확히 일치함을 의미합니다. 널(NULL) 값 스코어는 이미지가 카탈로그화되지 않았음을 의미합니다. 이 UDF의 거부된 버전은 이미지가 카탈로그화되지 않았을 때 스코어 -1을 리턴합니다.

예

평균 색상이 빨간색과 아주 유사한 테이블 컬럼에 카탈로그화된 이미지를 찾으십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
char Img_fnd[100];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT NAME
  INTO :Img_fnd
  FROM FABRIC
  WHERE (QBSCOREFROMNAME(SWATCH_IMG,
         'fshavgcol'))<0.1;
```

QbScoreFromStr

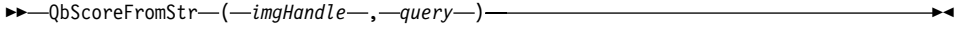
Image	Audio	Video
X		

이미지 특성이 조회 문자열의 그것에 대한 유사 정도를 표현하는 숫자인 이미지 스코어를 리턴합니다. 이미지 핸들이 속한 컬럼과 연관된 QBIC 카탈로그는 이미지의 스코어를 계산하는 데 사용됩니다. 스코어가 낮을수록 이미지의 특성은 조회 문자열의 특성과 더욱 일치합니다.

Include 파일

없음

구문



구문

거부된 버전

매개변수(데이터 유형)

imgHandle(DB2Image)

이미지 핸들.

query(VARCHAR(1024))

조회 문자열.

리턴 값(데이터 유형)

이미지 스코어(DOUBLE). 스코어의 범위는 0.0에서부터 무한에 가까운 매우 큰 수에 달할 수 있습니다. 스코어가 낮을수록 목표 이미지의 특성값은 조회에 지정된 특성값과 더욱 일치합니다. 스코어 0.0은 이미지가 정확히 일치함을 의미합니다. 널(Null) 값 스코어는 이미지가 카탈로그화되지 않았음을 의미합니다. 이 UDF의 거부된 버전은 이미지가 카탈로그화되지 않았을 때 스코어 -1을 리턴합니다.

예

평균 색상이 빨간색과 아주 유사한 테이블 컬럼에 카탈로그화된 이미지를 찾으십시오.

```
SELECT name
FROM fabric
WHERE (QbScoreFromStr(Swatch_Img,
       'QbColorFeatureClass color=<255, 0, 0>'))<0.1
```

QbScoreTBFromName

Image	Audio	Video
X		

이미지 컬럼용 스코어 테이블을 리턴합니다. 각각의 스코어는 이미지 특성이 조회 오브젝트의 그것에 대한 유사 정도를 표현하는 숫자입니다. 이미지 핸들이 속한 지정된 테이블 및 컬럼과 연관된 QBIC 카탈로그는 각각의 이미지의 스코어를 계산하는 데 사용됩니다. 이미지의 스코어가 낮을수록 그 이미지의 특성은 조회 오브젝트의 특성에 더 가까워집니다.

주:

1. **EEE 전용:** QbScoreTBFromName은 파티션된 데이터베이스 환경에서 지원되지 않습니다. QbQueryGetString API를 사용하여 조회 문자열을 확보한 후에 대신 QbScoreFromStr UDF를 사용하십시오.
2. QbScoreTBFromName은 파티션되지 않은 데이터베이스 환경의 경우, 나중에 거부될 것입니다. 조회를 다시 사용하려면, QbQueryGetString API를 사용하여 조회 문자열을 확보하고 응용프로그램에서 나중에 사용하기 위해 그 문자열을 저장해야 합니다.

Include 파일

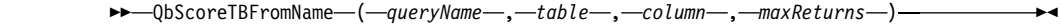
없음

구문

컬럼에 있는 카탈로그화된 모든 이미지의 스코어 리턴

구문

컬럼에 있는 특정 수의 카탈로그화된 이미지의 스코어 리턴



매개변수(데이터 유형)

queryName(VARCHAR(18))

조회 오브젝트 이름.

table(CHAR(18))

이미지 컬럼을 포함하는 테이블의 규정된 이름. 테이블 스키마가 DB2 Extender 서비스를 시작하기 위해 사용되는 사용자 ID와 같을 경우 규정되지 않은 테이블 이름을 사용할 수 있습니다.

column(CHAR(18))

이미지 컬럼 이름.

maxReturns(INTEGER)

결과 테이블이 리턴할 핸들의 최대 수. 값을 지정하지 않는 경우, 리턴되는 이미지 핸들의 수는 100입니다.

리턴 값(데이터 유형)

컬럼에 있는 이미지용 이미지 핸들과 스코어 테이블. 결과 테이블에는 두 개의 컬럼이 있습니다. 이미지 핸들이 있는 IMAGE_ID(DB2Image)와 스코어가 있는 SCORE(DOUBLE). 결과 테이블은 스코어에 대한 오름차순으로 정렬됩니다. 스코어의 범위는 0.0에서부터 무한에 가까운 매우 큰 수에 달할 수 있습니다. 스코어가 낮을수록 이미지의 특성값은 조회에 지정된 특성값과 더욱 일치합니다. 스코어 0.0은 이미지가 정확히 일치함을 의미합니다. 스코어 -1은 이미지가 카탈로그화되지 않았음을 의미합니다.

예

테이블 컬럼에 있는 이미지 텍스처를 조회 오브젝트에 지정된 텍스처와 비교하십시오. 이미지 핸들과 스코어를 리턴하십시오.

```
SELECT name, description
 INTO :hvName, :hvDesc
 FROM fabric
 WHERE CAST (swatch_img as varchar(250)) IN
  (SELECT CAST (image_id as varchar(250)) FROM TABLE
  (QbScoreTBFromName
      'fstxtr',
      'clothes.fabric',
      'swatch_img'))
  AS T1));
```

QbScoreTBFromStr

Image	Audio	Video
X		

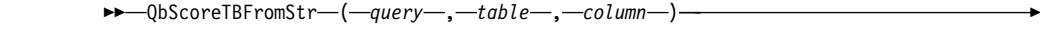
이미지 컬럼에서 스코어 테이블을 리턴합니다. 각각의 스코어는 이미지 특성이 조회 문자열에 지정된 것에 대한 유사 정도를 표현하는 숫자입니다. 이미지 핸들이 속한 테이블 및 컬럼과 연관된 QBIC 카탈로그는 각각의 이미지 스코어를 계산하는 데 사용됩니다. 이미지의 스코어가 낮을수록 그 이미지의 특성은 조회 문자열의 특성에 더 일치합니다.

Include 파일

없음

구문

컬럼에 있는 카탈로그화된 모든 이미지의 스코어 리턴



구문

컬럼에 있는 특정 수의 카탈로그화된 이미지의 스코어 리턴

매개변수(데이터 유형)

query(VARCHAR(1024))

조회 문자열.

table(CHAR(18))

이미지 컬럼을 포함하는 테이블의 규정된 이름. 테이블 스키마가 DB2 Extender 서비스를 시작하기 위해 사용되는 사용자 ID와 같을 경우 규정되지 않은 테이블 이름을 사용할 수 있습니다.

column(CHAR(18))

조회할 이미지 컬럼.

maxReturns(INTEGER)

결과 테이블이 리턴할 핸들의 최대 수. 값이 지정되지 않으면, 리턴되는 이미지 핸들의 최대 수는 100입니다.

리턴 값(데이터 유형)

컬럼에 있는 이미지용 이미지 핸들과 스코어 테이블. 결과 테이블에는 두 개의 컬럼이 있습니다. 이미지 핸들이 있는 IMAGE_ID(DB2Image)와 스코어가 있는 SCORE(DOUBLE). 결과 테이블은 스코어에 따라 오름차순으로 정렬됩니다. 스코어의 범위는 0.0에서부터 무한에 가까운 매우 큰 수에 달할 수 있습니다. 스코어가 낮을수록 이미지의 특성값은 조회에 지정된 특성값과 더욱 일치합니다. 스코어 0.0은 이미지가 정확히 일치함을 의미합니다. 스코어 -1은 이미지가 카탈로그화되지 않았음을 의미합니다.

예

테이블 컬럼에서 텍스처가 서버 파일에 있는 이미지의 텍스처와 가장 유사한 10개의 카탈로그화된 이미지를 찾으십시오.

```
SELECT name, description
FROM fabric
WHERE CAST (swatch_img as varchar(250)) IN
 (SELECT CAST (image_id as varchar(250)) FROM TABLE
 (QbScoreTBFromStr
 (QbTextureFeatureClass file=<server,"patterns/ptrn07.gif">'
      'clothes.fabric',
      'swatch_img',
      10))
  AS T1));
```

Replace

Image	Audio	Video
X	X	X

데이터베이스에 저장된 이미지, 오디오 또는 비디오의 내용을 갱신하고 그것의 주석을 갱신합니다.

Include 파일

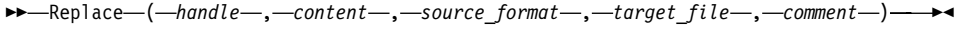
이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

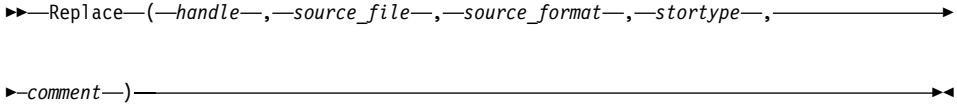
구문

버퍼나 클라이언트 파일로부터 내용 갱신 및 주석 갱신



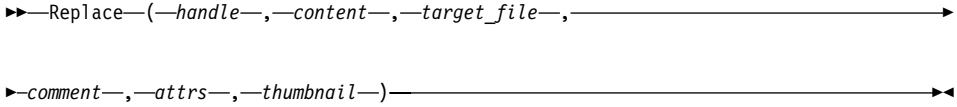
구문

서버 파일로부터 내용 갱신 및 주석 갱신



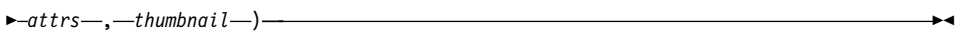
Include 파일

버퍼나 클라이언트 파일로부터 사용자 제공 속성으로 내용 갱신 및 주석 갱신



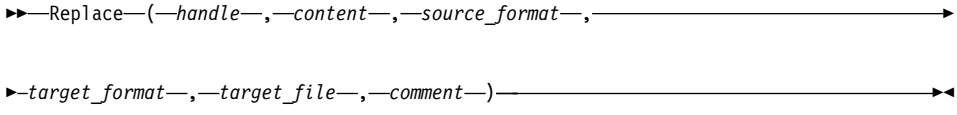
Include 파일

서버 파일로부터 사용자 제공 속성으로 내용 갱신 및 주석 갱신



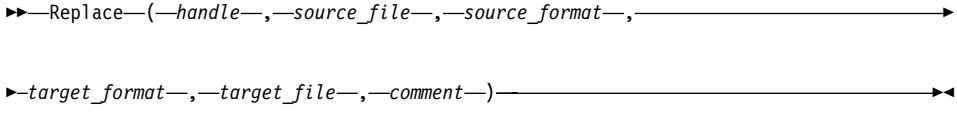
구문

형식 변환을 사용하여 버퍼나 클라이언트 파일로부터 내용 갱신 및 주석 갱신(이미지 전용)



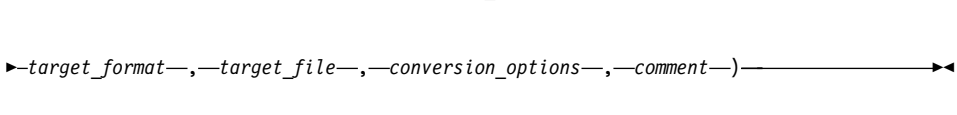
구문

형식 변환을 사용하여 서버 파일로부터 내용 갱신 및 주석 갱신(이미지 전용)



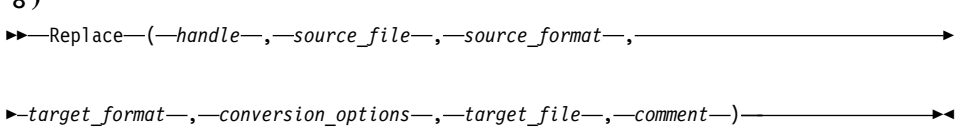
구문

형식 변환 및 추가 변경을 사용하여 버퍼나 클라이언트 파일로부터 내용 갱신 및 주석 갱신(이미지 전용)



구문

형식 변환 및 추가 변경을 사용하여 서버로부터 내용 갱신 및 주석 갱신(이미지 전용)



매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)
이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

source_file(LONG VARCHAR)
이미지, 오디오 또는 비디오 갱신용 내용이 있는 파일 이름.

target_file(LONG VARCHAR)
갱신되는 이미지, 오디오 또는 비디오 갱신용 내용이 있는 파일 이름.

create_target(INTEGER)
소스 내용이 서버 파일에 있는 경우, 목표 파일이 작성되는지 표시하는 값. 값은 0 또는 1일 수 있습니다. 값 0은 목표 파일이 작성되지 않음을 의미합니다.(사실상, 검색되지 않습니다.) 값이 1이면 목표 파일이 작성됨을 의미합니다(목표 파일이 이미 존재하는 경우, 이 값의 결과는 파일을 겹쳐쓰는 것입니다). 소스 내용이 BLOB라면 목표 파일은 언제나 작성됩니다.(파일이 이미 존재한다면, 겹쳐쓰입니다.)

target_format(VARCHAR(8))
검색 후의 이미지 형식. 소스 이미지 형식은 적절하게 변환될 것입니다. 내용이 형식 변환을 사용하여 갱신된다면, 목표 파일로의 경로가 DB2IMAGEPATH와 DB2MMPATH 환경 변수로 지정되어야 합니다. MPG1 형식에 대해 MPG1, mpg1, MPEG1 또는 mpeg1을 지정할 수 있습니다. MPG2 형식에 대해 MPG2, mpg2, MPEG2 또는 mpeg2를 지정할 수 있습니다.

content(BLOB(2G) AS LOCATOR)
이미지, 오디오 또는 비디오 갱신용 내용이 있는 호스트 변수. 호스트 변수는 BLOB, BLOB_FILE 또는 BLOB_LOCATOR 유형이 가능합니다. DB2는 데이터 유형을 BLOB_LOCATOR로 진행시키고 LOB 위치 지정자를 Replace UDF로 전달합니다.

source_format(VARCHAR(8))
이미지, 오디오 또는 비디오 갱신용 소스 형식. 널(NULL) 값이나 빈 문자열 또는 이미지 전용으로 문자열 ASIS이 지정될 수 있으며, 세 가지 경우 모두에 Extender는 자동으로 형식 결정을 시도합니다. MPG1 형식에

대해 MPG1, mpg1, MPEG1 또는 mpeg1을 지정할 수 있습니다. MPG2 형식에 대해 MPG2, mpg2, MPEG2 또는 mpeg2를 지정할 수 있습니다.

comment(LONG VARCHAR)

주석.

attrs(LONG VARCHAR FOR BIT DATA)

이미지, 오디오 또는 비디오의 속성

thumbnail(LONG VARCHAR FOR BIT DATA)

이미지, 비디오 프레임의 썸네일(이미지와 비디오 전용)

conversion_options(VARCHAR(100))

갱신시 이미지에 적용되는, 회전이나 암축과 같은 변경사항을 지정합니다. 지원되는 변환 옵션에 대해서는 99 페이지의 표6을 참조하십시오.

리턴 값(데이터 유형)

갱신되는 이미지, 오디오 또는 비디오 핸들(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO).

예

이미지 형식을 BMP에서 GIF로 변환하여 employee 테이블의 picture 컬럼에 있는 Anita Jones의 이미지를 갱신하고 주석을 갱신하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
  long hvStorageType;
EXEC SQL END DECLARE SECTION;

hvStorageType = MMDB_STORAGE_TYPE_INTERNAL;

EXEC SQL UPDATE EMPLOYEE
  SET PICTURE = REPLACE(PICTURE,
        '/employee/newimg/ajones.bmp',
        'BMP',
        'GIF',
        :hvStorageType,
        'Anita''s new picture')
  WHERE NAME='Anita Jones';
```

SamplingRate

Image	Audio	Video
	X	X

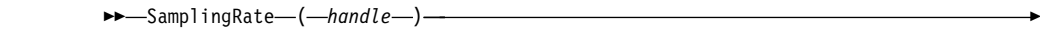
WAVE나 AIFF, 또는 비디오에 있는 오디오 트랙의 샘플링율을 초당 샘플 수로 리턴합니다.

Include 파일

오디오 dmbaudio.h

비디오 dmbvideo.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO 또는 DB2VIDEO)
오디오나 비디오 핸들이 있는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

비디오나 WAVE 또는 AIFF 오디오의 샘플링율(INTEGER). 기타 형식의 오디오에 대해 널(NULL) 값을 리턴합니다.

예

employee 테이블의 sound 컬럼에서 샘플링율이 44.1 KHz인 오디오 모두의 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME (SOUND)
     INTO :hvAud_fname
     FROM EMPLOYEE
     WHERE SAMPLINGRATE(SOUND) = 44100;
```

Size

Image	Audio	Video
X	X	X

이미지, 오디오 또는 비디오 크기(바이트 단위)를 리턴합니다.

Include 파일

이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)
이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

이미지, 오디오 또는 비디오의 바이트 크기(INTEGER).

예

employee 테이블의 picture 컬럼에서 310 KB 이상 크기의 이미지 모두의 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE SIZE(PICTURE) > 310000;
```

Thumbnail

Image	Audio	Video
X		X

데이터베이스에 저장된 이미지 또는 비디오 프레임의 썸네일 크기 버전을 리턴하거나 갱신합니다.

Include 파일

이미지 dmbimage.h

비디오 dmbvideo.h

구문

썸네일 검색

구문

썸네일 갱신

매개변수(데이터 유형)

handle(DB2IMAGE 또는 DB2VIDEO)

이미지, 비디오 핸들이 있는 컬럼 이름이나 호스트 변수.

new_thumbnail (LONG VARCHAR FOR BIT DATA)

썸네일 갱신을 위한 소스 내용.

리턴 값(데이터 유형)

검색의 경우, 검색된 썸네일의 내용(LONG VARCHAR FOR BIT DATA). 갱신의 경우, 이미지 또는 비디오 핸들(DB2IMAGE 또는 DB2VIDEO).

예

employee 테이블에 저장된 Anita Jones 이미지의 썸네일을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 struct{
       short len;
       char data [32000];
         }hvThumbnail;
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT THUMBNAIL(PICTURE)
     INTO :hvThumbnail
     FROM EMPLOYEE
     WHERE NAME = 'Anita Jones';
```

employee 테이블에 있는 Anita Jones 비디오와 연관된 썸네일을 갱신하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
     struct {
       short len;
       char data[10000];
     }hvThumbnail;
EXEC SQL END DECLARE SECTION;

/*  Create thumbnail and    */
/*  store in hvThumbnail    */

EXEC SQL UPDATE EMPLOYEE
   SET VIDEO=THUMBNAIL(
                VIDEO,
                :hvThumbnail)
   WHERE NAME='Anita Jones';
```

TicksPerQNote

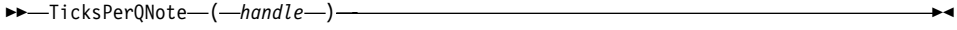
Image	Audio	Video
	X	

1/4 노트당 틱 수로 레코드된 MIDI 오디오의 시계 속도를 리턴합니다.

Include 파일

dmbaudio.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO)

오디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

MIDI 오디오의 1/4 노트당 시계의 틱 수(SMALLINT). 기타 형식의 오디오에 대해 널(NULL) 값을 리턴합니다.

예

employee 테이블의 sound 컬럼에서 1/4 노트당 200번 이상의 시계 틱 수 속도로 레코드된 MIDI 오디오 파일 이름 모두를 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
      INTO :hvAud_fname
      FROM EMPLOYEE
      WHERE FORMAT(SOUND)='MIDI'
      AND TICKSPERQNOTE(SOUND)>200;
```

TicksPerSec

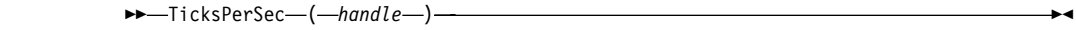
Image	Audio	Video
	X	

초당 틱 수로 레코드된 MIDI 오디오의 시계 속도를 리턴합니다.

Include 파일

dmbaudio.h

구문



매개변수(데이터 유형)

handle(DB2AUDIO)

오디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

MIDI 오디오의 초당 시계의 틱 수(SMALLINT). 기타 형식의 오디오에 대해 널(NULL) 값을 리턴합니다.

예

employee 테이블의 sound 컬럼에서 초당 시계의 틱 수가 50번 이하의 속도로 레코드된 employee인 테이블의 audio 컬럼에서 MIDI 오디오 파일 이름 모두를 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvAud_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(SOUND)
       INTO :hvAud_fname
       FROM EMPLOYEE
       WHERE FORMAT(SOUND)='MIDI'
       AND TICKSPERSEC(SOUND)<50;
```

Updater

Image	Audio	Video
X	X	X

데이터베이스 테이블에 이미지, 오디오 또는 비디오를 마지막으로 갱신한 사람의 사용자 ID를 리턴합니다.

Include 파일

이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)

이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

이미지, 오디오 또는 비디오 형식을 마지막으로 갱신한 사람의 사용자 ID(VARCHAR(8).

예

Robert Smith에 대해 employee 테이블의 video 컬럼에 저장된 비디오를 마지막에 갱신한 사람의 사용자 ID를 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
char hvUpdater[30];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT UPDATER(VIDEO)
     INTO :hvUpdater
     FROM EMPLOYEE
     WHERE NAME='rsmith';
```

UpdateTime

Image	Audio	Video
X	X	X

데이터베이스 테이블에 이미지, 오디오 또는 비디오가 마지막으로 갱신된 시간을 표시하는 시간소인을 나타냅니다.

Include 파일

이미지 dmbimage.h

오디오 dmbaudio.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2IMAGE, DB2AUDIO 또는 DB2VIDEO)

이미지, 오디오 또는 비디오 핸들을 포함하는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

이미지, 오디오 또는 비디오의 마지막 갱신시 시간소인(TIMESTAMP)

예

employee 테이블의 picture 컬럼에서 최근 이틀 동안 갱신된 이미지용 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE(CURRENT TIMESTAMP -
     UPDATETIME(PICTURE))< 2;
```

Width

Image	Audio	Video
X		X

이미지, 비디오 프레임의 픽셀 단위 폭을 리턴합니다.

Include 파일

이미지 dmbimage.h

비디오 dmbvideo.h

구문

매개변수(데이터 유형)

handle(DB2IMAGE 또는 DB2VIDEO)
이미지, 비디오 핸들이 있는 컬럼 이름이나 호스트 변수.

리턴 값(데이터 유형)

픽셀 단위의 폭(INTEGER)

예

300 픽셀보다 폭이 좁은 employee 테이블의 picture 컬럼에 모든 이미지의 파일 이름을 확보하십시오.

```
EXEC SQL BEGIN DECLARE SECTION;
 char hvImg_fname[251];
EXEC SQL END DECLARE SECTION;

EXEC SQL SELECT FILENAME(PICTURE)
     INTO :hvImg_fname
     FROM EMPLOYEE
     WHERE WIDTH(PICTURE)<300;
```

제16장 응용프로그램 프로그래밍 인터페이스(API)

이 장은 DB2 Extender 관리 API용 참조 정보를 제공합니다. API는 알파벳 순서로 수록되어 있습니다.

다음 정보가 각 API에 대해 제공됩니다.

- API를 제공하는 Extender
- 간략한 설명
- API를 사용하는 데 필요한 권한부여
- API용 라이브러리 파일
- UDF용 Include(헤더) 파일
- API 호출 구문
- API 매개변수 설명
- API에 의해 리턴되는 값
- 사용 예

DBaAdminGetInaccessibleFiles

Image	Audio	Video
	X	

사용자 테이블의 오디오 컬럼 내에서 참조된 액세스 불가능한 파일의 이름을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaAdminGetInaccessibleFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정

된 규정자를 가진 모든 테이블이 탐색됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스의 모든 테이블이 검색됩니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블 내에서 참조되는 액세스 불가능한 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

SQL_ERROR 또는 기타 SQL 리턴 코드

DB2에서 리턴된 오류.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

MMDB_RC_NO_AUTH

사용자에게 이 API를 호출할 적절한 권한이 없습니다.

예

사용자 ID rsmith가 소유하는 테이블의 오디오 컬럼 내에서 참조된 모든 액세스 불가능한 파일을 나열합니다.

```
#include <dmbaudio.h>
long idx;

rc = DBaAdminGetInaccessibleFiles("rsmith",
       &count, &filelist);
```

DBaAdminGetReferencedFiles

Image	Audio	Video
	X	

사용자 테이블의 오디오 컬럼 내에서 참조된 파일의 이름을 리턴합니다. 파일을 액세스할 수 없는 경우(예를 들어, 환경 변수 스펙을 사용하여 파일 이름을 해결할 수 없음), 파일 이름에 별표(*)가 선행됩니다. 이 API는 FILEREF 데이터 구조의 FILENAME 필드를 사용하지 않으므로 이것을 널(NULL)로 설정합니다. 응용프로그램은 이 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaAdminGetReferencedFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정된 규정자를 가진 모든 테이블이 탐색됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스의 모든 테이블이 탐색됩니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블 내에서 참조되는 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

MMDB_RC_NO_AUTH

사용자에게 이 API를 호출할 적절한 권한이 없습니다.

예

ajones가 소유하는 테이블의 오디오 컬럼 내에서 참조된 모든 파일을 나열합니다.

```
#include <dmbaudio.h>
long idx;

rc = DBaAdminGetReferencedFiles("ajones",
       &count, &fileList);
```

DBaAdminIsFileReferenced

Image	Audio	Video
	X	

지정된 파일을 참조하는 사용자 테이블에 있는 오디오 컬럼 항목의 목록을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaAdminIsFileReferenced(
      char *qualifier,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정된 규정자를 가진 모든 테이블이 탐색됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스의 모든 테이블이 탐색됩니다.

fileName(입력)

참조된 파일 이름.

count(출력)

출력 목록의 항목 갯수.

tableList(출력)

지정된 파일을 참조하는 테이블 항목의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

MMDB_RC_NO_AUTH

사용자는 이 API를 호출할 적절한 권한이 없습니다.

예

파일 /audios/asmith.wav를 참조하는 현재 데이터베이스 내의 모든 테이블에 있는 오디오 컬럼의 항목을 나열합니다.

```
#include <dmbaudio.h>
long idx;

rc = DBaAdminIsFileReferenced(NULL,
       "/audios/asmith.wav",
       &count, &tableList);
```

DBaAdminReorgMetadata

Image	Audio	Video
	X	

오디오 관련 메타데이터 테이블을『정리』합니다. 예를 들면,

- 오디오 메타데이터 테이블 내에서 더 이상 사용되지 않는 공간을 회수합니다.
- 더 이상 존재하지 않는 오디오 파일에 대한 오디오 메타데이터 테이블 내의 참조를 삭제합니다.

응용프로그램은 이 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaAdminReorgMetadata(
      char *qualifier
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정

된 규정자를 가진 모든 테이블은 정리됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스 내의 모든 테이블은 정리됩니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램은 데이터베이스에 유효한 연결을 갖고 있지 않습니다.

MMDB_RC_NO_AUTH

사용자는 이 API를 호출할 적절한 권한이 없습니다.

예

사용자 ID rsmith가 소유하는 테이블 내의 오디오 컬럼에 대한 메타데이터 테이블을 제거합니다.

```
#include <dmbaudio.h>

rc = DBaAdminReorgMetadata("rsmith");
```

DBaDisableColumn

Image	Audio	Video
	X	

오디오 컬럼(DB2Audio 데이터)으로 사용 불가능하게 하여 오디오 데이터를 수용하지 못하게 합니다. 컬럼 항목의 내용은 널(NULL)에 설정되며, 이 컬럼과 연관된 메타데이터는 삭제됩니다. 이 컬럼에 대해 Audio Extender에 의해 정의된 모든 트리거 역시 삭제됩니다. 새 행은 사용 불가능하게 된 컬럼을 포함하는 테이블에 삽입될 수 있으며, 새 행은 유형 DB2Audio에 의해 정의된 데이터를 포함할 수도 있으나 새 행과 관련된 아무런 메타데이터(관리 지원 테이블 내)도 없습니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다. 이는 이 API를 호출한 후에 SQL COMMIT문을 발행하는 것이 권장됩니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaDisableColumn(
      char *tableName,
      char *colName,
      );
```

매개변수

tableName(입력)

오디오 컬럼을 포함하는 테이블 이름.

colName(입력)

오디오 컬럼의 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

오디오(DB2Audio 데이터)용 employee 테이블의 sound 컬럼을 사용 불가능하게 합니다.

```
#include <dmbaudio.h>

rc = DBaDisableColumn("employee", "sound");
```

DBaDisableDatabase

Image	Audio	Video
	X	

오디오(DB2Audio 데이터)으로 데이터베이스를 사용 불가능화하여 오디오 데이터를 수용하지 못하게 합니다. DB2Audio용으로 정의된 데이터베이스 내의 모든 테이블도 역시 사용 불가능화됩니다. 이 데이터베이스용으로 Audio Extender에 의해 정의된 모든 메타데이터 및 UPF 역시 삭제됩니다. 새 행은 유형 DB2Audio로 정의된 데이터베이스 내 테이블에 삽입될 수도 있으나 새 행과 관련된 아무런 메타데이터(관리 지원 테이블 내)도 없습니다. 이 API를 호출한 후 SQL COMMIT 문을 실행하도록 권장합니다.

권한

DBADM, SYSADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaDisableDatabase(
      );
```

매개변수

DBaDisableDatabase에는 매개변수가 없습니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램은 데이터베이스에 유효한 연결을 갖고 있지 않습니다.

예

오디오(DB2Audio 데이터)용으로 현재 데이터베이스를 사용 불가능하게 합니다.

```
#include <dmbaudio.h>

rc = DBaDisableDatabase();
```

DBaDisableTable

Image	Audio	Video
	X	

오디오(DB2Audio 데이터)용으로 테이블을 사용 불가능하게 하여 오디오 데이터를 수용하지 못하게 합니다. DB2Audio용으로 정의된 테이블 내의 모든 컬럼도 역시 사용 불가능화됩니다. 이 테이블용으로 Audio Extender에 의해 정의된 메타데이터의 일부가 삭제됩니다. 새 행은 유형 DB2Audio로 정의된 테이블에 삽입될 수도 있으나 새 행과 관련된 아무런 메타데이터(관리 지원 테이블 내)도 없습니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다. 이 API를 호출한 후에 SQL COMMIT문을 발행하도록 권장됩니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaDisableTable(
      char *tableName
      );
```

매개변수

tableName(입력)

오디오 컬럼을 포함하는 테이블 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

오디오(DB2Audio 데이터)용으로 employee 테이블을 사용 불가능하게 합니다.

```
#include <dmbaudio.h>

rc = DBaDisableTable("employee");
```

DBaEnableColumn

Image	Audio	Video
	X	

오디오(DB2Audio 데이터)용으로 컬럼을 사용 가능하게 합니다. API는 이 컬럼과 메타데이터 테이블 사이의 관계를 정의하고 관리합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다. 이 API를 호출한 후에 SQL COMMIT문을 발행하도록 권장됩니다.

권한

Control, Alter, SYSADM, DBADM

API 매개변수에 지정된 테이블 공간 및 버퍼 풀에 관한 사용 특권이 또한 필요합니다.

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaEnableColumn(
      char *tableName,
      char *colName,
      );
```

매개변수

tableName(입력)

오디오 컬럼을 포함하는 테이블 이름.

colName(입력)

오디오 컬럼의 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED

컬럼은 이미 사용 가능하게 되었습니다.

MMDB_RC_WRONG_SIGNATURE

지정된 컬럼의 데이터 유형이 올바르지 않습니다. 사용자 정의 데이터 유형 MMDBSYS.DB2AUDIO이 예상됩니다.

MMDB_RC_COLUMN_DOESNOT_EXIST

컬럼이 지정된 테이블 내에서 지정되지 않았습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_NOT_ENABLED

데이터베이스 또는 테이블을 사용할 수 없습니다.

예

오디오(DB2Audio 데이터)용으로 employee 테이블의 sound 컬럼을 사용 가능하게 합니다.

```
#include <dmbaudio.h>

rc = DBaEnableColumn("employee", "sound");
```

DBaEnableDatabase

Image	Audio	Video
	X	

오디오(DB2Audio 데이터)용으로 데이터베이스를 사용 가능하게 합니다. 이 API는 데이터베이스당 한번씩 호출됩니다. 이것은 DB2 사용자 정의 유형, DB2Audio를 데이터베이스 관리 프로그램에 정의합니다. 이것은 또한 DB2Audio 데이터를 조작하는 모든 UDF를 생성합니다. 이 API를 호출한 후 SQL COMMIT문을 발행하도록 권장합니다.

권한

DBADM, SYSADM, SYSCTRL

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaEnableDatabase(
      char *tableSpace
      );
```

매개변수

tableSpace(입력)

관리 테이블이 저장된 컨테이너의 집합인 테이블 공간의 이름. 테이블 공간 스펙은 *datats, indexts, longts*의 세 부분으로 되어 있으며, 여기서 *datats*는 메타데이터 테이블이 생성되는 테이블 공간입니다. *indexts*는 메

타데이터 테이블에 대한 색인이 생성되는 테이블 공간입니다. *longts*는 (LONG VARCHAR 및 LOB 데이터 유형을 포함한 것과 같은) 메타데이터 테이블 내의 긴 컬럼의 값이 저장되는 테이블 공간입니다. 테이블 공간 스펙의 어떤 부분에 대해 널(NULL) 값을 제공하는 경우에 그 부분에 대해 기본 테이블 공간이 사용됩니다.

EEE 전용: Extender에 대해 데이터베이스를 사용 가능화할 때, 지정한 테이블 공간은 파티션된 데이터베이스 시스템의 모든 노드가 포함되는 노드 그룹에 정의되어 있어야 합니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED

데이터베이스는 이미 사용 가능하게 되었습니다.

MMDB_RC_API_NOT_SUPPORTED_FOR_SERVER

연결되는 서버는 이 명령을 지원하지 않습니다.

MMDB_WARN_NOT_ALL_NODES

지정된 테이블 공간이 해당 Extender와 연관된 노드를 모두 포함하지는 않습니다. **(EEE 전용)**

MMDB_RC_NOT_SAME_NODEGROUP

지정된 테이블 공간이 동일한 노드 그룹에 있지 않습니다**(EEE 전용)**.

예

테이블 공간 MYTS 내 오디오(DB2Audio 데이터)용으로 현재 데이터베이스를 사용 가능하게 합니다. 색인 및 긴 테이블 공간용으로 기본값을 사용합니다.

```
#include <dmbaudio.h>

rc = DBaEnableDatabase("myts,,");
```

DBaEnableDatabase

오디오(DB2Audio 데이터)용으로 현재 데이터베이스를 사용 가능하게 합니다. 기본 테이블 공간을 사용합니다.

```
#include <dmbaudio.h>

rc = DBaEnableDatabase(NULL);
```

DBaEnableTable

Image	Audio	Video
	X	

오디오(DB2Audio 데이터)용으로 테이블을 사용 가능하게 합니다. 이 API는 테이블당 한번씩 호출됩니다. 테이블 내의 오디오 컬럼용 속성을 저장하고 관리하기 위해 메타데이터 테이블을 생성합니다. 잠금 가능성을 방지하려면 응용프로그램은 반드시 이 API를 호출하기 전에 트랜잭션을 확약해야 합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다. 이 API를 호출한 후에 SQL COMMIT문을 발행하도록 권장됩니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaEnableTable(
      char *tableSpace,
      char *tableName
      );
```

매개변수

tableSpace(입력)

관리 테이블이 저장되는 컨테이너의 집합인 테이블 공간의 이름. 테이블 공

간 스펙은 *datats, indexts, longts*의 세 부분으로 되어 있으며, 여기서 *datats*는 메타데이터 테이블이 생성되는 테이블 공간입니다. *indexts*는 메타데이터 테이블에 대한 색인이 생성되는 테이블 공간입니다. *longts*는 (LONG VARCHAR 및 LOB 데이터 유형을 포함한 것과 같은) 메타데이터 테이블 내의 긴 컬럼의 값이 저장된 테이블 공간입니다.

테이블 공간 스펙의 임의의 부분에 대해 널(NULL) 값을 제공하는 경우, 이 부분에 대해 기본 테이블 공간이 사용됩니다.

EEE 전용: 지정된 테이블 공간은 사용자 테이블과 동일한 노드 그룹에 있어야 합니다.

tableName(입력)
오디오 컬럼을 포함할 테이블 이름.

오류 코드

MMDB_SUCCESS
API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH
호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED
테이블은 이미 사용 가능하게 되었습니다.

MMDB_RC_NOT_CONNECTED
응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_TABLE_DOESNOT_EXIST
테이블이 존재하지 않습니다.

MMDB_RC_TABLESPACE_NOT_SAME_NODEGROUP
지정된 테이블 공간이 사용자 테이블과 동일한 노드 그룹에 있지 않습니다(**EEE 전용**).

예

테이블 공간 MYTS 내 오디오(DB2Audio 데이터)용으로 employee 테이블을 사용 가능하게 합니다. 색인 및 긴 테이블 공간에 대해 기본값을 사용합니다.

```
#include <dmbaudio.h>

rc = DBaEnableTable("myts,,",
       "employee");
```

오디오(DB2Audio 데이터)용으로 employee 테이블을 사용할 수 있게 합니다. 기본 테이블 공간을 사용합니다.

```
#include <dmbaudio.h>

rc = DBaEnableTable(NULL,
       "employee");
```

DBaGetError

Image	Audio	Video
	X	

마지막 오류의 설명을 리턴합니다. 기타 다른 API가 오류 코드를 리턴한 뒤에 이 API를 호출하십시오.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaGetError(
      SQLINTEGER *sqlcode,
      char *errorMsgText
      );
```

매개변수

sqlcode(출력)

일반적인 SQL 오류 코드.

errorMsgText(출력)

SQL 오류 메시지 텍스트.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

예

SQL 오류 코드를 errCode에 저장하고, 메시지 텍스트를 errMsg에 저장하여 마지막 오류 코드를 확보합니다.

```
#include <dmbaudio.h>

rc = DBaGetError(&errCode, &errMsg);
```

DBaGetInaccessibleFiles

Image	Audio	Video
	X	

사용자 테이블의 오디오 컬럼 내에서 참조된 액세스 불가능한 파일의 이름을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

모든 탐색된 사용자 테이블 및 연관된 관리 지원 테이블에 있는 사용 가능한 오디오 컬럼에 관한 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaGetInaccessibleFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테이블 이름이 지정된 경우 그 테이블은 액세스 불가능한 파일에 대한 참조용으로 탐색됩니다. 널(NULL) 값이 지정되면 지정된 규정자를 갖는 모든 테이블이 탐색됩니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블 내에서 참조되는 액세스 불가능한 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

테이블 employee의 오디오 컬럼 내에서 참조되는 모든 액세스 불가능한 파일을 나열합니다.

```
long idx;
#include <dmbaudio.h>

rc = DBaGetInaccessibleFiles("employee",
       &count, &filelist);
```

DBaGetReferencedFiles

Image	Audio	Video
	X	

사용자 테이블의 오디오 컬럼 내에서 참조된 파일의 이름을 리턴합니다. 파일을 액세스할 수 없는 경우(예를 들어, 환경 변수 스펙을 사용하여 파일 이름을 해결할 수 없음), 파일 이름에 별표(*)가 선행됩니다. 이 API는 FILEREF 데이터 구조의 FILENAME 필드를 사용하지 않으므로 이것을 널(NULL)로 설정합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

모든 탐색된 사용자 테이블 및 연관된 관리 지원 테이블에 있는 사용 가능한 오디오 컬럼에 관한 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaGetReferencedFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 테이블 이름. 테이블 이름이 지정된 경우 그 테이블은 파일에 참조용으로 탐색됩니다. 널(NULL) 값이 지정되면 현재 사용자 ID 데이터베이스에 의해 소유된 모든 테이블이 탐색됩니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블 내에서 참조되는 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

테이블 employee의 오디오 컬럼 내에서 참조되는 모든 파일을 나열합니다.

```
#include <dmbaudio.h>
long idx;

rc = DBaGetReferencedFiles("employee",
       &count, &filelist);
```

DBaIsColumnEnabled

Image	Audio	Video
	X	

컬럼이 오디오(DB2Audio 데이터)용으로 사용 가능하게 되었는지의 여부를 판별합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

사용자 테이블에 관한 SYSADM, DBADM, 테이블 소유자 또는 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaIsColumnEnabled(
      char *tableName,
      char *colName,
      short *status
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 테이블 이름.

colName(입력)

컬럼의 이름.

status(출력)

컬럼이 사용 가능한지를 표시합니다. 매개변수는 숫자 값을 리턴합니다. Extender 또한 상태를 표시하는 상수를 리턴합니다. 그 값과 상수는 다음과 같습니다.

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

-1 MMDB_INVALID_DATATYPE

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

employee 테이블의 sound 컬럼이 오디오용으로 사용 가능한지를 판별합니다.

```
#include <dmbaudio.h>

rc = DBaIsColumnEnabled("employee",
       "sound", &status);
```

DBaIsDatabaseEnabled

Image	Audio	Video
	X	

데이터베이스가 오디오(DB2Audio 데이터)용으로 사용 가능해졌는지의 여부를 판별합니다. 응용프로그램은 이 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaIsDatabaseEnabled(
      short *status
      );
```

매개변수

status(출력)

데이터베이스가 사용 가능한지를 표시합니다. 매개변수는 숫자 값을 리턴합니다. Extender 역시 상태를 표시하는 상수를 리턴합니다. 그 값과 상수는 다음과 같습니다.

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램은 데이터베이스에 유효한 연결을 갖고 있지 않습니다.

예

personnl 데이터베이스가 오디오용으로 사용 가능한지를 판별합니다.

```
#include <dmbaudio.h>

rc = DBaIsDatabaseEnabled(&status);
```

DBaIsFileReferenced

Image	Audio	Video
	X	

지정된 파일을 참조하는 테이블 항목의 목록을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

모든 탐색된 사용자 테이블 및 연관된 관리 지원 테이블에 있는 사용 가능한 오디오 컬럼에 관한 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaIsFileReferenced(
      char *tableName,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테이블 이름이 지정된 경우 그 테이블은 지정된 파일에 대한 참조용으로 탐색됩니다. 널(NULL) 값을 지정하는 경우, 현재 사용자 ID가 소유한 모든 테이블을 탐색합니다.

fileName(입력)

참조된 파일 이름.

count(출력)

출력 목록의 항목 갯수.

tableList(출력)

지정된 파일을 참조하는 테이블 항목의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

/audios/ajones.wav 파일을 참조하는 employee 테이블의 오디오 컬럼 내의 항목을 나열합니다.

```
#include <dmbaudio.h>
long idx;

rc = DBaIsFileReferenced(NULL,
       "/audios/ajones.wav",
       &count, &tableList);
```

DBaIsTableEnabled

Image	Audio	Video
	X	

테이블이 오디오(DB2Audio 데이터)용으로 사용 가능하게 되었는지의 여부를 판별합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaIsTableEnabled(
      char *tableName,
      short *status
      );
```

매개변수

tableName(입력)

테이블 이름.

status(출력)

테이블이 사용 가능한지를 표시합니다. 매개변수는 숫자 값을 리턴합니다. Extender 또한 상태를 표시하는 상수를 리턴합니다. 그 값과 상수는 다음과 같습니다.

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

-1 MMDB_INVALID_DATATYPE

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

employee 테이블이 오디오(DB2Audio 데이터)용으로 사용 가능한지의 여부를 판별합니다.

```
#include <dmbaudio.h>

rc = DBaIsTableEnabled("employee", &status);
```

DBaPlay

Image	Audio	Video
	X	

클라이언트상의 오디오 플레이어를 열고 오디오 클립을 실행시킵니다. 클립은 오디오 컬럼이나 외부 파일에 저장 가능합니다.

- 오디오 클립이 외부 파일에 저장된 경우에 파일 이름이나 오디오 핸들을 API로 전달할 수 있습니다. API는 파일 위치를 결정하기 위해 클라이언트 환경 변수 DB2AUDIOPATH를 사용합니다. 파일은 반드시 클라이언트 워크스테이션로부터 액세스 가능해야 합니다.
- 오디오 클립이 컬럼에 저장되어 있는 경우 반드시 API로 오디오 핸들을 전달해야 합니다. 응용프로그램을 데이터베이스에 연결해야 하며, 오디오 클립이 저장된 테이블에 대한 읽기 액세스 권한이 있어야 합니다.

오디오가 컬럼에 저장된 경우, Extender는 임시 파일을 작성하고 오브젝트의 내용을 컬럼에서 파일로 복사합니다. Extender는 또한 오디오가 외부 파일에 저장되고 상대적인 파일 이름을 환경 변수 값을 사용하여 해결할 수 없는 경우 또는 파일을 클라이언트 머신에서 액세스할 수 없는 경우 임시 파일을 작성할 수도 있습니다. 임시 파일은 DB2AUDIOTEMP 환경 변수에 지정된 디렉토리에 작성됩니다. 그러면 Extender는 임시 파일에서 오디오를 재생합니다.

권한

컬럼에서 오디오 클립을 실행시키는 경우 사용자 테이블에 대한 Select 권한

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

컬럼에 저장된 오디오 재생

```
long DBaPlay(
      char *playerName,
      MMDB_PLAY_HANDLE,
      DB2Audio *audioHandle,
      waitFlag
      );
```

구문

파일로 저장된 오디오 재생

```
long DBaPlay(
      char *playerName,
      MMDB_PLAY_FILE,
      char *fileName,
      waitFlag
      );
```

매개변수

playerName(입력)

오디오 플레이어의 이름. 널(NULL)에 지정된 경우 DB2AUDIOPLAYER 환경 변수에 의해 지정된 기본 오디오 플레이어가 사용됩니다.

MMDB_PLAY_HANDLE(입력)

오디오가 BLOB으로 저장되었음을 표시하는 상수.

MMDB_PLAY_FILE(입력)

오디오가 클라이언트로부터 액세스 가능한 파일로 저장되었음을 표시하는 상수.

audioHandle(입력)

오디오의 핸들. 이 매개변수는 컬럼에서 오디오 컬럼을 실행할 때 반드시 전달되어야 합니다. 오디오 핸들이 외부 파일을 표현하는 경우, 클라이언트 환경 변수 DB2VIDEOPATH가 파일 위치를 결정하기 위해 사용됩니다.

fileName(입력)

오디오를 포함하는 파일 이름.

waitFlag(입력)

계속하기 전에 프로그램이 사용자가 플레이어를 닫기를 기다리는지를 표시하는 상수. MMDB_PLAY_WAIT는 응용프로그램과 같은 스레드에서 플레이어를 실행합니다. MMDB_PLAY_NO_WAIT는 분리된 스레드에서 플레이어를 실행합니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

audioHandle에 의해 식별된 오디오를 재생합니다. 응용프로그램과 같은 스레드에서 기본 플레이어를 실행합니다.

```
#include <dmbaudio.h>

rc = DBaPlay(NULL, MMDB_PLAY_HANDLE,
       audioHandle, MMDB_PLAY_WAIT);
```

DBaPrepareAttrs

Image	Audio	Video
	X	

사용자 제공 오디오 속성을 준비합니다. 이 API는 오디오 오브젝트가 사용자 제공 속성과 함께 저장되거나 갱신될 때 사용됩니다. 서버에서 수행되는 UDF 코드는 항상『big endian』형식의 데이터를 예상하며, 이것은 대부분의 UNIX 플랫폼에서 사용하는 형식입니다. 오디오 오브젝트가『little endian』형식으로 저장되거나 갱신된 경우(즉, 비-UNIX 클라이언트로부터), 저장 또는 갱신 요청을 수행하기 전에 DBaPrepare API를 사용해야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudio.lib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
void DBaPrepareAttrs(
      MMDBAudioAttrs *audAttr
      );
```

매개변수

audAttr(입력)

오디오의 사용자 제공 속성.

예

사용자 제공 오디오 속성을 준비합니다.

```
#include <dmbaudio.h>

DBaPrepareAttrs(&imgattr);
```

DBaReorgMetadata

Image	Audio	Video
	X	

오디오 관련 메타데이터 테이블을 『정리』합니다. 예를 들면,

- 오디오 메타데이터 테이블 내에서 더 이상 사용되지 않는 공간을 회수합니다.
- 더 이상 존재하지 않는 오디오 파일에 대한 오디오 메타데이터 테이블 내의 참조를 삭제합니다.

응용프로그램은 이 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

Alter, Control, SYSADM, SYSCTRL, SYSMAINT, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbaudiolib	libdmbaudio.a(AIX) libdmbaudio.sl(HP-UX) libdmbaudio.so(Solaris)

Include 파일

dmbaudio.h

구문

```
long DBaReorgMetadata(
      char *tableName
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테

이블 이름이 지정된 경우, 정리(cleanup)는 지정된 사용자 테이블과 관련된 오디오 메타데이터 테이블에 대해 수행됩니다. 널(NULL) 값을 지정하면, 현재 사용자 ID가 소유하는 모든 테이블 내의 오디오 컬럼용 메타데이터 테이블이 정리됩니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램은 데이터베이스에 유효한 연결을 갖고 있지 않습니다.

예

employee 테이블 내의 audio 컬럼용 모든 메타데이터 테이블을 정리합니다.

```
#include <dmbaudio.h>

rc = DBaReorgMetadata("employee");
```

DBiAdminGetInaccessibleFiles

Image	Audio	Video
X		

사용자 테이블의 이미지 컬럼 내에서 참조된 액세스 불가능한 파일의 이름을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiAdminGetInaccessibleFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정

된 규정자를 가진 모든 테이블이 탐색됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스 내의 모든 테이블이 탐색됩니다.

count(출력)
출력 목록의 항목 갯수.

fileList(출력)
테이블 내에서 참조되는 액세스 불가능한 파일의 목록.

오류 코드

MMDB_SUCCESS
API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED
응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_NO_AUTH
사용자에게 이 API를 호출할 적절한 권한이 없습니다.

MMDB_RC_MALLOC
시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

사용자 ID rjones가 소유하는 테이블의 이미지 컬럼 내에서 참조된 모든 액세스 불가능한 파일을 나열합니다.

```
#include <dmbimage.h>
long idx;

rc = DBiAdminGetInaccessibleFiles
       ("rjones", &count, &filelist);
```

DBiAdminGetReferencedFiles

Image	Audio	Video
X		

사용자 테이블의 이미지 컬럼 내에서 참조된 파일의 이름을 리턴합니다. 파일을 액세스할 수 없는 경우(예를 들어, 환경 변수 스펙을 사용하여 파일 이름을 해결할 수 없음), 파일 이름에 별표(*)가 선행됩니다. 이 API는 FILEREF 데이터 구조의 FILENAME 필드를 사용하지 않으므로 이것을 널(NULL)로 설정합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiAdminGetReferencedFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정된 규정자를 가진 모든 테이블이 탐색됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스의 모든 테이블이 탐색됩니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블 내에서 참조되는 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_NO_AUTH

사용자에게 이 API를 호출할 적절한 권한이 없습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

ajones가 소유하는 테이블의 이미지 컬럼 내에서 참조된 모든 파일을 나열합니다.

```
#include <dmbimage.h>
long idx;

rc = DBiAdminGetReferencedFiles("ajones",
       &count, &filelist);
```

DBiAdminIsFileReferenced

Image	Audio	Video
X		

지정된 파일을 참조하는 사용자 테이블에 있는 이미지 컬럼 항목의 목록을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiAdminIsFileReferenced(
      char *qualifier,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정된 규정자를 가진 모든 테이블이 탐색됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스의 모든 테이블이 검색됩니다.

fileName(입력)

참조된 파일 이름.

count(출력)

출력 목록의 항목 갯수.

tableList(출력)

지정된 파일을 참조하는 테이블 항목의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_NO_AUTH

사용자는 이 API를 호출할 적절한 권한이 없습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

/images/asmith.bmp 파일을 참조하는 현재 데이터베이스 내의 모든 테이블에 있는 이미지 컬럼 항목을 나열합니다.

```
#include <dmbimage.h>
long idx;

rc = DBiAdminIsFileReferenced(NULL,
       "/images/asmith.bmp",
       &count, &tableList);
```

DBiAdminReorgMetadata

Image	Audio	Video
X		

이미지 관련 메타데이터 테이블을 『정리(cleanup)』합니다.

- 이미지 메타데이터 테이블 내에서 더이상 사용되지 않는 공간을 회수합니다.
- 더이상 존재하지 않는 이미지 파일에 대한 이미지 메타데이터 테이블 내의 조회를 삭제합니다.

응용프로그램은 이 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiAdminReorgMetadata(
      char *qualifier
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정

된 규정자를 가진 모든 테이블은 정리됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스 내의 모든 테이블은 정리됩니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램은 데이터베이스에 유효한 연결을 갖고 있지 않습니다.

MMDB_RC_NO_AUTH

사용자는 이 API를 호출할 적절한 권한이 없습니다.

예

사용자 ID rsmith가 소유하는 테이블 내의 이미지 컬럼에 대한 메타데이터 테이블을 정리합니다.

```
#include <dmbimage.h>

rc = DBiAdminReorgMetadata("rsmith");
```

DBiBrowse

Image	Audio	Video
X		

클라이언트상에 이미지 브라우저를 열고 이미지를 표시합니다. 이미지는 이미지 컬럼이나 외부 파일에 저장 가능합니다.

- 이미지가 외부 파일에 저장된 경우에 파일 이름이나 이미지 핸들을 API로 전달할 수 있습니다. API는 파일 위치를 결정하기 위해 클라이언트 환경 변수 DB2IMAGEPATH를 사용합니다. 파일은 반드시 클라이언트 워크스테이션으로부터 액세스 가능해야 합니다.
- 이미지가 컬럼에 저장되어 있는 경우 반드시 API로 이미지 핸들을 전달해야 합니다. 응용프로그램을 데이터베이스에 연결해야 하며 이미지가 저장된 테이블에 대한 읽기 액세스 권한이 있어야 합니다.

브라우저가 직접 이미지에 액세스할 수 없는 경우, Extender가 DB2IMAGETEMP 환경 변수에 지정된 디렉토리에 임시 파일을 작성합니다. 그러면 Extender는 임시 파일에서 이미지를 표시합니다.

권한

컬럼에서 이미지를 열람하는 경우 사용자 테이블에 대한 Select 권한

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

컬럼에 저장된 이미지 열람

```
long DBiBrowse(
      char *browserName,
      MMDB_PLAY_HANDLE,
      DB2Image *imageHandle,
      waitFlag
      );
```

구문

파일로 저장된 이미지 열람

```
long DBiBrowse(
      char *browserName,
      MMDB_PLAY_FILE,
      char *fileName,
      waitFlag
      );
```

매개변수

browserName(입력)

이미지 브라우저의 이름. 널(NULL)에 지정된 경우, DB2IMAGEBROWSER 환경 변수에 의해 지정된 기본 이미지 브라우저가 사용됩니다.

MMDB_PLAY_HANDLE(입력)

이미지가 BLOB으로 저장되었음을 표시하는 상수.

MMDB_PLAY_FILE(입력)

이미지가 클라이언트로부터 액세스 가능한 파일로 저장되었음을 표시하는 상수.

imageHandle(입력)

이미지의 핸들. 이 매개변수는 컬럼에서 이미지를 열람할 때 반드시 전달되어야 합니다. 이미지 핸들이 외부 파일을 표현할 경우, 파일 위치를 결정하기 위해 클라이언트 환경 변수 DB2IMAGEPATH가 사용됩니다.

fileName(입력)

이미지를 포함하는 파일 이름.

waitFlag(입력)

계속하기 전에 프로그램이 사용자가 브라우저를 닫기를 기다리는지를 표

시하는 상수. MMDB_PLAY_WAIT는 응용프로그램과 같은 스레드에서 브라우저를 실행합니다. MMDB_PLAY_NO_WAIT는 분리된 스레드에서 브라우저를 실행합니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

imageHandle에 의해 식별된 이미지를 재생합니다. 응용프로그램과 같은 스레드에서 기본 브라우저를 실행합니다.

```
#include <dmbimage.h>

rc = DBiBrowse(NULL, MMDB_PLAY_HANDLE,
       imageHandle, MMDB_PLAY_WAIT);
```

DBiDisableColumn

Image	Audio	Video
X		

이미지(DB2Image 데이터)용으로 컬럼을 사용 불가능하게 하여 이미지 데이터를 수용하지 못하게 합니다. 컬럼 항목의 내용은 널(NULL)에 설정되며 이 컬럼과 연관된 메타데이터는 삭제됩니다. 이 컬럼과 연관된 QBIC 카탈로그가 또한 삭제됩니다. 이 컬럼으로 Image Extender에 의해 정의된 모든 트리거 역시 삭제됩니다. 새 행은 사용 불가능하게 된 컬럼을 포함하는 테이블에 포함될 수 있으며, 새 행은 유형 DB2Image으로 정의된 데이터를 포함할 수도 있으나 새 행과 관련된 아무런 메타데이터(관리 지원 테이블 내)도 없습니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiDisableColumn(
      char *tableName,
      char *colName,
      );
```

매개변수

tableName(입력)

이미지 컬럼을 포함할 테이블 이름.

colName(입력)

이미지 컬럼의 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

이미지(DB2Image 데이터)용으로 employee 테이블의 picture 컬럼을 사용 가능하게 합니다.

```
#include <dmbimage.h>

rc = DBiDisableColumn("employee",
       "picture");
```

DBiDisableDatabase

Image	Audio	Video
X		

이미지(DB2Image 데이터)용으로 데이터베이스를 사용 불가능화하여 이미지 데이터를 수용하지 못하게 합니다. DB2Image용으로 정의된 데이터베이스 내의 모든 테이블도 역시 사용 불가능화됩니다. 데이터베이스 서버용으로 Image Extender에서 정의한 메타데이터 및 UDF가 삭제됩니다. 새 행은 유형 DB2Image으로 정의된 데이터베이스 내 테이블에 삽입될 수도 있으나, 새 행과 관련된 아무런 메타데이터(관리 지원 테이블 내)도 없습니다.

권한

DBADM, SYSADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiDisableDatabase(
      );
```

매개변수

DBiDisableDatabase에는 매개변수가 없습니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램은 데이터베이스에 유효한 연결을 갖고 있지 않습니다.

예

이미지(DB2Image 데이터)용으로 현재 데이터베이스를 사용 불가능화합니다.

```
#include <dmbimage.h>

rc = DBiDisableDatabase();
```

DBiDisableTable

Image	Audio	Video
X		

이미지(DB2Image 데이터)용으로 테이블을 사용 불가능하게 하여 이미지 데이터를 수용하지 못하게 합니다. DB2Image용으로 정의된 테이블 내의 모든 컬럼도 역시 사용 불가능화됩니다. 테이블용으로 Image Extender에 의해 정의된 메타데이터의 일부가 삭제됩니다. 테이블의 이미지 컬럼과 연관된 모든 QBIC 카탈로그가 또한 삭제됩니다. 새 행은 유형 DB2Image으로 정의된 테이블에 삽입될 수도 있으나 새 행과 관련된 아무런 메타데이터(관리 지원 테이블 내)도 없습니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiDisableTable(
      char *tableName
      );
```

매개변수

tableName(입력)

이미지 컬럼을 포함하는 테이블 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

이미지(DB2Image 데이터)용으로 employee 테이블을 사용 불가능하게 합니다.

```
#include <dmbimage.h>

rc = DBiDisableTable("employee");
```

DBiEnableColumn

Image	Audio	Video
X		

이미지(DB2Image 데이터)용으로 컬럼을 사용 가능하게 합니다. API는 이 컬럼과 메타데이터 테이블 사이의 관계를 정의하고 관리합니다. 이 API를 호출하기 전에, 응용프로그램을 데이터베이스에 연결하고 사용자 테이블을 확약해야 합니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiEnableColumn(
      char *tableName,
      char *colName,
      );
```

매개변수

tableName(입력)

이미지 컬럼을 포함할 테이블 이름.

colName(입력)

이미지 컬럼의 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED

컬럼은 이미 사용 가능하게 되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_WRONG_SIGNATURE

지정된 데이터 유형 컬럼이 올바르지 않습니다. 사용자 정의 유형 MMDBSYS.DB2IMAGE가 예상됩니다.

MMDB_RC_COLUMN_DOESNOT_EXIST

컬럼이 지정된 테이블 내에서 지정되지 않았습니다.

MMDB_RC_NOT_ENABLED

데이터베이스 또는 테이블을 사용할 수 없습니다.

예

이미지용으로 employee 테이블의 picture 컬럼을 사용 가능하게 합니다.

```
#include <dmbimage.h>

rc = DBiEnableColumn("employee",
       "picture");
```

DBiEnableDatabase

Image	Audio	Video
X		

이미지(DB2Image 데이터)용으로 데이터베이스를 사용 가능하게 합니다. 이 API는 데이터베이스당 한번씩 호출됩니다. 이것은 DB2 사용자 정의 유형, DB2Image를 데이터베이스 관리 프로그램에 정의합니다. 이것은 역시 DB2Image 데이터를 조작하는 모든 UDF를 생성합니다.

권한

DBADM, SYSADM, SYSCTRL

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiEnableDatabase(
      char *tableSpace
      );
```

매개변수

tableSpace(입력)

관리 테이블이 저장된 컨테이너의 집합인 테이블 공간의 명칭. 테이블 공간 스펙은 *datats, indexts, longts*의 세 부분으로 되어 있으며, 여기서 *datats*는 메타데이터 테이블이 생성되는 테이블 공간입니다. *indexts*는 메타데이터 테이블에 대한 색인이 생성되는 테이블 공간입니다. *longts*는

(LONG VARCHAR 및 LOB 데이터 유형을 포함한 것과 같은) 메타데이터 테이블 내의 긴 컬럼의 값이 저장된 테이블 공간입니다. 테이블 공간 스펙의 어떤 부분에 대해 널(NULL) 값을 제공하는 경우에 그 부분에 대한 기본 테이블 공간이 사용됩니다.

EEE 전용: Extender에 대해 데이터베이스를 사용 가능화할 때 지정한 테이블 공간은 파티션된 데이터베이스 시스템의 모든 노드가 포함되는 노드 그룹에 정의되어 있어야 합니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED

데이터베이스는 이미 사용 가능하게 되었습니다.

MMDB_RC_API_NOT_SUPPORTED_FOR_SERVER

연결되는 서버는 이 명령을 지원하지 않습니다.

MMDB_WARN_NOT_ALL_NODES

지정된 테이블 공간이 해당 Extender와 연관된 노드를 모두 포함하지는 않습니다. **(EEE 전용)**

MMDB_RC_NOT_SAME_NODEGROUP

지정된 테이블 공간이 동일한 노드 그룹에 있지 않습니다**(EEE 전용)**.

예

MYTS로 이름 지정된 테이블 공간 내 이미지(DB2Video 데이터)용으로 현재 데이터베이스를 사용 가능하게 합니다. 색인 및 긴 테이블 공간용으로 기본값을 사용합니다.

```
#include <dmbimage.h>

rc = DBiEnableDatabase("myts,,");
```

이미지(DB2Image 데이터)용으로 현재 데이터베이스를 사용 가능하게 합니다. 기본값 테이블 공간을 사용합니다.

```
#include <dmbimage.h>

rc = DBiEnableDatabase(NULL);
```

DBiEnableTable

Image	Audio	Video
X		

이미지(DB2Image 데이터)용으로 테이블을 사용 가능하게 합니다. 이 API는 테이블당 한번씩 호출됩니다. 테이블 내의 이미지 컬럼용 속성을 저장하고 관리하기 위해 메타데이터 테이블을 생성합니다. 잠금 가능성을 방지하려면 응용프로그램은 반드시 이 API를 호출하기 전에 트랜잭션을 확약해야 합니다. 이 API를 호출하기 전에, 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiEnableTable(
      char *tableSpace,
      char *tableName
      );
```

매개변수

tableSpace(입력)

관리 테이블이 저장되는 컨테이너의 집합인 테이블 공간의 이름. 테이블 공간 스펙은 *datats, indexts, longts*의 세 부분으로 되어 있으며, 여기서

*datats*는 메타데이터 테이블이 생성되는 테이블 공간입니다. *indexts*는 메타데이터 테이블에 대한 색인이 생성되는 테이블 공간입니다. *longts*는 (LONG VARCHAR 및 LOB 데이터 유형을 포함한 것과 같은) 메타데이터 테이블 내의 긴 컬럼의 값이 저장된 테이블 공간입니다.

테이블 공간 스펙의 임의의 부분에 대해 널(NULL) 값을 제공하는 경우, 이 부분에 대해 기본 테이블 공간이 사용됩니다.

EEE 전용: 지정된 테이블 공간은 사용자 테이블과 동일한 노드 그룹에 있어야 합니다.

tableName(입력)
이미지 컬럼을 포함할 테이블 이름.

오류 코드

MMDB_SUCCESS
API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH
호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED
테이블은 이미 사용 가능하게 되었습니다.

MMDB_RC_NOT_CONNECTED
응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_TABLE_DOESNOT_EXIST
테이블이 존재하지 않습니다.

MMDB_RC_TABLESPACE_NOT_SAME_NODEGROUP
지정된 테이블 공간이 사용자 테이블과 동일한 노드 그룹에 있지 않습니다. **(EEE 전용)**

예

테이블 공간 MYTS에서 이미지(DB2Image 데이터)용으로 employee 테이블을 사용 가능화합니다. 색인 및 긴 테이블 공간에 대해 기본값을 사용합니다.

```
#include <dmbimage.h>

rc = DBiEnableTable("myts,,",
       "employee");
```

이미지(DB2Image 데이터)용 employee 테이블을 사용할 수 있게 합니다. 기본 테이블 공간을 사용합니다.

```
#include <dmbimage.h>

rc = DBiEnableTable(NULL,
       "employee");
```

DBiGetError

Image	Audio	Video
X		

마지막 오류의 설명을 리턴합니다. 기타 다른 API가 오류 코드를 리턴한 뒤에 이 API를 호출하십시오.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiGetError(
      SQLINTEGER *sqlcode,
      char *errorMsgText
      );
```

매개변수

sqlcode(출력)

일반적인 SQL 오류 코드.

errorMsgText(출력)

SQL 오류 메시지 텍스트.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

예

SQL 오류 코드를 errCode에 저장하고 메시지 텍스트를 errMsg에 저장하여 마지막 오류를 확보합니다.

```
#include <dmbimage.h>

rc = DBiGetError(&errCode, &errMsg);
```

DBiGetInaccessibleFiles

Image	Audio	Video
X		

사용자 테이블의 이미지 컬럼 내에서 참조된 액세스 불가능한 파일의 이름을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

모든 탐색된 사용자 테이블 및 연관된 관리 지원 테이블에 있는 사용 가능한 이미지 컬럼에 관한 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiGetInaccessibleFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테이블 이름이 지정된 경우 그 테이블은 액세스 불가능한 파일에 대한 참조용으로 탐색됩니다. 널(NULL) 값이 지정되면 지정된 규정자와 함께 모든 테이블은 탐색됩니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블 내에서 참조되는 액세스 불가능한 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

employee 테이블의 이미지 컬럼 내에서 참조된 모든 액세스 불가능한 파일을 나열합니다.

```
#include <dmbimage.h>
long idx;

rc = DBiGetInaccessibleFiles("employee",
       &count, &filelist);
```

DBiGetReferencedFiles

Image	Audio	Video
X		

사용자 테이블의 이미지 컬럼 내에서 참조된 파일의 이름을 리턴합니다. 파일을 액세스할 수 없는 경우(예를 들어, 환경 변수 스펙을 사용하여 파일 이름을 해결할 수 없음), 파일 이름에 별표(*)가 선행됩니다. 이 API는 FILEREF 데이터 구조의 FILENAME 필드를 사용하지 않으므로 이것을 널(NULL)로 설정합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

모든 탐색된 사용자 테이블 및 연관된 관리 지원 테이블에 있는 사용 가능한 이미지 컬럼에 관한 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiGetReferencedFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테이블 이름이 지정된 경우 그 테이블은 파일에 참조용으로 탐색됩니다. 널(NULL) 값을 지정하는 경우, 현재 사용자 ID가 소유한 모든 테이블을 탐색합니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블 내에서 참조되는 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

employee 테이블의 이미지 컬럼 내에서 참조된 모든 파일을 나열합니다.

```
#include <dmbimage.h>
long idx;

rc = DBiGetReferencedFiles("employee",
       &count, &filelist);
```

DBiIsColumnEnabled

Image	Audio	Video
X		

컬럼이 이미지(DB2Image 데이터)용으로 사용 가능하게 되었는지의 여부를 판별합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

사용자 테이블에 관한 SYSADM, DBADM, 테이블 소유자 또는 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiIsColumnEnabled(
      char *tableName,
      char *colName,
      short *status
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 테이블 이름.

colName(입력)

컬럼의 이름.

status(출력)

컬럼이 사용 가능한지를 표시합니다. 매개변수는 숫자 값을 리턴합니다. Extender 또한 상태를 표시하는 상수를 리턴합니다. 그 값과 상수는 다음과 같습니다.

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

-1 MMDB_INVALID_DATATYPE

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED

컬럼은 이미 사용 가능하게 되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

employee 테이블의 picture 컬럼이 이미지용으로 사용 가능한지의 여부를 판별합니다.

```
#include <dmbimage.h>

rc = DBiIsColumnEnabled("employee",
       "picture", &status);
```

DBiIsDatabaseEnabled

Image	Audio	Video
X		

데이터베이스가 이미지(DB2Image 데이터)용으로 사용 가능해졌는지의 여부를 판별합니다. 응용프로그램은 이 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiIsDatabaseEnabled(
      short *status
      );
```

매개변수

status(출력)

데이터베이스가 사용 가능한지를 표시합니다. 매개변수는 숫자 값을 리턴합니다. Extender 또한 상태를 표시하는 상수를 리턴합니다. 그 값과 상수는 다음과 같습니다.

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램은 데이터베이스에 유효한 연결을 갖고 있지 않습니다.

예

personnl 데이터베이스가 이미지용으로 사용 가능한지를 판별합니다.

```
#include <dmbimage.h>

rc = DBiIsDatabaseEnabled(&status);
```

DBiIsFileReferenced

Image	Audio	Video
X		

지정된 파일을 참조하는 이미지 컬럼 내 테이블 항목의 목록을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

모든 탐색된 사용자 테이블 및 연관된 관리 지원 테이블에 있는 사용 가능한 이미지 컬럼에 관한 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiIsFileReferenced(
      char *tableName,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테이블 이름이 지정된 경우 그 테이블은 지정된 파일에 대한 참조용으로 탐색됩니다. 널(NULL) 값을 지정하는 경우, 현재 사용자 ID가 소유한 모든 테이블을 탐색합니다.

fileName(입력)

참조된 파일 이름.

count(출력)

출력 목록의 항목 갯수.

tableList(출력)

지정된 파일을 참조하는 테이블 항목의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

/images/ajones.bmp 파일을 참조하는 employee 테이블의 이미지 컬럼에 있는 항목을 나열합니다.

```
#include <dmbimage.h>
long idx;

rc = DBiIsFileReferenced(NULL,
       "/images/ajones.bmp",
       &count, &tableList);
```

DBiIsTableEnabled

Image	Audio	Video
X		

테이블이 이미지(DB2Image 데이터)용으로 사용 가능하게 되었는지의 여부를 판별합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiIsTableEnabled(
      char *tableName,
      short *status
      );
```

매개변수

tableName(입력)

테이블 이름.

status(출력)

테이블이 사용 가능한지를 표시합니다. 매개변수는 숫자 값을 리턴합니다. Extender 또한 상태를 표시하는 상수를 리턴합니다. 그 값과 상수는 다음과 같습니다.

1 MMDB_IS_ENABLED

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

employee 테이블이 이미지용으로 사용 가능한지의 여부를 판별합니다.

```
#include <dmbimage.h>

rc = DBiIsTableEnabled("employee",
       &status);
```

DBiPrepareAttrs

Image	Audio	Video
X		

사용자 제공 이미지 속성을 준비합니다. 이 API는 이미지 오브젝트가 사용자 제공 속성으로 저장되거나 갱신될 때 사용됩니다. 서버에서 수행되는 UDF 코드는 항상『big endian』형식의 데이터를 예상하며, 이것은 대부분의 UNIX 플랫폼에서 사용하는 형식입니다. 이미지 오브젝트가『little endian』형식으로 저장되거나 갱신된 경우(즉, 비-UNIX 클라이언트로부터), 저장 또는 갱신 요청을 수행하기 전에 DBiPrepare API를 사용해야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
void DBiPrepareAttrs(
      MMDBImageAttrs *imgAttr
      );
```

매개변수

imgAttr(입력)

이미지의 사용자 제공 속성.

예

사용자 제공 이미지 속성을 준비합니다.

```
#include <dmbimage.h>

DBiPrepareAttrs(&imgattr);
```

DBiReorgMetadata

Image	Audio	Video
X		

이미지 관련 메타데이터 테이블을『정리』합니다. 예를 들면,

- 이미지 메타데이터 테이블 내에서 더이상 사용되지 않는 공간을 회수합니다.
- 더이상 존재하지 않는 이미지 파일에 대한 이미지 메타데이터 테이블 내의 참조를 삭제합니다.

응용프로그램은 이 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

Alter, Control, SYSADM, SYSCTRL, SYSMAINT, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbimage.lib	libdmbimage.a(AIX) libdmbimage.sl(HP-UX) libdmbimage.so(Solaris)

Include 파일

dmbimage.h

구문

```
long DBiReorgMetadata(
      char *tableName
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테

이블 이름이 지정된 경우, 정리는 지정된 사용자 테이블과 관련된 이미지 메타데이터 테이블에 대해 수행됩니다. 널(NULL) 값을 지정하면, 현재 사용자 ID가 소유하는 모든 테이블 내의 이미지 컬럼용 메타데이터 테이블이 정리됩니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램은 데이터베이스에 유효한 연결을 갖고 있지 않습니다.

예

employee 테이블 내의 이미지 컬럼용 메타테이블을 정리합니다.

```
#include <dmbimage.h>

rc = DBiReorgMetadata("employee");
```

DBvAdminGetInaccessibleFiles

Image	Audio	Video
		X

사용자 테이블의 비디오 컬럼 내에서 참조된 액세스 불가능한 파일의 이름을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvAdminGetInaccessibleFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정되면 지정된 규

정자를 갖는 모든 테이블이 탐색됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스의 모든 테이블이 탐색됩니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블 내에서 참조되는 액세스 불가능한 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_NO_AUTH

사용자에게 이 API를 호출할 적절한 권한이 없습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

사용자 ID rsmith가 소유하는 테이블의 비디오 컬럼 내에서 참조된 모든 액세스 불가능한 파일을 나열합니다.

```
#include <dmbvideo.h>
long idx;

rc = DBvAdminGetInaccessibleFiles
       ("rsmith", &count,
       &filelist);
```

DBvAdminGetReferencedFiles

Image	Audio	Video
		X

사용자 테이블의 비디오 컬럼 내에서 참조된 파일의 이름을 리턴합니다. 파일을 액세스할 수 없는 경우(예를 들어, 환경 변수 스펙을 사용하여 파일 이름을 해결할 수 없음), 파일 이름에 별표(*)가 선행됩니다. 이 API는 FILEREF 데이터 구조의 FILENAME 필드를 사용하지 않으므로 이것을 널(NULL)로 설정합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvAdminGetReferencedFiles(
      char *qualifier,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정된 규정자를 갖는 모든 테이블이 탐색됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스의 모든 테이블이 탐색됩니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블 내에서 참조되는 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_NO_AUTH

사용자에게 이 API를 호출할 적절한 권한이 없습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

ajones가 소유하는 테이블의 비디오 컬럼 내에서 참조된 모든 파일을 나열합니다.

```
#include <dmbvideo.h>
long idx;

rc = DBvAdminGetReferencedFiles
       ("ajones", &count,
       &filelist);
```

DBvAdminIsFileReferenced

Image	Audio	Video
		X

지정된 파일을 참조하는 사용자 테이블에 있는 비디오 컬럼 항목의 목록을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 자원을 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvAdminIsFileReferenced(
      char *qualifier,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정된 규정자를 가진 모든 테이블이 탐색됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스의 모든 테이블이 참색됩니다.

fileName(입력)

참조된 파일 이름.

count(출력)

출력 목록의 항목 갯수.

tableList(출력)

지정된 파일을 참조하는 테이블 항목의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_NO_AUTH

사용자는 이 API를 호출할 적절한 권한이 없습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

/videos/asmith.bmp 파일을 참조하는 현재 데이터베이스 내의 모든 테이블에 있는 이미지 컬럼 항목을 나열합니다.

```
#include <dmbvideo.h>
long idx;

rc = DBvAdminIsFileReferenced(NULL,
       "/videos/asmith.mpg",
       &count, &tableList);
```

DBvAdminReorgMetadata

Image	Audio	Video
		X

비디오 관련 메타데이터 테이블을 『정리』합니다. 예를 들면,

- 비디오 메타데이터 테이블 내에서 더이상 사용되지 않는 공간을 회수합니다.
- 더이상 존재하지 않는 비디오 파일에 대한 비디오 메타데이터 테이블 내의 참조를 삭제합니다.

응용프로그램은 이 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

SYSADM, SYSCTRL, SYSMAINT

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvAdminReorgMetadata(
      char *qualifier
      );
```

매개변수

qualifier(입력)

유효한 사용자 ID 또는 널(NULL) 값. 사용자 ID가 지정된 경우에 지정

된 규정자를 가진 모든 테이블이 정리됩니다. 널(NULL) 값이 지정되면 현재 데이터베이스 내의 모든 테이블이 정리됩니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_NO_AUTH

사용자는 이 API를 호출할 적절한 권한이 없습니다.

예

사용자 ID rsmith가 소유하는 테이블 내의 비디오 컬럼에 대한 메타데이터 테이블을 정리합니다.

```
#include <dmbvideo.h>

rc = DBvAdminReorgMetadata("rsmith");
```

DBvBuildStoryboardFile

Image	Audio	Video
		X

컷 카탈로그 파일을 생성하고 파일 내에 비디오 내의 모든 컷용 항목을 만듭니다. 소스 비디오는 데이터베이스나 파일 내에 있을 수 있습니다. 각 컷용으로, API는 컷 번호, 시작 프레임 번호, 끝 프레임 번호 및 적어도 하나의 대표 프레임용 정보를 저장합니다. DBvStoryboardCtrl 데이터 구조 내의 값이 얼마나 많은 대표 프레임이 컷을 위해 식별되는지를 결정합니다. 그 길이가 DBvStoryboardCtrl 내의 임계값 미만인 컷용으로 API는 하나의 대표 프레임을 식별합니다. 길이가 DBvStoryboardCtrl 내의 하위와 상위 임계값 사이인 컷용으로 API는 두 개의 대표 프레임을 식별합니다. 길이가 DBvStoryboardCtrl 내의 상위 임계값을 초과하는 컷용으로, API는 세 개의 대표 프레임을 식별합니다. 대표 프레임 정보는 프레임의 번호와 프레임 내용을 포함하고 있는 파일의 이름을 포함합니다. 이 정보는 스토리보드 즉, 비디오의 시각적 요약을 표시하기 위해 사용될 수 있습니다.

권한

Insert, Control

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvBuildStoryboardFile(
      char *fileName,
      DBvIOType *video,
      DBvShotControl *shotCtrl,
      DBvStoryBoardCtrl *sbCtrl
            );
```

매개변수

catalogName(입력)

컷 카탈로그 파일의 이름에 대한 포인터.

video(입력)

비디오 구조에 대한 포인터.

shotCtrl(입력)

컷 제어 구조에 대한 포인터.

sbCtrl(입력)

스토리보드 제어 구조에 대한 포인터.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_INVALID_CATALOG

카탈로그가 유효하지 않거나 존재하지 않습니다.

예

hotshots이라는 컷 카탈로그 파일을 생성하고 비디오 내의 모든 컷에 대한 데이터를 입력합니다.

```
#include <dmbshot.h>

rc = DBvBuildStoryboardFile("hotshots",
       video, &shotCtrl, &sbCtrl);
```

DBvBuildStoryboardTable

Image	Audio	Video
		X

컷 카탈로그 내에 비디오 내의 모든 컷용 항목을 만듭니다. 소스 비디오는 데이터베이스나 파일 내에 있을 수 있습니다. 컷 카탈로그는 데이터베이스 내에 있습니다. 각 컷용으로 API는 핸들 또는 소스 비디오용 파일 정보를 저장합니다. 또한 컷 번호, 시작 프레임 번호, 끝 프레임 번호 및 적어도 하나의 대표 프레임용 정보를 저장합니다. DBvStoryboardCtrl 데이터 구조 내의 값이 얼마나 많은 대표 프레임이 컷을 위해 식별되는지를 결정합니다. 그 길이가 DBvStoryboardCtrl 내의 임계값 미만인 컷용으로 API는 하나의 대표 프레임을 식별합니다. 길이가 DBvStoryboardCtrl 내의 하위와 상위 임계값 사이인 컷용으로 API는 두 개의 대표 프레임을 식별합니다. 길이가 DBvStoryboardCtrl 내의 상위 임계값을 초과하는 컷용으로, API는 세 개의 대표 프레임을 식별합니다. 대표 프레임 정보는 그 프레임의 번호와 프레임 데이터를 포함합니다. 컷 카탈로그에 저장된 대표 프레임 정보는 스토리보드 즉, 비디오의 시각적 요약을 표시하기 위해 사용될 수 있습니다.

응용프로그램은 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

Insert, Control

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvBuildStoryboardTable(
      char *catalogName,
      DBvIOType *video,
      DBvShotControl *shotCtrl,
      DBvStoryBoardCtrl *sbCtrl,
      SQLHDBC hdbc
      );
```

매개변수

catalogName(입력)

컷 카탈로그의 이름에 대한 포인터.

video(입력)

비디오 구조에 대한 포인터.

shotCtrl(입력)

컷 제어 구조에 대한 포인터.

sbCtrl(입력)

스토리보드 제어 구조에 대한 포인터.

hdbc(입력)

SQL 연결로부터 데이터베이스 핸들.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_INVALID_CATALOG

카탈로그가 유효하지 않거나 존재하지 않습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

비디오용으로 hotshot이라는 이름의 컷 카탈로그 내에 항목을 생성합니다.

```
#include <dmbshot.h>

rc = DBvBuildStoryboardTable("hotshots",
       video, &shotCtrl, &sbCtrl, hdbc);
```

DBvClose

Image	Audio	Video
		X

장면 변경 검출을 위해 열렸던 비디오 파일을 닫습니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbmpeg.lib	libdmbmpeg.a(AIX) libdmbmpeg.sl(HP-UX) libdmbmpeg.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvClose(
      DB2vIOType *video
      );
```

매개변수

video(입력)

비디오 구조에 대한 포인터.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_CANNOT_CLOSE

비디오 파일을 닫을 수 없었습니다.

예

비디오 장면 변경 검출용으로 이전에 열렸던 비디오 파일을 닫습니다.

```
#include <dmbshot.h>

rc = DBvClose(video);
```

DBvCreateIndex

Image	Audio	Video
		X

파일 내에 저장된 비디오에 대한 색인을 생성합니다. 색인은 Video Extender에 의해 비디오 내의 컷과 프레임에 액세스하기 위해 사용됩니다. 색인은 소스 비디오 파일과 동일한 디렉토리 내에 플랫 파일로 저장됩니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbmpeg.lib	libdmbmpeg.a(AIX) libdmbmpeg.sl(HP-UX) libdmbmpeg.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvCreateIndex(
      char *fileName
      );
```

매개변수

fileName(입력)

비디오 파일 이름에 대한 포인터.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_OPEN_VIDEO

비디오 파일을 처리용으로 열 수 없었습니다.

MMDB_RC_INDEX_FAIL

색인을 빌드할 수 없었습니다.

예

\videos\ajones.mpg 파일 내에 비디오용 색인을 만듭니다.

```
#include <dmbshot.h>

rc = DBvCreateIndex("\videos\ajones.mpg");
```

DBvCreateIndexFromVideo

Image	Audio	Video
		X

비디오용으로 색인을 생성합니다. 비디오는 반드시 컷 검출용으로 먼저 열려야 합니다. 색인은 Video Extender에 의해 비디오 내의 컷과 프레임에 액세스하기 위해 사용됩니다. 색인은 플랫 파일에 저장됩니다. 파일 이름은 DBvIOType 데이터 구조 내에 저장됩니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvCreateIndexFromVideo(
      DBvIOType *video
      );
```

매개변수

video(갱신)

비디오 구조에 대한 포인터.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_OPEN_VIDEO

비디오 파일을 처리용으로 열 수 없었습니다.

MMDB_RC_INDEX_FAIL

색인을 빌드할 수 없었습니다.

예

비디오용으로 색인을 생성합니다.

```
#include <dmbshot.h>

rc = DBvCreateIndexFromVideo(video);
```

DBvCreateShotCatalog

Image	Audio	Video
		X

프레임 번호와 같은 컷에 대한 정보를 포함하는 테이블의 집합인 컷 카탈로그를 생성합니다.

응용프로그램은 반드시 db2video 및 db2image용으로 사용 가능한 데이터베이스에 연결되어야 합니다.

권한

Create, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvCreateShotCatalog(
      char *catalogName,
      SQLHDBC hdbc
      );
```

매개변수

catalogName(입력)

생성되는 컷 카탈로그의 이름.

hdbc(입력)

SQL 연결로부터 데이터베이스 핸들.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

hotshots이라고 이름 지정된 컷 카탈로그를 생성합니다.

```
#include <dmbshot.h>

rc = DBvCreateShotCatalog("hotshots", hdbc);
```

DBvDeleteShot

Image	Audio	Video
		X

카탈로그로부터 컷을 삭제합니다.

권한

Insert, Control

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvDeleteShot(
      char *catalogName ,
      char *shotHandle,
      SQLHDBC hdbc
      );
```

매개변수

catalogName(입력)

카탈로그의 이름.

shotHandle(입력)

컷 핸들.

hdbc(입력)

SQL 연결로부터 데이터베이스 핸들.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_ACCESS

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_INVALID_CATALOG

카탈로그가 유효하지 않거나 존재하지 않습니다.

예

hotshots 카탈로그에서 컷의 핸들을 사용하여 컷을 삭제합니다.

```
#include <dmbshot.h>

rc = DBvDeleteShot("hotshots", shot,
       hdbc);
```

DBvDeleteShotCatalog

Image	Audio	Video
		X

컷 카탈로그를 삭제합니다.

권한

Control, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvDeleteShotCatalog(
      char *catalogName,
      SQLHDBC hdbc
      );
```

매개변수

catalogName(입력)

삭제되는 컷 카탈로그의 이름.

hdbc(입력)

SQL 연결로부터 데이터베이스 핸들.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_ACCESS

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_INVALID_CATALOG

카탈로그가 유효하지 않거나 존재하지 않습니다.

예

컷 카탈로그 hotshots을 삭제합니다.

```
#include <dmbshot.h>

rc = DBvDeleteShotCatalog("hotshots",
       hdbc);
```

DBvDetectShot

Image	Audio	Video
		X

비디오 파일 내의 다음 컷을 탐색합니다. 컷이 검출되면 검출된 컷 내의 첫번째 프레임의 프레임 번호와 프레임 데이터를 기록합니다. 컷이 검출되었는지를 확인하려면 반드시 shotDetected 플래그를 검사해야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvDetectShot(
      DBvIOType *video,
      unsigned long *start_frame,
      char *shotDetected,
      DBvShotControl *shotCtrl,
      DBvShotType *shot,
            );
```

매개변수

video(갱신)

비디오 구조에 대한 포인터.

start_frame(입출력)

탐색 시작점으로 사용되는 프레임 번호. 리턴시, 이 매개변수는 그 다음 컷을 찾기 시작하는 위치로 갱신됩니다.

shotDetected(출력)

컷 검출 플래그: 1= 프레임이 검출되었습니다, 0= 프레임이 검출되지 않았습니다.

shotCtrl(입력)

컷 제어 데이터에 대한 포인터.

shot(출력)

검출된 컷 및 컷 데이터에 대한 포인터.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_EOF

파일의 끝에 도달.

MMDB_NO_INDEX

비디오 색인이 존재하지 않습니다.

예

프레임 1에서부터 비디오 파일 내의 다음 컷을 탐색합니다.

```
#include <dmbshot.h>

long start_frame=1;

rc = DBvDetectShot(video, start_frame&Detected,
       &shotCtrl, &shot);
```

DBvDisableColumn

Image	Audio	Video
		X

비디오(DB2Video 데이터)용으로 컬럼을 사용 불가능하게 하여 비디오 데이터를 수용하지 못하게 합니다. 컬럼 항목의 내용은 널(NULL)에 설정되며, 이 컬럼과 연관된 메타데이터는 삭제됩니다. 이 컬럼용으로 Video Extender에 의해 정의된 모든 트리거 역시 삭제됩니다. 새 행은 사용 불가능하게 된 컬럼을 포함하는 테이블에 포함될 수 있으며, 새 행은 유형 DB2Video로 정의된 데이터를 포함할 수도 있으나 새 행과 관련된 아무런 메타데이터(관리 지원 테이블 내)도 없습니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvDisableColumn(
      char *tableName,
      char *colName,
      );
```

매개변수

tableName(입력)

비디오 컬럼을 포함하는 테이블 이름.

colName(입력)

비디오 컬럼의 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

비디오(DB2Video 데이터)용으로 employee 테이블의 tv_ads 컬럼을 사용 불가능하게 합니다.

```
#include <dmbvideo.h>

rc = DBvDisableColumn("employee",
       "tv_ads");
```

DBvDisableDatabase

Image	Audio	Video
		X

비디오(DB2Video 데이터)용으로 데이터베이스를 사용 불가능화하여 비디오 데이터를 수용하지 못하게 합니다. DB2Video용으로 정의된 데이터베이스 내의 모든 테이블도 역시 사용 불가능화됩니다. 이 데이터베이스용으로 Video Extender에 의해 정의된 메타데이터 및 UDF가 삭제됩니다. 새 행은 유형 DB2Video에 의해 정의된 데이터베이스 내 테이블에 삽입될 수도 있으나, 새 행과 관련된 아무런 메타데이터(관리 지원 테이블 내)도 없습니다.

권한

DBADM, SYSADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvDisableDatabase(
      );
```

매개변수

DBvDisableDatabase에는 매개변수가 없습니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

비디오(DB2Video 데이터)용으로 현재 데이터베이스를 사용 불가능화합니다.

```
#include <dmbvideo.h>

rc = DBvDisableDatabase();
```

DBvDisableTable

Image	Audio	Video
		X

비디오(DB2Video 데이터)용으로 테이블을 사용 불가능하게 하여 비디오 데이터를 수용하지 못하게 합니다. DB2Video용으로 정의된 테이블의 모든 컬럼도 역시 사용 불가능하게 됩니다. 이 테이블용으로 Video Extender에 의해 정의된 메타데이터의 일부가 삭제됩니다. 새 행은 유형 DB2Video에 의해 정의된 테이블에 삽입될 수도 있으나 새 행과 관련된 아무런 메타데이터(관리 지원 테이블 내)도 없습니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvDisableTable(
      char *tableName
      );
```

매개변수

tableName(입력)

비디오 컬럼을 포함하는 테이블 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

비디오(DB2Video 데이터)용으로 employee 테이블을 사용 불가능하게 합니다.

```
#include <dmbvideo.h>

rc = DBvDisableTable("employee");
```

DBvEnableColumn

Image	Audio	Video
		X

비디오(DB2Video 데이터)용으로 컬럼을 사용 가능하게 합니다. API는 이 컬럼과 메타데이터 테이블 사이의 관계를 정의하고 관리합니다. 이 API를 호출하기 전에, 응용프로그램을 데이터베이스에 연결하고 사용자 테이블을 확약해야 합니다.

권한

Control, Alter, SYSADM, DBADM

API 매개변수에 지정된 테이블 공간 및 버퍼 풀에 관한 사용 특권이 또한 필요합니다.

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvEnableColumn(
      char *tableName,
      char *colName,
      );
```

매개변수

tableName(입력)

비디오 컬럼을 포함하는 테이블 이름.

colName(입력)

비디오 컬럼의 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED

컬럼은 이미 사용 가능하게 되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_WRONG_SIGNATURE

지정된 데이터 유형이 올바르지 않습니다. 사용자 정의 데이터 유형 MMDBSYS.DB2VIDEO가 예상됩니다.

MMDB_RC_COLUMN_DOESNOT_EXIST

컬럼이 지정된 테이블 내에서 정의되지 않았습니다.

MMDB_RC_NOT_ENABLED

데이터베이스 또는 테이블을 사용할 수 없습니다.

예

비디오(DB2Video 데이터)용으로 employee 테이블의 video 컬럼을 사용 가능하게 합니다.

```
#include <dmbvideo.h>

rc = DBvEnableColumn("employee",
       "video");
```

DBvEnableDatabase

Image	Audio	Video
		X

비디오(DB2Video 데이터)용으로 데이터베이스를 사용 가능하게 합니다. 이 API는 데이터베이스당 한번씩 호출됩니다. 이것은 DB2 사용자 정의 유형, DB2Video를 데이터베이스 관리 프로그램에 정의합니다. 이것은 역시 DB2Video 데이터를 조작하는 모든 UDF를 생성합니다.

권한

DBADM, SYSADM, SYSCTRL

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvEnableDatabase(
      char *tableSpace
      );
```

매개변수

tableSpace(입력)

관리 테이블이 저장된 컨테이너의 집합인 테이블 공간의 명칭. 테이블 공간 스펙은 *datats, indexts, longts*의 세 부분으로 되어 있으며, 여기서 *datats*는 메타데이터 테이블이 생성되는 테이블 공간입니다. *indexts*는 메타데이터 테이블에 대한 색인이 생성되는 테이블 공간입니다. *longts*는

(LONG VARCHAR 및 LOB 데이터 유형을 포함한 것과 같은) 메타데이터 테이블 내의 긴 컬럼의 값이 저장되는 테이블 공간입니다. 테이블 공간 스펙의 어떤 부분에 대해 널(NULL) 값을 제공하는 경우에 그 부분에 대해 기본 테이블 공간이 사용됩니다.

EEE 전용: Extender에 대해 데이터베이스를 사용 가능화할 때 지정한 테이블 공간은 파티션된 데이터베이스 시스템의 모든 노드가 포함되는 노드 그룹에 정의되어 있어야 합니다.

오류 코드

MMDB_SUCCESS
API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH
호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED
데이터베이스는 이미 사용 가능하게 되었습니다.

MMDB_RC_API_NOT_SUPPORTED_FOR_SERVER
연결되는 서버는 이 명령을 지원하지 않습니다.

MMDB_WARN_NOT_ALL_NODES
지정된 테이블 공간이 해당 Extender와 연관된 노드를 모두 포함하지는 않습니다. **(EEE 전용)**

MMDB_RC_NOT_SAME_NODEGROUP
지정된 테이블 공간이 동일한 노드 그룹에 있지 않습니다**(EEE 전용)**.

예

MYTS라고 이름 지정된 테이블 공간 내 비디오(DB2Video 데이터)용으로 현재 데이터베이스를 사용 가능하게 합니다. 색인 및 긴 테이블 공간용으로 기본값을 사용합니다.

```
#include <dmbvideo.h>

rc = DBvEnableDatabase("myts,,");
```

비디오(DB2Video 데이터)용으로 현재 데이터베이스를 사용 가능하게 합니다. 기본 테이블 공간을 사용합니다.

```
#include <dmbvideo.h>

rc = DBvEnableDatabase(NULL);
```

DBvEnableTable

Image	Audio	Video
		X

비디오(DB2Video 데이터)용으로 테이블을 사용 가능하게 합니다. 이 API는 테이블당 한번씩 호출됩니다. 테이블의 비디오 컬럼용 속성을 저장하고 관리하기 위해 메타데이터 테이블을 생성합니다. 잠금 가능성을 방지하려면 응용프로그램은 반드시 이 API를 호출하기 전에 트랜잭션을 확약해야 합니다. 이 API를 호출하기 전에, 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

Control, Alter, SYSADM, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvEnableTable(
      char *tableSpace,
      char *tableName
      );
```

매개변수

tableSpace(입력)

관리 테이블이 저장되는 컨테이너의 집합인 테이블 공간의 이름. 테이블 공간 스펙은 *datats, indexts, longts*의 세 부분으로 되어 있으며, 여기서

*datats*는 메타데이터 테이블이 생성되는 테이블 공간입니다. *indexts*는 메타데이터 테이블에 대한 색인이 생성되는 테이블 공간입니다. *longts*는 (LONG VARCHAR 및 LOB 데이터 유형을 포함한 것과 같은) 메타데이터 테이블 내의 긴 컬럼의 값이 저장되는 테이블 공간입니다.

테이블 공간 스펙의 임의의 부분에 대해 널(NULL) 값을 제공하는 경우, 해당 부분에 대한 기본 테이블 공간이 사용됩니다.

EEE 전용: 지정된 테이블 공간은 사용자 테이블과 동일한 노드 그룹에 있어야 합니다.

tableName(입력)

비디오 컬럼을 포함할 테이블 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_WARN_ALREADY_ENABLED

테이블은 이미 사용 가능하게 되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_TABLE_DOESNOT_EXIST

테이블이 존재하지 않습니다.

MMDB_RC_TABLESPACE_NOT_SAME_NODEGROUP

지정된 테이블 공간이 사용자 테이블과 동일한 노드 그룹에 있지 않습니다**(EEE 전용)**.

예

테이블 공간 MYTS의 비디오(DB2Audio 데이터)용으로 employee 테이블을 사용 가능하게 합니다. 색인 및 긴 테이블 공간에 대해 기본값을 사용합니다.

```
#include <dmbvideo.h>

rc = DBvEnableTable("myts,,",
       "employee");
```

비디오(DB2Video 데이터)용으로 employee 테이블을 사용할 수 있게 합니다. 기본 테이블 공간을 사용합니다.

```
#include <dmbvideo.h>

rc = DBvEnableTable(NULL,
       "employee");
```

DBvFrameDataTo24BitRGB

Image	Audio	Video
		X

MPEG와 같은, YUV 색상-값 형식에서 24비트 RGB 형식으로 비디오 프레임을 변환합니다. 사용자는 API 호출을 발생하기 전에 반드시 목표 버퍼를 할당해야만 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbmpeg.lib	libdmbmpeg.a(AIX) libdmbmpeg.sl(HP-UX) libdmbmpeg.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvFrameDataTo24BitRGB(
      unsigned char *RGB,
      DBvFrameData *fd,
      unsigned long dx,
      unsigned long dy
      );
```

매개변수

RGB(출력)

목표 RGB 버퍼에 대한 포인터.

fd(입력)

전환되는 프레임 데이터에 대한 포인터.

dx(입력)

프레임 폭.

dy(입력)

프레임 높이.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

예

비디오 프레임을 MPEG에서 24비트 RGB로 변환합니다.

```
#include <dmbshot.h>

rc = DBvFrameDataTo24BitRGB(RGB, &video->fd,
       video->dx, video->dy);
```

DBvGetError

Image	Audio	Video
		X

마지막 오류의 설명을 리턴합니다. 기타 다른 API가 오류 코드를 리턴한 뒤에 이 API를 호출하십시오.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvGetError(
      SQLINTEGER *sqlcode,
      char *errorMsgText
      );
```

매개변수

sqlcode(출력)

일반적인 SQL 오류 코드.

errorMsgText(출력)

SQL 오류 메시지 텍스트.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

예

SQL 오류 코드를 errCode에 저장하고 메시지 텍스트를 errMsg에 저장하여 마지막 오류를 확보합니다.

```
#include <dmbvideo.h>

rc = DBvGetError(&errCode, &errMsg);
```

DBvGetFrame

Image	Audio	Video
		X

비디오 파일 내에 현재 프레임을 확보합니다. 프레임 데이터는 DBvFrameData 비디오 구조에 리턴됩니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbmpeg.lib	libdmbmpeg.a(AIX) libdmbmpeg.sl(HP-UX) libdmbmpeg.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvGetFrame(
      DBvIOType *video
      );
```

매개변수

video(갱신)

비디오 구조에 대한 포인터.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_EOF
파일의 끝에 도달.

예

비디오 파일 내에 현재 프레임을 확보합니다.

```
#include <dmbshot.h>

rc = DBvGetFrame(video);
```

DBvGetInaccessibleFiles

Image	Audio	Video
		X

사용자 테이블의 비디오 컬럼에서 참조되었던 액세스 불가능한 파일의 이름을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 정보 소스를 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

모든 탐색된 사용자 테이블 및 연관된 관리 지원 테이블에 있는 사용 가능한 비디오 컬럼에 관한 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvGetInaccessibleFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테이블 이름이 지정된 경우 그 테이블은 액세스 불가능한 파일에 대한 참조용으로 탐색됩니다. 널(NULL) 값이 지정되면 지정된 규정자를 갖는 모든 테이블이 탐색됩니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블에서 참조된 액세스 불가능한 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

employee 테이블의 video 컬럼에서 참조된 액세스 불가능한 모든 파일을 나열합니다.

```
#include <dmbvideo.h>
long idx;

rc = DBvGetInaccessibleFiles("employee",
       &count, &filelist);
```

DBvGetReferencedFiles

Image	Audio	Video
		X

사용자 테이블의 비디오 컬럼에서 참조되었던 파일의 이름을 리턴합니다. 파일을 액세스할 수 없는 경우(예를 들어, 환경 변수 스펙을 사용하여 파일 이름을 해결할 수 없음), 파일 이름에 별표(*)가 선행됩니다. 이 API는 FILEREF 데이터 구조의 FILENAME 필드를 사용하지 않으므로 이것을 널(NULL)로 설정합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 정보 소스를 할당해제하는 것이 중요합니다. 특히, 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

모든 탐색된 사용자 테이블 및 연관된 관리 지원 테이블에 있는 사용 가능한 비디오 컬럼에 관한 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvGetReferencedFiles(
      char *tableName,
      long *count,
      FILEREF *(*fileList)
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테이블 이름이 지정된 경우 그 테이블은 파일에 참조용으로 탐색됩니다. 널(NULL) 값을 지정하는 경우, 현재 사용자 ID가 소유한 모든 테이블을 탐색합니다.

count(출력)

출력 목록의 항목 갯수.

fileList(출력)

테이블에서 참조된 파일의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

employee 테이블의 video 컬럼에서 참조된 모든 파일을 나열합니다.

```
#include <dmbvideo.h>
long idx;

rc = DBvGetReferencedFiles("employee",
       &count, &filelist);
```

DBvInitShotControl

Image	Audio	Video
		X

컷 제어 데이터 구조 내의 값을 초기화합니다.

권한

Insert, Control

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvInitShotControl(
      DBvShotControl *shotCtrl,
      );
```

매개변수

shotCtrl(입력)

컷 제어 데이터 구조에 대한 포인터.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_ACCESS

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

컷 제어 데이터 구조 내의 값을 초기화합니다.

```
#include <dmbshot.h>

rc = DBvInitShotControl(shotCtrl);
```

DBvInitStoryboardCtrl

Image	Audio	Video
		X

컷 제어 데이터 구조 내의 값을 초기화합니다.

권한

Insert, Control

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvInitStoryboardCtrl(
      DBvStoryboardCtrl *sbCtrl,
      );
```

매개변수

shotCtrl(입력)

컷 제어 데이터 구조에 대한 포인터.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_ACCESS

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

스토리보드 제어 데이터 구조 내의 값을 초기화합니다.

```
#include <dmbshot.h>

rc = DBvInitStoryboardCtrl(shotCtrl);
```

DBvInsertShot

Image	Audio	Video
		X

컷을 컷 카탈로그로 삽입합니다.

권한

Insert, Control

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvInsertShot(
      char *catalogName,
      DBvShotType *shot,
      DBvIOType *video,
      char *shotHandle,
      SQLHDBC hdbc
      );
```

매개변수

catalogName(입력)

카탈로그의 이름.

shot(입력)

카탈로그에 삽입될 확장된 컷에 대한 포인터.

shotHandle(입력)

컷 핸들.

hdbc(입력)

SQL 연결로부터 데이터베이스 핸들.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_ACCESS

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_INVALID_CATALOG

카탈로그가 유효하지 않거나 존재하지 않습니다.

예

hotshots라고 하는 컷 카탈로그에 컷을 삽입합니다.

```
rc = DBvInsertShot(
"hotshots",                    /* shot catalog name */
shot,                          /* pointer to shot structure */
video,                         /* pointer to video structure */
shotHandle,                    /* pointer to shot handle */
hdbc);                         /* database connection handle */
```

DBvIsColumnEnabled

Image	Audio	Video
		X

컬럼이 비디오(DB2Video 데이터)용으로 사용 가능하게 되었는지의 여부를 판별합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

사용자 테이블에 관한 SYSADM, DBADM, 테이블 소유자 또는 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvIsColumnEnabled(
      char *tableName,
      char *colName,
      short *status
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 테이블 이름.

colName(입력)

컬럼의 이름.

status(출력)

컬럼이 사용 가능한지를 표시합니다. 매개변수는 숫자 값을 리턴합니다. Extender 또한 상태를 표시하는 상수를 리턴합니다. 그 값과 상수는 다음과 같습니다.

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

-1 MMDB_INVALID_DATATYPE

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

employee 테이블의 video 컬럼이 비디오용으로 사용 가능한지의 여부를 결정합니다.

```
#include <dmbvideo.h>

rc = DBvIsColumnEnabled("employee",
       "video", &status);
```

DBvIsDatabaseEnabled

Image	Audio	Video
		X

데이터베이스가 비디오(DB2Video 데이터)용으로 사용 가능해졌는지의 여부를 판별합니다. 응용프로그램은 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvIsDatabaseEnabled(
      short *status
      );
```

매개변수

status(출력)

데이터베이스가 사용 가능한지를 표시합니다. 매개변수는 숫자 값을 리턴합니다. Extender 또한 상태를 표시하는 상수를 리턴합니다. 그 값과 상수는 다음과 같습니다.

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

personnl 데이터베이스가 비디오용으로 사용 가능한지를 판별합니다.

```
#include <dmbvideo.h>

rc = DBvIsDatabaseEnabled(&status);
```

DBvIsFileReferenced

Image	Audio	Video
		X

지정된 파일을 참조하는 비디오 컬럼 내 테이블 항목의 목록을 리턴합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

호출한 뒤에 이 API에 의해 할당된 정보 소스를 할당해제하는 것이 중요합니다. 특히, 파일 목록의 각 항목 파일 이름 필드 뿐만 아니라 파일 목록 데이터 구조를 할당해제해야 합니다.

권한

모든 탐색된 사용자 테이블 및 연관된 관리 지원 테이블에 있는 사용 가능한 비디오 컬럼에 관한 SELECT 특권

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvIsFileReferenced(
      char *tableName,
      char *fileName,
      long *count,
      FILEREF *(*tableList)
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테이블 이름이 지정된 경우 그 테이블은 지정된 파일에 대한 참조용으로 탐색됩니다. 널(NULL) 값을 지정하는 경우, 현재 사용자 ID가 소유한 모든 테이블을 탐색합니다.

fileName(입력)

참조된 파일 이름.

count(출력)

출력 목록의 항목 갯수.

tableList(출력)

지정된 파일을 참조하는 테이블 항목의 목록.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_MALLOC

시스템은 결과를 리턴하기 위해 메모리를 할당할 수 없습니다.

예

/videos/ajones.mpg 파일을 참조하는 employee 테이블의 비디오 컬럼 내의 항목을 나열합니다.

```
#include <dmbvideo.h>
long idx;

rc = DBvIsFileReferenced(NULL,
       "/videos/ajones.mpg",
       &count, &tableList);
```

DBvIsIndex

Image	Audio	Video
		X

비디오 색인의 존재를 점검합니다. 응용프로그램은 이 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbmpeg.lib	libdmbmpeg.a(AIX) libdmbmpeg.sl(HP-UX) libdmbmpeg.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvIsIndex(
      char *fileName,
      short *status
      );
```

매개변수

fileName(입력)

참조된 파일 이름.

status(출력)

색인의 존재 여부를 표시합니다. 값 1은 색인이 존재함을 뜻하고 값 0은 색인이 없다는 것을 뜻합니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_ERROR

상태가 유효하지 않습니다.

예

비디오 파일 \videos\ajones.mpg용 색인의 존재 여부를 점검합니다.

```
#include <dmbshot.h>

rc = DBvIsIndex("\videos\ajones.mpg", &status);
```

DBvIsTableEnabled

Image	Audio	Video
		X

테이블이 비디오(DB2Video 데이터)용으로 사용 가능하게 되었는지의 여부를 판별합니다. 이 API를 호출하기 전에 응용프로그램을 데이터베이스에 연결해야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvIsTableEnabled(
      char *tableName,
      short *status
      );
```

매개변수

tableName(입력)

테이블 이름.

status(출력)

테이블이 사용 가능한지를 표시합니다. 매개변수는 숫자 값을 리턴합니다. Extender 또한 상태를 표시하는 상수를 리턴합니다. 그 값과 상수는 다음과 같습니다.

1 MMDB_IS_ENABLED

0 MMDB_IS_NOT_ENABLED

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

employee 테이블이 비디오용으로 사용 가능한지의 여부를 판별합니다.

```
#include <dmbvideo.h>

rc = DBvIsTableEnabled("employee",
       &status);
```

DBvMergeShots

Image	Audio	Video
		X

두 컷을 한 컷으로 병합합니다. 결과 컷은 컷 핸들과 첫번째 컷의 시작 프레임을 사용합니다. 두 컷 중 더 큰 최종 프레임이 결과 컷에서 사용됩니다. 두 번째 컷 핸들이 지시하는 행은 삭제됩니다.

권한

Control, Select, Delete, Update

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvMergeShots(
      char *catalogName,
      char *shotHandle1,
      char *shotHandle2,
      SQLHDBC hdbc
      );
```

매개변수

catalogName(입력)

컷 카탈로그의 이름.

shotHandle1(입력)

첫번째 컷의 핸들.

shotHandle2(입력)

두 번째 컷의 핸들.

hdbc(입력)

SQL 연결로부터 데이터베이스 핸들.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_CANNOT_MERGE

컷을 병합할 수 없습니다.

MMDB_RC_INVALID_CATALOG

카탈로그가 유효하지 않거나 존재하지 않습니다.

예

hotshots 카탈로그 내의 shopHandle1과 shotHandle2 핸들을 갖는 컷을 병합합니다.

```
#include <dmbshot.h>

rc = DBvMergeShots("hotshots", shotHandle1,
       shotHandle2, hdbc);
```

DBvOpenFile

Image	Audio	Video
		X

DBvIOType 구조용으로 공간을 할당하고 픽셀(pixel) 액세스용 비디오 파일을 엽니다. 비디오가 성공적으로 열리면 첫번째 프레임 번호(프레임 0)를 지시합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbmpeg.lib	libdmbmpeg.a(AIX) libdmbmpeg.sl(HP-UX) libdmbmpeg.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvOpenFile(
      DBvIOType **video,
      char *fileName,
      );
```

매개변수

video(출력)

비디오 구조 포인터에 대한 포인터.

fileName(입력)

열릴 비디오 파일의 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_CANNOT_OPEN

비디오 파일을 열지 못했습니다.

MMDB_RC_NO_MEMORY

메모리가 부족합니다.

MMDB_RC_NO_INDEX

비디오 임의 액세스 색인이 없습니다.

예

비디오 파일 \videos\ajones.mpg을 엽니다.

```
#include <dmbshot.h>

rc = DBvOpenFile(&videoa,
       "\videos\ajones.mpg");
```

DBvOpenHandle

Image	Audio	Video
		X

DBvIOType 구조용으로 공간을 할당하고 픽셀(pixel) 액세스용 비디오 핸들을 엽니다. 구조는 첫번째 프레임 번호(프레임 0)를 지시합니다. 비디오는 BLOB이 될 수 있습니다. 비디오는 DB2VIDEOTEMP 환경 변수에 의해 지정된 디렉토리 내의 임시 파일에 복사됩니다. isIdx 플래그는 임의 액세스 색인의 존재에 기초하여 설정됩니다.

권한

Select

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvOpenHandle(
      DBvIOType **video,
      DB2Video *videoHandle
      SQLHDBC hdbc
      );
```

매개변수

video(출력)

비디오 구조에 대한 포인터.

videoHandle(입력)

비디오 핸들.

hdbc(입력)

SQL 연결로부터 데이터베이스 핸들.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_CANNOT_OPEN

비디오 파일을 열지 못했습니다.

MMDB_RC_NO_MEMORY

메모리가 부족합니다.

MMDB_RC_NO_INDEX

비디오 임의 액세스 색인이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_INVALID_HANDLE

비디오 핸들이 유효하지 않습니다.

예

비디오 포인터를 사용하여 videoHandle들을 엽니다.

```
#include <dmbshot.h>

rc = DBvOpenHandle(&oa, videoHandle, hdbc);
```

DBvPlay

Image	Audio	Video
		X

클라이언트상의 비디오 플레이어를 열고 비디오를 재생합니다. 비디오는 비디오 컬럼이나 외부 파일에 저장 가능합니다.

- 비디오가 외부 파일에 저장된 경우에 파일 이름이나 비디오 핸들을 API에 전달할 수 있습니다. API는 파일 위치를 결정하기 위해 클라이언트 환경 변수 DB2VIDEOPATH를 사용합니다. 파일은 반드시 클라이언트 워크스테이션으로부터 액세스 가능해야 합니다.
- 비디오가 컬럼 내에 저장되어 있는 경우 반드시 API로 비디오 핸들을 전달해야 합니다. 응용프로그램을 데이터베이스에 연결해야 하며, 비디오가 저장된 테이블에 대한 읽기 액세스 권한이 있어야 합니다.

비디오가 컬럼에 저장된 경우, Extender는 임시 파일을 작성하고 오브젝트의 내용을 컬럼에서 파일로 복사합니다. Extender는 또한 비디오가 외부 파일에 저장되고 상대적인 파일 이름을 환경 변수 값을 사용하여 해결할 수 없는 경우 또는 파일을 클라이언트 머신에서 액세스할 수 없는 경우 임시 파일을 작성할 수도 있습니다. 임시 파일은 DB2VIDEOTEMP 환경 변수에 지정된 디렉토리에 작성됩니다. 그러면 Extender는 임시 파일에서 비디오를 재생합니다.

권한

컬럼에서 비디오를 재생하는 경우 사용자 테이블에 대한 Select 권한

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

컬럼에 저장된 비디오 재생

```
long DBvPlay(
      char *playerName,
      MMDB_PLAY_HANDLE,
      DB2Video *videoHandle,
      waitFlag
      );
```

구문

파일로 저장된 비디오 재생

```
long DBvPlay(
      char *playerName,
      MMDB_PLAY_FILE,
      char *fileName,
      waitFlag
      );
```

매개변수

playerName(입력)

비디오 플레이어의 이름. 널(NULL)에 지정된 경우 DB2VIDEOPLAYER 환경 변수에 의해 지정된 기본 비디오 플레이어가 사용됩니다.

MMDB_PLAY_HANDLE(입력)

비디오가 컬럼에 저장되었음을 표시하는 상수.

MMDB_PLAY_FILE(입력)

비디오가 클라이언트로부터 액세스 가능한 파일로 저장되었음을 표시하는 상수.

videoHandle(입력)

비디오의 핸들. 이 매개변수는 컬럼에서 비디오를 재생할 때 반드시 전달되어야 합니다. 비디오 핸들이 외부 파일을 나타낼 경우, 파일 위치를 결정하기 위해 클라이언트 환경 변수 DB2VIDEOPATH가 사용됩니다.

fileName(입력)

비디오를 포함하는 파일 이름. API는 파일 위치를 결정하기 위해 클라이

언트 환경 변수 DB2VIDEOPATH를 사용합니다. 파일은 반드시 클라이언트 워크스테이션으로부터 액세스 가능해야 합니다.

waitFlag(입력)

계속하기 전에 프로그램이 사용자가 플레이어를 닫기를 기다리는지를 표시하는 상수. MMDB_PLAY_WAIT는 응용프로그램과 같은 스레드에서 플레이어를 실행합니다. MMDB_PLAY_NO_WAIT는 분리된 스레드에서 플레이어를 실행합니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

videoHandle에 의해 식별된 비디오를 재생합니다. 응용프로그램과 같은 스레드에서 기본 플레이어를 실행합니다.

```
#include <dmbvideo.h>

rc = DBvPlay(NULL, MMDB_PLAY_HANDLE, videoHandle,
       MMDB_PLAY_WAIT);
```

DBvPrepareAttrs

Image	Audio	Video
		X

사용자 제공 비디오 속성을 준비합니다. 이 API는 비디오 오브젝트가 사용자 제공 속성으로 저장되거나 갱신될 때 사용됩니다. 서버에서 실행되는 UDF 코드는 항상 『big endian』 형식의 데이터를 예상하며, 이것은 대부분의 UNIX 플랫폼에서 사용하는 형식입니다. 비디오 오브젝트가 『little endian』 형식으로 저장되거나 갱신된 경우(즉, 비-UNIX 클라이언트로부터), 저장 또는 갱신 요청을 수행하기 전에 DBvPrepare API를 사용해야 합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
void DBvPrepareAttrs(
      MMDBVideoAttrs *vidAttr
      );
```

매개변수

vidAttr(입력)

비디오의 사용자 제공 속성.

예

사용자 제공 비디오 속성을 준비합니다.

```
#include <dmbvideo.h>

DBvPrepareAttrs(&vidattr);
```

DBvReorgMetadata

Image	Audio	Video
		X

비디오 관련 메타데이터 테이블을 『정리』합니다. 예를 들면,

- 비디오 메타데이터 테이블 내에서 더이상 사용되지 않는 공간을 회수합니다.
- 더이상 존재하지 않는 비디오 파일에 대한 비디오 메타데이터 테이블 내의 참조를 삭제합니다.

응용프로그램은 API를 호출하기 전에 반드시 데이터베이스에 연결되어야 합니다.

권한

Alter, Control, SYSADM, SYSCTRL, SYSMAINT, DBADM

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbvideo.lib	libdmbvideo.a(AIX) libdmbvideo.sl(HP-UX) libdmbvideo.so(Solaris)

Include 파일

dmbvideo.h

구문

```
long DBvReorgMetadata(
      char *tableName,
      );
```

매개변수

tableName(입력)

규정되거나 규정되지 않은 또는 널(NULL) 입력 가능한 테이블 이름. 테이블 이름이 지정된 경우, 정리(cleanup)는 지정된 사용자 테이블과 관련

된 비디오 메타데이터 테이블에 대해 수행됩니다. 널(NULL) 값이 지정되면 현재 사용자 ID에 의해 소유된 모든 테이블 내의 비디오 컬럼용 메타데이터 테이블이 정리됩니다.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NO_AUTH

호출자는 적절한 액세스 권한이 없습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

예

employee 테이블 내의 비디오 컬럼용 메타데이터 테이블을 정리합니다.

```
#include <dmbvideo.h>

rc = DBvReorgMetadata("employee");
```

DBvSetFrameNumber

Image	Audio	Video
		X

현재 프레임을 지정된 프레임 번호에 설정합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbmpeg.lib	libdmbmpeg.a(AIX) libdmbmpeg.sl(HP-UX) libdmbmpeg.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvSetFrameNumber(
      DBvIOType *video
      unsigned long frameNumber
      );
```

매개변수

video(입력)

비디오 구조에 대한 포인터.

frameNumber(입력)

요청된 프레임의 번호.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_FRAME_NOT_FOUND

요청된 프레임을 발견할 수 없었습니다.

MMDB_NO_INDEX

비디오 색인이 존재하지 않습니다.

예

현재 프레임을 비디오 파일 내의 프레임 번호 85에 설정합니다.

```
#include <dmbshot.h>

rc = DBvSetFrameNumber(video, 85);
```

DBvSetShotComment

Image	Audio	Video
		X

컷 내의 읽기 전용 주석을 갱신합니다.

권한

Control, Update

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvSetShotComment(
      char *catalogName,
      char *shotHandle,
      char *comment,
      SQLHDBC hdbc
      );
```

매개변수

catalogName(입력)

카탈로그의 이름.

shotHandle(입력)

갱신될 컷의 핸들.

comment(입력)

컷용 새 주석.

hdbc(입력)

SQL 연결로부터 데이터베이스 핸들.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_CANNOT_UPDATE

API가 컷을 갱신할 수 없습니다.

MMDB_RC_INVALID_CATALOG

카탈로그가 유효하지 않거나 존재하지 않습니다.

예

카탈로그 hotshot 내의 shotHandle을 갖는 컷을 설명하는 주석을 변경합니다.

```
#include <dmbshot.h>

rc = DBvSetShotComment("hotshot", shotHandle,
       "This is a hot shot.", hdbc);
```

DBvUpdateShot

Image	Audio	Video
		X

카탈로그 내의 비디오 컷의 속성을 대체합니다. 주석을 제외한 모든 속성은 DBvShotType 구조 내의 속성으로 대체됩니다. 비고(remark) 포인터가 널(NULL)인 경우, 존재하는 비고는 변경되지 않고 남아 있게 됩니다.

권한

Control, Update

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbshot.lib	libdmbshot.a(AIX) libdmbshot.sl(HP-UX) libdmbshot.so(Solaris)

Include 파일

dmbshot.h

구문

```
long DBvUpdateShot(
      char *catalogName,
      DBvShotType *shot,
      char *shotHandle,
      SQLHDBC hdbc
      );
```

매개변수

catalogName(입력)

카탈로그의 이름.

shot(입력)

컷의 속성을 포함한 컷 정보 구조에 대한 포인터.

shotHandle(입력)

컷 핸들.

hdbc(입력)

SQL 연결로부터 데이터베이스 핸들.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RC_NOT_CONNECTED

응용프로그램이 데이터베이스에 대한 유효한 연결을 가지고 있지 않습니다.

MMDB_RC_CANNOT_UPDATE

API가 컷을 갱신할 수 없습니다.

MMDB_RC_NO_SHOT

컷이 존재하지 않습니다.

MMDB_RC_INVALID_CATALOG

카탈로그가 유효하지 않거나 존재하지 않습니다.

예

hotshots 카탈로그 내의 컷의 속성을 갱신합니다.

```
#include <dmbshot.h>

rc = DBvUpdateShot("hotshots", shot,
       shothandle, hdbc);
```

DMBRedistribute(EEE 전용)

Image	Audio	Video
X		

노드 그룹에 노드가 추가되거나 노드 그룹에서 노드가 제거될 때, 또는 노드 그룹에 대한 새로운 파티션 맵이 설치될 때, QBIC 특성 데이터를 재분산합니다.

권한

이 API는 인스턴스 소유 ID에서 수행해야 합니다.

라이브러리 파일

Windows	AIX 및 Solaris
dmbrd.lib	libdmbrd.a(AIX)
	libdmbrd.so(Solaris)

Include 파일

dmbrdst.h

구문

```
long DMBRedistribute (
       char  *pNodeGroupName,
       char  DataRedistOption  /* ""continue"" use CONTINUE parameter */
       );
                               /* blank:start redistribution */
```

매개변수

pNodeGroupName(입력)

재분산할 노드 그룹의 이름.

오류 코드

MMDB_SUCCESS

API 호출이 성공적으로 처리되었습니다.

MMDB_RD_NO_CONTINUE

CONTINUE 매개변수 없이 다시 제출하십시오.

MMDB_RD_CONTINUE

CONTINUE 매개변수를 사용하여 다시 제출하십시오.

예

groupone 노드 그룹의 QBIC Extender 데이터를 재분산합니다.

```
#include <dmbrdst.h>

rc = DMBRedistribute(groupone,"continue");
```

QbAddFeature

Image	Audio	Video
X		

현재 열려 있는 카탈로그에 특성을 추가합니다. QbAddFeature는 데이터베이스에 지정된 특성에 대한 특성 테이블을 작성합니다. 사용자 테이블 내의 이미지 컬럼에 이미지를 추가한 후에, 특성 테이블에 각 이미지용 항목을 추가하고 이미지를 분석하는 QbReCatalogColumn API를 사용하십시오.

권한

Alter

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbAddFeature(
      QbCatalogHandle cHdl,
      char *featureName
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

featureName(입력)

특성의 이름. 다음 특성이 Image Extender와 함께 제공됩니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

qbicECCatalogReadOnly

카탈로그가 읽기 전용으로 열려 있습니다.

qbicECDupFeature

특성이 이미 카탈로그에 있습니다.

qbiECinvalidFeatureClass

지정한 특성이 유효한 이름 형식이 아닙니다.

예

핸들 CatHdl에 의해 식별된 카탈로그에 QbColorFeatureClass 특성을 추가합니다.

```
#include <dmbqbapi.h>

rc = QbAddFeature(CatHdl,
       QbColorFeatureClass);
```

QbCatalogColumn

Image	Audio	Video
X		

카탈로그화되지 않은 사용자 테이블 이미지 컬럼의 이미지를 카탈로그화합니다. API는 특성 테이블에 각 이미지용으로 항목을 추가하고 이미지를 분석합니다. API가 이미지를 분석할 때, 이미지 데이터를 생성하고 특성 테이블 내의 이미지 항목 내에 저장합니다. 특성용 기본 매개변수가 사용됩니다. 카탈로그는 반드시 열려 있어야 합니다.

권한

Insert

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbCatalogColumn(
      QbCatalogHandle cHdl
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

qbicECInvalidCatalog

지정된 핸들 또는 테이블 컬럼이 카탈로그에 대해 유효하지 않습니다.

qbicECCatalog Errors

개별 이미지를 카탈로그화는 동안 오류가 발생하였으며, 해당 오류가 기록되었습니다. 구간 복원(rollback)은 발생하지 않습니다.

qbicECImageNotFound

이미지를 발견할 수 없거나 이미지에 액세스할 수 없습니다.

qbicECCatalogRO

카탈로그는 읽기 전용입니다.

qbicECSQLError

SQL 오류가 발생했습니다.

예

```
#include <dmbqbapi.h>

rc = QbCatalogColumn(CatHdl);
```

QbCatalogImage

Image	Audio	Video
X		

전체 이미지를 카탈로그화합니다. API는 특성 테이블에 각 이미지용 항목을 추가하고 이미지를 분석합니다. API가 이미지를 분석할 때, 이미지 데이터를 생성하고 특성 테이블 내의 이미지 항목 내에 저장합니다. 이미지 핸들은 반드시 현재 QBIC 카탈로그와 연관된 이미지 컬럼에서 기인해야 합니다. 이미지는 현재 지정된 특성 클래스에 따라 카탈로그화됩니다. 카탈로그 내의 특성에 대해 기본 매개변수가 사용됩니다.

권한

Insert

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbCatalogImage(
      QbCatalogHandle cHdl,
      char *imgHandle
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

imgHandle(입력)

이미지의 핸들.

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

qbicECImageNotFound

이미지를 발견할 수 없거나 이미지에 액세스할 수 없습니다.

qbicECCatalogRO

카탈로그는 읽기 전용입니다.

예

Img_hdl 핸들에 의해 식별된 이미지를 카탈로그화합니다.

```
#include <dmbqbapi.h>

rc = QbCatalogColumn(CatHdl, Img_hdl);
```

QbCloseCatalog

Image	Audio	Video
X		

카탈로그를 닫습니다. API는 열린 카탈로그 핸들과 할당된 자원을 할당해제합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbCloseCatalog(
      QbCatalogHandle cHdl
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

예

핸들 CatHdl에 의해 식별된 카탈로그를 닫습니다.

```
#include <dmbqbapi.h>

rc = QbCloseCatalog(CatHdl);
```

QbCreateCatalog

Image	Audio	Video
X		

지정된 이미지 컬럼에 대해 현재 연결된 데이터베이스에서 카탈로그를 작성합니다. 컬럼은 이미지 데이터용으로 사용 가능해야 합니다. API는 규정자로 사용되는 카탈로그용 이름을 생성합니다.

권한

Alter

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbCreateCatalog(
     char *tableName,
     char *columnName,
     SQLINTEGER autoCatalog,
     char *reserved
     );
```

매개변수

tableName(입력)

이미지 컬럼을 포함하는 테이블 이름.

columnName(입력)

카탈로그를 만드는 이미지 컬럼의 이름.

autoCatalog(입력)

이미지 컬럼에 추가된 이미지가 자동으로 카탈로그화될지의 여부 즉, 특성 테이블에 추가되고 분석되는지의 여부를 표시합니다. 자동 카탈로그화에는 1을 지정하고 그렇지 않으면 0을 지정합니다. 자동 카탈로그화를 설정하지 않을 경우, QbCatalogColumn 또는 QbCatalogImage API를 사용하여 이미지 컬럼에 추가하는 이미지를 카탈로그화하십시오.

reserved(입력)

현재 사용되지 않습니다.

오류 코드

qbicECSqlError

SQL 오류가 발생했습니다.

qbicECNotEnabled

데이터베이스, 테이블 또는 컬럼이 DB2Image 데이터 유형에 대해 사용 가능하지 않습니다.

qbicECDupCatalog

카탈로그가 이미 존재합니다.

qbicECunsupportedOption

지원되지 않는 옵션을 지정하였습니다.

qbicECerrorParameterTooLong

매개변수가 처리하기에 너무 깁니다.

qbicECqerr

QBIC 오류가 발생하였으며, 메시지가 생성되었습니다.

qbicECqerrUnknown

내부 QBIC 오류가 발생하였으며, 일반 오류 메시지가 생성되었습니다.

예

employee 테이블의 picture 컬럼에 이미지용 카탈로그를 생성합니다. 자동 카탈로그화를 작동 상태로 설정합니다.

QbCreateCatalog

```
#include <dmbqbapi.h>

rc = QbCreateCatalog("employee",
        "picture", 1);
```

QbDeleteCatalog

Image	Audio	Video
X		

현재 데이터베이스에서 지정된 카탈로그를 삭제합니다.

권한

Alter

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbDeleteCatalog(
      char *tableName,
      char *columnName
      );
```

매개변수

tableName(입력)

이미지 컬럼을 포함할 테이블 이름.

columnName(입력)

카탈로그와 연관된 이미지 컬럼의 이름.

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

qbicECCatalogInUse

카탈로그가 다른 사람에 의해 사용되고 있습니다.

qbicECCatalogRO

카탈로그는 읽기 전용입니다.

qbicECSystem

시스템 오류가 발생했습니다.

qbicECSqlError

SQL 오류가 발생했습니다.

예

employee 테이블의 picture 컬럼과 연관된 QBIC 카탈로그를 삭제합니다.

```
#include <dmbqbapi.h>

rc=QbDeleteCatalog("employee", "picture");
```

QbGetCatalogInfo

Image	Audio	Video
X		

다음 정보를 포함하는 QbCatalogInfo 구조를 리턴합니다.

- 카탈로그가 소속된 사용자 테이블 및 이미지 컬럼의 이름.
- 카탈로그에 포함된 특성의 번호.
- 자동 카탈로그화가 설정되었는지의 여부.

권한

Select

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbGetCatalogInfo(
      QbCatalogHandle cHdl,
      QbCatalogInfo *catInfo
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

catInfo(출력)

카탈로그 정보 구조.

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

예

핸들 CatHdl에 의해 식별된 카탈로그에 관한 정보를 확보하고 정보를 catInfo라는 이름의 구조로 리턴합니다.

```
#include <dmbqbapi.h>

rc = QbGetCatalogInfo(CatHdl, &catInfo);
```

QbListFeatures

Image	Audio	Video
X		

현재 카탈로그에 포함되어 있는 사용중인 특성의 목록을 리턴합니다. 목록은 할당한 버퍼에 리턴됩니다.

권한

Select

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbListFeatures(
      QbCatalogHandle cHdl,
      SQLINTEGER bufSize,
      SQLINTEGER *count,
      char *featureNames
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

bufSize(입력)

버퍼의 *크기*. 필요한 버퍼의 *크기*를 예측하려면 QbGetCatalogInfo API에

의해 리턴된 특성 카운트를 사용할 수 있으며, 가장 긴 특성 이름의 길이에 카운트를 곱할 수 있습니다. 버퍼에 저장된 특성 이름은 공백 문자에 의해 분리됩니다.

count(출력)
리턴된 특성 이름의 번호.

featureNames(출력)
버퍼 내 특성 이름의 배열.

오류 코드

qbicECIvalidHandle
카탈로그 핸들이 유효하지 않습니다.

qbicECTruncateData
리턴 버퍼가 너무 작기 때문에 리턴된 데이터가 절단되었습니다.

예

핸들 CatHdl에 의해 식별된 카탈로그의 사용중인 특성의 목록을 확보합니다. 특성 이름 배열에 정보를 저장합니다.

우선, 목록을 위해 필요한 버퍼 크기인 bufSize를 계산합니다. CatInfo 구조의 특성의 수를 리턴하기 위해 QbGetCatalogInfo API를 사용합니다. 그리고 가장 긴 특성 이름의 크기인 qbiMaxFeatureName 상수에 그 숫자를 곱합니다.

```
#include <dmbqbapi.h>

rc = QbGetCatalogInfo(CatHdl, &catInfo);

bufSize =
   catInfo.featureCount*qbiMaxFeatureName;

rc = QbListFeatures(CatHdl, bufSize,
       count, featureNames);
```

QbOpenCatalog

Image	Audio	Video
X		

특정 이미지 컬럼용으로 QBIC 카탈로그를 엽니다. 읽기 모드나 갱신 모드에서 카탈로그를 열 수 있습니다. API는 열린 카탈로그용으로 핸들을 리턴합니다. 그러면 카탈로그를 관리하고 상주시키기 위해 다른 API에서 그 핸들을 사용합니다.

카탈로그 사용을 끝낸 후에 카탈로그를 닫았는지 확인하십시오.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbOpenCatalog(
      char *tableName,
      char *columnName,
      SQLINTEGER mode,
      QbCatalogHandle *cHdl
      );
```

매개변수

tableName(입력)

이미지 컬럼을 포함하는 테이블의 이름.

columnName(입력)

이미지 컬럼의 이름.

mode(입력)

카탈로그를 열고 있는 모드. 유효한 값은 qbiRead 및 qbiUpdate입니다.

cHdl(출력)

카탈로그의 핸들에 대한 포인터.

오류 코드

qbicECCatalogNotFound

카탈로그가 발견되지 않았습니다.

qbicECCatalogInUse

카탈로그가 다른 사람에 의해 사용되고 있습니다.

qbicECOpenFailed

카탈로그를 열 수 없었습니다.

qbicECNotEnabled

카탈로그가 사용 가능하지 않습니다.

qbicECNoCatalogFound

어떠한 카탈로그도 발견되지 않았습니다.

qbicECSqlError

SQL 오류가 발생했습니다.

qbicECSystem

시스템 오류가 발생했습니다.

예

읽기 모드로 employee 테이블의 picture 컬럼용 카탈로그를 엽니다.

```
#include <dmbqbapi.h>

rc=QbOpenCatalog("employee", "picture",
     qbiread, &CatHdl);
```

QbQueryAddFeature

Image	Audio	Video
X		

QBIC 카탈로그에 지정된 특성을 추가합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryAddFeature(
      QbQueryHandle qObj,
      char *featureName
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

featureName(입력)

추가될 조회 특성 이름. 다음 특성이 Image Extender와 함께 제공됩니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass

- QbDrawFeatureClass
- QbTextureFeatureClass

오류 코드

qbiECinvalidQueryHandle

사용자가 지정한 조회 오브젝트 핸들은 유효한 조회 오브젝트를 참조하지 않습니다

qbiECunknownFeatureClass

지정한 특성이 인식된 특성 클래스 명칭이 아닙니다.

qbiECinvalidFeatureClass

지정한 특성이 유효한 이름 형식이 아닙니다.

qbiECfeaturePresent

지정한 특성이 이미 조회 오브젝트의 구성원입니다

qbiECallocation

시스템이 충분한 메모리를 할당할 수 없습니다.

예

qoHandle 핸들에 의해 식별된 조회 오브젝트에 QbColorFeatureClass 특성을 추가합니다.

```
#include <dmbqbapi.h>

rc = QbQueryAddFeature(qoHandle,
        "QbColorFeatureClass");
```

QbQueryCreate

Image	Audio	Video
X		

조회 오브젝트를 생성하고 핸들을 리턴합니다. 조회 오브젝트를 조작하기 위해 다른 API에서 그 핸들을 사용할 수 있습니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryCreate(
      QbQueryHandle *qObj
      );
```

매개변수

qObj(출력)

조회 핸들에 대한 포인터. 실패한 경우, 이 핸들은 0으로 설정됩니다.

오류 코드

qbiECallocation

시스템이 충분한 메모리를 할당할 수 없습니다.

예

조회 오브젝트를 생성하고 qoHandle에 핸들을 리턴합니다.

```
#include <dmbqbapi.h>

rc = QbQueryCreate(&qoHandle);
```

QbQueryDelete

Image	Audio	Video
X		

이름 지정되지 않은 조회 오브젝트를 삭제합니다. API는 해당 조회 오브젝트에 의해 사용된 모든 메모리 및 추가된 특성을 할당해제합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryDelete(
      QbQueryHandle qObj
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

오류 코드

qbiECinvalidQueryHandle

지정한 조회 오브젝트 핸들은 유효한 조회를 참조하지 않습니다.

예

qoHandle 핸들에 의해 식별된 조회 오브젝트를 삭제합니다.

```
#include <dmbqbapi.h>
rc = QbQueryDelete(qoHandle);
```

QbQueryGetFeatureCount

Image	Audio	Video
X		

조회 오브젝트에 추가된 특성의 수를 리턴합니다. 아래 특성이 Image Extender와 함께 제공됩니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryGetFeatureCount(
      QbQueryHandle qObj,
      SQLINTEGER* count
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

count(출력)

현재 있는 특성의 수로 설정될 변수에 대한 포인터.

오류 코드

qbiECinvalidQueryHandle

사용자가 지정한 조회 오브젝트 핸들은 유효한 조회 오브젝트를 참조하지 않습니다.

예

qoHandle 핸들에 의해 식별된 조회 오브젝트용 특성의 수를 리턴합니다.

```
#include <dmbqbapi.h>

rc = QbQueryGetFeatureCount(qoHandle,
       &count);
```

QbQueryGetString

Image	Audio	Video
X		

조회에서 조회 문자열을 리턴합니다. 응용프로그램에서 UDF에 대한 입력에 조회 문자열을 사용할 수 있습니다. 예를 들어, UDF QbScoreFromStr이나 API QbQueryStringSearch에서 이를 사용할 수 있습니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryGetString(
      QbQueryHandle qObj,
      (char*)*  queryString
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

queryString(출력)

조회 오브젝트의 조회 문자열에 대한 포인터.

오류 코드

qbiECinvalidQueryHandle

지정한 조회 핸들이 유효한 조회를 참조하지 않습니다.

예

핸들 qrHandle에 의해 식별된 조회 오브젝트의 조회 문자열을 리턴합니다.

```
#include <dmbqbapi.h>

SQLRETURN  rc;
char *queryString;
QbQueryHandle qrHandle

rc = QbQueryGetString(qrHandle, &queryString);
if (rc == 0)  {
   ...                  /* use the returned queryString for input to UDFs */
   free((void*)queryString);  /* you must free queryString  */
   queryString=(char*)0;
}
```

QbQueryListFeatures

Image	Audio	Video
X		

조회 오브젝트의 특성의 현재 목록을 리턴합니다. API는 목록을 할당한 버퍼에 리턴됩니다. 아래 특성이 Image Extender와 함께 제공됩니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryListFeatures(
      QbQueryHandle qObj,
      SQLINTEGER bufSize,
      SQLINTEGER* count,
      char *featureNames
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

bufSize(입력)

특성 이름 버퍼의 *크기*. 버퍼 *크기*로 qbiMaxFeatureName 상수를 사용합니다. 조회 오브젝트 특성은 문자열 이름에 의해 식별됩니다.

count(출력)

리턴된 특성 이름의 수.

featureNames(출력)

조회 오브젝트용 특성 이름의 배열에 대한 포인터. 배열은 할당한 버퍼에 저장됩니다.

오류 코드

qbiECinvalidQueryHandle

지정한 조회 핸들이 유효한 조회를 참조하지 않습니다.

예

qoHandle에 의해 식별된 조회 오브젝트의 특성의 수를 리턴합니다. 필요한 버퍼의 *크기*를 결정하기 위해 qbiMaxFeatureName 상수를 사용합니다. 특성 이름을 특성 버퍼에 리턴하고 특성의 수를 retCount 변수에 리턴합니다.

```
#include <dmbqbapi.h>

bufSize = qbiMaxFeatureName;

rc = QbQueryListFeatures(qoHandle, bufSize,
        &retCount, feats);
```

QbQueryNameCreate

Image	Audio	Video
X		

조회 오브젝트를 이름 지정하고 저장하여 UDF에서 사용할 수 있게 해줍니다. 사용자는 조회 오브젝트의 이름을 제공하며, 설명을 제공할 수도 있습니다.

주:

1. **EEE 전용:** QbQueryNameCreate는 파티션된 데이터베이스 환경에서 지원되지 않습니다.
2. QbQueryNameCreate는 파티션되지 않은 데이터베이스 환경의 경우 나중 릴리스에서 거부될 것입니다. 조회를 저장하려면, QbQueryGetString을 사용하여 조회 문자열을 확보하고 응용프로그램에서 나중에 사용하기 위해 그 문자열을 저장해야 합니다.

권한

없음.

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryNameCreate(
      QbQueryHandle qObj,
      char *name,
      char *description
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

name(입력)

조회 오브젝트의 이름. 이름은 18문자까지 가능합니다.

description(입력)

조회 오브젝트의 간략한 설명으로, 250문자까지 가능합니다

오류 코드

qbiECinvalidQueryHandle

지정한 조회 오브젝트 핸들이 유효한 조회를 참조하지 않습니다.

예

QbQueryCreat API로 생성된 조회 오브젝트에 이름과 설명을 부여합니다.

```
#include <dmbqbapi.h>

rc = QbQueryNameCreate(qHandle,
       "fshavgcol",
       "average color query, 10/15/96");
```

QbQueryNameDelete

Image	Audio	Video
X		

조회 오브젝트를 삭제합니다. 조회 오브젝트는 반드시 QbQueryNameCreate API를 사용하여 이름 지정되고 저장되어야 합니다.

주:

1. **EEE 전용:** QbQueryNameDelete는 파티션된 데이터베이스 환경에서 지원되지 않습니다.
2. QbQueryNameDelete는 파티션되지 않은 데이터베이스 환경의 경우 나중 릴리스에서 거부될 것입니다.

권한

없음.

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryNameDelete(
      char *name
      );
```

매개변수

name(입력)

삭제할 조회 오브젝트의 이름.

오류 코드

qbiECinvalidQueryHandle

사용자가 지정한 조회 오브젝트 핸들은 유효한 조회 오브젝트를 참조하지 않습니다.

예

fshavgcol로 이름 지정된 조회 오브젝트를 삭제합니다.

```
#include <dmbqbapi.h>

rc = QbQueryNameDelete("fshavgcol",);
```

QbQueryNameSearch

Image	Audio	Video
X		

조회 오브젝트에 포함된 탐색 기준에 필적하는 이미지용 QBIC 카탈로그를 탐색합니다. 조회 오브젝트는 그 이름에 의해 식별됩니다. 이미지 핸들과 QBIC 탐색 스코어를 포함하는 결과는 클라이언트 메모리의 결과 배열에 저장됩니다. 결과는 스코어에 따라 정렬됩니다.

주:

1. **EEE 전용:** QbQueryNameSearch는 파티션된 데이터베이스 환경에서 지원되지 않습니다.
2. QbQueryNameSearch는 파티션되지 않은 데이터베이스 환경의 경우 나중 릴리스에서 거부될 것입니다. 조회를 저장하려면, QbQueryGetString을 사용하여 조회 문자열을 확보하고 응용프로그램에서 나중에 사용하기 위해 그 문자열을 저장해야 합니다.

권한

Select

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryNameSearch(
      char *qName,
      char *tableName,
      char *columnName,
      SQLINTEGER maxReturns,
      QbQueryScope* scope,
      SQLINTEGER resultType,
      SQLINTEGER* count,
      QbResult* returns
      );
```

매개변수

qName(입력)

조회 오브젝트의 이름.

tableName(입력)

탐색하고자 하는 이미지의 컬럼을 포함하고 있는 테이블 이름.

columnName(입력)

이미지 컬럼의 이름. 컬럼은 이미지 데이터용으로 사용 가능해야 합니다.

maxReturns(입력)

리턴할 최대 이미지 수.

scope(입력)(예약됨)

널(NULL)에 설정되어야 합니다.

resultType(입력)(예약됨)

qbiArray에 설정되어야 합니다.

count(출력)

리턴된 이미지의 수에 대한 포인터. 영(0)이 리턴된 경우, 이미지 컬럼이 조회 오브젝트의 모든 특성용으로 카탈로그화되었는지 확인하십시오.

returns(출력)

리턴된 결과를 보관하고 있는 QbResult 구조의 배열에 대한 포인터. 예상되는 모든 결과를 보관하기에 충분히 큰 버퍼를 할당했는지를 확인하십시오.

오류 코드

qbiECinvalidQueryHandle

지정한 조회 오브젝트 핸들이 유효한 조회 오브젝트를 참조하지 않습니다.

예

employee 테이블의 picture 컬럼에 카탈로그화된 이미지에 대해서 FSHAVGCOL 조회를 수행합니다. 6개를 초과한 이미지가 리턴되지 않도록 확인합니다.

```
#include <dmbqbapi.h>

rc = QbQueryNameSearch("fshavgcol",
       "employee", "picture",
       6, 0, qbiArray, &count, &returns);
```

QbQueryRemoveFeature

Image	Audio	Video
X		

조회 오브젝트에서 조회 특성을 제거하고 모든 연관된 메모리를 할당해제합니다. 아래 특성이 Image Extender와 함께 제공됩니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryRemoveFeature(
      QbQueryHandle qObj,
      char *featureName
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

featureName(입력)

제거될 특성 이름.

오류 코드

qbiECinvalidQueryHandle

사용자가 지정한 조회 오브젝트 핸들은 유효한 조회 오브젝트를 참조하지 않습니다.

qbiECinvalidFeatureClass

지정한 특성이 유효한 이름 형식이 아닙니다.

qbiECfeatureNotPresent

지정한 특성이 조회 오브젝트의 구성원이 아닙니다.

예

qoHandle에 의해 식별된 조회 오브젝트로부터 QbColorFeatureClass 특성을 제거합니다.

```
#include <dmbqbapi.h>

rc = QbQueryRemoveFeature(qoHandle,
        "QbColorFeatureClass");
```

QbQuerySearch

Image	Audio	Video
X		

조회 오브젝트에 포함된 탐색 기준에 필적하는 이미지용 QBIC 카탈로그를 탐색합니다. 조회 오브젝트는 조회 오브젝트 핸들에 의해 식별됩니다. 이미지 핸들과 QBIC 탐색 스코어를 포함하는 결과는 클라이언트 메모리의 결과 배열 안에 저장됩니다. 스코어에 따라 정렬됩니다.

권한

Select

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQuerySearch(
      QbQueryHandle qObj,
      char *tableName,
      char *columnName,
      SQLINTEGER maxReturns,
      QbQueryScope* scope,
      SQLINTEGER resultType,
      SQLINTEGER* count,
      QbResult* returns
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

tableName(입력)

탐색하고자 하는 이미지의 컬럼을 포함하고 있는 테이블 이름.

columnName(입력)

이미지 컬럼의 이름. 컬럼은 이미지 데이터용으로 사용 가능해야 합니다.

maxReturns(입력)

리턴하고자 하는 이미지의 최대수.

scope(입력)(예약됨)

0(널(NULL))으로 설정되어야 합니다.

resultType(입력)(예약됨)

qbiArray에 설정되어야 합니다.

count(출력)

리턴된 이미지의 수에 대한 포인터. 영(0)이 리턴된 경우, 이미지 컬럼이 조회 오브젝트의 모든 특성용으로 카탈로그화되었는지 확인하십시오.

returns(출력)

리턴된 결과를 보관하고 있는 QbResult 구조의 배열에 대한 포인터. 예상되는 모든 결과를 보유하기에 충분히 큰 버퍼를 할당했는지를 확인하십시오.

오류 코드

qbiECinvalidQueryHandle

사용자가 지정한 조회 오브젝트 핸들은 유효한 조회 오브젝트를 참조하지 않습니다.

예

employee 테이블의 picture 컬럼의 카탈로그화된 이미지를 조회합니다. 6개를 초과한 이미지가 리턴되지 않도록 확인합니다.

```
#include <dmbqbapi.h>

rc = QbQuerySearch(qHandle, "employee",
       "picture", 6, 0, qbiArray,
       &count, &returns);
```

QbQuerySetFeatureData

Image	Audio	Video
X		

조회 오브젝트의 특성용 이미지 데이터의 소스를 설정합니다. 조회 오브젝트에 특성을 추가한 후에만 데이터 소스를 설정할 수 있습니다. 데이터 소스는 사용자 테이블, 파일 또는 워크스테이션 버퍼 내의 이미지가 될 수 있습니다. 파티션되지 않은 데이터베이스 환경에서만 데이터 소스로 클라이언트 파일이나 워크스테이션 버퍼를 사용할 수 있습니다. 또한, 평균 색상에 대한 데이터나 색상 특성을 정확하게 지정할 수 있습니다.

QbQuerySetFeatureData를 사용하여 서버 파일에서 이미지 데이터에 대한 소스를 설정한 다음에 QbQueryStringSearch를 사용하십시오. QbQuerySearch는 QbQuerySetFeatureData로 설정된 서버 파일의 이미지 데이터에 대해 소스를 사용하지 않습니다.

다음 특성이 Image Extender와 함께 제공됩니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQuerySetFeatureData(
      QbQueryHandle qObj,
      char *featureName,
      QbImageSource* imgSource
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

featureName(입력)

설정될 특성 이름.

imgSource(입력)

이미지 소스 구조에 대한 포인터. 이미지 소스용으로 널(NULL)을 설정한 경우, 정보가 특성 내에서 변경되어서는 안된다는 것을 의미합니다. 자세한 내용은 170 페이지의 『데이터 소스 구조 사용』을 참조하십시오.

오류 코드

qbiECinvalidQueryHandle

사용자가 지정한 조회 오브젝트 핸들은 유효한 조회 오브젝트를 참조하지 않습니다.

qbiECunknownFeatureClass

지정한 특성이 인식되는 특성 클래스 이름이 아닙니다.

qbiECinvalidFeatureClass

지정한 특성이 유효한 이름 형식이 아닙니다.

qbiECfeatureNotPresent

지정한 특성이 조회 오브젝트의 구성원이 아닙니다

qbiECfileUnreadable

이미지 소스 파일을 발견하거나 읽을 수 없습니다.

예

조회 오브젝트에 히스토그램 색상 특성에 대한 데이터 소스를 설정합니다. 특성에 대한 데이터 소스는 클라이언트 워크스테이션상의 파일입니다.

```
#include <dmbqbapi.h>

QbQueryHandle qoHandle;
QbImageSource imgSource;

imgSource.sourceType = qbiSource_ClientFile;
strcpy(featureName, "QbColorHistogramFeatureClass");
strcpy(imgSource.clientFile, "/tmp/image.gif");

rc = QbQuerySetFeatureData(qoHandle, featureName, &imgSource);
```

QbQuerySetFeatureWeight

Image	Audio	Video
X		

조회 오브젝트의 지정된 특성의 가중치를 설정합니다.

권한

없음

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQuerySetFeatureWeight(
      QbQueryHandle qObj,
      sqldouble* weight
      );
```

매개변수

qObj(입력)

조회 오브젝트의 핸들.

weight(출력)

특성 가중치를 설정할 변수에 대한 포인터.

오류 코드

qbiECinvalidQueryHandle

사용자가 지정한 조회 오브젝트 핸들이 유효한 오브젝트를 참조하지 않습니다.

예

qoHandle 핸들에 의해 식별된 조회 오브젝트에 평균 색상 특성용 가중치를 설정합니다.

```
#include <dmbqbapi.h>

weight=2.0
rc = QbQuerySetFeatureWeight(qoHandle, "QbColorFeatureClass", &weight);
```

QbQueryStringSearch

Image	Audio	Video
X		

조회 문자열에 포함된 탐색 기준에 필적하는 이미지용 QBIC 카탈로그를 탐색합니다. 이미지 핸들과 QBIC 탐색 스코어를 포함하는 결과는 클라이언트 메모리의 결과 배열 안에 저장됩니다. 스코어에 따라 정렬됩니다.

권한

Select

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqqry.lib	libdmbqqry.a(AIX) libdmbqqry.sl(HP-UX) libdmbqqry.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbQueryStringSearch(
      char *queryString,
      char *tableName,
      char *columnName,
      SQLINTEGER maxReturns,
      QbQueryScope* scope,
      SQLINTEGER resultType,
      SQLINTEGER* count,
      QbResult* returns
      );
```

매개변수

queryString(입력)

조회 문자열.

tableName(입력)

탐색하고자 하는 이미지의 컬럼을 포함하고 있는 테이블 이름.

columnName(입력)

이미지 컬럼의 이름. 컬럼은 이미지 데이터용으로 사용 가능해야 합니다.

maxReturns(입력)

리턴할 최대 이미지 수.

scope(입력)(예약됨)

0(널(NULL))으로 설정되어야 합니다.

resultType(입력)(예약됨)

qbiArray에 설정되어야 합니다.

count(출력)

리턴된 이미지의 수에 대한 포인터. 영(0)이 리턴된 경우, 이미지 컬럼이 조회 문자열의 모든 특성용으로 카탈로그화되었는지를 확인하십시오.

returns(출력)

리턴된 결과를 보관하고 있는 QbResult 구조의 배열에 대한 포인터. 예상되는 모든 결과를 보관하기에 충분히 큰 버퍼를 할당했는지를 확인하십시오.

오류 코드

qbiECinvalidQueryString

지정한 조회 문자열이 유효하지 않습니다.

예

employee 테이블의 picture 컬럼의 카탈로그화된 이미지를 조회합니다. 6개를 초과한 이미지가 리턴되지 않도록 확인합니다.

```
#include <dmbqbapi.h>

rc = QbQueryStringSearch("QbColorFeatureClass color=<255, 0, 0>"
       "employee",
       "picture", 6, 0, qbiArray,
       &count, &returns);
```

QbReCatalogColumn

Image	Audio	Video
X		

새 특성용으로 열린 QBIC 카탈로그 내의 모든 존재하는 이미지를 재분석합니다. 특성의 기본 매개변수가 사용됩니다. 이미 이미지를 가진 카탈로그에 새 특성을 추가할 때 이 API를 사용하십시오.

권한

Update, Insert

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbReCatalogColumn (
      QbCatalogHandle cHdl
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

qbicECInvalidCatalog

지정된 핸들 또는 테이블 컬럼이 카탈로그에 대해 유효하지 않습니다.

qbicECCatalog Errors

개별 이미지를 카탈로그화는 동안 오류가 발생하였으며, 해당 오류가 기록되었습니다. 구간 복원(rollback)은 발생하지 않습니다.

qbicECImageNotFound

이미지를 발견할 수 없거나 이미지에 액세스할 수 없습니다.

qbicECCatalogRO

카탈로그는 읽기 전용입니다.

qbicECSQLError

SQL 오류가 발생했습니다.

예

새 특성용으로 열린 QBIC 카탈로그 내의 모든 존재하는 이미지를 재분석합니다.

```
#include <dmbqbapi.h>

rc = QbReCatalogColumn(CatHdl);
```

QbRemoveFeature

Image	Audio	Video
X		

열린 카탈로그로부터 지정된 특성을 삭제합니다.

권한

Alter

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbRemoveFeature(
      QbCatalogHandle cHdl,
      char *featureName
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

featureName(입력)

특성의 이름.

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

qbicECCatalogReadOnly

카탈로그가 읽기 전용으로 열려 있습니다.

qbicECFeatureNotFound

특성이 카탈로그에 없습니다.

qbiECinvalidFeatureClass

지정한 특성이 유효한 이름 형식이 아닙니다.

예

핸들 CatHdl에 의해 식별된 카탈로그에서 QbColorHistogramFeatureClass 특성을 제거합니다.

```
#include <dmbqbapi.h>

rc=QbRemoveFeature(CatHdl,
     "QbColorHistogramFeatureClass");
```

QbSetAutoCatalog

Image	Audio	Video
X		

이미지 컬럼으로 가져온 이미지를 자동으로 카탈로그화합니다. API는 특성 테이블에 각 이미지용으로 항목을 추가하고 이미지를 분석합니다. API가 이미지를 분석할 때, 이미지 데이터를 생성하고 특성 테이블 내의 이미지 항목 내에 저장합니다.

자동 카탈로그화를 작동 상태로 두지 않는 경우에, 이미지 컬럼에 이미지를 추가한 후에 QbCatalogColumn 또는 QbCatalogImage API를 사용하십시오.

권한

Alter

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbSetAutoCatalog(
      QbCatalogHandle cHdl
      SQLINTEGER autoCatalog
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

autoCatalog(입력)

이미지 컬럼에 추가된 이미지가 자동으로 특성 테이블에 추가되고 분석될 지의 여부를 표시합니다. 자동 카탈로그화에는 1을 지정하고, 그렇지 않으면 0을 지정합니다.

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

예

핸들 CatHdl에 의해 식별된 카탈로그용으로 자동 카탈로그화를 설정합니다.

```
#include <dmbqbapi.h>

rc=QbSetAutoCatalog(CatHdl, 1);
```

QbUncatalogImage

Image	Audio	Video
X		

카탈로그에서 이미지를 제거합니다. 이미지 핸들은 반드시 열린 QBIC 카탈로그와 연관된 이미지 컬럼에서 기인해야 합니다. 이미지는 열린 카탈로그에서 제거될 것입니다. 이미지 속성 테이블 내의 상응하는 행이 이미지가 카탈로그화되지 않았음을 표시합니다.

권한

Delete

라이브러리 파일

OS/2 및 Windows	AIX, HP-UX 및 Solaris
dmbqbapi.lib	libdmbqbapi.a(AIX) libdmbqbapi.sl(HP-UX) libdmbqbapi.so(Solaris)

Include 파일

dmbqbapi.h

구문

```
SQLRETURN QbUncatalogImage(
      QbCatalogHandle cHdl,
      char *imgHandle
      );
```

매개변수

cHdl(입력)

카탈로그의 핸들에 대한 포인터.

imgHandle(입력)

이미지의 핸들. 사용자 테이블에서 이 핸들을 검색할 수 있습니다.

오류 코드

qbicECIvalidHandle

카탈로그 핸들이 유효하지 않습니다.

qbicECImageNotFound

이미지를 발견할 수 없거나 이미지에 액세스할 수 없습니다.

qbicECCatalogRO

카탈로그는 읽기 전용입니다.

예

핸들 CatHdl에 의해 식별된 카탈로그에서 핸들 Img_hdl에 의해 식별된 이미지를 제거합니다.

```
#include <dmbqbapi.h>

rc=QbUncatalogImage(CatHdl, Img_hdl);
```

제17장 클라이언트용 관리 명령

이 장은 클라이언트용 DB2 Extender 관리 명령을 입력하는 방법을 설명합니다. 또한 클라이언트용 각 DB2 Extender 관리 명령에 관한 참조 정보를 제공합니다.

DB2 Extender 관리 명령 입력

대화식 모드 또는 명령 모드에서 db2ext 명령행 처리기에 DB2 Extender 관리 명령을 제출할 수 있습니다. 대화식 모드는 db2ext 프롬프트에 의해 특징지어집니다. 이 모드에서 사용자는 단지 DB2 Extender 관리 명령만 입력할 수 있습니다. 명령 모드에서 운영 체제 명령 프롬프트로부터 명령을 입력할 수 있습니다. DB2 명령과 운영 체제 명령 뿐만 아니라 DB2 Extender 명령도 입력할 수 있습니다.

DB2 명령 프롬프트에서는 DB2 Extender 명령을 입력하지 마십시오.

대화식 모드에서 db2ext 명령행 처리기(CLP)를 시작하려면 다음을 수행하십시오.

클라이언트	조치
OS/2	DB2 Extender 폴더의 DB2EXT 명령행 처리기(CLP) 아이콘을 더블 클릭하거나 OS/2 명령 프롬프트에서 DB2EXT 명령을 입력하십시오.
AIX, HP-UX, Solaris	운영 체제 명령 프롬프트에서 DB2EXT 명령을 입력하십시오.
Windows	DB2 Extender 폴더 내의 DB2EXT 명령행 처리기 아이콘을 더블 클릭하거나 DB2 명령 창에서 DB2EXT 명령을 입력하십시오.

대화식 모드를 종료하려면 quit 또는 terminate 명령을 입력하십시오. quit 명령은 대화식 모드를 종료하지만 DB2에 대한 현재 연결을 유지합니다. terminate 명령은 대화식 모드를 종료하고 DB2에 대한 현재 연결을 제거합니다.

명령 모드에서 DB2 Extender 명령을 제출하려면, 운영 체제 명령행에서 명령을 입력하십시오. 반드시 db2ext를 각 DB2 Extender 명령 앞에 써야 합니다. 예를 들면, 다음과 같습니다.

```
db2ext enable database for db2image using mydataspace, myindxspace, mylongspace
```

DB2 Extender 명령에 대한 온라인 도움말 확보

모든 DB2 Extender 명령에 대한 온라인 도움말을 확보하려면 다음을 입력하십시오.

```
db2ext ?
```

ADD QBIC FEATURE

Image	Audio	Video
X		

현재 카탈로그 내의 지정된 특성용 특성 테이블을 생성합니다. 카탈로그에 존재하는 이미지가 Image Extender에 의해 자동적으로 재분석되지 않습니다.

권한

Alter, Control, SYSADM, DBADM

명령 구문

명령 매개변수

feature_name

사용자가 QBIC 카탈로그에 추가한 특성의 이름. 다음 특성이 Image Extender와 함께 제공됩니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

예

현재 열린 카탈로그에 QbColorFeatureClass 특성을 추가합니다.

```
add qbic feature qbcolorfeatureclass
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

카탈로그는 반드시 열려 있어야 합니다.

CATALOG QBIC COLUMN

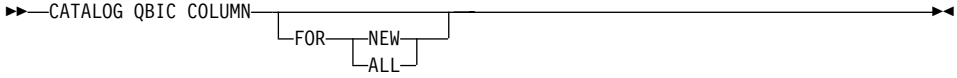
Image	Audio	Video
X		

이미지 컬럼의 이미지를 카탈로그화하고 특성 데이터를 사용하여 현재 열린 QBIC 카탈로그를 갱신합니다. 이미지 컬럼의 모든 이미지에 대한 카탈로그를 갱신할 수도 있으며, 카탈로그가 마지막으로 분석된 이후로 이미지 컬럼에 추가된 새 이미지에 대한 카탈로그만 갱신할 수도 있습니다.

권한

Insert, Control, SYSADM, DBADM

명령 구문



명령 매개변수

없음

예

현재 카탈로그로 새 이미지, 즉 카탈로그화되지 않은 이미지를 카탈로그화합니다.

```
catalog qbic column for new
```

사용법 참고

NEW가 지정되면, Image Extender는 단지 카탈로그화되지 않은 이미지만을 사용하여 카탈로그를 갱신합니다. ALL이 지정되면, Image Extender는 현재 카탈로그에 대한 이미지 컬럼에 있는 모든 이미지를 분석합니다. NEW가 기본값입니다.

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

카탈로그는 반드시 열려 있어야 합니다.

CLOSE QBIC CATALOG

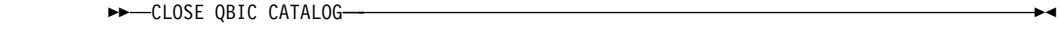
Image	Audio	Video
X		

QBIC 카탈로그를 닫습니다.

권한

없음

명령 구문



명령 매개변수

없음

예

현재 카탈로그를 닫습니다.

```
close qbic catalog
```

사용법 참고

QBIC 카탈로그는 반드시 열려 있어야 합니다.

CONNECT

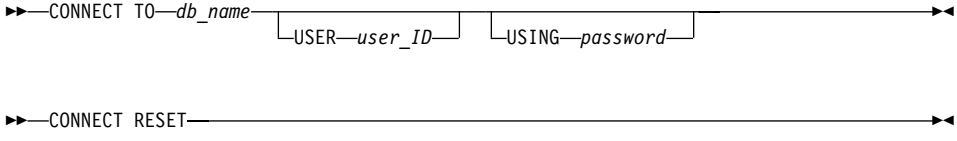
Image	Audio	Video
X	X	X

데이터베이스에 연결합니다. Extender는 DB2 연결과 별도로, 데이터베이스에 독립적인 연결을 요구합니다.

권한

Connect

명령 구문



명령 매개변수

db_name
데이터베이스의 이름.

user_ID
데이터베이스에 연결되도록 공인된 사용자 ID.

password
사용자 ID용 암호.

RESET
모든 보류된 변경을 확약한 후에 데이터베이스에서 연결을 끊습니다.

예

PERSONNL 데이터베이스에 연결합니다. 사용자 ID는 anita이며 암호는 anitapas입니다.

```
connect to personnl user anita
   using anitapas
```

사용법 참고

데이터베이스는 SHARE 모드로 연결됩니다.

어떤 다른 Extender 명령을 수행하기 전에 이 명령을 수행하십시오.

CREATE QBIC CATALOG

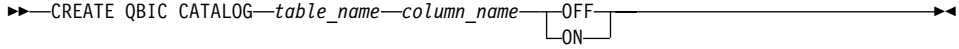
Image	Audio	Video
X		

지정된 DB2IMAGE 컬럼용으로 현재 데이터베이스에 QBIC 카탈로그를 생성합니다. Extender는 자동적으로 카탈로그 이름을 생성합니다.

권한

Alter, Control, SYSADM, DBADM

명령 구문



명령 매개변수

table_name

DB2IMAGE 사용 가능한 테이블의 이름.

column_name

DB2IMAGE 사용 가능한 컬럼의 이름.

OFF 이미지를 수동으로 카탈로그화합니다.

ON 이미지를 자동으로 카탈로그화합니다.

tablespace_name

QBIC 카탈로그에 대한 테이블 공간 스펙 및 색인 옵션. 스펙에는 다음과 같은 네 가지 부분이 있습니다.

- 특성 데이터를 포함하는 카탈로그 테이블에 대한 테이블 공간명. 테이블 공간을 지정해야 합니다. 테이블 공간이 세그먼트화된 테이블 공간이어야 합니다.

- 카탈로그 테이블에 작성된 색인의 경우, 사용중인 블록, 사용 가능 블록, gbpcache 블록 또는 유형 2 파티션되지 않은 색인에 대한 색인 옵션의 임의의 조합. 이 스펙은 선택적입니다. 이 부분을 지정하지 않는 경우 기본값을 갖게 됩니다.
- 카탈로그 로그 테이블에 대한 테이블 공간 이름. 테이블 공간은 단순한 테이블 공간이거나 세그먼트화된 테이블 공간일 수 있습니다. 이 스펙은 선택적입니다. 로그 테이블에 대한 테이블 공간을 지정하지 않는 경우, 특성 데이터 테이블에 대해 지정된 테이블 공간이 사용됩니다.
- 로그 데이터 테이블에 작성된 색인의 경우, 사용중인 블록, 사용 가능 블록, gbpcache 블록 또는 유형 2 파티션되지 않은 색인에 대한 색인 옵션의 임의의 조합. 이 스펙은 선택적입니다. 이 부분을 지정하지 않는 경우 기본값을 갖게 됩니다.

예

자동 카탈로그화를 ON으로 설정한 상태에서 employee 테이블에서 picture 컬럼용 QBIC 카탈로그를 생성합니다.

```
create qbic catalog employee picture on
```

사용법 참고

ON을 지정하면 컬럼으로 가져온 이미지는 자동적으로 연관된 QBIC 카탈로그 내로 카탈로그화됩니다. 기본값은 OFF입니다.

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

DELETE QBIC CATALOG

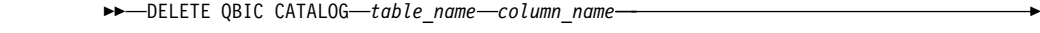
Image	Audio	Video
X		

모든 QBIC 탐색 지원 데이터를 포함한 QBIC 카탈로그를 삭제합니다.

권한

Alter, Control, SYSADM, DBADM

명령 구문



명령 매개변수

table_name

DB2IMAGE 사용 가능한 테이블의 이름.

column_name

DB2IMAGE 사용 가능한 컬럼의 이름.

예

employee 테이블의 picture 컬럼과 연관된 카탈로그를 삭제합니다.

```
delete qbic catalog employee picture
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

DISABLE COLUMN

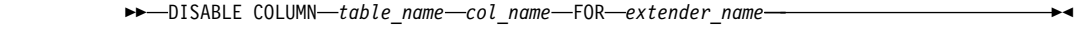
Image	Audio	Video
X	X	X

지정된 미디어 데이터를 저장한 것에서 지정된 컬럼을 사용 불가능하게 합니다.

권한

SYSADM, DBADM, Control, Alter

명령 구문



명령 매개변수

table_name

현재 데이터베이스 내의 테이블 이름.

col_name

사용 불가능하게 하고자 하는 컬럼 이름.

extender_name

컬럼을 사용 불가능하게 하고자 하는 Extender 이름. 유효한 Extender 이름은 db2image, db2audio 및 db2video입니다.

예

employee 테이블의 picture 컬럼을 사용 불가능하게 하여 이미지 데이터를 보관할 수 없게 합니다.

```
disable column employee photo for db2image
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

컬럼을 사용 불가능하게 한 경우:

- 컬럼이 지정된 Extender용으로 데이터를 저장할 수 없습니다. 테이블의 다른 컬럼이 멀티미디어 데이터 유형용으로 사용 가능한지의 여부에는 영향을 미치지 않습니다.
- 컬럼 항목의 내용은 널(NULL)에 설정되어 있으며 관리 테이블의 상응하는 행은 삭제됩니다.
- 컬럼과 연관된 트리거가 삭제됩니다.

DISABLE DATABASE

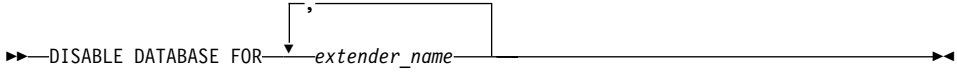
Image	Audio	Video
X	X	X

미디어 데이터 저장에서 현재 데이터베이스를 사용 불가능하게 합니다.

권한

SYSADM, DBADM

명령 구문



명령 매개변수

extender_name

현재 데이터베이스를 사용 불가능하게 하고자 하는 Extender 이름. 유효한 Extender 이름은 db2image, db2audio 및 db2video입니다.

예

이미지 데이터 보관에서 현재 데이터베이스를 사용 불가능하게 합니다.

```
disable database for db2image
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

데이터베이스를 사용 불가능하게 했을 때, 시스템은

- 지정된 Extender용으로만 사용 가능한 모든 테이블을 사용 불가능하게 합니다.
- 지정된 Extender용 UDF 관리 지원 테이블을 삭제합니다.

DISABLE TABLE

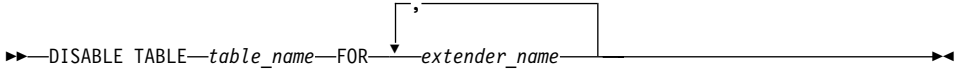
Image	Audio	Video
X	X	X

미디어 데이터 저장에서 지정된 테이블을 사용 불가능하게 합니다.

권한

SYSADM, DBADM, Control, Alter

명령 구문



명령 매개변수

table_name

현재 데이터베이스에서 사용 불가능하게 할 테이블의 이름.

extender_name

테이블을 사용 불가능하게 하고자 하는 Extender 이름. 유효한 Extender 이름은 db2image, db2audio 및 db2video입니다.

예

이미지 데이터 보관에서 employee 테이블을 사용 불가능하게 합니다.

```
disable table employee for db2image
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

테이블을 사용 불가능하게 했을 때, 시스템은

- 지정된 Extender용으로 사용 가능한 테이블의 모든 컬럼을 사용 불가능하게합니다.
- 테이블과 연관된 관리 지원 테이블을 삭제합니다.

DISCONNECT SERVER AT NODENUM(EEE 전용)

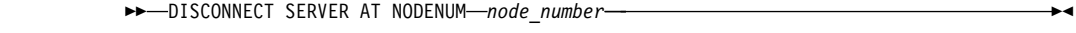
Image	Audio	Video
X	X	X

모든 데이터베이스에 대해 지정된 노드에서 서버를 연결해제합니다.

권한

SYSADM, SYSCTRL, SYSMAINT, DBADM

명령 구문



명령 매개변수

node_number

서버에서 연결해제할 노드.

예

노드 번호 2에 있는 모든 데이터베이스에서 서버를 연결해제합니다.

```
disconnect server at nodenum 2
```

사용법 참고

모든 노드에 대한 모든 데이터베이스에서 서버를 연결해제하려면 DMBSTOP 명령을 사용하십시오.

DISCONNECT SERVER FOR DATABASE(EEE 전용)

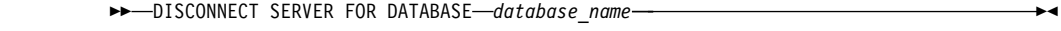
Image	Audio	Video
X	X	X

지정된 데이터베이스의 모든 노드에서 서버를 연결해제합니다.

권한

SYSADM, SYSCTRL, SYSMAINT, DBADM

명령 구문



명령 매개변수

database_name
서버에서 연결해제할 데이터베이스.

예

이름이 MY_DATABASE인 데이터베이스에서 서버를 연결해제합니다.

```
disconnect server for database my_database
```

사용법 참고

모든 노드에 대한 모든 데이터베이스에서 서버를 연결해제하려면 DMBSTOP 명령을 사용하십시오.

DISCONNECT SERVER FOR DATABASE AT NODENUM(EEE 전용)

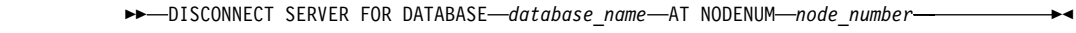
Image	Audio	Video
X	X	X

지정된 노드의 지정된 데이터베이스에서 서버를 연결해제합니다.

권한

SYSADM, SYSCTRL, SYSMAINT, DBADM

명령 구문



명령 매개변수

database_name

서버에서 연결해제할 데이터베이스.

node_number

서버에서 연결해제할 노드.

예

노드 번호 2에 있는 이름이 MY_DATABASE인 데이터베이스에서 서버를 연결해제합니다.

```
disconnect server for database my_database at nodenum 2
```

사용법 참고

모든 노드에 대한 모든 데이터베이스에서 서버를 연결해제하려면 DMBSTOP 명령을 사용하십시오.

ENABLE COLUMN

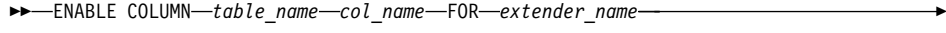
Image	Audio	Video
X	X	X

미디어 데이터를 저장하기 위해 지정된 컬럼을 사용 가능하게 합니다.

권한

SYSADM, DBADM, Control, Alter

명령 구문



명령 매개변수

table_name
현재 데이터베이스 내의 테이블 이름.

col_name
사용 가능하게 하고자 하는 컬럼 이름.

extender_name
테이블을 사용 가능하게 하고자 하는 Extender 이름. 유효한 Extender 이름은 db2image, db2audio 및 db2video입니다.

예

employee 테이블의 photo 컬럼을 사용 가능하게 하여 이미지 데이터를 보관하게 합니다.

```
enable column employee photo for db2image
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

ENABLE DATABASE

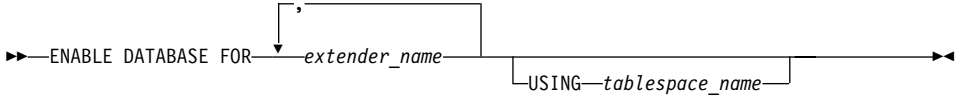
Image	Audio	Video
X	X	X

지정된 테이블 공간을 사용하여 미디어 데이터를 보관하기 위해 현재 데이터베이스를 사용 가능하게 합니다.

권한

SYSADM, SYSCTRL, DBADM

명령 구문



명령 매개변수

extender_name

현재 데이터베이스를 사용 가능하게 하고자 하는 Extender 이름. 유효한 Extender 이름은 db2image, db2audio 및 db2video입니다.

tablespace_name

관리 테이블이 저장되는 컨테이너의 집합인 테이블 공간 이름. 테이블 공간 이름은 *datats, indexts, longts*의 세 부분으로 되어 있으며, 여기서 *datats*는 메타데이터 테이블이 생성되는 테이블 공간 이름입니다. *indexts*는 메타데이터 테이블에 대한 색인이 생성되는 테이블 공간 이름입니다. *longts*는 (LONG VARCHAR 및 LOB 데이터 유형을 포함한 것과 같은) 메타데이터 테이블 내의 긴 컬럼의 값이 저장되는 테이블 공간 이름입니다. 테이블 공간 이름의 특정 부분에 널(NULL) 값을 제공하면, 그 부분에 대한 기본 테이블 공간 이름이 사용됩니다. 지정된 테이블 공간은 파티션된 데이터베이스 시스템의 모든 노드가 포함되는 노드 그룹에 정의되어 있어야 합니다.

예

이미지 데이터를 보관하기 위해 현재 데이터베이스를 사용 가능하게 합니다.

```
enable database for db2image using mydataspace, myindxspace, mylongspace
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

테이블 공간이 지정되지 않은 경우에 시스템은 관리 테이블용으로 USERSPACE1 테이블 공간을 사용합니다.

ENABLE TABLE

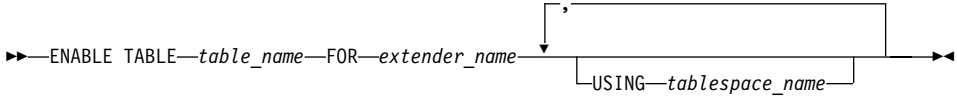
Image	Audio	Video
X	X	X

지정된 테이블 공간을 사용하여 미디어 데이터를 보관하기 위해 지정된 테이블을 사용 가능하게 합니다.

권한

SYSADM, DBADM, Control, Alter

명령 구문



명령 매개변수

table_name

현재 데이터베이스에서 사용 가능하게 하고자 하는 테이블 이름.

extender_name

테이블을 사용 가능하게 하고자 하는 Extender 이름. 유효한 Extender 이름은 db2image, db2audio 및 db2video입니다.

tablespace_name

관리 테이블이 저장되는 컨테이너의 집합인 테이블 공간의 이름. 테이블 공간 스펙은 *datats, indexts, longts*의 세 부분으로 되어 있으며, 여기서 *datats*는 메타데이터 테이블이 생성되는 테이블 공간입니다. *indexts*는 메타데이터 테이블에 대한 색인이 생성되는 테이블 공간입니다. *longts*는 (LONG VARCHAR 및 LOB 데이터 유형을 포함한 것과 같은) 메타데이터 테이블 내의 긴 컬럼의 값이 저장되는 테이블 공간입니다. 테이블 공간 스펙의 임의의 부분에 널(NULL) 값을 제공하는 경우, 그 부분에 대한 기본 테이블 공간이 사용됩니다.

테이블 공간 스펙의 임의의 부분에 대해 널(NULL) 값을 제공하는 경우, 이 부분에 대한 기본 테이블 공간이 사용됩니다.

EEE 전용: 지정된 테이블 공간은 사용자 테이블과 동일한 노드 그룹에 있어야 합니다.

예

이미지 데이터를 보관하기 위해 employee 테이블을 사용 가능하게 합니다.

```
enable table employee for db2image
   using mydataspace, myindxspace, mylongspace
```

이미지 데이터를 보관하기 위해 employee 테이블을 사용 가능하게 합니다. 다음과 같은 기본 테이블 공간을 사용하십시오.

```
enable table employee for db2image
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

테이블 공간이 지정되지 않은 경우, 시스템은 현재 데이터베이스를 사용 가능화할 때 정의된 테이블 공간을 사용합니다.

GET EXTENDER STATUS

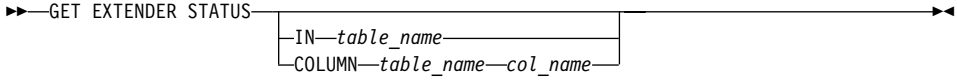
Image	Audio	Video
X	X	X

드문 경우이나 컬럼, 테이블 또는 현재 데이터베이스가 사용 가능한 Extender의 이름을 표시합니다.

권한

없음

명령 구문



명령 매개변수

table_name

현재 데이터베이스의 테이블 이름.

col_name

컬럼의 이름.

예

데이터베이스에서 사용할 수 있는 Extender의 이름을 표시합니다.

```
get extender status
```

employee 테이블의 상태를 표시합니다.

```
get extender status in employee
```

employee 테이블의 ADDRESS 컬럼의 상태를 표시합니다.

```
get extender status column employee address
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

GET INACCESSIBLE FILES

Image	Audio	Video
X	X	X

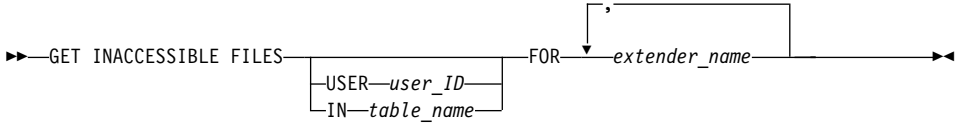
특정 규정자를 가지고 있는 테이블이나 현재 데이터베이스 내의 모든 테이블에 의해 액세스할 수는 없지만 참조되는 모든 미디어 파일을 나열합니다.

권한

현재 데이터베이스 내의 모든 테이블, 즉 사용자가 USER 또는 IN을 지정하지 않은 경우: SYSADM, SYSCTRL, SYSMAINT, DBADM.

특정한 테이블(IN을 지정한 경우) 또는 규정자에 속하는 테이블(USER를 지정한 경우)의 경우: Select

명령 구문



명령 매개변수

user_ID

액세스할 수 없는 파일을 나열하려고 하는 현재 데이터베이스 내의 테이블 규정자.

table_name

액세스할 수 없는 파일을 나열하려고 하는 현재 데이터베이스 내의 테이블 이름.

extender_name

Extender의 이름. 유효한 Extender 이름은 db2image, db2audio 및 db2video입니다.

예

데이터베이스 내의 테이블에 의해 참조되지만 액세스할 수 없는 모든 이미지 파일을 나열합니다.

```
get inaccessible files
   for db2image
```

anita 규정자로 테이블에서 참조되었으나 액세스할 수 없는 모든 이미지 파일을 나열합니다.

```
get inaccessible files
   user anita for db2image
```

employee 테이블의 항목에 의해 참조되었으나 액세스할 수 없는 모든 이미지 파일을 나열합니다.

```
get inaccessible files
   in employee FOR db2image
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

테이블을 지정한 경우에 명령은 그 테이블에 대해 액세스 불가능한 파일을 나열합니다. 규정자를 지정한 경우에 명령은 단지 그 규정자를 가진 테이블에 대해 액세스 불가능한 파일만을 나열합니다. 아무 것도 지정하지 않는 경우, 명령은 현재 데이터베이스 내의 모든 테이블에 대해 액세스할 수 없는 파일을 나열합니다.

GET QBIC CATALOG INFO

Image	Audio	Video
X		

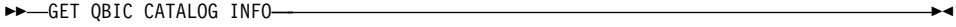
현재 열려 있는 카탈로그에 대한 다음 정보를 리턴합니다.

- 카탈로그가 연관된 사용자 테이블 및 이미지 컬럼의 이름.
- 카탈로그 내의 특성 이름.
- 카탈로그 내의 특성 수.
- 자동 분석이 작동중인지의 여부.

권한

Select, Control, SYSADM, DBADM

명령 구문



명령 매개변수

없음

예

현재 열려 있는 QBIC 카탈로그에 관한 정보를 확보합니다.

```
get qbic catalog info
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

카탈로그는 반드시 열려 있어야 합니다.

GET REFERENCED FILES

Image	Audio	Video
X	X	X

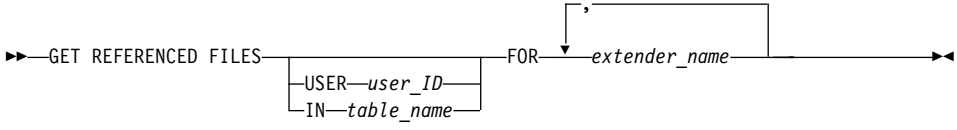
특정 규정자를 가지고 있는 테이블 또는 현재 데이터베이스 내의 모든 테이블에서 에서 참조되는 모든 미디어 파일과 컬럼 이름을 나열합니다.

권한

현재 데이터베이스 내의 모든 테이블 즉, USER 또는 IN을 지정하지 않는 경우: SYSADM, SYSCTRL, SYSMAINT, DBADM.

특정한 테이블(IN을 지정한 경우) 또는 규정자에 속하는 테이블(USER를 지정한 경우)의 경우: Select

명령 구문



명령 매개변수

user_ID

참조된 파일을 나열하려는 데이터베이스의 테이블 규정자. 명령은 단지 그 규정자를 가지고 있는 테이블만을 탐색합니다.

table_name

참조된 파일을 나열하려는 현재 데이터베이스의 테이블 이름. 명령은 그 테이블만을 탐색합니다.

extender_name

Extender의 이름. 유효한 Extender 이름은 db2image, db2audio 및 db2video입니다.

예

데이터베이스 내의 모든 테이블에 있는 테이블 항목에서 참조하는 모든 이미지 파일을 나열합니다.

```
get referenced files
   for db2image
```

테이블에서 anita 규정자를 가지고 있는 항목에 의해 참조된 모든 이미지 파일을 나열합니다.

```
get referenced files
   user anita for db2image
```

employee 테이블의 항목에 의해 참조된 모든 이미지 파일을 나열합니다.

```
get referenced files
   in employee for db2image
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

어떤 매개변수도 지정하지 않는 경우, 명령은 데이터베이스 내의 모든 테이블을 탐색합니다.

GET SERVER STATUS

Image	Audio	Video
X	X	X

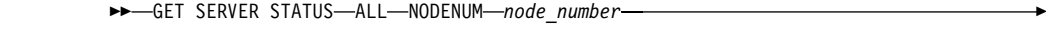
현재 데이터베이스나 모든 데이터베이스에 대한 Extender 서버 상태를 표시합니다.

EEE 전용: 노드를 지정하면, 이 명령은 그 노드에서의 Extender 서버 상태(현재 데이터베이스나 모든 데이터베이스에 대한)만을 표시합니다.

권한

없음

명령 구문



명령 매개변수

ALL 모든 데이터베이스의 상태를 표시합니다.

node_number
노드 번호. 명령은 이 노드의 상태를 표시합니다**(EEE 전용)**.

예

현재 데이터베이스에 대한 Extender 서버의 상태를 표시합니다.

```
get server status
```

모든 데이터베이스에 대한 Extender 서버의 상태를 표시합니다.

```
get server status all
```

모든 데이터베이스에 대한 노드 번호 2의 Extender 서버 상태를 표시합니다.

```
get server status all nodenum 2
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

매개변수를 지정하지 않을 경우, 이 명령은 db2nodes.cfg 파일에 나열된 현재 데이터베이스의 모든 노드의 상태를 표시합니다.

OPEN QBIC CATALOG

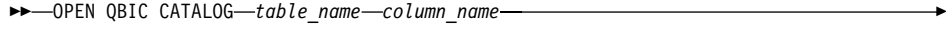
Image	Audio	Video
X		

지정된 DB2IMAGE 컬럼을 위해 카탈로그를 엽니다. 데이터베이스는 항상 갱신 모드로 카탈로그를 열려고 할 것입니다. 카탈로그가 이미 갱신 모드에 있는 경우 카탈로그는 읽기 모드로 열릴 것입니다.

권한

Connect

명령 구문



명령 매개변수

table_name
DB2IMAGE 사용 가능한 테이블의 이름.

column_name
DB2IMAGE 사용 가능한 컬럼의 이름.

예

employee 테이블의 picture 컬럼용 QBIC 카탈로그를 엽니다.

```
open qbic catalog employee picture
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

이 명령은 열린 카탈로그를 모두 닫히게 할 것입니다.

QUIT

Image	Audio	Video
X	X	X

대화식 모드의 명령 항목용 db2ext 명령행 처리기를 닫습니다. DB2 연결은 유지되고, 따라서 여전히 명령 모드의 db2ext 명령행 처리기에 명령을 제출할 수 있습니다.

권한

없음

명령 구문

명령 매개변수

없음

예

대화식 모드용 명령행 인터페이스를 닫습니다.

```
quit
```

사용법 참고

QUIT은 데이터베이스로의 연결을 유지합니다.

RECONNECT SERVER AT NODENUM(EEE 전용)

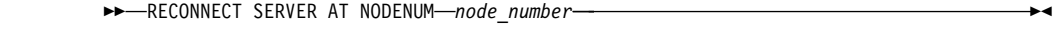
Image	Audio	Video
X	X	X

모든 데이터베이스 상의 지정된 노드에 서버를 재연결합니다.

권한

SYSADM, SYSCTRL, SYSMAINT, DBADM

명령 구문



명령 매개변수

node_number
서버에 재연결할 노드.

예

노드 번호 2에 있는 모든 데이터베이스에 서버를 재연결합니다.

```
reconnect server at nodenum 2
```

사용법 참고

모든 노드 상의 모든 데이터베이스에서 서버를 재연결하려면, DMBSTART 명령을 사용하십시오.

RECONNECT SERVER FOR DATABASE(EEE 전용)

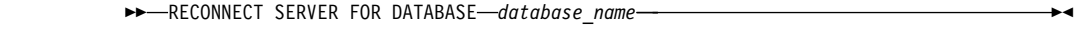
Image	Audio	Video
X	X	X

지정된 데이터베이스의 모든 노드에 서버를 재연결합니다.

권한

SYSADM, SYSCTRL, SYSMAINT, DBADM

명령 구문



명령 매개변수

database_name

서버에 재연결할 데이터베이스.

예

이름이 MY_DATABASE인 데이터베이스에 서버를 재연결합니다.

```
disconnect server for database my_database
```

사용법 참고

모든 노드 상의 모든 데이터베이스에 서버를 재연결하려면, DMBSTART 명령을 사용하십시오.

RECONNECT SERVER FOR DATABASE AT NODENUM(EEE 전용)

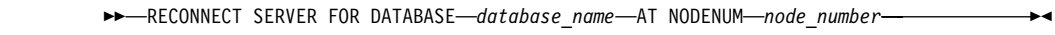
Image	Audio	Video
X	X	X

지정된 노드의 지정된 데이터베이스에 서버를 재연결합니다.

권한

SYSADM, SYSCTRL, SYSMAINT, DBADM

명령 구문



명령 매개변수

database_name

서버에 재연결할 데이터베이스.

node_number

서버에 재연결할 노드.

예

노드 번호 2에 있는 이름이 MY_DATABASE인 데이터베이스에 서버를 재연결합니다.

```
reconnect server for database my_database at nodenum 2
```

사용법 참고

모든 노드 상의 모든 데이터베이스에 서버를 재연결하려면, DMBSTART 명령을 사용하십시오.

REDISTRIBUTE NODEGROUP(EEE 전용)

Image	Audio	Video
X		

노드 그룹에 노드가 추가되거나 노드 그룹에서 노드가 제거될 때, 또는 노드 그룹에 대한 새로운 파티션 맵이 설정될 때, Extender 데이터를 다시 분산시킵니다.

권한

SYSADM, DBADM

명령 구문

명령 매개변수

nodegroup

다시 분산시킬 노드 그룹의 이름.

CONTINUE

재분산 프로세스가 오류를 리턴하면, 명령 응답에서 제공하는 지시사항에 따라 CONTINUE 매개변수가 있는 명령이나 CONTINUE 매개변수가 없는 명령을 재수행할 수 있습니다. 이 옵션은 시스템에 처음부터 시작하지 말고 정지된 부분에서 계속하도록 지시합니다.

예

my_nodegroup이라는 노드 그룹을 재분산시킵니다.

```
redistribute nodegroup my_nodegroup
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

REDISTRIBUTE NODEGROUP

DB2의 REDISTRIBUTE 명령 다음에 처음으로 REDISTRIBUTE NODEGROUP 명령을 실행할 때 CONTINUE 매개변수를 사용해서는 안됩니다. 이 때 CONTINUE 매개변수를 사용하면, 오류가 기록되고 재분산이 처음부터 시작됩니다.

데이터 무결성을 유지하려면, 한번에 한 개의 노드 그룹을 재분산시켜야 합니다. 하나의 노드 그룹이 재분산을 완료한 후에 다른 노드 그룹의 재분산을 시작하십시오.

REDISTRIBUTE NODEGROUP이 실패할 경우, 다음 디렉토리 중 하나에 있는 ″redist.log″ 파일에서 자세한 설명을 참조할 수 있습니다.

- **Unix:** /<home-instance>/dmb/redist
- **Windows:**
  \\<instance_owning_machine>\DB2<instance_name>\<instance_name>\dmb\redist

REMOVE QBIC FEATURE

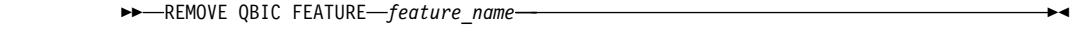
Image	Audio	Video
X		

열린 카탈로그로부터 지정된 특성의 특성 테이블을 삭제합니다.

권한

Alter, Control, SYSADM, DBADM

명령 구문



명령 매개변수

feature_name

사용자가 QBIC 카탈로그에서 제거할 특성의 이름. 다음 특성이 Image Extender와 함께 제공됩니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

예

현재 열려 있는 카탈로그에서 QbColorFeatureClass 특성을 제거합니다.

```
remove qbic feature qbcolorfeatureclass
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

카탈로그는 반드시 열려 있어야 합니다.

REORG

Image	Audio	Video
X	X	X

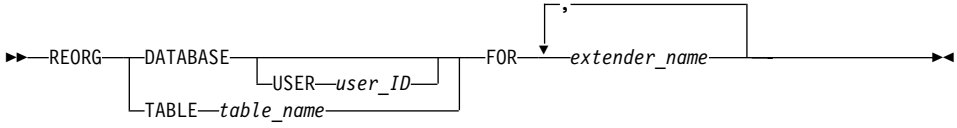
특정 테이블과 연관된 관리 테이블(관리 테이블 및 속성 테이블) 또는 특정 규정자를 가진 테이블 또는 현재 데이터베이스 내의 모든 테이블을 정리합니다.

권한

특정 테이블(REORG TABLE을 수행하는 경우) 또는 특정 규정자를 가진 테이블(REORG DATABASE를 수행하는 경우)의 경우: SYSADM, SYSCTRL, SYSMAINT, DBADM, Control.

데이터베이스 내의 모든 테이블(REORG DATABASE를 수행하는 경우)의 경우: SYSADM, SYSCTRL, SYSMAINT, DBADM.

명령 구문



명령 매개변수

user_ID

테이블의 규정자.

table_name

그 관리 테이블을 정리하고자 하는 현재 데이터베이스 내의 테이블 이름.

extender_name

Extender의 이름. 유효한 Extender 이름은 db2image, db2audio 및 db2video입니다.

예

현재 데이터베이스용으로 이미지 관리 테이블을 재구성하고 정리합니다.

```
reorg database for db2image
```

anita 규정자를 가진 모든 테이블의 이미지 관리 테이블을 재구성하고 정리합니다.

```
reorg database user anita for db2image
```

employee 테이블용 이미지 관리 테이블을 재구성하고 정리합니다.

```
reorg table employee for db2image
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

SET QBIC AUTOCATALOG

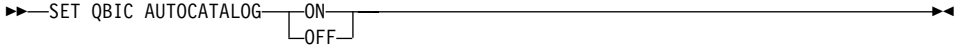
이미지	Audio	Video
X		

컬럼 내로 가져올 때 자동으로 이미지를 카탈로그화합니다. 이미지는 컬럼과 연관된 QBIC 카탈로그에 추가됩니다.

권한

Alter, Control, SYSADM, DBADM

명령 구문



명령 매개변수

없음

예

자동 카탈로그화를 작동 상태로 설정합니다.

```
set qbic autocatalog on
```

사용법 참고

QBIC 카탈로그는 반드시 열려 있어야 합니다.

START SERVER(비EEE 전용)

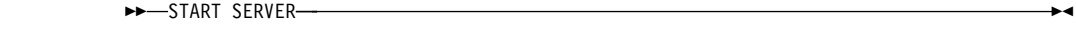
Image	Audio	Video
X	X	X

현재 데이터베이스용 Extender 서버를 시작합니다.

권한

SYSADM, SYSCTRL, SYSMAINT, DBADM

명령 구문



명령 매개변수

없음

예

현재 데이터베이스용 Extender 서버를 시작합니다.

```
start server
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

STOP SERVER(EEE 이외에서만)

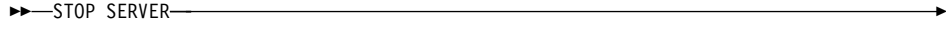
Image	Audio	Video
X	X	X

현재 데이터베이스용 Extender 서버를 멈춥니다.

권한

SYSADM, SYSCTRL, SYSMAINT, DBADM

명령 구문



명령 매개변수

없음

예

현재 데이터베이스용 Extender 서버를 멈춥니다.

```
stop server
```

사용법 참고

이 명령을 사용하기 전에 데이터베이스에 연결하십시오.

TERMINATE

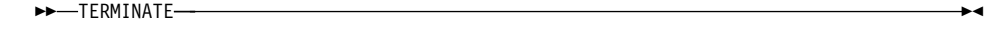
Image	Audio	Video
X	X	X

db2ext 명령행 처리기를 종료하고 DB2에 대한 연결을 삭제합니다.

권한

없음

명령 구문



명령 매개변수

없음

예

db2ext 명령행 처리기를 종료합니다.

```
quit
```

사용법 참고

TERMINATE는 데이터베이스에 대한 연결을 삭제합니다.

제18장 서버용 관리 명령

이 장의 명령은 서버 운영 체제의 명령행에서 수행됩니다. 이 명령을 DB2 명령행 또는 db2ext 명령행에서 수행하지 마십시오. 서버 시스템을 종료하고 재시작할 때마다 DMBSTART 명령을 수행하십시오.

EEE 전용: 또한 다중 파티션 데이터베이스 환경에서 DMBSTART 및 DMBSTOP 서버 명령을 수행할 수도 있습니다. 다중 파티션 데이터베이스 환경에서 서버 명령을 수행할 때 노드 번호를 지정하지 않으면, 이 명령은 모든 노드에 적용됩니다. 노드 번호를 지정하면, 지정된 노드에만 명령이 적용됩니다.

EEE 전용: DMBSTAT 명령은 다중 파티션 환경에서는 수행할 수 없습니다. 클라이언트 명령 GET SERVER STATUS ALL을 수행함으로써, 다중 파티션 환경에서 서버 상태를 점검할 수 있습니다.

DMBICRT

Image	Audio	Video
X	X	X

DB2 Extender 인스턴스를 작성합니다. DB2의 여러 인스턴스를 가지고 있으면, DB2 Extender 서버의 여러 인스턴스를 작성해야 합니다. UNIX에서, DB2 Extender 클라이언트를 설치할 때 클라이언트 인스턴스를 작성합니다. 클라이언트 인스턴스를 작성하면 DB2 Extender 사용을 위한 환경이 설정됩니다.

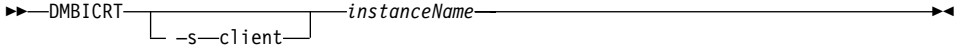
권한

SYSADM

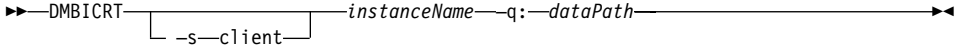
UNIX에서, 루트 권한을 가지고 있어야 합니다.

명령 구문

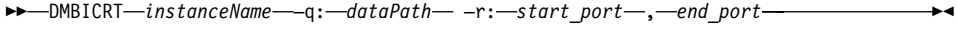
파티션되지 않은 데이터베이스 환경에서:



UNIX의 파티션된 데이터베이스 환경에서:



Windows의 파티션된 데이터베이스 환경에서:



명령 매개변수

instanceName 기존의 DB2 인스턴스 이름. DB2 인스턴스가 이 이름으로 존재하지 않으면, 이를 작성할 것인지 묻는 프롬프트가 표시됩니다.

–s client 클라이언트 전용 인스턴스를 작성할 것을 지정합니다. 이 매개변

수를 사용할 경우, *instanceName*은 클라이언트의 사용자 ID입니다. 클라이언트 인스턴스를 작성하면 클라이언트용 환경이 설정됩니다**(UNIX 전용)**.

dataPath 공유 디렉토리나 파일 시스템의 이름. 그 디렉토리는 모든 노드에서 존재해야 합니다. 이 디렉토리는 UNIX에서 DB2MMDATAPATH 환경 변수로, 그리고 Windows에서는 레지스트리로 설정됩니다**(EEE 전용)**.

start_port, end_port
사용할 TCP/IP 포트의 범위. 포트 범위는 작업하는 노드 수와 같거나 더 많아야 합니다. 포트 번호는 Windows 레지스트리에 기록됩니다**(Windows EEE 전용)**.

예

파티션되지 않은 데이터베이스 환경에서 DB2 인스턴스 DEVINST에 대한 DB2 Extender 서버의 인스턴스를 작성합니다.

```
dmbicrt  devinst
```

사용법 참고

DMBICRT 명령은 인스턴스에 의해 사용되는 파일에 대해 DB2 Extender 인스턴스 디렉토리를 작성합니다. 이 디렉토리의 이름은 다음과 같습니다.

- *install_directory*\INSTANCE*instance_name*. 여기서 *install_directory*는 DB2 Extender를 설치한 디렉토리입니다**(Windows, OS/2)**.
- *INSTHOME*/dmb. 여기서 *INSTHOME*은 인스턴스 소유자의 홈 디렉토리입니다 **(UNIX)**.

DMBICRT 명령을 사용할 때 지정된 이름의 DB2 인스턴스가 존재하지 않으면, 이를 작성하도록 하는 프롬프트가 표시됩니다.

EEE 전용:

참여하는 노드의 루트 사용자 ID에서 DMBICRT를 실행할 수 있어도, 같은 노드를 사용하여 모든 DB2 Extender 서버 인스턴스를 작성하는 것이 바람직합니다. 노드는 DB2 인스턴스를 작성하기 위해 사용되는 노드와, 그리고 DB2

인스턴스 디렉토리가 상주하는 노드와 같아야 합니다. 다른 노드를 사용하여 DB2 Extender 서버 인스턴스를 작성할 경우, 노드에 저장되는 인스턴스 목록이 완전하지 않을 수도 있습니다.

*dataPath*로 지정된 공유 디렉토리나 파일 시스템은 UNIX에서는 *$INSTHOME*/dmb/dmbprofile에 DB2MMDATAPATH 환경 변수의 값으로, Windows에서는 다음의 레지스트리의 키로 저장됩니다.

```
\\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2 Extenders\PROFILES
      \instance_name\DB2MMDATAPATH
```

UNIX에서, 인스턴스를 작성하려면 먼저 /etc/services 파일에 포트 범위를 추가해야 합니다. 다음 구문을 사용하여 두 항목을 파일에 추가하십시오.

- DMB_*instance_name start_port*
- DMB_*instance_name*_END *end_port*

범위는 파티션된 데이터베이스 환경에서 모든 노드에 대해 충분히 커야 합니다.

DB2 Extender 서버의 인스턴스를 작성하기 전에 DB2 인스턴스를 작성해야 합니다.

파티션된 데이터베이스 환경에서 Text Extender 인스턴스를 작성하려면, *DB2 Universal Database Text Extender 관리 및 프로그래밍*에 설명된 대로 TXTICRT 명령을 사용하십시오.

DMBIDROP

Image	Audio	Video
X	X	X

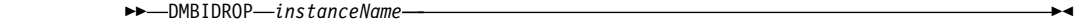
DB2 Extender 인스턴스를 삭제합니다.

권한

SYSADM

UNIX에서, 루트 권한을 가지고 있어야 합니다.

명령 구문



명령 매개변수

instanceName 삭제할 DB2 Extender 인스턴스의 이름.

예

DEVINST라고 하는 DB2 Extender 서버를 제거합니다.

```
dmbidrop  devinst
```

사용법 참고

이 명령을 실행하기 전에, 다음을 수행하십시오.

- 인스턴스와 모든 db2ext 명령행 처리기를 사용하는 모든 응용프로그램을 중지하십시오.
- Extender 서비스를 중지하십시오.

DMBIDROP 명령은 DB2 Extender 인스턴스 디렉토리를 제거하고, 인스턴스 목록에서 해당 인스턴스 항목을 제거하며, 인스턴스에 대한 기타 정보를 제거합니다.

DMBIDROP 명령은 DB2 Extender 인스턴스만 제거하고, 연관된 DB2 인스턴스는 제거하지 않습니다. 명시적으로 DB2 인스턴스를 제거해야 합니다.

DB2 Extender 서버 인스턴스와 연관되는 DB2 인스턴스를 삭제해도 DB2 Extender 서버 인스턴스는 제거되지 않습니다. 그러나, 이를 사용할 수는 없습니다.

(EEE 전용) DB2 Extender 인스턴스를 제거할 경우, UNIX에서는 /etc/services 파일에서, Windows에서는 \WINNT\system32\drivers\etc\Services 파일에서 그 인스턴스에 대한 시작 및 종료 포트 항목을 제거해야 합니다. 이러한 항목은 DMB_*instanceName*mpp 및 DMB_*instanceName*mpp_END입니다.

DMBILIST

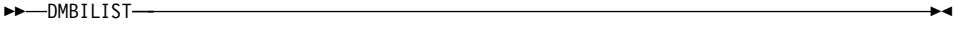
Image	Audio	Video
X	X	X

DB2 Extender의 모든 인스턴스를 나열합니다.

권한

없음

명령 구문



명령 매개변수

없음

예

DB2 Extender 인스턴스를 나열합니다.

```
dmbilist
```

DMBIMIGR

Image	Audio	Video
X	X	X

(UNIX 전용) DB2 Extender 인스턴스를 이전 릴리스에서 현재 릴리스로 이주합니다.

권한

루트 권한을 가지고 있어야 합니다.

명령 구문

명령 매개변수

instanceName 이주할 DB2 Extender 인스턴스의 이름.

예

OLDINST라고 하는 DB2 Extender 인스턴스를 이주합니다.

```
dmbimigr  oldinst
```

사용법 참고

이 명령을 실행하기 전에, 다음을 수행하십시오.

- DB2 Extender의 현재 릴리스가 설치되어 있어야 합니다.
- 연관된 DB2 인스턴스를 이주해야 합니다.

각각의 DB2 Extender 인스턴스에 대해 DMBIMIGR을 한번 실행하십시오. DMBILIST를 사용하여 인스턴스를 나열하십시오.

DMBSTART

Image	Audio	Video
X	X	X

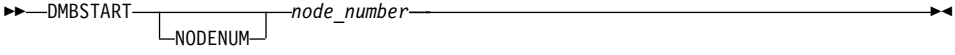
Extender 인스턴스용 모든 Extender 서비스를 시작합니다.

EEE 전용: 노드를 지정하면, 이 명령은 그 노드에서만 Extender 서비스를 시작합니다. DMBSTART는 또한 각 노드에서 노드 작성/삭제 기능을 시작합니다. 노드 작성/삭제 기능은 DB2에 대해 정의된 노드가 Extender에 대해 정의된 노드와 일치하는지 점검합니다. 이들이 일치하지 않을 경우, 노드 작성/삭제 기능은 필요에 따라 노드를 추가하거나 제거합니다.

권한

SYSADM

명령 구문



명령 매개변수

node_number

Extender 서비스를 시작할 노드(**EEE 전용**).

예

Extender 서비스를 시작합니다.

```
dmbstart
```

노드 번호 2에서 Extender 서비스를 시작합니다.

```
dmbstart nodenum 2
```

사용법 참고

이 명령은 다음 경우에 수행됩니다.

- 인스턴스 소유자로서 로그온되어 있는 동안(AIX, HP-UX 또는 Solaris).
- DB2INSTANCE 환경 변수가 시작하고자 하는 인스턴스와 동일한 창에서(OS/2 또는 Windows에서).
- 서버 시스템을 종료하고 재시작할 때마다.

단일 파티션 환경에서, DMBSTART는 DB2 인스턴스가 수행중이 아닐 경우 이 인스턴스를 시작하기도 합니다.

EEE 전용:

다중 파티션 환경에서, DMBSTART는 DB2 인스턴스를 시작하지 않습니다. 파티션된 환경에서 DMBSTART를 수행하기 전에 DB2를 시작해야 합니다.

DMBSTART가 실패할 경우, 다음을 확인하십시오.

- DBD2MMDATAPATH 변수의 값이 올바른지.
- 변수의 파일 시스템이나 공유 디렉토리가 존재하는지, 그리고 모든 노드에서 액세스할 수 있는지.

DMBSTAT

Image	Audio	Video
X	X	X

사용 가능한 데이터베이스와 그 데이터베이스용 Extender 서비스가 수행 중인지를 표시합니다.

권한

없음

명령 구문

명령 매개변수

없음

예

Extender 서비스의 상태를 표시합니다.

```
dmbstat
```

DMBSTOP

Image	Audio	Video
X	X	X

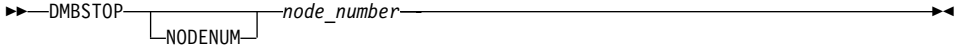
Extender 인스턴스용 모든 Extender 서비스를 중단합니다.

EEE 전용: 노드를 지정하면, DMBSTOP은 그 노드에서만 Extender 서비스를 중단합니다.

권한

SYSADM

명령 구문



명령 매개변수

node_number

Extender 서비스를 중단할 노드**(EEE 전용)**.

예

Extender 서비스를 중단합니다.

```
dmbstop
```

노드 번호 2에서 Extender 서비스를 중단합니다.

```
dmbstop nodenum 2
```

사용법 참고

이 명령은 다음 경우에 수행됩니다.

- 인스턴스 소유자로서 로그온되어 있는 동안(AIX, HP-UX 또는 Solaris에서).
- DB2INSTANCE 환경 변수가 중단하고자 하는 인스턴스와 동일한 창에서(OS/2 또는 Windows에서).

DMBSTOP은 DB2 인스턴스를 중단시키지 않습니다.

EEE 전용: 유지보수 모드에서 작업 중인 때를 제외하고, 특정 노드에서 DMBSTOP을 수행하지 마십시오. 또한, 노드가 작동 중지 상태일 때는 이 노드에서 Extender 활동이 트리거되지 않도록 해야 합니다. 그러지 않을 경우, 예기치 못한 동작이 발생할 수 있습니다.

제19장 진단 정보

DB2 Extender UDF를 호출하는 프로그램을 포함하여, 사용자 프로그램 내의 모든 Embedded SQL문과 DB2 CLI 호출은 Embedded SQL문 또는 DB2 CLI 호출이 성공적으로 실행되었는지를 표시하는 코드를 생성합니다. 관리 API와 같은 기타 DB2 Extender API도 역시 성공 여부를 표시하는 코드를 리턴합니다. 사용자 프로그램은 반드시 이 리턴 코드를 점검하고 반응해야 합니다.

또한 사용자 프로그램에서 이들 코드를 보완하는 정보를 검색할 수도 있습니다. 이것은 SQLSTATE 정보 및 오류 메시지를 포함합니다. 사용자는 프로그램에서 문제를 격리시키고 해결하기 위해 이 진단 정보를 사용할 수 있습니다.

가끔 문제의 소스가 쉽게 진단되지 않을 수도 있습니다. 이런 경우에 문제를 격리시키고 해결하기 위해 서비스 담당자에게 정보를 제공해야 할 필요가 있을 수 있습니다. DB2 Extender는 Extender 활동을 기록하는 추적 기능을 포함합니다. 추적 정보는 서비스 담당자에게 유용한 입력이 될 수 있습니다. IBM 서비스 담당자의 지시 하에서만 추적 기능을 사용해야 합니다.

이 장은 이 진단 정보에 액세스하는 방법을 설명합니다. 이 장은 다음 사항을 설명합니다.

- DB2 Extender UDF 리턴 코드 및 API 리턴 코드 처리법
- 추적 제어법

이 장은 또한 Extender에 의해 리턴되는 SQLSTATE 및 오류 메시지를 나열하고 설명합니다.

UDF 리턴 코드 처리

Embedded SQL문은 SQLCA 구조의 SQLCODE, SQLWARN 및 SQLSTATE 필드에 코드를 리턴합니다. 이 구조는 SQLCA Include 파일에서 정의됩니다. (SQLCA 구조 및 SQLCA Include 파일에 관한 자세한 정보는 *DB2 응용프로그램 개발 안내서*를 참조하십시오.)

DB2 CLI 호출은 SQLError 기능을 이용하여 검색할 수 있는 SQLCODE 및 SQLSTATE 값을 리턴합니다. (SQLError 기능을 사용한 오류 정보 검색에 관한 자세한 정보는 *CLI Guide and Reference*를 참조하십시오.)

SQLCODE 값이 0이면 명령문이 성공적으로 실행되었음을 의미합니다.(경고 상태가 동반될 수 있음). 양수 SQLCODE 값은 명령문이 성공적으로 실행되었으나 경고를 동반함을 나타냅니다. (Embedded SQL문은 SQLWARN 필드에 0 또는 양수의 SQLCODE 값과 연관된 경고를 리턴합니다.) 음수의 SQLCODE 값은 오류 조건이 발생했음을 의미합니다.

DB2는 각 SQLCODE 값과 메시지를 연관시킵니다. DB2 Extender UDF가 경고나 오류 조건을 만나면 SQLCODE 메시지에 포함시키기 위해 연관된 정보를 DB2에 전달합니다.

SQLSTATE 값은 SQLCODE 메시지를 보완하는 코드를 포함하고 있습니다. DB2 Extender에서 리턴되는 SQLSTATE 코드 각각에 대한 설명은 587 페이지의 『SQLSTATE 코드』를 참조하십시오.

DB2 Extender UDF를 호출하는 Embedded SQL문 및 DB2 CLI 호출은 이 UDF들에 고유한 SQLCODE 메시지와 SQLSTATE 값을 리턴할 수도 있지만 DB2는 다른 Embedded SQL문 또는 기타 DB2 CLI 호출에 대한 것과 동일한 방법으로 이 값들을 리턴합니다. 따라서 이 값에 액세스하는 방법은 DB2 Extender UDF를 시작하지 않는 Embedded SQL문 또는 DB2 CLI 호출의 방법과 동일합니다.

Extender에 의해 리턴 가능한 SQLSTATE 값 및 연관된 메시지의 메시지 번호에 대해 보려면 587 페이지의 『SQLSTATE 코드』를 참조하십시오. 각 메시지에 대한 정보를 보려면 592 페이지의 『메시지』를 참조하십시오.

API 리턴 코드 처리

각 DB2 Extender API 호출은 코드를 리턴합니다. 0의 리턴 코드는 API 호출이 성공적으로 처리되었음을 의미합니다. 0 외의 리턴 코드는 API 호출이 성공적으로 처리되었지만 경고 조건을 만났거나, 오류 조건으로 인해 성공적으로 처리되지 못했음을 나타냅니다.

291 페이지의 『제16장 응용프로그램 프로그래밍 인터페이스(API)』는 기호 값을 나열하고 DB2 Extender API에 의해 리턴 가능한 각 코드에 대해 설명하고 있습니다.

API에 의해 발생하는 오류에 관한 부가적 정보를 검색할 수 있습니다. 이 부가적인 정보를 검색하려면 DB*x*GetError API를 사용하십시오. 여기서 *x*는 Audio Extender용이며 i는 Image Extender용이며 v는 Video Extender용입니다. DB*x*GetError API는 오류가 발생한 마지막 DB2 Extender API에 대한 SQL 오류 코드와 연관된 메시지를 리턴합니다. SQL 오류 코드에 대한 정보는 *DB2 메시지 참조서*를 참조하십시오. 592 페이지의 『메시지』를 보면 *x*GetError API에 의해 리턴될 수 있는 각 메시지에 관한 정보를 참조할 수 있습니다.

예를 들어, C 응용프로그램의 다음 명령문은 테이블을 Audio Extender용으로 사용 가능하게 하고 오류를 점검합니다.

```
rc=DBaEnableTable((char *)NULL, "employee");

rc=DBaGetError(&errCode, &errMsg);
```

SQLSTATE 코드

표16은 DB2 Extender에 의해 리턴될 수 있는 SQLSTATE 값을 나열하고 설명합니다. 각 SQLSTATE 값의 설명은 기호 표시를 포함하고 있습니다. 표는 또한 각 SQLSTATE 값과 연관된 메시지 번호를 나열하고 있습니다. 각 메시지에 대한 정보를 보려면 592 페이지의 『메시지』를 참조하십시오.

표 16. SQLSTATE 코드 및 연관된 메시지 번호

SQLSTATE	메시지 번호	설명
00000		MMDB_SQLSTATE_OK 성공적
01H01	DMB0211W	MMDB_SQLSTATE_WARN_NO_OVERWRITE 파일 겹쳐쓰기가 일어나지 않습니다.
38A00	DMB0101E	MMDB_SQLSTATE_AUDIO_NULL_PARM UDF에 대한 입력 매개변수는 널(NULL)이 될 수 없습니다.
38A02	DMB0209E	MMDB_SQLSTATE_AUDIO_OPEN_HDR_ERROR 오디오 파일 헤더를 여는 동안 오류가 발생했습니다.

표 16. SQLSTATE 코드 및 연관된 메시지 번호 (계속)

SQLSTATE	메시지 번호	설명
38A03	DMB0209E	MMDB_SQLSTATE_AUDIO_BAD_WAVE_HDR 올바르지 않은 웨이브 파일이 제공되었습니다.
38V00	DMB0101E	MMDB_SQLSTATE_VIDEO_NULL_PARM UDF에 대한 입력 매개변수는 널(NULL)이 될 수 없습니다.
38V02	DMB0051E	MMDB_SQLSTATE_VIDEO_OPEN_HDR_ERROR 비디오 파일 헤더를 여는 동안 오류가 발생했습니다.
38V03	DMB0105E	MMDB_SQLSTATE_VIDEO_BAD_MPEG1_HDR 올바르지 않은 mpeg1 파일이 제공되었습니다.
38V04	DMB0104E	MMDB_SQLSTATE_VIDEO_BLOB_TOO_SHORT 제공된 비디오 버퍼가 너무 작습니다.
38V05	DMB0106E	MMDB_SQLSTATE_VIDEO_BAD_AVI_HDR 올바르지 않은 AVI 파일이 제공되었습니다.
38V07	DMB0106E	MMDB_SQLSTATE_VIDEO_BAD_QT_HDR 올바르지 않은 퀵타임(Quicktime) 파일이 제공되었습니다.
38600	DMB0075E DMB0101E DMB0102E DMB0103E DMB0210E	MMDB_SQLSTATE_INVALID_INPUT UDF에 대한 입력 매개변수가 올바르지 않습니다.
38601	DMB0009E	MMDB_SQLSTATE_MALLOC_FAIL 메모리 할당이 실패했습니다.
38602	DMB0386E	MMDB_SQLSTATE_CANNOT_COLLOCATE 사용자 데이터와 나란히 배치할 수 없습니다.
38603	DMB0077E	MMDB_SQLSTATE_READ_FILE_FAIL 파일을 읽을 수 없습니다.
38604	DMB0080E	MMDB_SQLSTATE_WRITE_FILE_FAIL 파일에 쓸 수 없습니다.
38610	DMB0070E	MMDB_SQLSTATE_INVALID_HANDLE 미디어 컬럼이 올바르지 않은 데이터를 포함하고 있습니다.
38611	DMB0073E	MMDB_SQLSTATE_INVALID_SESSION_HANDLE 올바르지 않은 UDF 세션 핸들
38612	DMB0074E	MMDB_SQLSTATE_INVALID_STATEMENT_HANDLE 올바르지 않은 UDF문 핸들
38613	DMB0083E	MMDB_SQLSTATE_INVALID_IMPORT_REQUEST 가져오기용 요청이 올바르지 않습니다.

표 16. SQLSTATE 코드 및 연관된 메시지 번호 (계속)

SQLSTATE	메시지 번호	설명
38615	DMB0071E	MMDB_SQLSTATE_CONNECT_FAIL 데이터베이스에 연결하는 동안 오류가 발생했습니다.
38617	DMB0071E	MMDB_SQLSTATE_ALLOC_STMT_FAIL 새 명령문 핸들을 할당하는 동안 오류가 발생했습니다.
38618	DMB0208E DMB0138E	MMDB_SQLSTATE_FREE_STMT_FAIL 명령문을 할당해제하는 동안 오류가 발생했습니다.
38619	DMB0208E DMB0132E	MMDB_SQLSTATE_BIND_FAIL 바인딩 동안 오류가 발생했습니다.
38620	DMB0208E	MMDB_SQLSTATE_BIND_COLUMN_FAIL 컬럼을 바인딩하는 동안 오류가 발생했습니다.
38621	DMB0208E	MMDB_SQLSTATE_BIND_FILE_FAIL 파일을 바인딩하는 동안 오류가 발생했습니다.
38622	DMB0208E DMB0132E	MMDB_SQLSTATE_SET_PARAM_FAIL 매개변수를 설정하는 동안 오류가 발생했습니다.
38623	DMB0208E DMB0131E	MMDB_SQLSTATE_PREPARE_FAIL SQL문을 준비하는 동안 오류가 발생했습니다.
38624	DMB0208E DMB0133E DMB0172E	MMDB_SQLSTATE_EXECUTE_FAIL 명령문을 실행하는 동안 오류가 발생했습니다.
38625	DMB0208E DMB0133E	MMDB_SQLSTATE_EXEC_DIRECT_FAIL UDF 내에서 직접적으로 SQL문을 실행하는 동안 오류가 발생했습니다.
38626	DMB0208E DMB0133E	MMDB_SQLSTATE_FETCH_FAIL 데이터의 새 행을 검색하는 동안 오류가 발생했습니다.
38627	DMB0208E	MMDB_SQLSTATE_COMMIT_FAIL 트랜잭션을 확약하는 동안 오류가 발생했습니다.
38628	DMB0208E	MMDB_SQLSTATE_GET_LENGTH_FAIL 문자열 길이를 검색하는 동안 오류가 발생했습니다.
38629	DMB0208E	MMDB_SQLSTATE_GET_SUBSTRING_FAIL 문자열 값의 부분을 검색하는 동안 오류가 발생했습니다.
38650	DMB0077E DMB0079E	MMDB_SQLSTATE_COPY_BLOB_2_FILE_FAIL BLOB을 파일에 복사하는 동안 오류가 발생했습니다.
38651	DMB0086E	MMDB_SQLSTATE_BLOB_BUFFER_TOO_SMALL 제공된 버퍼가 너무 작습니다.

표 16. SQLSTATE 코드 및 연관된 메시지 번호 (계속)

SQLSTATE	메시지 번호	설명
38652	DMB0082E	MMDB_SQLSTATE_BUILD_HANDLE 미디어 컬럼 데이터를 구성하는 동안 오류가 발생했습니다.
38653	DMB0083E	MMDB_SQLSTATE_INVALID_INSERT_VIA_SELECT 선택을 통한 삽입 요청이 올바르지 않습니다.
38654	DMB0081E	MMDB_SQLSTATE_INVALID_OFFSET_SIZE 오프세트 크기가 올바르지 않습니다.
38655	DMB0068E	MMDB_SQLSTATE_METATABLE_DOESNOT_EXIST 요구받은 메타데이터 테이블이 존재하지 않습니다.
38670	DMB0134E DMB0103E	MMDB_SQLSTATE_UNKNOWN_FORMAT 저장된 미디어가 알려지지 않은 형식을 갖고 있습니다.
38671	DMB0135E	MMDB_SQLSTATE_CREATE_THUMBNAIL_FAIL 썸네일을 작성하는 동안 오류가 발생했습니다.
38672	DMB0114E	MMDB_SQLSTATE_FORMAT_CONVERSION_FAIL 파일 형식을 변환하는 동안 오류가 발생했습니다.
38673	DMB0363E	MMDB_SQLSTATE_INVALID_UPDATE 갱신 UDF가 테이블을 참조하지 않고 호출될 때 오류가 발생했습니다.
38674	DMB0361E	MMDB_SQLSTATE_NOT_ENABLED 가져오기 UDF가 Extender에 대해 사용할 수 없는 컬럼에 적용될 때 오류가 발생했습니다.
38675	DMB0129E	MMDB_SQLSTATE_VIDEO_INTERNAL Video Extender UDF 내의 내부 오류
38676	DMB0129E	MMDB_SQLSTATE_AUDIO_INTERNAL Audio Extender UDF 내의 내부 오류
38677	DMB0129E	MMDB_SQLSTATE_IMAGE_INTERNAL
38678	DMB0089E DMB0208E	MMDB_SQLSTATE_BASE_INTERNAL_ERROR 공유 계층 내의 내부 오류
38681	DMB0108E	MMDB_SQLSTATE_IMPORT_ENV_NOT_SETUP 가져오기용 환경 변수가 제대로 설정되지 않았습니다.
38682	DMB0111E	MMDB_SQLSTATE_STORE_ENV_NOT_SETUP 저장 조작용 환경 변수가 제대로 설정되지 않았습니다.
38683	DMB0107E	MMDB_SQLSTATE_EXPORT_ENV_NOT_SETUP 내보내기 조작용 환경 변수가 제대로 설정되지 않았습니다.
38684	DMB0117E	MMDB_SQLSTATE_TEMP_ENV_NOT_SETUP 임시 파일 작성 환경 변수가 제대로 설정되지 않았습니다.

표 16. SQLSTATE 코드 및 연관된 메시지 번호 (계속)

SQLSTATE	메시지 번호	설명
38686	DMB0109E	MMDB_SQLSTATE_CANT_RESOLVE_IMPORT_FILE 가져오기 파일 이름을 해결하는 동안 오류가 발생했습니다.
38687	DMB0112E	MMDB_SQLSTATE_CANT_RESOLVE_STORE_FILE 저장 파일 이름을 해결하는 동안 오류가 발생했습니다.
38688	DMB0110E	MMDB_SQLSTATE_CANT_RESOLVE_EXPORT_FILE 내보내기 파일 이름을 해결하는 동안 오류가 발생했습니다.
38689	DMB0116E	MMDB_SQLSTATE_CANT_CREATE_TMP_FILE 임시 파일을 작성하는 동안 오류가 발생했습니다.
38690	DMB0076E	MMDB_SQLSTATE_OPEN_IMPORT_FILE_FAIL 가져오기 파일을 열 수가 없습니다.
38691	DMB0115E	MMDB_SQLSTATE_OPEN_STORE_FILE_FAIL 저장 파일을 열 수가 없습니다.
38692	DMB0114E	MMDB_SQLSTATE_OPEN_EXPORT_FILE_FAIL 내보내기 파일을 열 수가 없습니다.
38693	DMB0118E	MMDB_SQLSTATE_OPEN_TEMP_FILE_FAIL 임시 파일을 열 수가 없습니다.
38694	DMB0117E	MMDB_SQLSTATE_OPEN_CONTENT_FILE_FAIL 내용 파일을 열 수가 없습니다.
38695	DMB0135E	MMDB_SQLSTATE_OPEN_THUMBNAIL_FILE_FAIL 썸네일 파일을 열 수가 없습니다.
38696	DMB0135E	MMDB_SQLSTATE_READ_THUMBNAIL_FILE_FAIL 썸네일 파일을 읽을 수 없습니다.
38697	DMB0207E	MMDB_SQLSTATE_OVERWRITE_NOT_ALLOWED 겹쳐쓰기 조작이 수행되지 않았습니다.
38699	DMB0171E	MMDB_SQLSTATE_QUERY_NAME_NOT_FOUND 해당 이름의 조회를 찾을 수 없었습니다.
38700		MMDB_SQLSTATE_NO_MANAGEBLOB
38701		MMDB_SQLSTATE_UDFLOCATOR_FAIL
38702		MMDB_SQLSTATE_SQL_FAIL
38703		MMDB_SQLSTATE_INVALID_UPDATE
38704		MMDB_SQLSTATE_NOT_ENABLED
38705	DMB0366E DMB0382E	MMDB_SQLSTATE_QBIC_QUERY_FAIL_TO_BUILD 조회 빌드 중에 장애가 발생했습니다.

표 16. SQLSTATE 코드 및 연관된 메시지 번호 (계속)

SQLSTATE	메시지 번호	설명
38706	DMB0205E	MMDB_SQLSTATE_QBIC_TABLE_COLUMN_PAIR_NOT_VALID QBIC 카탈로그에 액세스하려고 할 때 장애가 발생했습니다. 카탈로그에서 이미지 핸들이 발견되지 않았거나, 테이블 이름 및 컬럼 이름의 조합이 카탈로그를 가지고 있지 않습니다.
38707	DMB0383E	MMDB_SQLSTATE_QBIC_QUERY_EXECUTE_FAILED 조회 실행 중에 장애가 발생했습니다.
38708		MMDB_SQLSTATE_QBIC_UKNOWN_ERROR QBIC에서 알 수 없는 장애가 발생했습니다.
38709	DMB0208E	MMDB_COPY_FILE_TO_LOCATOR_FAILURE 파일을 LOB 위치 지정자에 복사할 때 오류가 발생했습니다.
38710	DMB0534E	MMDB_SQLSTATE_QBIC_UNSUPPORTED_UDF UDF는 지원되지 않습니다.

메시지

DMB0001E DB2 Extender 서버가 연결되지 않았습니다. 이유: "<code>".

원인: 시도된 조작이 DB2 Extender 서비스의 실행을 요청합니다.

조치: 서버에서, 운영 체제용 명령행에 DMBSTART 명령을 실행하십시오.

DMB0003W DB2 Extender 추적 기능이 이 세션에 대해 실행 중입니다.

원인: 추적 기능이 시스템 자원을 사용하고 있습니다.

조치: 시스템의 성능이 영향을 받은 경우에 추적을 사용 중지하기를 원할 수도 있습니다.

DMB0004I 이 프로그램은 인스턴스 소유자 "<name>"만이 실행할 수 있습니다.

원인: DB2 Extender 서버는 반드시 인스턴스를 작성한 사용자 ID에서 시작되어야 합니다.

조치: 인스턴스를 작성한 사용자 ID에서 DMBSTART 명령을 실행하십시오.

DMB0005E 현재 데이터베이스가 "<extender-name>" Extender에 대해 사용 가능하지 않습니다.

원인: 특정 DB2 Extender에 대해 데이터베이스를 사용 가능하게 해야 하는 조작이 시도되었습니다. 예를 들어, DB2IMAGE 데이터에 대해 테이블을 사용 가능하게 하고자 하면 반드시 먼저 테이블이 DB2IMAGE 데이터에 대해 저장된 데이터베이스를 사용 가능하게 해야 합니다.

조치: 원하는 Extender 데이터 유형에 대해 데이터베이스를 사용 가능하게 하고 다시 시도하십시오.

DMB0006E "<name>" 사용자는 이 API를 호출할 권한이 없습니다.

원인: 응용프로그램 프로그래밍 인터페이스에 대한 호출이 해당 API에 필요한 레벨의 권한이 없는 사

용자 ID에 의해 시도되었습니다.

조치: 다른 사용자 ID로부터 응용프로그램을 실행하거나 데이터베이스 관리자를 통해 초기 사용자 ID의 권한 레벨을 변경하십시오.

DMB0007E ″<table-name>″ 사용자 테이블이 <extender-name>″ Extender에 대해 사용 가능하지 않습니다.

원인: 조작이 시도된 테이블이 해당 DB2 Extender에 대해 사용 가능하지 않습니다. 예를 들어, 테이블 내의 컬럼이 오디오에 대해 사용 가능하게 되기 전에 반드시 오디오 데이터를 보관하기 위해 테이블이 사용 가능하게 되어야 합니다.

조치: 먼저 테이블이 Extender에 대해 사용 가능한지 확인하십시오. 그런 뒤에 컬럼을 사용 가능하게 하십시오.

DMB0008E 저장 프로시듀어 ″<name>″을 실행하는 동안 오류가 발생했습니다.

원인: 메시지 내에서 식별된 저장 프로시듀어 내에 오류가 있거나, 환경에 문제가 있습니다.

조치: 응용프로그램을 확인하고 다시 시도하십시오.

DMB0009E 메모리 할당 오류가 발생했습니다.

원인: 시스템이 시도된 조작을 지원하는 데 필요한 메모리를 할당할 수 없었습니다.

조치: 시스템이 조작을 완성하기 위해 충분한 메모리를 가지고 있는지를 확인하십시오.

DMB0010E ″<extender-name>″ Extender는 UDT ″<name>″에 대해 이미 정의되어 있습니다.

원인: 사용자 정의 유형(UDT)의 이름이 다른 DB2 Extender에 대해 정의된 UDT에 대해 이미 사용되었습니다.

조치: UDT에 대해 다른 이름을 선택하십시오.

DMB0011E ″<column-name>″ 사용자 컬럼을 ″<extender-name>″ Extender에 대해 사용 가능하게 할 수 없습니다. 사용자 컬럼의 정의가 그 Extender와 연관된 구별 유형 ″MMDBSYS.<name>″과 호환되지 않습니다.

원인: 표시된 컬럼이 메시지에 표시된 데이터 유형에 대해 정의되지 않았으므로, 해당 Extender에 대해 사용 가능화할 수 없습니다.

조치: 사용 가능하게 하고자 하는 컬럼이 Extender에 상응하는 데이터 유형을 사용하여 정의되었는지를 확인하십시오.

MB0012E 지정된 사용자 테이블 ″<table-name>″이 없습니다.

원인: 지정된 이름의 테이블이 존재하지 않습니다.

조치: 테이블 이름과 그 존재 여부를 점검하십시오.

DMB0013E ″<column-name>″ 컬럼은 ″<table-name>″ 테이블에 대하여 정의되어 있지 않습니다.

원인: 시도된 조작이 식별된 테이블 내에 존재하지 않는 컬럼 이름을 참조했습니다.

조치: 테이블 이름과 컬럼 이름을 점검하십시오.

DMB0014W ″<table-name>″ 사용자 테이블의 ″<column-name>″ 컬럼은 ″<extender-name>″ Extender에 대하여 이미 사용 가능합니다.

원인: 그 컬럼이 이미 사용 가능화된 Extender에 대해 컬럼을 사용 가능하게 하려고 시도했습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0015W 데이터베이스가 이미 "<extender-name>" Extender에 대해 사용 가능합니다.

원인: 그 데이터베이스가 이미 사용 가능화된 Extender에 대해 데이터베이스를 사용 가능하게 하려고 시도했습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0016W "<table-name>" 사용자 테이블은 "<extender-name>" Extender에 대해 이미 사용 가능합니다.

원인: 그 테이블이 이미 사용 가능화된 Extender에 대해 테이블을 사용 가능하게 하려고 시도했습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0017E "<table-name>" 사용자 테이블은 "<extender-name>" Extender에 대해 이미 사용 가능합니다. 그러나 관련 메타데이터 테이블 "<table-name>" 또는 "<table-name>" 중 적어도 하나가 없습니다.

원인: 테이블과 관련 관리 지원(메타데이터) 테이블 중 하나 이상이 손상되었거나 파괴되었습니다. 이 메타데이터 테이블 없이, 사용자 테이블은 해당 Extender 유형의 데이터에 대해 사용될 수 없습니다.

조치: 사용자 테이블을 사용 불가능하게 하고 그 Extender에 대해 다시 사용 가능하게 하십시오.

DMB0018E 시스템이 "<table-name>" 테이블의 "<column-name>" 컬럼에 대해 고유한 트리거 이름을 작성할 수 없습니다.

원인: 시스템이 사용자 테이블 내의 컬럼을 사용 가능하게 하려고 시도할 때, DB2 Extender에서 사용하는 트리거 작성 중 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 데이터베이스 관리자와 문의하고 IBM 서비스에 문의하십시오.

DMB0019I "<Count>"개 파일이 "<extender-name>" Extender에 대한 "<table-name>" 테이블에서 참조됩니다.

원인: 메시지가 특정 Extender에 대한 사용자 테이블에서 참조하는 외부 미디어 파일의 수를 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0020I "<Count>"개 파일이 "<extender-name>" Extender에 대한 "<name>" 테이블 스키마가 있는 테이블에서 참조됩니다.

원인: 메시지가 특정 스키마 이름을 갖는 사용자 테이블에 의해 참조된 외부 미디어 파일의 수를 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0021I "<extender-name>" Extender에 대해 "<table-name>" 테이블에 참조되는 "<count>"개의 액세스 불가능한 파일이 있습니다.

원인: 메시지가 특정 Extender에 대한 사용자 테이블에 의해 참조되었으나, 액세스 불가능한 외부 미

디어 파일의 수를 표시합니다. 파일이 지워졌을 수도 있습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0022I ″<Extender-name>″ Extender에 의해 참조되는 ″<count>″개의 액세스 불가능한 파일이 있습니다.

원인: 메시지가 다음과 같은 외부 미디어 파일의 수를 표시합니다.

- 현재 데이터베이스 내의 사용자 테이블에 의해 참조된 파일.
- 특정 Extender 미디어 유형(비디오와 같은)의 파일.
- 액세스 불가능한 파일. 예를 들어, 파일이 지워졌을 수도 있습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0023I ″<Extender-name>″ Extender에 대해 테이블 스키마 ″<name>″이 있는 테이블에서 참조되는 ″<count>″개의 액세스 불가능한 파일이 있습니다.

원인: 메시지가 지정된 스키마 이름을 갖는 사용자 테이블에 의해 참조되었으나, 액세스 불가능한 외부 미디어 파일의 수를 표시합니다. 파일이 지워졌을 수도 있습니다. 메시지는 또한 테이블이 사용 가능한 Extender의 수를 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0024I 현재 데이터베이스가 ″<count>″개의 Extender에 대해 사용 가능합니다.

원인: 메시지가 현재 데이터베이스가 사용 가능하게 된 DB2 Extender의 수를 나열합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0025I ″<table-name>″ 테이블은 ″<count>″개의 Extender에 대해 사용 가능합니다.

원인: 메시지가 표시된 테이블이 사용 가능하게 된 DB2 Extender의 수를 나열합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0026I ″<table-name>″ 테이블의 ″<column-name>″ 컬럼은 ″<count>″개의 Extender에 대해 사용 가능합니다.

원인: 메시지가 표시된 컬럼이 사용 가능하게 된 DB2 Extender의 수를 나열합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0027I 현재 데이터베이스가 ″<extender-name>″ Extender에 대해 사용 가능합니다.

원인: 메시지가 현재 데이터베이스가 사용 가능하게 된 DB2 Extender를 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0028I ″<table-name>″ 테이블은 ″<extender-name>″ Extender에 대해 사용 가능합니다.

원인: 메시지가 사용자 테이블이 보관하기 위해 사용 가능하게 된 미디어 데이터 유형을 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0029I ″<table-name>″ 테이블의 ″<column-name>″ 컬럼은 ″<extender-name>″ Extender에 대해 사용 가능합니다.

원인: 메시지가 사용자 컬럼이 보관하기 위해 사용 가능하게 된 미디어 데이터 유형을 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0030E 현재 데이터베이스를 ″<extender-name>″ Extender에 대해 사용 가능하게 할 수 없습니다. RC = ″<code>.″

원인: 데이터베이스가 존재하지 않거나 사용자에게 데이터베이스를 사용 가능하게 할 권한이 없습니다.

조치: 데이터베이스가 존재하는지와 사용자에게 데이터베이스를 사용 가능하게 할 권한이 있는지 확인하십시오.

DMB0031E 테이블을 ″<extender-name>″ Extender에 대해 사용 가능하게 할 수 없습니다. RC = ″<code>.″

원인: 데이터베이스가 존재하지 않거나 테이블이 사용 가능하지 않거나 또는 사용자에게 데이터베이스를 사용 가능하게 할 권한이 없습니다.

조치: 데이터베이스가 존재하는지와 데이터베이스와 테이블이 모두 Extender에 대해 사용 가능한지를 확인하십시오. 사용자에게 테이블을 사용 가능하게 할 권한이 있는지 확인하십시오.

DMB0032E 컬럼을 ″<extender-name>″ Extender에 대해 사용 가능하게 할 수 없습니다. RC = ″<code>.″

원인: 컬럼이 이 Extender에 대한 데이터 유형을 사용하여 정의되지 않았거나 컬럼이 존재하지 않거나 또는 테이블이 사용 가능하지 않거나 또는 사용자에게 컬럼을 사용 가능하게 할 권한이 없습니다.

조치: 컬럼이 올바른 데이터 유형을 사용하여 정의되었는지를 확인하십시오. 테이블이 사용 가능한지와 사용자에게 컬럼을 사용 가능하게 할 권한이 있는지 확인하십시오.

DMB0033E 이 명령을 실행할 권한이 없습니다.

원인: 사용자 ID가 이 명령을 실행하기 위해 필요한 레벨의 권한을 갖고 있지 않습니다.

조치: 다른 사용자 ID로부터 명령을 실행하거나 데이터베이스 관리자를 통해 현 사용자 ID의 권한 레벨을 변경하십시오.

DMB0034I 데이터베이스 ″<database-name>″에 대한 DB2 Extender 서버가 정상적으로 시작되었습니다.

원인: 서버가 현재 데이터베이스에 대해 정상적으로 시작되었습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0035I 데이터베이스 "<database-name>" 에 대한 DB2 Extender 서버가 중지되었습니다.

원인: 서버가 현재 데이터베이스에 대해 정상적으로 중지되었습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0036E DB2 Extender 서버를 시작 또는 중지할 수 없습니다. DB2 Extender 서버 디먼이 실행되고 있지 않습니다. 데이터베이스 관리자에게 문의하십시오.

원인: DB2 Extender 서버를 시작하거나 중지하는 중에 오류가 발생했습니다. DB2 Extender 서버 디먼이 실행되고 있지 않는 것 같습니다.

조치: 데이터베이스 관리자에게 문의하십시오.

DMB0037E USE 세션 핸들이 올바르지 않습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0038E USE문 핸들이 올바르지 않습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0039E USE 오류: "<error>."

원인: 내부 오류가 발생했습니다.

조치: 연관된 오류 메시지에 포함된 지시에 따라 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0040E SQL 오류: "<error>"

원인: 내부 오류가 발생했습니다.

조치: 연관된 오류 메시지에 포함된 지시에 따라 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0041W 현재 데이터베이스가 새로 지정된 테이블 공간을 사용하여 "<extender-name>" Extender에 대해 다시 사용 가능하게 되었습니다.

원인: 현재 데이터베이스가 이전에 사용 가능하게 되었을 때, 다른 테이블 공간을 사용했습니다. 데이터베이스는 이제 관리 지원 테이블에 대한 새 테이블 공간과 함께 사용 가능하게 되었습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0042E "<table-name>" 테이블의 "<column-name>" 컬럼이 "<extender-name>" Extender에 대해 사용 가능하지 않습니다.

원인: 표시된 컬럼이 조작이 시도된 Extender에 대해 사용 가능하지 않습니다. 예를 들어, 현재 표시된 Extender에 대해 사용 가능하게 되지 않은 컬럼을 사용 불가능하게 하려고 시도했을 수도 있습니다.

조치: 컬럼이 메시지 내에서 표시된 Extender에 대해 사용 가능하게 되었는지 확인하십시오.

DMB0043I 현재 데이터베이스는 ″<extender-name>″ Extender에 대해 사용 불가능합니다.

원인: 사용 불가능화 조작이 성공했습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0044I ″<table-name>″ 테이블은 ″<extender-name>″ Extender에 대해 사용 불가능합니다.

원인: 사용 불가능화 조작이 성공했습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0045I ″<table-name>″ 테이블의 ″<column-name>″ 컬럼은 ″<extender-name>″ Extender에 대해 사용 불가능합니다.

원인: 사용 불가능화 조작이 성공했습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0046E 현재 데이터베이스를 ″<extender-name>″ Extender에 대해 사용 불가능화할 수 없습니다. RC = ″<code>.″

원인: 데이터베이스가 존재하지 않거나 Extender에 대해 사용 가능하지 않거나 또는 사용자에게 데이터베이스를 사용 불가능하게 할 권한이 없습니다.

조치: 테이블이 존재하는지와 Extender에 대해 사용 가능한지 확인하십시오. 사용자에게 데이터베이스를 사용 불가능하게 할 권한이 있는지 확인하십시오.

DMB0047E 테이블을 ″<extender-name>″ Extender에 대해 사용 불가능하게 할 수 없습니다. RC = ″<code>.″

원인: 테이블이 존재하지 않거나 Extender에 대해 사용 가능하지 않거나 또는 사용자에게 테이블을 사용 불가능하게 할 권한이 없습니다.

조치: 테이블이 존재하는지와 Extender에 대해 사용 가능한지 확인하십시오. 사용자에게 테이블을 사용 불가능하게 할 권한이 있는지 확인하십시오.

DMB0048E 컬럼을 ″<extender-name>″ Extender에 대해 사용 불가능하게 할 수 없습니다. RC = ″<code>″

원인: 컬럼이 메시지 내에 표시된 Extender에 대해 사용 가능하지 않으므로, 해당 Extender에 대해 사용 불가능하게 할 수 없습니다.

조치: Extender 이름이 맞는지 확인하고 사용자 컬럼이 사용 불가능하게 하고자 하는 컬럼이 맞는지를 확인하십시오.

DMB0049E 이 명령을 실행할 권한이 없습니다.

원인: 사용자 ID가 이 명령을 실행하기 위해 필요한 레벨의 권한을 갖고 있지 않습니다.

조치: 다른 사용자 ID로부터 응용프로그램을 실행하거나 데이터베이스 관리자를 통해 현 사용자 ID의 권한 레벨을 변경하십시오.

DMB0050E ″<table-name>″ 테이블에 대한 ″<authority-level>″ 권한이 없습니다.

원인: 조작을 하려면 시도한 사용자 ID보다 높은 레벨의 권한이 필요합니다.

조치: 올바른 권한을 가진 사용자 ID로부터 조작을 실행하거나 데이터베이스 관리자를 통해 현 사용자 ID의 권한 레벨을 변경하십시오.

DMB0051E 잘못된 미디어 파일 헤더입니다.

원인: 시스템이 미디어 파일의 헤더를 판독하거나 열 수 없습니다. 파일이 손상되었거나 미디어 파일이 아닙니다.

조치: 파일이 미디어 파일이고 손상되지 않았는지 확인하십시오.

DMB0052I ″<database-name>″ 데이터베이스에 대한 DB2 Extender 서버가 정상적으로 시작되었습니다.

원인: 서버가 정상적으로 시작되었습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0053I ″<database-name>″ 데이터베이스에 대한 DB2 Extender 서버가 정상적으로 중지되었습니다.

원인: 서버가 정상적으로 중지되었습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0054E DB2 Extender 서버가 데이터베이스에 연결할 수 없거나 DB2 명령문 핸들을 할당할 수 없습니다. 데이터베이스 ″<database-name>″에 대한 DB2 Extender 서버가 실행되고 있지 않습니다.

원인: DB2 Extender 서버가 데이터베이스에 연결할 수 없거나 DB2문 핸들을 할당할 수 없습니다. 데이터베이스에 대한 DB2 Extender 서버가 실행되고 있지 않는 것 같습니다.

조치: 데이터베이스에 대한 DB2 Extender 서버가 실행 중인지 확인하십시오. 만약 실행 중이 아니면 데이터베이스에 대한 특정 Extender를 시작하거나 시스템 관리자에게 Extender 서비스를 재시작하도록 요청하십시오.

DMB0055I ″command-name″ 명령이 정상적으로 완료되었습니다.

원인: 명령이 정상적으로 완료되었습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0056E 예기치 않은 토큰 ″<token>″이 ″<keyword>″ 다음에 발견되었습니다. 예상되는 토큰은 다음을 포함할 수 있습니다. ″<extendername>

원인: 명령은 메시지 내에서 표시된 토큰 대신 DB2 Extender의 이름을 예상했습니다.

조치: 명령의 구문을 따라서 다시 시도하십시오.

DMB0057E 테이블 공간 "<table-space-name>"이 올바르지 않습니다.

원인: 메시지 내의 테이블 공간이 존재하지 않습니다.

조치: 테이블 공간 이름과 존재 여부를 확인하십시오.

DMB0058I "<count>"개의 파일이 "<Extender-name>" Extender에 의해 참조됩니다.

원인: 메시지가 특정 Extender에 대한 사용자 테이블에서 참조하는 외부 미디어 파일의 수를 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0059E "<Name>"은 DB2 Extender의 유효한 이름이 아닙니다. 유효한 Extender 이름에는 "<extender-name>" DB2VIDEO, DB2AUDIO 및 DB2IMAGE가 포함됩니다.

원인: Extender 이름의 철자가 잘못되었습니다.

조치: Extender 이름을 확인하십시오.

DMB0060E "<Function>"의 올바른 구문: "<syntax>"

원인: 입력한 명령의 구문이 잘못되었습니다.

조치: 메시지에 설명된 대로 구문을 따르십시오.

DMB0061E "<Keyword>" 뒤에 오는 테이블 이름 "<name>"이 올바르지 않습니다.

원인: 명령은 테이블의 이름을 예상했습니다.

조치: 명령의 구문을 따라서 다시 시도하십시오.

DMB0062E "<Keyword>" 뒤에 오는 컬럼 이름 "<name>"이 올바르지 않습니다.

원인: 명령은 컬럼의 이름을 예상했습니다.

조치: 명령의 구문을 따라서 다시 시도하십시오.

DMB0064E 시스템이 "<Keyword>" 뒤에 오는 토큰 "<token>"을 인식하지 못합니다.

원인: 명령은 메시지 내에서 표시된 토큰 외의 다른 것을 예상했습니다.

조치: 명령의 구문을 따라서 다시 시도하십시오.

DMB0065E "<Keyword>" 뒤에 오는 사용자 ID "<indentifier>"는 올바르지 않습니다.

원인: 명령은 올바른 사용자 ID를 예상했습니다.

조치: 원하는 사용자 ID를 확인하고 다시 시도하십시오.

DMB0066E ″<Keyword>″ 뒤에 오는 암호 ″<password>″는 올바르지 않습니다.

원인: 명령은 메시지 내에 표시된 토큰 대신 유효한 사용자 ID를 예상했습니다.

조치: 암호를 확인하고 다시 시도하십시오.

DMB0067E 입력한 명령이 올바르지 않습니다.

원인: 명령 이름의 철자가 잘못되었거나, 구문이 틀립니다.

조치: 명령의 구문을 따라서 다시 시도하십시오.

DMB0068E 메타데이터 테이블이 존재하지 않습니다.

원인: 함수가 데이터 오브젝트에 대해 존재해야 하는 관리 지원(메타데이터) 테이블의 사용을 시도했습니다. 메타데이터 테이블이 손상되었거나 지워졌을 수 있습니다.

조치: 메타데이터 테이블의 이름과 존재 여부를 점검하십시오. 메타데이터 테이블이 실수로 지워졌거나 손상된 경우, 데이터 오브젝트를 사용 불가능화한 다음 다시 사용 가능하게 하십시오.

DMB0069E DB 이름 ″<name>″이 올바르지 않습니다.

원인: 이 이름을 가진 데이터베이스는 존재하지 않습니다.

조치: 데이터베이스의 이름과 존재 여부를 점검하십시오.

DMB0070E 핸들이 올바르지 않습니다.

원인: 응용프로그램에 전달된 핸들 값이 손상되었을 수 있습니다.

조치: Extender 핸들 값이 변경되지 않았는지 응용프로그램을 확인하십시오.

DMB0071E ″<Database-name>″에 연결할 수 없습니다.

원인: 데이터베이스에 대한 DB2 Extender 서버가 시작되지 않은 것 같습니다.

조치: 서버의 상태를 점검하십시오. 서버가 실행 중이 아닌 경우에는 DMB 명령행에서 START SERVER 명령을 이용하여 서버를 재시작하십시오.

DMB0072E UDF SQL 서버가 DB로부터 연결을 차단할 수 없습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0073E USE 세션 핸들이 올바르지 않습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0074E USE문 핸들이 올바르지 않습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0075E 파일 이름을 지정하십시오.

원인: 조작은 미디어 파일 이름을 요구합니다.

조치: 미디어 파일 이름을 입력하십시오.

DMB0076E 가져오기 파일을 열 수 없습니다.

원인: 가져오기 파일이 누락되었거나 손상되었습니다.

조치: 가져오기 파일의 이름과 존재 여부를 확인하십시오.

DMB0077E 내용 파일 열기/읽기를 할 수 없습니다.

원인: Extender 핸들이 존재하지 않거나 손상된 파일을 지시합니다. 파일이 Extender에 액세스 불가능하게 되었습니다.

조치: 파일 이름을 찾으려면 FILENAME UDF를 사용하고 또는 내용 파일의 존재 여부를 확인하십시오.

DMB0078E 내보내기 파일을 작성할 수 없습니다.

원인: 내보내기 파일이 누락되었거나 손상되었습니다.

조치: 내보내기 파일의 이름과 존재 여부를 확인하십시오.

DMB0079E BLOB을 파일로 복사할 수 없습니다.

원인: 파일이 BLOB을 받아들일 수 없습니다. BLOB을 저장할 충분한 저장 공간이 없는 것 같습니다.

조치: 사용 가능한 저장영역에 BLOB의 크기를 비교하고, 필요하면 저장영역을 증가시키십시오.

DMB0080E 파일에 쓸 수 없습니다.

원인: 파일이 손상 또는 존재하지 않거나, 또는 이름의 철자가 잘못되었습니다.

조치: 파일의 이름과 존재 여부를 확인하십시오.

DMB0081E 오프세트 또는 크기가 올바르지 않습니다.

원인: 조작이 데이터 구조 내에서 예상된 데이터를 찾지 못했습니다. 필드 크기나 오프세트가 올바르지 않습니다.

조치: 오프세트와 필드 크기를 확인하십시오.

DMB0082E 핸들을 빌드할 수 없습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0083E "<extender-name>"과 "<extender-name>"이 호환되지 않습니다.

원인: 메시지 내에 지정된 두 Extender는 이 사용에서 호환되지 않습니다. 완전 또는 부속선택에 의

한 삽입 조작이 올바르지 않습니다.

조치: 소스 오브젝트가 사용 가능한 동일 Extender에 대해 목표 오브젝트가 사용 가능한지 확인하십시오.

DMB0084E 가져오기 요청이 다음에서 올바르지 않습니다. 파일이름, 내용, 저장 유형.

원인: 가져오기 조작이 실패했습니다. 파일 이름, 내용 또는 저장 유형이 올바르지 않습니다.

조치: 데이터를 확인하고 다시 시도하십시오.

DMB0085E 갱신 요청이 다음에서 올바르지 않습니다. 파일이름, 내용, 저장 유형.

원인: 갱신 조작이 실패했습니다. 파일 이름, 내용 또는 저장 유형이 올바르지 않습니다.

조치: 데이터를 확인하고 다시 시도하십시오.

DMB0086E 요청된 크기가 너무 큽니다.

원인: 요청한 크기가 UDF에 대한 최대 blob 크기보다 큽니다.

조치: 더 작은 크기를 요청하십시오.

DMB0087E 파일 이름이 올바르지 않습니다.

원인: 그런 이름의 파일이 없습니다.

조치: 파일의 이름과 존재 여부를 확인하십시오.

DMB0088E 핸들 값이 널(NULL)입니다.

원인: UDF는 널(NULL)값 이외의 핸들을 예상했습니다.

조치: 응용프로그램이 유효한 핸들을 확보하는지와 그것을 UDF에 전달하는지 확인하십시오.

DMB0089E 핸들 값이 존재하지 않습니다.

원인: UDF에 전달한 핸들이 올바르지 않습니다.

조치: 응용프로그램이 유효한 핸들을 전달하는지 확인하십시오.

DMB0090E 데이터가 절단되었습니다.

원인: 파일 또는 버퍼가 데이터를 받아들이기에는 너무 작습니다.

조치: 파일 또는 버퍼의 크기를 증가시키십시오.

DMB0091W 파일에 내용이 이미 있습니다.

원인: 파일이 이미 내용을 갖고 있습니다. 내용은 겹쳐쓰기될 것입니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0092E "<column-name>" 컬럼에 대해 시도된 삽입 조작이 올바르지 않습니다. 컬럼은 "<extender-name>" Extender에 대해 사용 가능합니다.

원인: 삽입되는 데이터의 데이터 유형이 컬럼이 사용 가능한 Extender와 다릅니다.

조치: 소스 오브젝트가 사용 가능하게 된 동일한 Extender에 대해 목표 오브젝트가 사용 가능하게 되었는지를 확인하십시오.

DMB0093E ″<column-name>″ 컬럼에 대해 시도된 갱신 조작이 올바르지 않습니다. 컬럼은 ″<extender-name>″ Extender에 대해 사용 가능합니다.

원인: 갱신되는 데이터의 데이터 유형이 컬럼이 사용 가능한 Extender와 다릅니다.

조치: 소스 오브젝트가 사용 가능한 동일 Extender에 대해 목표 오브젝트가 사용 가능한지 확인하십시오.

DMB0094I ″<table-name>″ 테이블이 없습니다.

원인: 시스템이 그런 이름의 테이블을 찾을 수 없습니다. 다른 데이터베이스에 존재할 수도 있습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0095W ″<table-name>″ 테이블은 ″<extender-name>″ Extender에 대해 사용 가능하지 않습니다.

원인: 테이블이 Extender에 대해 사용 가능하지 않습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0096W ″<table-name>″ 테이블의 ″<column-name>″ 컬럼이 ″<extender-name>″ Extender에 대해 사용 가능하지 않습니다.

원인: 시스템은 컬럼이 사용 가능하게 되기를 예상했습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0097W 현재 데이터베이스는 ″<extender-name>″ Extender에 대해 사용 가능하지 않습니다.

원인: 시스템은 데이터베이스가 사용 가능하게 되기를 예상했습니다.

조치: 메시지 내에서 표시된 Extender에 대해 데이터베이스를 사용 가능하게 하십시오.

DMB0098E 사용자에게 ″<table-name>″ 테이블에 대한 ″<authority-level>″ 권한이 없습니다.

원인: 조작을 하려면 시도한 사용자 ID보다 높은 레벨의 권한이 필요합니다.

조치: 테이블을 소유한 사용자 ID로부터 조작을 실행하거나 데이터베이스 관리자에게 현 사용자 ID의 권한 레벨을 변경해 줄 것을 요청하십시오.

DMB0099E 트랜잭션을 확약할 수 없습니다.

원인: 현재 데이터베이스에 대한 Extender 서버가 중지되었을 수도 있습니다.

조치: 서버의 상태를 점검하십시오. 서버가 실행 중이 아닌 경우에는 db2ext 명령행에서 START SERVER 명령을 사용하여 서버를 재시작하십시오.

DMB0100E ″<name>″은 올바른 테이블 이름이 아닙니다.

원인: 그런 이름의 테이블은 존재하지 않습니다.

조치: 테이블의 이름과 존재 여부를 확인하고 다시 시도하십시오.

DMB0101E 올바르지 않은 널(NULL) 매개변수입니다.

원인: 명령은 널(NULL)값 이외의 매개변수를 예상했습니다.

조치: 구문을 확인하고 다시 시도하십시오.

DMB0102E 저장 유형이 올바르지 않습니다.

원인: DB2 Extender의 경우, 저장 유형은 미디어 데이터가 저장되는 위치를 표시합니다.

조치: 외부(파일 내)를 표시하려면 0, 외부(데이터베이스 내)를 표시하려면 1을 지정하십시오.

DMB0103E 지원되지 않는 형식입니다.

원인: DB2 Extender는 이 오브젝트의 형식을 지원하지 않습니다.

조치: 오브젝트를 지원되는 형식으로 변환하십시오.

DMB0104E 비디오 내용 버퍼가 너무 작습니다.

원인: 비디오 클립이 할당된 버퍼에 비해 너무 큽니다.

조치: 더 큰 버퍼를 할당하십시오.

DMB0105E MPEG1 헤더가 올바르지 않습니다.

원인: MPEG1 파일의 헤더가 누락되었거나 손상되었습니다.

조치: 파일이 MPEG1 파일인지 확인하십시오.

DMB0106E AVI 헤더가 올바르지 않습니다.

원인: AVI 파일의 헤더가 누락되었거나 손상되었습니다.

조치: 파일이 AVI 파일인지 확인하십시오.

DMB0107E 내보내기 환경이 설정되지 않았습니다.

원인: DB2 Extender에서 내보내기 환경용 환경 변수가 올바르게 설정되지 않았습니다.

조치: 환경 변수가 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』에 설명된 대로, 제대로 설정되었는지 확인하십시오.

DMB0108E 가져오기 환경이 설정되지 않았습니다.

원인: DB2 Extender에서 가져오기 환경용 환경 변수가 올바르게 설정되지 않았습니다.

조치: 환경 변수가 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』에 설명된 대로, 제대로 설정되었는지 확인하십시오.

DMB0109E 가져오기 파일을 해석할 수 없습니다.

원인: 그런 이름의 가져오기 파일이 없습니다.

조치: 파일의 이름과 존재 여부를 확인하고 환경 변수가 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』에 설명된 대로 제대로 설정되었는지 확인하십시오.

DMB0110E 내보내기 파일을 해석할 수 없습니다.

원인: 그런 이름의 내보내기 파일이 없습니다.

조치: 파일의 이름과 존재 여부를 확인하고 환경 변수가 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』에 설명된 대로 제대로 설정되었는지 확인하십시오.

DMB0111E 저장 환경이 설정되지 않았습니다.

원인: 저장 환경용 환경 변수가 올바르게 설정되지 않았습니다.

조치: 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』에 설명된 대로 환경 변수가 제대로 설정되었는지 확인하십시오.

DMB0112E 저장 파일을 해석할 수 없습니다.

원인: 그런 이름의 저장 파일이 없습니다.

조치: 파일의 이름과 존재 여부를 확인하고 환경 변수가 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』에 설명된 대로 제대로 설정되었는지 확인하십시오.

DMB0113E 가져오기 파일을 열 수 없습니다.

원인: 누군가에 의해 파일이 잠겼거나 파일이 누락 또는 손상된 것 같습니다.

조치: 파일의 이름, 존재 여부 및 상태를 확인하고 권한 레벨을 확인하십시오.

DMB0114E 내보내기 파일을 열 수 없습니다.

원인: 누군가에 의해 파일이 잠겼거나 파일이 누락 또는 손상된 것 같습니다.

조치: 파일의 이름, 존재 여부 및 상태를 확인하고 권한 레벨을 확인하십시오.

DMB0115E 저장 파일을 열 수 없습니다.

원인: 시스템이 파일을 작성하려고 시도하고 있으나, 해당 파일이 이미 존재합니다. 서버가 파일을 겹쳐쓰기할 권한을 갖고 있지 않습니다.

조치: 파일의 이름, 존재 여부 및 상태를 확인하고 권한 레벨을 확인하십시오.

DMB0116E 임시 파일을 작성할 수 없습니다.

원인: 임시 파일을 작성할 충분한 저장 공간이 없는 것 같습니다.

조치: Extender용 임시 환경 변수가 제대로 설정되었는지 확인하십시오. 필요하면 저장영역을 증가시키십시오.

DMB0117E 임시 환경이 설정되지 않았습니다.

원인: 임시 환경용 환경 변수가 올바르게 설정되지 않았습니다.

조치: 환경 변수가 633 페이지의 『부록A. DB2 Extender의 환경 변수 설정』에 설명된 대로, 제대로 설정되었는지 확인하십시오.

DMB0118E 임시 파일을 열 수 없습니다.

원인: 임시 파일이 겹쳐쓰기 되었거나 손상되었을 수 있습니다.

조치: 환경 변수가 633 페이지의『부록A. DB2 Extender의 환경 변수 설정』에 설명된 대로, 제대로 설정되었는지 확인하십시오.

DMB0119I dmbsrv 서버가 "<count>"가지 연결로 "<name>"에 대해 시작하고 있습니다.

원인: 메시지는 서버가 시작될 때 몇 개의 연결이 수행되는지를 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0120E dmbsrv 서버가 "<count>"가지 연결로 "<name>"에 대해 시작하는 데 실패했습니다.

원인: DB2가 아직 시작되지 않았거나 데이터베이스가 존재하지 않거나 시스템이 허용되는 연결을 모두 사용한 것 같습니다.

조치: DB2가 시작되었는지와 데이터베이스가 존재하는지를 확인하십시오. 문제가 계속되면 사용권을 더 확보하기 위해 IBM에 문의하십시오.

DMB0121I dmbsrv 서버가 "<count>"가지 연결로 "<name>"에 대해 시작되었습니다.

원인: 메시지는 서버가 시작될 때 몇 개의 연결이 수행되는지를 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0122I dmbssd 서버가 대기 상태입니다.

원인: 서버가 응용프로그램 실행을 위한 대기 상태입니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0129E 올바르지 않은 조작: "<operation-name>".

원인: 그런 이름의 명령이나 API가 존재하지 않습니다.

조치: 명령이나 API를 확인하고 다시 시도하십시오.

DMB0130E "<column-name>" 컬럼을 SQL문에 바인드하는 데 실패했습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0131E SQL prepare문에 실패했습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0132E SQL set 매개변수에 실패했습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0133E SQL execute문에 실패했습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0134E 파일 형식 변환에 실패했습니다.

원인: 저장된 멀티미디어 데이터의 형식이 형식 변환에 대해 지원되지 않습니다.

조치: 이 파일의 형식을 변환할 수 없습니다.

DMB0135E 썸네일 열기/읽기를 할 수 없습니다.

원인: 썸네일 파일이 누락되었거나 손상되었습니다.

조치: 썸네일 파일의 이름, 존재 여부 및 무결성을 확인하십시오.

DMB0136E 바인드 파일을 찾을 수 없습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0137E DB "<database-name>"에 연결할 수 없습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0138E SQL문을 할당해제할 수 없습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0139E "<keyword>" 뒤에 오는 특성 이름 "<name>"이 올바르지 않습니다.

원인: Image Extender는 이 명령에서 올바른 특성 이름을 예상했습니다.

조치: 올바른 특성 이름으로 명령을 다시 시도하십시오. 올바른 특성 이름은 다음을 포함합니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

DMB0141E "<keyword>" 다음의 규정자 "<identifier>"가 올바르지 않습니다.

원인: 시스템이 명령 내의 규정자를 식별할 수 없습니다.

조치: 규정자를 점검하고 다시 시도하십시오.

DMB0142E 아무런 카탈로그도 열려 있지 않습니다.

원인: DB2 Extender에서 현재 명령은 열릴 QBIC 카탈로그가 필요합니다.

조치: OPEN QBIC CATALOG 명령으로 QBIC 카탈로그를 여십시오.

DMB0143I "<table-name>" 테이블의 "<column-name>" 컬럼에 대한 QBIC 카탈로그에서, 자동 카탈로그화가 "<status>"으로 설정되었습니다. "<count>"개 특성이 있습니다.

원인: 메시지는 QBIC 카탈로그에 특정 이미지 컬럼용으로 정의된 특성의 수와 자동 카탈로그화가 작동 중인지를 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0145E 조회 핸들이 올바르지 않습니다.

원인: API 호출에서 사용한 조회 핸들이 올바르지 않습니다.

조치: 올바른 조회 핸들을 확보했는지 응용프로그램을 점검하십시오.

DMB0146E 특성 클래스 이름 "<feature-class>"가 올바르지 않습니다.

원인: 그런 이름의 특성 클래스가 없습니다. 유효한 특성 이름은 다음을 포함합니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

조치: 특성 이름을 정정하고 다시 시도하십시오.

DMB0147E 특성 클래스 이름 "<feature-class>"가 누락되었거나 올바르지 않습니다.

원인: 올바른 특성 이름은 다음을 포함합니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

조치: 특성 이름을 정정하고 다시 시도하십시오.

DMB0148E "<feature-name>" 특성은 이미 조회의 구성원입니다.

원인: 조회는 이미 메시지 내에서 표시된 특성을 지원합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0149E "<feature-name>" 특성은 조회의 구성원이 아닙니다.

원인: 조회가 지정된 특성 이름을 포함하지 않습니다.

조치: 특성에 액세스하는 다른 API를 호출하기 전에 조회에 특성을 추가하려면, QbQueryAddFeature API를 사용하십시오.

DMB0150E 시스템이 메모리를 할당할 수 없습니다.

원인: 시스템이 시도된 조작을 지원하기 위해 필요한 메모리를 할당할 수 없습니다.

조치: 시스템이 조작을 완성하기 위해 충분한 메모리를 가지고 있는지를 확인하십시오.

DMB0151E 리턴 값에 대한 포인터가 널(NULL)입니다.

원인: 리턴 값에 대한 포인터가 널(NULL)이 아니어야 하므로 API 호출이 정상적으로 완료되지 않았습니다.

조치: 올바른 매개변수가 API 호출에 제공되었는지와 구문을 제대로 따랐는지 확인하십시오.

DMB0152E 목록 리턴 값에 대한 포인터가 널(NULL)입니다.

원인: 리턴 값에 대한 포인터가 널(NULL)이 아니어야 하므로 API 호출이 정상적으로 완료되지 않았습니다.

조치: 올바른 매개변수가 API 호출에 제공되었는

지와 구문을 제대로 따랐는지 확인하십시오.

DMB0153E 범위 매개변수는 예약되어 있으며 0이어야 합니다.

원인: 매개변수가 차후 사용을 위해 예약되어 있습니다.

조치: 범위를 0으로 설정하십시오.

DMB0154E 특성 클래스 이름에 대한 포인터가 올바르지 않습니다.

원인: API 호출은 입력 특성 클래스 이름에 대해 올바른 포인터를 예상했습니다.

조치: 올바른 매개변수가 API 호출에 제공되었는지와 구문을 제대로 따랐는지 확인하십시오.

DMB0155I 버퍼 크기 0이 "<feature-name>" 함수로 전달되었습니다.

원인: API 호출은 정보를 리턴하기 위해 버퍼가 필요합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0156E QbImageSource 포인터가 널(NULL)입니다.

원인: 널(NULL) 값은 구조가 변경되어서는 안된다는 것을 표시합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0157E QbImageSource 유형 "<type>"이 올바르지 않습니다.

원인: 이 DB2 Extender API에 의해 참조된 데이터 구조의 데이터 유형이 잘못되었습니다.

조치: 구조의 데이터 유형은 QbImageSource여야 합니다.

DMB0159E QbImageSource 이미지 버퍼에 대한 포인터가 널(NULL)입니다.

원인: API 호출은 포인터의 리턴을 예상했습니다.

조치: API 호출 및 버퍼가 제대로 지정되었는지 응용프로그램을 점검하십시오.

DMB0160I 이미지 버퍼 또는 파일의 길이가 0입니다.

원인: 길이가 0입니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0161E 테이블 또는/및 컬럼 이름에 대한 포인터가 널(NULL)입니다.

원인: API 호출은 포인터의 제공을 예상했습니다.

조치: API 호출에 대한 입력이 제대로 지정되었는지 응용프로그램을 점검하십시오.

DMB0162I requestedHits를 0으로 설정합니다.

원인: requestedHits을 0으로 설정하면 아무 것도 돌아오지 않습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0163I 이 함수는 아직 지원되지 않습니다.

원인: 이 함수는 아직 지원되지 않습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0164E 시스템이 조회(<query-name>)를 처리할 수 없습니다.

원인: 조회가 작성될 때 오류가 발생했습니다.

조치: 명령이나 API에 대한 입력을 확인하고 다시 시도하십시오.

DMB0165E 시스템이 조회(<query-name>)를 실행할 수 없습니다.

원인: 조회가 작성될 때 오류가 발생했습니다.

조치: 명령이나 API에 대한 입력을 확인하고 다시 시도하십시오.

DMB0166E "<name>"을 실행하는 동안 명령문 오류 "<error>"가 "<name>"에서 발견되었습니다.

원인: 내부 IBM 오류가 발생했습니다.

조치: 데이터베이스 관리자에게 문의하십시오.

DMB0167E QbGenericImageDataClass를 읽는 동안 오류가 발생했습니다.(<error>).

원인: 내부 IBM 오류가 발생했습니다.

조치: 데이터베이스 관리자에게 문의하십시오.

DMB0168E 조회의 "<feature-name>" 특성이 탐색 전에 설정되지 않았습니다.

원인: 조회에 지정된 특성이 없으므로 조회가 실행되지 않았습니다.

조치: QbAddFeature API 또는 ADD QBIC FEATURE 명령을 사용하여 조회에 특성을 추가하십시오.

DMB0169E 호출-레벨 인터페이스에서 다음 오류가 발생했습니다. "<error>".

원인: CLI 오류.

조치: 메시지 텍스트 내의 지시를 따르십시오.

DMB0170E 조회 이름 "<query-name>"이 이미 사용 중입니다.

원인: 그런 이름을 가진 다른 조회가 존재합니다.

조치: 다른 이름을 선택하십시오.

DMB0171E 조회 이름 "<query-name>"이 저장되지 않았습니다.

원인: 조회를 작성한 뒤에 시스템이 그 조회를 저장할 수 없었습니다.

조치: 쓰기 권한과 조회를 저장할 충분한 저장영역이 있는지 확인하십시오.

DMB0172E SQL 오류: "<error>".

원인: 내부 오류가 발생했습니다.

조치: 연관된 오류 메시지에 포함된 지시에 따라 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0173E "<catalog-name>" 카탈로그가 열리나 읽기 전용입니다.

원인: 다른 누군가가 이미 카탈로그를 쓰기 모드에서 열었거나 그에 대한 쓰기 권한이 없으므로 카탈로그를 갱신할 수 없습니다.

조치: 다른 사용자가 끝낼 때까지 기다리거나, 다른 사용자 ID로 응용프로그램을 실행하거나 데이터베이스 관리자를 통해 현 사용자 ID의 권한 레벨을 변경하십시오.

DMB0174E 시스템 오류 "<error>"가 발생했습니다.

원인: 내부 IBM 오류가 발생했습니다.

조치: 연관된 오류 메시지에 포함된 지시에 따라 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0175I "<information>" 이미지를 찾을 수 없었습니다.

원인: 조회에 일치하는 이미지가 발견되지 않았습니다. 데이터베이스가 비어 있을 수도 있습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0176I 컬럼에 이미 QBIC 카탈로그가 있습니다. "<table-name column-name>".

원인: 그런 이름을 가진 다른 카탈로그가 존재합니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0177E 시스템이 카탈로그를 열 수 없습니다. 오류 메시지: "<error>".

원인: 카탈로그가 손상되었습니다.

조치: 메시지 텍스트의 지시를 따르십시오.

DMB0178E 시스템이 카탈로그를 삭제할 수 없습니다. 오류 메시지: "<error>".

원인: 카탈로그가 존재하지 않거나, 손상되었습니다.

조치: 카탈로그의 이름과 존재 여부를 확인하고 다시 시도하십시오.

DMB0179E 카탈로그 핸들이 올바르지 않습니다. "<error>".

원인: API 호출에서 사용한 카탈로그 핸들이 올바르지 않습니다.

조치: 올바른 카탈로그 핸들인지 응용프로그램을 점검하십시오.

DMB0180I 카탈로그에 대한 액세스가 거부됩니다. "<error>".

원인: 액세스가 거부됩니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0181I 카탈로그가 사용 중입니다. "<error>".

원인: 다른 조작이 이 카탈로그를 사용하고 있습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0184I 추적이 이미 시작되었습니다.

원인: 추적이 이미 시작되었습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0185I 추적이 아직 시작되지 않았습니다.

원인: 추적이 아직 시작되지 않았습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0186I 추적이 <time>"에 "<directory-name>" 디렉토리로부터 시작되었습니다. 추적 파일은 "<file-name>"입니다. "<count> 바이트의 추적 데이터가 기록되었습니다.

원인: 추적이 작동 중입니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0187E 시스템이 쓰기에 필요한 "<file-name>" 파일을 열 수 없기 때문에 통신을 설정할 수 없습니다.

원인: 사용자가 환경 변수 DB2INSTANCE에 의해 설명된 현재 인스턴스의 소유자가 아니거나, DB2MMTOP과 같은 환경 변수가 제대로 설정되지 않았습니다.

조치: 인스턴스를 소유한 사용자 ID로 로그인하십시오. 환경 변수가 제대로 설정되었는지 확인하십시오.

DMB0188I 추적 디먼 작성시 오류가 발생했습니다. "<error>"

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0189I 추적이 이미 정상적으로 시작되었습니다.

원인: 추적이 이미 시작되었습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0190E 추적을 시작할 수 없습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0191E 환경 변수 "<name>"을 설정해야 합니다.

원인: 시스템 구성이 바르지 않습니다.

조치: 변수를 설정하고 다시 시도하십시오.

DMB0192I 추적이 정상적으로 중단되었습니다.

원인: 추적이 중단되었습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0193E 시스템이 "<file-name>" 파일에 쓰기를 할 수 없습니다.

원인: 지정된 파일의 디렉토리에 대한 쓰기 권한이 없습니다.

조치: 권한을 확보하려면 데이터베이스 관리자에게 문의하십시오.

DMB0194E 시스템이 "<file-name>" 파일 읽기를 할 수 없습니다.

원인: 파일이 존재하지 않거나 파일에 대한 읽기 권한이 없습니다.

조치: 파일이 존재하는지와 파일에 대한 읽기 권한이 있는지 확인하십시오.

DMB0198E 입력 파일의 추적 코드 "<code>"를 알 수 없습니다. 입력 파일이 손상되었을 수도 있습니다.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0199E 참조되는 테이블 중 어느 것에 대해서도 "<authority-level>" 권한이 없습니다.

원인: 사용자 ID가 이 조작을 위해 필요한 레벨의 권한을 갖고 있지 않습니다.

조치: 다른 사용자 ID로부터 조작을 수행하거나 데이터베이스 관리자를 통해 현 사용자 ID의 권한 레벨을 변경하십시오.

DMB0200W 참조되는 테이블 중 적어도 하나에 대해 "<authority-level>" 권한이 없습니다.

원인: 사용자 ID가 일부 테이블에 필요한 권한 레벨을 갖고 있지 않습니다.

참조되는 파일을 나열 중인 경우, 나열되는 파일이 사용자가 선택 권한을 갖는 테이블에 의해 참조되었습니다. 선택 권한을 가지고 있지 않은 시스템에 파일이 있는 경우에, 참조되는 파일이 나열되지 않습니다.

메타데이터를 재구성하고 있는 경우에, 시스템은 사용자가 제어 권한을 가지고 있는 테이블에 대한 메타데이터만을 재구성합니다.

조치: 모든 파일을 확보하려면 다른 사용자 ID로 조작을 수행하거나 데이터베이스 관리자를 통해 현 사용자 ID의 권한 레벨을 변경하십시오.

DMB0201I "<feature-name>" 특성이 이미 존재합니다.

원인: 그런 이름을 가진 특성이 이미 QBIC 카탈로그 내에 포함되어 있습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0202E 특성 이름 "<feature-namef>"이 올바르지 않습니다.

원인: 그런 이름의 특성 클래스가 없습니다. 유효한 특성 이름은 다음을 포함합니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

조치: 특성 이름을 정정하고 다시 시도하십시오.

DMB0203E "<feature-name>" 특성이 없습니다.

원인: 그런 이름의 특성 클래스가 없거나 특성 클래스가 QBIC 카탈로그에 포함되어 있지 않습니다. 유효한 특성 이름은 다음을 포함합니다.

- QbColorFeatureClass
- QbColorHistogramFeatureClass
- QbDrawFeatureClass
- QbTextureFeatureClass

조치: 특성 이름을 정정하고 다시 시도하십시오.

DMB0204E ″<column-name>″ 컬럼이 DB2IMAGE에 대해 사용 가능하지 않습니다.

원인: 컬럼이 Image Extender에 대해 사용 가능하지 않습니다.

조치: 컬럼이 DB2 Image Extender에 대해 사용 가능한지 확인하십시오.

DMB0205E <table-name column-name>″에 대한 카탈로그가 없습니다.

원인: 지정된 컬럼과 연관된 QBIC 카탈로그가 없습니다.

조치: 기타 다른 QBIC 카탈로그 조작을 수행하기 전에 컬럼에 대한 QBIC 카탈로그를 작성하십시오.

DMB0206W 지정된 컬럼이 어떤 Extender에서도 사용 가능하지 않습니다.

원인: 컬럼이 존재하지 않거나 데이터 유형이 Extender와 호환되지 않을 수 있습니다.

조치: 컬럼이 올바른 데이터 유형을 사용하여 정의되었는지를 확인하십시오.

DMB0207E 파일을 겹쳐 쓸 수 없습니다.

원인: 파일은 이미 존재하나, EXPORT UDF가 겹쳐 쓸 수 없습니다.

조치: 파일을 다른 파일 이름으로 내보내기하거나 EXPORT UDF가 파일을 겹쳐 쓸 수 있도록 하십시오.

DMB0208E sqlcode=<code> clistate=<code>.

원인: 내부 오류가 발생했습니다.

조치: 조작을 반복하십시오. 오류가 다시 발생하는 경우 IBM 서비스에 문의하십시오.

DMB0209E 오디오 헤더가 올바르지 않습니다.

원인: 오디오 파일의 헤더가 누락되었거나 손상되었습니다.

조치: 오디오 파일의 형식이 DB2 Extender에 의해 지원되는지 확인하십시오.

DMB0211W 파일이 겹쳐 쓰기 없이 존재합니다.

원인: 지정된 목표 파일이 이미 존재하며 겹쳐 쓰여지지 않습니다.

조치: 아무런 조치도 필요하지 않습니다.

DMB0212E resultType 매개변수는 예약되어 있으며 0이어야 합니다.

원인: 매개변수가 차후 사용을 위해 예약되어 있습니다.

조치: resultType을 0으로 설정하십시오.

DMB0214E 조회 이름에 대한 포인터가 올바르지 않습니다.

원인: API 호출은 입력 조회 이름에 대해 올바른 포인터를 예상했습니다.

조치: 올바른 매개변수가 API 호출에 제공되었는지와 구문을 제대로 따랐는지 확인하십시오.

DMB0352E 명령행 환경이 초기화되지 않았습니다.

원인: 명령행 환경이 db2ext 명령행 처리기를 실행하도록 초기화되지 않았습니다.(이 메시지는 Windows NT 및 Windows 95 환경에만 적용됩니다.)

조치: DB2CLP 창을 열려면 db2cmd 명령을 발행하고 그 창 내에서 db2 명령행 처리기를 실행하려면 db2ext 명령을 발행하십시오.

DMB0353E db2ext 명령행 처리기 백그라운드 프로세스와 통신할 수 없습니다.

원인: db2ext 명령행 처리기에 대한 백그라운드 프로세스가 실행 중이지만 db2ext 명령행 처리기는 백그라운드 프로세스와 통신할 수 없습니다.

조치: 다른 창에서 db2ext 명령을 발행하도록 시도하십시오.

DMB0354E db2ext 명령행 처리기 백그라운드 프로세스를 시작할 수 없습니다.

원인: db2ext 명령행 처리기에 대한 백그라운드 프로세스가 실행 중이지만 db2ext 명령행 처리기는 백그라운드 프로세스와 통신할 수 없습니다.

조치: 백그라운드 프로세스에 대한 실행프로그램 모듈(db2extb 또는 db2extb.exe)이 존재하며 그 디렉토리가 PATH 환경 변수에 있는지 점검하십시오.

DMB0355E db2ext 명령행 처리기 백그라운드 프로세스가 시간종료되었습니다.

원인: db2ext 명령행 처리기에 대한 백그라운드 프로세스가 정상적으로 시작되었지만 db2ext 명령행 처리기가 허용된 시간 내에 백그라운드 프로세스와 통신할 수 없습니다.

조치: 다른 창에서 db2ext 명령을 발행하도록 시도하십시오.

DMB0356E db2ext 명령행 처리기 백그라운드 프로세스와 통신할 수 없습니다.

원인: db2ext 명령행 처리기가 백그라운드 프로세스에 요청을 보냈으나 요청이 받아들여지지 않았습니다.

조치: db2ext 명령행 처리기에 대한 백그라운드 프로세스가 아직 실행 중인지를 확인하십시오.

DMB0357E db2ext 명령행 처리기 백그라운드 프로세스가 응답하지 않습니다.

원인: db2ext 명령행 처리기가 백그라운드 프로세스에 요청을 보냈으나 백그라운드 프로세스가 허용된 시간 내에 응답하지 않았습니다.

조치: db2ext 명령행 처리기에 대한 백그라운드 프로세스가 아직 실행 중인지를 확인하십시오.

DMB0359E db2ext 명령행 처리기 백그라운드 프로세스 요청 대기행렬 또는 입력 대기행렬이 시간종료 기간 내에 작성되지 않았습니다.

원인: db2ext 명령행 처리기에 대한 백그라운드 프로세스가 허용된 시간 내에 메시지 대기행렬을 작성하지 않았습니다.(이 메시지는 UNIX 환경에만 적용됩니다.)

조치: DB2 인스턴스 홈 디렉토리가 있는 디스크가 가득 차지 않았는지 확인하십시오.(백그라운드 프로세스는 메시지 대기행렬용 파일을 작성하기 위해

이 홈 디렉토리가 필요합니다.) 디스크가 가득 차지 않은 경우에, 너무 많은 db2extb 프로세스를 시작했는지를 점검하십시오. 많은 창에서 db2ext 명령행 처리기를 실행했을 수 있습니다. 백그라운드 프로세스는 사용자가 db2ext 명령행 처리기 요청을 명령 모드에서 처음으로 발행한 창에서 실행됩니다. 더이상 필요하지 않을 때 db2ext 명령행 처리기를 종료하려면 db2ext terminate 명령을 발행하십시오. 백엔드 프로세스용 메시지 대기행렬은 사용자가 terminate 명령을 발행한 경우에만 삭제됩니다.

DMB0361E 컬럼 또는 테이블이 사용 가능하지 않습니다.

원인: 가져오기 UDF가 지정되었으나 지정된 테이블 컬럼이 Extender에 대해 사용 가능하지 않습니다.

조치: 테이블 컬럼을 사용 가능하게 하고 재시도하십시오.

DMB0363E 테이블 및 컬럼 이름이 누락되었습니다.

원인: 갱신 UDF가 호출되었지만 테이블은 지정되지 않았습니다.

조치: 테이블을 지정하고 재시도하십시오.

DMB0364E "<extender-name>" Extender가 테이블 공간 "<tablespace-name>"에 대해 이미 정의되었습니다.

원인: 지정된 데이터베이스, 테이블 또는 컬럼이 이미 지정된 테이블 공간이 아닌 다른 테이블 공간을 사용하는 Extender에 대해 사용 가능하게 되었습니다.

조치: 테이블 공간 스펙이 올바른지 점검하십시오.

DMB0365E "<schema-name>"."<table-name>"에 대한 메타데이터 테이블인 "<metadata-table-name>"과 "<metadata-table-name>"에 대해 CONTROL 특권이 없습니다.

원인: 지정된 사용자 테이블용 메타데이터 테이블에 대해 필요한 CONTROL 특권이 없으므로 요청이 거부되었습니다.

조치: 데이터베이스 관리자에게 메타데이터 테이블에 대한 CONTROL 특권을 부여하도록 하십시오.

DMB0366E 특성 이름이 예상됩니다.

원인: 특성 이름이 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0367E 색상|색상 히스토그램|파일이 예상됩니다.

원인: 『색상』, 『히스토그램』, 또는 『파일』이 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0368E ','가 예상됩니다.

원인: ','가 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0369E 파일이 올바르지 않음.

원인: 조회 문자열에 지정된 파일이 올바르지 않습니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0370E 파일 이름이 예상됩니다.

원인: 파일 이름이 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0371E 서버|클라이언트가 예상됩니다.

원인: 『서버』 또는 『클라이언트』가 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0372E '('가 예상됩니다.

원인: '('가 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0373E ')'가 예상됩니다.

원인: ')'가 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0374E 백분율이 예상됩니다.

원인: 백분율 값이 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0375E 색상이 예상됩니다.

원인: 빨간색, 초록색 및 파란색 값이 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0376E '='가 예상됩니다.

원인: '='가 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0377E '<'이 예상됩니다.

원인: '<'이 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0378E '>'이 예상됩니다.

원인: '>'이 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0379E 'and'가 예상됩니다.

원인: 'and'가 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0380E 가중치가 예상됩니다.

원인: 가중치가 조회 문자열에서 예상됩니다.

조치: 조회 문자열을 정정하고 다시 시도하십시오.

DMB0381E 특성이 설정되지 않았습니다.

원인: 특성이 QBIC 카탈로그에 추가되지 않았습니다.

조치: 특성을 QBIC 카탈로그에 추가하고 이미지를 다시 카탈로그화하십시오.

DMB0382E 조회를 빌드할 수 없었습니다.

원인: 현재 데이터베이스에 대한 Extender 서버가 중지되었을 수도 있습니다.

조치: 서버의 상태를 점검하십시오. 서버가 실행 중이 아닌 경우에는 db2ext 명령행에서 START SERVER 명령을 사용하여 서버를 재시작하십시오.

DMB0383E 조회를 실행할 수 없었습니다.

원인: 현재 데이터베이스에 대한 Extender 서버가 중지되었을 수도 있습니다.

조치: 서버의 상태를 점검하십시오. 서버가 실행 중이 아닌 경우에는 db2ext 명령행에서 START SERVER 명령을 사용하여 서버를 재시작하십시오.

DMB0384E 다음 항목을 확보할 수 없었습니다.

원인: 목록의 끝에 도달했습니다.

조치: 응용프로그램이 목록의 끝 이상의 항목을 검색하려고 시도한 것이 아닌지 확인하십시오.

DMB0386E 사용자 데이터와 나란히 배치할 수 없습니다.

원인: SQL API sqluihsh()가 0이 아닌 리턴 코드를 리턴했습니다.

조치: 재시도하십시오. 문제가 계속되면 IBM 지원 부서에 문의하십시오.

DMB0387E 지정된 테이블 공간에 대한 노드 그룹이 사용자 테이블의 노드 그룹과 다릅니다.

원인: 테이블을 사용 가능화하기 위해 입력으로서 전달되는 하나 이상의 테이블 공간(즉 메타데이터 테이블, 색인 또는 BLOB용)이 사용자 테이블이 정의된 노드 그룹과는 다른 노드 그룹에 정의되어 있습니다.

조치: 사용 가능화되는 사용자 테이블과 동일한 노드 그룹에 정의된 테이블 공간을 사용하십시오.

DMB0388E 보통, 긴 또는 색인 테이블 공간이 동일한 노드 그룹에 정의되어 있지 않습니다.

원인: 데이터베이스를 사용 가능화하기 위해 입력으로서 전달되는 하나 이상의 테이블 공간(즉, 메타데이터 테이블, 색인 또는 BLOB용)이 다른 테이블 공간과 동일한 노드 그룹에 정의되어 있지 않습니다.

조치: 동일한 노드 그룹에 정의된 테이블 공간을 사용하십시오.

DMB0389W 지정된 테이블 공간의 노드 그룹이 모든 파티션 서버를 포함하지 않습니다.

원인: 입력으로서 전달되는 테이블 공간이 파티션 서버를 모두 다 포함하지 않는 노드 그룹에 정의되어 있습니다.

조치: 아무런 조치도 필요하지 않습니다. 단, 테이블 공간이 모든 파티션 서버를 포함하는 노드 그룹에 정의되어 있으면, 가져오기 및 갱신 UDF가 보다 효율적으로 수행됩니다. 이러한 특징은 특히 Extender 응용프로그램이 BLOB 형식으로 미디어

내용을 저장할 때 두드러집니다.

DMB0391I 이 명령은 DB2 UDB 클라이언트가 DB2 UDB 서버에 액세스 중일 때에만 실행할 수 있습니다.

원인: db2ext 명령행 처리기가 DB2 UDB 서버에 연결되어 있지 않거나, DB2 UDB 클라이언트가 db2ext 명령행 처리기를 시작하지 않았습니다. 예를 들어, START SERVER 명령은 db2ext 명령행 처리기가 비 DB2 Extended Enterprise Edition 서버에 연결되어 있는 경우에만 올바릅니다.

조치: 현재 클라이언트/서버 구성에서는 이 명령을 발행하지 마십시오.

DMB0392I 이 명령은 DB2 UDB 클라이언트가 DB2 UDB Extended Enterprise Edition 서버에 액세스 중일 때에만 실행할 수 있습니다. 예를 들어, DISCONNECT SERVER 명령은 db2ext 명령행 처리기가 DB2 Extended Enterprise Edition 서버에 연결되어 있는 경우에만 올바릅니다.

원인: db2ext 명령행 처리기가 DB2 UDB Extended Enterprise Edition 서버에 연결되어 있지 않거나, DB2 UDB 클라이언트가 db2ext 명령행 처리기를 시작하지 않았습니다.

조치: 현재 클라이언트/서버 구성에서는 이 명령을 발행하지 마십시오.

DMB0402E "<command-name>" 명령에 대한 "<option-name>" 옵션은 응용프로그램이 DB2 "<server-type>" 서버에 연결된 경우에만 올바릅니다.

원인: db2ext 명령행 처리기가 해당 옵션을 지원하는 서버 유형에 연결되어 있지 않으므로, 지정된 매개변수가 올바르지 않습니다. 예를 들어, db2ext 명령행 처리기가 DB2 Extended Enterprise Edition 서버에 연결되어 있는 경우에만 GET SERVER STATUS 명령에 NODENUM <nodenum> 매개변수를 지정할 수 있습니다.

조치: 현재 클라이언트/서버 구성에서는 이 명령-매개변수 조합을 발행하지 마십시오.

DMB0411E 올바르지 않은 기본 포트입니다.

원인: 인스턴스 작성 중 올바르지 않은 TCP/IP 포트 번호를 기본 포트로서 입력했습니다.

조치: 올바른 구문은 dmbicrt -r:base_port,end_port -t:base_port,end_port입니다. 매개변수를 정정하고 명령을 재시도하십시오.

DMB0412E 올바르지 않은 종료 포트입니다.

원인: 인스턴스 작성 중 올바르지 않은 TCP/IP 포트 번호를 종료 포트로 입력했습니다.

조치: 올바른 구문은 dmbicrt -r:base_port,end_port -t:base_port,end_port입니다. 매개변수를 정정하고 명령을 재시도하십시오.

DMB0413E DB2 Extender 설치 경로를 해석할 수 없습니다.

원인: 인스턴스 작성 프로그램이 ″DMBPATH″ 환경 변수의 값을 찾을 수 없었습니다.

조치: ″DMBPATH″ 변수를 설정하고 응용프로그램을 재시도하십시오.

DMB0414E 컴퓨터 호스트 이름을 해석할 수 없습니다.

원인: 컴퓨터의 이름을 해석하는 동안 내부 오류가 발생했습니다.

조치: IBM 지원 부서에 문의하십시오.

DMB0415E 이 머신에 대한 노드 번호를 해석할 수 없습니다.

원인: 인스턴스 작성이 실행 중인 머신이 ″db2nodes.cfg″ 파일에 나열되어 있지 않습니다.

조치: ″db2nodes.cfg″에 해당 머신을 추가한 다음 응용프로그램을 재시도하십시오.

DMB0416E 이 프로그램은 루트로 시작해야 합니다. 계속할 수 없습니다.

원인: 프로그램을 실행 중인 사용자 ID가 루트 권한을 가지고 있지 않습니다.

조치: 루트로서 로그온하여 응용프로그램을 재시도하십시오.

DMB0417E 이 프로그램은 관리자 권한을 가진 사용자로 실행해야 합니다. 계속할 수 없습니다.

원인: 프로그램을 실행 중인 사용자 ID가 관리자 권한을 가지고 있지 않습니다.

조치: 관리 권한이 있는 사용자 ID로 로그온하여, 응용프로그램을 재시도하십시오.

DMB0418E 사용자 ″<userid>″에 관한 정보를 확보할 수 없습니다.

원인: 작성 중인 인스턴스와 연관된 사용자 정보를 확보하려고 시도할 때 내부 오류가 발생했습니다.

조치: 작성 중인 인스턴스와 동일한 이름을 가지는 올바른 사용자 ID가 있는지 확인한 후, 응용프로그램을 재시도하십시오.

DMB0419E AIV Extender 디렉토리 ″<directory_name>을 작성할 수 없습니다. 리턴 코드 = <code>

원인: 지정된 디렉토리를 작성하는 중 오류가 발생했습니다. 리턴 코드는 운영 체제에서 리턴된 오류를 나타냅니다.

조치: 디렉토리 이름에 지정된 파일 시스템/드라이브가 있는지와 디렉토리 작성이 허용되는지 확인하십시오.

DMB0420E AIV Extender 디렉토리 "<directory_name>"에 대한 링크를 작성할 수 없습니다. 리턴 코드 = <code>

원인: 지정된 심볼 링크를 작성하는 중 오류가 발생했습니다. 리턴 코드는 운영 체제에서 리턴된 오류를 나타냅니다.

조치: 디렉토리 이름에 지정된 파일 시스템/드라이브가 있는지와 디렉토리 작성이 허용되는지 확인하십시오.

DMB0421E "<file_name>" 파일을 열 수 없습니다. 리턴 코드 = <code>

원인: 지정된 파일을 여는 중 오류가 발생했습니다. 리턴 코드는 운영 체제에서 리턴된 오류를 나타냅니다.

조치: 파일이 있는지와 파일 열기가 허용되는지 확인하십시오.

DMB0422E "<file_name>" 파일을 쓸 수 없습니다. 리턴 코드 = <code>

원인: 지정된 파일에 쓰는 중 오류가 발생했습니다. 리턴 코드는 운영 체제에서 리턴된 오류를 나타냅니다.

조치: 파일이 있는지와 그 파일로의 쓰기가 허용되는지 확인하십시오.

DMB0424E "db2nodes.cfg" 파일을 찾을 수 없습니다.

원인: DB2 파일 "db2nodes.cfg"를 찾을 수 없습니다.

조치: 올바른 버전의 DB2 UDB Extended Enterprise Edition을 설치했는지 확인하고 응용프로그램을 재시도하십시오.

DMB0426E 오류: "<registry_key>" 키를 여는 중 "<error_code>"가 발생했습니다.

원인: 지정된 레지스트리 키를 여는 중 오류가 발생했습니다.

조치: 리턴 코드를 기록하고 IBM 지원 부서에 문의하십시오.

DMB0427E "<variable>" 변수가 프로파일 레지스트리에서 설정되지 않았습니다.

원인: 지정된 값이 Windows NT 레지스트리에 없습니다.

조치: 유효한 DB2 Extender 변수의 이름을 지정했는지 확인하십시오.

DMB0430E DB2 레지스트리 값을 찾을 수 없습니다.

원인: DB2에 의해 사용된 레지스트리 값을 찾을 수 없습니다.

조치: 올바른 버전의 DB2 UDB Extended Enterprise Edition이 설치되어 있는지 확인하고 응용프로그램을 재시도하십시오.

DMB0431E Extender 레지스트리 키 "<registry_key>"를 작성할 수 없습니다.

원인: Extender 레지스트리 키를 작성하는 중 오류가 발생했습니다.

조치: IBM 지원 부서에 문의하십시오.

DMB0432E Extender 레지스트리 키 "<registry_key>"에 대한 값을 설정할 수 없습니다.

원인: Extender 레지스트리 키 값을 설정하는 중 내부 오류가 발생했습니다.

조치: IBM 지원 부서에 문의하십시오.

DMB0435E 제어 파일 "<control_file>"에 액세스할 수 없습니다.

원인: 지정된 제어 파일을 찾을 수 없었습니다.

조치: IBM 지원 부서에 문의하십시오.

DMB0443E "<directory_name>" 디렉토리를 열 수 없습니다. 리턴 코드 = <code>

원인: 지정된 디렉토리를 여는 중에 오류가 발생했습니다. 리턴 코드는 운영 체제에서 리턴된 오류를 나타냅니다.

조치: 디렉토리 이름에 지정된 파일 시스템/드라이브가 있는지와 디렉토리 열기가 허용되는지 확인하십시오.

DMB0449W -q:datapath는 DB2 Extender 인스턴스 작성을 위해 필수입니다.

원인: DB2 Extender 인스턴스를 작성할 때 `-q` 매개변수를 지정하지 않았습니다.

조치: 이 매개변수를 지정하고 응용프로그램을 재시도하십시오.

DMB0450W 지정된 "<port>" 포트의 하나 이상이 이미 사용 중입니다.

원인: 서비스 파일에 이미 사용 중인 것으로 나열되어 있는 포트를 DB2 Extender용으로 지정했습니다.

조치: 사용 중이 아닌 포트를 지정하고 응용프로그램을 재시도하십시오.

DMB0452E "db2nodes.cfg" 파일에 노드 번호 "<node_num>"이 없습니다.

원인: `db2nodes.cfg` 파일에서 이 머신의 노드 번호를 찾을 수 없습니다.

조치: `db2nodes.cfg` 파일에 노드 번호를 추가한 다음 응용프로그램을 재시도하십시오.

DMB0460W TCP/IP 포트가 사용 가능한지를 판별할 수 없습니다.

원인: 지정된 TCP/IP 포트가 이미 사용 중인지 확인하는 중에 오류가 발생했습니다.

조치: 지정된 포트가 다른 응용프로그램에서 사용 중인 것으로 서비스 파일에 나열되지 않는지 확인하십시오.

DMB0462E 이 노드를 초기화할 수 없습니다. 리턴 코드 = <code>

원인: Extender 시작시 현재 노드를 초기화하는 중에 오류가 발생했습니다.

조치: IBM 지원 부서에 문의하십시오.

DMB0495E 이 AIV Extender 버전은 긴 이름을 지원하지 않습니다.

원인: Extender 관리 API를 호출하거나 db2ext 명령행 명령을 발행할 때 긴 식별자를 지정했습니다. 이 AIV Extender 버전에서 지원되는 식별자의 최대 길이는 다음과 같습니다.

- 지역 권한 부여 ID(AUTHID) — 8자
- 테이블 스키마(TABSCHEMA) — 8자
- 테이블 이름(TABNAME) — 18자
- 컬럼 이름 — 18자

API 호출이나 명령에서 짧은 식별자가 사용되는지 확인하십시오.

DMB0496E 올바르지 않은 테이블 이름 또는 컬럼 이름이 지정되었습니다.

원인: Extender 관리 API를 호출하거나 db2ext 명령행 명령을 발행할 때 올바르지 않은 식별자를 지정했습니다. 가능한 원인은 식별자 이름이 너무 길었기 때문입니다. DB2 UDB에서의 이름 길이에 대한 정보는 *빠른 시작*을 참조하십시오.

API 호출이나 명령에서 짧은 식별자가 사용되는지 확인하십시오.

DMB497E DB2MMDATAPATH에 대한 액세스가 거부되었습니다.

원인: **(EEE 전용)** 모든 노드에서 액세스할 수 없는 디렉토리나 공유 이름을 지정했습니다. DB2 Extender의 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 존재해야 하며 모든 노드에서 액세스할 수 있어야 합니다. 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 모든 노드에 존재하며 액세스할 수 있는지 확인하십시오.

DMB498E DB2MMDATAPATH 경로 중 최소한 한 부분이 디렉토리가 아닙니다.

원인: **(EEE 전용)** 노드상의 디렉토리가 아닌 디렉토리나 공유 이름을 지정했습니다. DB2 Extender의 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 존재해야 하며 모든 노드에서 액세스할 수 있어야 합니다. 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 모든 노드에 존재하며 액세스할 수 있는지 확인하십시오.

DMB499E DB2MMDATAPATH 문자열이 너무 깁니다.

원인: **(EEE 전용)** DB2MMDATAPATH 변수가 너무 길어지는 디렉토리나 공유 이름을 지정했습니다. DB2 Extender의 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 존재해야 하며 모든 노드에서 액세스할 수 있어야 합니다. 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 모든 노드에 존재하며 액세스할 수 있는지 확인하십시오.

DMB500E DB2MMDATAPATH 디렉토리가 없습니다.

원인: **(EEE 전용)** 노드상에 존재하지 않는 디렉토리나 공유 이름을 지정했습니다. DB2 Extender의 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 존재해야 하며 모든 노드에서 액세스할 수 있어야 합니다. 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 모든 노드에 존재하며 액세스할 수 있는지 확인하십시오.

DMB501E DB2MMDATAPATH의 알 수 없는 stat( ) 오류.

원인: (EEE 전용) 이 환경 변수에서 디렉토리나 공유 이름에 액세스하려고 할 때 문제점이 발생했습니다. DB2 Extender의 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 존재해야 하며 모든 노드에서 액세스할 수 있어야 합니다. 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 모든 노드에 존재하며 액세스할 수 있는지 확인하십시오.

DMB502E DB2MMDATAPATH는 있으나 디렉토리가 아닙니다.

원인: (EEE 전용) 디렉토리나 공유 이름에 대해 이름을 지정했지만 모든 노드상의 디렉토리나 공유 이름에 대한 이름이 아닙니다. DB2 Extender의 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 존재해야 하며 모든 노드에서 액세스할 수 있어야 합니다. 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 모든 노드에 존재하며 액세스할 수 있는지 확인하십시오.

DMB503E DB2MMDATAPATH는 있으나 읽을 수 없습니다.

원인: (EEE 전용) 모든 노드에서 읽을 수 없는 디렉토리나 공유 이름을 지정했습니다. DB2 Extender의 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 존재해야 하며 모든 노드에서 액세스할 수 있어야 합니다. 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 모든 노드에 존재하며 액세스할 수 있는지 확인하십시오.

DMB504E DB2MMDATAPATH는 있으나 쓸 수 없습니다.

원인: (EEE 전용) 모든 노드에서 쓸 수 없는 디렉토리나 공유 이름을 지정했습니다. DB2 Extender의 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 존재해야 하며 모든 노드에서 액세스할 수 있고 읽고 쓸 수 있어야 합니다. 인스턴스를 작성할 때 지정한 디렉토리나 공유 이름이 모든 노드에 존재하며 액세스할 수 있는지 확인하십시오.

DMB504E DB2MMDATAPATH가 설정되지 않았습니다.

원인: (EEE 전용) DB2 Extender 인스턴스를 작성할 때 환경 변수 DB2MMDATAPATH를 설정하지 않았습니다. 이것이 새로운 DB2 Extender 인스턴스일 경우, DMBIDROP를 사용하여 인스턴스를 제거한 후, `-q` 옵션을 올바르게 지정하여 이를 다시 작성하십시오.

이것이 새로운 DB2 Extender 인스턴스가 아니면, 다음을 수행하십시오.

- UNIX 환경에서:
  1. 디렉토리가 올바른지, 존재하는지, 그리고 모든 노드에서 액세스할 수 있는지 확인하십시오.
  2. *$INSTHOME*/dmb/dmbprofile을 수정하여 DB2MMDATAPATH를 디렉토리로 내보내기 하십시오.
- Windows 환경에서:
  1. 디렉토리의 공유 이름이 올바른지, 그리고 그 디렉토리가 존재하며 모든 노드에서 액세스할 수 있는지 확인하십시오.

2. IAV Extender 인스턴스 레지스트리에서 값으로 그 공유 이름을 갖는 레지스트리 항목 DB2MMDATAPATH를 추가하십시오. 키는 \\HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2 Extenders\PROFILE*instance_name*\DB2MMDATAPATH입니다.

DMB506E 인스턴스 이름이 설정되지 않았습니다.

원인: DMBSTART를 실행할 때 DB2INSTANCE 환경 변수가 설정되지 않았습니다. DMBSTART를 사용하여 DB2 Extender 서비스를 시작하기 전에 DB2START가 제대로 작동하는지 확인하십시오.

DMB507E 사용법: dmbssd *name* node *arguments*

원인: 내부 오류. IBM 대표에게 문의하십시오.

DMB508E 노드 번호는 0보다 크거나 같아야 합니다.

원인: 내부 오류. IBM 대표에게 문의하십시오.

DMB509E 이러한 프로그램은 수동으로 시작해서는 안됩니다.

원인: 내부 오류. IBM 대표에게 문의하십시오.

DMB512E 사용법: *arguments* dmbInstName.

원인: 내부 오류. IBM 대표에게 문의하십시오.

DMB513E *Name*은 올바른 인스턴스 이름이 아닙니다.

원인: DB2 Extender 인스턴스를 제거하려고 할 때 지정한 이름이 인스턴스의 이름으로 인식되지 않습니다. 올바른 인스턴스 이름을 지정했고 그런 이름의 디렉토리 *$INSTHOME*/dmb가 존재하는지 확인하십시오.

DMB514I 서버 또는 클라이언트가 이 인스턴스에 설치되지 않았습니다.

원인: DB2 Extender 인스턴스를 제거하려고 했으나, Extender가 설치되지 않았습니다. 설치가 올바르고 설치 디렉토리의 이름이 바뀌지 않았는지 확인하십시오.

DMB515I DB2 인스턴스가 제거되지 않았습니다. DB2IDROP을 호출하여 DB2 인스턴스를 제거할 수 있습니다.

원인: DB2 Extender 인스턴스를 제거할 때, 연관된 DB2 인스턴스가 제거되지 않았습니다. DB2IDROP을 사용하여 DB2 인스턴스를 제거하십시오.

DMB518E 예기치 않은 오류. 함수 = *function name*, 리턴 코드 = *return_code*.

원인: DB2 Extender 인스턴스를 작성하거나 제거하려고 할 때 예기치 않은 오류가 발생했습니다. 설치와 설정이 올바른지 확인하십시오.

DMB520E **이 프로그램을 루트로서 실행할 수 없습니다.**

원인: 이 조치를 수행할 수 있는 올바른 권한을 가지고 있는지 확인하십시오.

DMB521E *name*에 대한 허용권한 변경에 실패했습니다.

원인: 허용권한을 변경할 수 있는 올바른 권한을 가지고 있는지 확인하십시오.

DMB522E *name*에 대한 소유권 변경에 실패했습니다.

원인: 소유권을 변경할 수 있는 올바른 권한을 가지고 있는지 확인하십시오.

DMB523E *name*에 대한 그룹 소유권 변경에 실패했습니다.

원인: 그룹 소유권을 변경할 수 있는 올바른 권한을 가지고 있는지 확인하십시오.

DMB524E *name* 파일 또는 디렉토리가 이미 있습니다.

원인: 지정한 이름의 파일 또는 디렉토리가 이미 있습니다. 다른 이름을 선택하고 명령을 다시 실행하십시오.

DMB525E *name* 작성에 실패했습니다.

원인: 이 조치를 수행할 수 있는 올바른 권한을 가지고 있는지 확인하십시오.

DMB526E *name* 파일 또는 디렉토리가 누락되었습니다.

원인: 표시된 파일 또는 디렉토리를 찾을 수 없었습니다. 올바른 파일 이름이나 디렉토리 이름을 지정했는지 확인하십시오.

DMB527E *name* 파일 또는 디렉토리를 *name*에 복사하는 데 실패했습니다.

원인: 파일이나 디렉토리를 복사할 수 있는 올바른 권한을 가지고 있는지 확인하십시오. 복사할 수 있을 만큼 충분한 공간이 있는지 확인하십시오.

DMB528E *user_ID* 사용자 ID가 올바르지 않습니다.

원인: 올바르지 않은 사용자 ID를 지정했습니다. 사용자 ID를 확인한 후 명령을 다시 실행하십시오.

DMB529E *user_ID* 사용자 ID의 *group* 1차 그룹이 올바르지 않습니다.

원인: 사용자 ID에 대해 올바르지 않은 1차 그룹을 지정했습니다. 올바른 1차 그룹인지 확인한 후 명령을 다시 실행하십시오.

DMB530E *name* 인스턴스 이름이 올바르지 않습니다.

원인: 인스턴스를 작성하거나 인스턴스에 대해 작업할 때 올바르지 않은 이름을 지정했습니다. 올바른 인스턴스 이름인지 확인한 후 명령을 다시 실행하십시오.

DMB531E 지원되지 않는 운영 체제 *name*, 버전 *version_number*.

원인: 지원되지 않는 운영 체제 버전에서 이 명령을 실행하려고 했습니다. 이 조작을 수행하기 위한 요구사항을 확인하십시오.

DMB535E 지정된 파일에 액세스할 수 없습니다.

원인: 이 명령을 실행하기 전에 파일에 대한 액세스 권한을 가지고 있는지 확인하십시오.

DMB0533E API <*API name*>은 파티션된 데이터베이스 서버 환경에 대해 지원되지 않습니다.

원인: 파티션된 데이터베이스 환경에서 지정한 API를 사용할 수 없습니다.

조치: 응용프로그램에서 이 함수를 처리하는 방법에 대한 정보는 지정된 API의 절을 참조하십시오.

DMB0534E UDF가 지원되지 않습니다.

원인: 파티션된 데이터베이스 환경에서는 사용자 정의 함수(UDF)를 사용할 수 없습니다.

조치: 메시지 SQL0443N을 확인하여 문제점이 발견된 UDF를 판별하십시오. 응용프로그램에서 이 함수를 처리하는 방법에 대한 정보는 해당되는 UDF의 절을 참조하십시오.

진단 추적

DB2 Extender는 Extender 서버 활동을 기록하는 추적 기능을 포함합니다. IBM 서비스 담당자가 지시한 경우에만 추적 기능을 사용해야 합니다.

추적 기능은 DB2 Extender 구성요소의 시작 또는 종료, 또는 DB2 Extender 구성요소에 의한 오류 코드의 리턴과 같은 다양한 이벤트에 대해 서버 파일에 정보를 기록합니다. 추적 기능은 많은 이벤트에 대해 정보를 기록하므로, 예를 들면, 에러 조건을 조사할 때와 같이 필요한 때에만 사용되어야 합니다. 게다가, 추적 기능을 사용할 때에는 반드시 실행 중인 응용프로그램의 수를 제한해야 합니다. 실행 중인 응용프로그램의 수를 제한하는 것이 문제의 원인을 밝히는 것을 용이하게 합니다.

추적을 제어하려면 DMBTRC 명령을 사용하십시오. OS/2 서버, AIX 서버 또는 Windows NT 4.0 이상의 서버상의 명령행에서 명령을 발행할 수 있습니다. 명령을 발행하려면 반드시 SYSADM, SYSCTRL, 또는 SYSMINT 권한을 가지고 있어야합니다.

다음을 수행하려면 DMBTRC 명령을 사용하십시오.

- 추적 시작
- 추적 중지
- 더 쉽게 읽을 수 있도록 추적 정보 재형식화
- 추적 상태 표시

추적 시작

명령을 입력하여 추적을 시작할 수 있습니다.

dmbtrc on *path*

여기서, *path*는 추적 정보를 포함할 서버 파일의 경로입니다.

예를 들어, 다음 명령은 추적을 시작합니다.

```
dmbtrc on /tmp/trace.txt
```

추적 중지

명령을 입력하여 추적을 멈출 수 있습니다.

dmbtrc off

추적 정보 재형식화

추적 정보는 2진 형식으로 기록됩니다. 정보를 재형식화하고 다음 명령을 입력해서 좀 더 읽기 쉽도록 만들 수 있습니다.

dmbtrc format *input_file output_file*

여기서 *input_file*은 이진 형식으로 추적 정보를 포함하고 있는 파일이며 *output_file*은 재형식화된 정보를 포함할 파일입니다. *output_file* 매개변수는 선택적이며, 지정하지 않는 경우 새로 형식화된 정보가 화면에 표시됩니다.

예를 들어, 다음 명령은 추적 정보를 재형식화합니다.

```
dmbtrc format /tmp/trace.txt /tmp/fmttrace.txt
```

추적 상태 표시

아래 명령을 사용하십시오.

```
dmbtrc info
```

이것은 다음의 추적 상태 정보를 보여줍니다.

- 추적 기능의 사용 여부
- 추적 정보를 포함한 파일의 경로

제5부 부록 및 끝머리

부록A. DB2 Extender의 환경 변수 설정

DB2 Extender는 이미지, 오디오 또는 비디오 오브젝트의 저장, 검색 또는 갱신시 파일 이름 지정 방식에 융통성을 부여합니다. 또한 데이터베이스 테이블에서 이미지, 오디오 및 비디오 오브젝트를 표시 또는 재생하기 위해 프로그램을 지정하는 방법에도 융통성을 부여합니다.

파일 이름을 해석하기 위해 환경 변수가 사용되는 법

저장, 검색 및 갱신 조작을 위해 완전히 규정된 파일 이름(즉, 파일 이름이 뒤따르는 완전한 경로)을 지정할 수도 있지만 상대 파일 이름을 지정하는 것이 더 좋습니다. AIX, HP-UX 또는 Solaris에서 상대 파일 이름은 슬래쉬로 시작하지 않는 파일 이름입니다. OS/2 및 Windows에서 상대 파일 이름은 콜론과 백슬래쉬가 다음에 오는 드라이브 이름으로 시작하지 않는 파일 이름입니다.

상대 파일 이름을 지정한 경우에 Extender는 파일 이름을 해석하기 위해 다양한 클라이언트 및 서버 환경 변수 내의 디렉토리 스펙을 사용할 것입니다. 이렇게 함으로써 파일 이름을 변경하지 않고 클라이언트/서버 환경 내에서 파일을 옮길 수 있습니다. 완전히 규정된 파일 이름은 파일이 옮겨질 때마다 변경되어야 합니다.

표17은 파일 이름 해석시 Image, Audio 및 Video Extender에서 사용하기 위해 사용자가 설정할 수 있는 환경 변수를 나열하고 설명합니다.

표 17. DB2 Extender용 환경 변수

Image Extender	Audio Extender	Video Extender	설명
		서버 환경 변수	
DB2IMAGEPATH	DB2AUDIOPATH	DB2VIDEOPATH	서버 파일에서 저장, 검색 및 갱신 조작에 대한 소스 파일 이름을 해석하기 위해 사용됩니다.
DB2IMAGESTORE	DB2AUDIOSTORE	DB2VIDEOSTORE	서버 파일로의 저장 및 갱신 조작에 대한 목표 파일 이름을 해석하기 위해 사용됩니다.
DB2IMAGEEXPORT	DB2AUDIOEXPORT	DB2VIDEOEXPORT	서버 파일로의 검색 조작을 위한 목표 파일 이름을 해석하기 위해 사용됩니다.

표 17. DB2 Extender용 환경 변수 (계속)

Image Extender	Audio Extender	Video Extender	설명
DB2IMAGETEMP			임시 서버 파일을 작성하는 조작에 대한 목표 파일 이름을 해석하기 위해 사용됩니다. 그러나 TMP 환경 변수가 지정된 경우에는 TMP 디렉토리가 파일 이름을 해석하기 위해 사용됩니다.
		클라이언트 환경 변수	
DB2IMAGEPATH	DB2AUDIOPATH	DB2VIDEOPATH	클라이언트 파일에 대한 표시 및 재생 조작에 대한 소스 파일 이름을 해석하기 위해 사용됩니다.
DB2IMAGETEMP	DB2AUDIOTEMP	DB2VIDEOTEMP	임시 클라이언트 파일을 작성하는 조작에 대한 목표 파일 이름을 해석하기 위해 사용됩니다. 그러나 TMP 환경 변수가 지정된 경우에는 TMP 디렉토리가 파일 이름을 해석하기 위해 사용됩니다.

사용자가 지정된 Extender용으로 적당한 환경 변수를 설정하지 않은 경우에 Extender는 파일 이름을 해석하기 위해 다음의 환경 변수를 사용하게 됩니다.

환경 변수	설명
DB2MMPATH	저장, 검색 및 갱신 조작용 소스 파일 이름을 해석하기 위해 사용됩니다.
DB2MMSTORE	저장 및 갱신 조작용 목표 파일 이름을 해석하기 위해 사용됩니다.
DB2MMEXPORT	검색 조작용 목표 파일 이름을 해석하기 위해 사용됩니다.
DB2MMTEMP	임시 파일을 작성하는 조작용 파일 이름을 해석하기 위해 사용됩니다.

표시 또는 재생 프로그램을 식별하기 위해 환경 변수를 사용하는 법

파일 이름 해석을 비롯하여 환경 변수는 또한 Image Extender에 의해 검색된 이미지 오브젝트를 표시하고 Audio와 Video Extender에 의해 검색된 오디오 및 비디오 오브젝트를 재생하기 위한 프로그램 식별에 사용됩니다. 이러한 오브젝트를 표시하거나 재생하기 위해 각각 DBiBrowse, DBaPlay 및 DBvPlay API를 사용합니다. 각 API를 사용할 때, 표시나 재생 프로그램을 지정하거나 또는 기본 프로그램이 오브젝트를 표시 또는 재생하기를 원하는지를 표시할 수 있습니다.

DB2 Extender는 기본 표시나 재생 프로그램을 식별하기 위해 클라이언트에서 다음 환경 변수를 사용합니다.

환경 변수	설명
DB2IMAGEBROWSER	기본 이미지 표시 프로그램을 식별하기 위해 사용됩니다.
DB2AUDIOPLAYER	기본 오디오 플레이어 프로그램을 식별하기 위해 사용됩니다.
DB2VIDEOPLAYER	기본 비디오 플레이어 프로그램을 식별하기 위해 사용됩니다.

DB2MMDATAPATH 환경 변수의 사용법(EEE 전용)

DB2 Extender는 DB2MMDATAPATH 환경 변수를 사용하여 파티션된 데이터베이스 환경에서 다양한 조작의 위치를 해석합니다. 예를 들어, DB2 Image Extender는 DB2MMDATAPATH 값을 사용하여 파티션된 데이터베이스 환경에 QBIC 데이터를 저장합니다.

572 페이지의 『DMBICRT』와다음의 설치 『readme』 파일에 설명된 대로, DB2 Extender 인스턴스를 작성할 때 DB2MMDATAPATH를 설정합니다.

- aixeee 디렉토리의 install.txt(AIX에서 DB2 Extended Enterprise Edition와 함께 사용할 DB2 Extender 설치)
- soleee 디렉토리의 install.txt(Solaris Operating Environment에서 DB2 Extended Enterprise Edition와 함께 사용할 DB2 Extender 설치)

QBIC 특성과 색인 데이터를 저장하는 데에서 DB2MMDATAPATH 사용법의 한 예를 볼 수 있습니다. UNIX에서, DB2 Image Extender는 이러한 QBIC 데이터를 다음 디렉토리에 저장합니다.

db2mmdatapath /NODE*node_num*/QBIC/*database_name*

여기서, *db2mmdatapath*는 DB2MMDATAPATH 환경 변수의 값이고, *node_num*은 노드 번호이며, *database_name*은 데이터베이스 이름입니다.

다음 AIX 예를 고려하십시오. DB2MMDATAPATH가 /localfs/dmbdata로 설정되어 있다고 가정하십시오. 또한 sample이라는 이름의 데이터베이스가 노드 0, 2 및 5에 파티션되어 있다고 가정하십시오. 이 sample 데이터베이스의 경우, QBIC 데이터는 다음 디렉토리에 저장됩니다.

노드 0: /localfs/dmbdata/NODE0000/QBIC/sample

노드 2: /localfs/dmbdata/NODE0002/QBIC/sample

노드 5: /localfs/dmbdata/NODE0005/QBIC/sample

환경 변수 설정

AIX, HP-UX, Solaris, OS/2 및 Windows에서 환경 변수를 설정할 수 있습니다.

AIX, HP-UX, Solaris 서버 및 클라이언트에서 환경 변수 설정

AIX, HP-UX, Solaris에서, 환경 변수는 C 쉘, Korn 쉘 및 Bourne 쉘 스크립트에서 지정됩니다. DB2 Extender가 설치될 때, 서버의 환경 변수는 다음과 같이 설정됩니다.

C 쉘

```
setenv DB2MMPATH /usr/lpp/db2ext/samples:/tmp
setenv DB2MMTEMP /tmp
setenv DB2MMSTORE /tmp
setenv DB2MMEXPORT /tmp
```

Korn 및 Bourne 쉘

```
DB2MMPATH=/usr/lpp/db2ext/samples:/tmp
export DB2MMPATH

DB2MMSTORE=/tmp
```

```
export DB2MMSTORE

DB2MMEXPORT=/tmp
export DB2MMEXPORT

DB2MMTEMP=/tmp
export DB2MMTEMP
```

서버용 환경 변수는 DB2 Extender와 함께 배포된 샘플 프로그램 내에서 사용된 미디어 파일에 액세스하도록 허용하는 값으로 먼저 설정됩니다. (샘플 프로그램 및 미디어 파일에 관한 정보를 보려면 643 페이지의 『부록B. 샘플 프로그램 및 미디어 파일』을 참조하십시오.)

클라이언트 환경 변수는 사용자가 DB2 Extender를 AIX, HP-UX 또는 Solaris 클라이언트에 설치할 때 다음과 같이 설정됩니다.

C 쉘

```
setenv DB2MMPATH /tmp
setenv DB2MMTEMP /tmp
```

Korn 및 Bourne 쉘

```
DB2MMPATH=/tmp
export DB2MMPATH

DB2MMTEMP=/tmp
export DB2MMTEMP
```

파일 이름을 해석하는 데 사용되는 서버 및 클라이언트 환경 변수를 설정하십시오. 환경에 알맞는 변수를 설정하십시오. PATH 환경 변수에는 분리문자에 의해 분리되는 여러 디렉토리를 설정할 수 있습니다. STORE, EXPORT 및 TEMP 환경 변수에는 단 한 디렉토리만 설정됩니다.

각각 DB2IMAGEBROWSER, DB2AUDIOPLAYER 및 DB2VIDEOPLAYER 클라이언트 환경 변수에 적절한 이미지 표시, 오디오 재생 및 비디오 재생 프로그램의 이름을 지정하십시오.

다음과 같이 환경 변수의 초기 설정을 변경할 수 있습니다.

C 쉘

환경 변수를 설정하기 위해 SETENV 명령을 사용하십시오.

```
setenv env-var directory
```

예를 들면,

```
setenv DB2MMPATH /usr/lpp/db2ext/samples:/media
setenv DB2IMAGEPATH /employee/pictures:/images
setenv DB2AUDIOSTORE /employee/sounds
setenv DB2IMAGEBROWSER 'xv %s'
```

Bourne 쉘

환경 변수를 설정하기 위해 EXPORT 명령을 사용하십시오.

```
env-var=directory
export env-var
```

예를 들면,

```
DB2MMPATH=/usr/lpp/db2ext/samples:/media
export DB2MMPATH

DB2IMAGEPATH=/employee/pictures:/images
export DB2IMAGEPATH

DB2AUDIOSTORE=/employee/sounds
export DB2AUDIOSTORE
```

Korn 쉘

환경 변수를 설정하기 위해 EXPORT 명령을 사용하십시오.

```
export env-var=directory
```

예를 들면,

```
export DB2MMPATH=/usr/lpp/db2ext/samples:/media
export DB2IMAGEPATH=/employee/pictures:/images
export DB2AUDIOSTORE=/employee/sounds
```

OS/2 서버 및 클라이언트에서 환경 변수 설정

OS/2에서 환경 변수는 CONFIG.SYS 파일에 추가되며 설치되는 동안 자동적으로 설정됩니다.

OS/2 서버에서 DB2 Extender를 설치한 경우에 서버 환경 변수는 다음과 같이 설정됩니다.

```
SET DB2MMPATH=install-dir\SAMPLES; temp-file-dir
SET DB2MMSTORE=temp-file-dir
SET DB2MMEXPORT=temp-file-dir
SET DB2MMTEMP=temp-file-dir
```

여기서 *install-dir*는 설치 디렉토리이며 *temp-file-dir*은 임시 파일 디렉토리입니다. 기본 설치 디렉토리는 C:\DMB이며, 기본 임시 파일 디렉토리는 C:\DMB\TMP입니다. 설치하는 동안 어느 디렉토리나 위치를 변경할 수 있습니다. 임시 파일 디렉토리의 위치를 올바르게 지정하는 것이 중요합니다.

서버용 환경 변수는 DB2 Extender와 함께 배포된 샘플 프로그램 내에서 사용된 미디어 파일에 액세스하도록 허용하는 값으로 먼저 설정됩니다. (샘플 프로그램 및 미디어 파일에 관한 정보를 보려면 643 페이지의 『부록B. 샘플 프로그램 및 미디어 파일』을 참조하십시오.)

OS/2 클라이언트에서 DB Extender를 설치한 경우에 클라이언트 환경 변수는 다음과 같이 설정됩니다.

```
SET DB2MMPATH=temp-file-dir
SET DB2MMTEMP=temp-file-dir
```

환경 변수를 재설정하려면 SET 명령을 사용하십시오. PATH 환경 변수에는 분리 문자에 의해 분리되는 여러 디렉토리를 설정할 수 있습니다. STORE, EXPORT 및 TEMP 환경 변수에는 단 한 디렉토리만 설정됩니다.

각각 DB2IMAGEBROWSER, DB2AUDIOPLAYER 및 DB2VIDEOPLAYER 클라이언트 환경 변수에 적절한 이미지 표시, 오디오 플레이어 및 비디오 플레이어 프로그램을 지정하려면 SET 명령을 사용하십시오.

다음과 같이 SET 명령을 지정하십시오.

```
SET env-var=directory
```

예를 들면,

```
SET DB2MMPATH=C:\DMB\SAMPLES;\D:\MEDIA
SET DB2IMAGEPATH=C:\EMPLOYEE\PICTURES;D:\IMAGES
SET DB2AUDIOSTORE=C:\EMPLOYEE\SOUNDS
SET DB2IMAGEBROWSER=ib.exe %s
```

Windows 서버 및 클라이언트에서 환경 변수 설정

Windows에서 환경 변수를 설정하는 방법은 파티션되지 않은 환경에서 DB2 Extender를 사용하는지, 아니면 파티션된 데이터베이스 환경(즉, Windows용 DB2 Extended Enterprise Edition이 있는)에서 DB2 Extender를 사용하는 지에 따라 다릅니다.

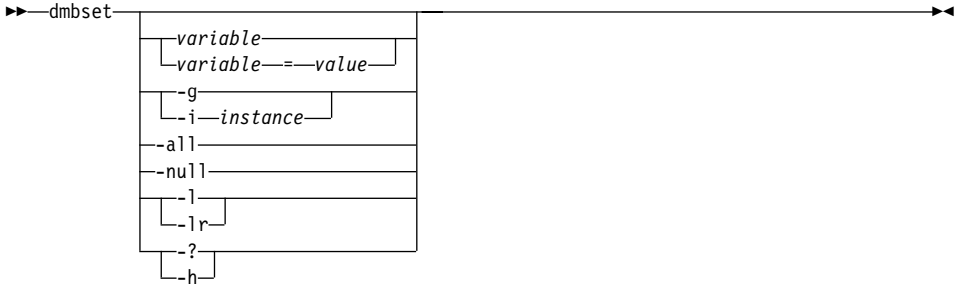
Windows 파티션되지 않은 데이터베이스 환경에서 환경 변수 설정(비 EEE 전용)

Windows에서 환경 변수는 시스템 레지스트리에 저장됩니다. Windows 제어판을 열고 시스템 아이콘을 선택하여 변수를 설정할 수 있습니다. 시스템 등록 정보 대화 상자에서 환경 탭을 선택하십시오. 환경 변수와 해당 값을 포함하는 두 개의 창이 있습니다. 맨 위 창에는 모든 사용자에게 영향을 미치는 변수가 표시됩니다. 맨 아래 창에는 현재 사용자에게만 영향을 미치는 변수가 표시됩니다.

Windows 파티션된 데이터베이스 환경에서 환경 변수 설정(EEE 전용)

Windows 파티션된 환경에서는, DB2 Extender에서 사용되는 모든 변수가 시스템 레지스트리의 개별 영역에 저장됩니다. Extender 변수를 검사하고 변경하기 위해 DMBSET이라고 하는 프로그램이 제공됩니다.

이 프로그램의 구문은 다음과 같습니다.



변수의 값을 조회하려면, dmbset *variable_name*을 입력하십시오. 예를 들면,

```
dmbset DB2MMPATH
```

변수의 값을 설정하려면, dmbset *variable_name=value*를 입력하십시오. 예를 들면,

```
dmbset DB2MMPATH=C:\DMB\SAMPLES
```

정의된 인스턴스에 대한 모든 변수의 값을 표시하려면, dmbset -i *instance_name*을 입력하십시오. 예를 들면,

```
dmbset -i dmbinst1
```

값을 널(NULL)로 설정하려면, dmbset *variable_name -null*을 입력하십시오. 예를 들면,

```
dmbset DB2MMPATH -null
```

모든 인스턴스에서 사용되는 변수의 값을 표시하려면, `dmbset -g`를 입력하십시오.

DB2 Extender에서 사용되는 모든 변수의 이름을 나열하려면, `dmbset -lr`을 입력하십시오.

레지스트리에 정의된 모든 인스턴스 프로파일 이름을 나열하려면, `dmbset -l`을 입력하십시오.

파티션된 데이터베이스 환경에서는 여러 가지 방식으로 DB2 Extender의 환경 변수를 설정할 수 있습니다. 예를 들어, 다음 형식으로 DB2MMDATAPATH를 제외한 모든 환경 변수의 값을 지정할 수 있습니다.

- 일반 이름지정 규칙 이름: *machine_name**share_name*. 예를 들면,

  ```
  \\harmony\JimsShr
  ```

- 드라이브:경로. 예를 들면,

  ```
  f:\media
  ```

- 기타: *share_name**directory_name*. 예를 들면,

  ```
  JimsShr\images
  ```

부록B. 샘플 프로그램 및 미디어 파일

다양한 샘플 프로그램이 DB2 Extender에 포함되어 있습니다. 샘플 프로그램은 Extender에 의해 제공되는 이미지, 오디오 및 비디오 파일을 사용합니다. 대부분의 샘플 프로그램은 C로 작성되며, 모든 C 샘플 프로그램은 콜 레벨 인터페이스(CLI) 형식으로 되어 있습니다. 몇 개의 Java 샘플 프로그램과 Net.Data 샘플 매크로 파일도 제공됩니다.

샘플 프로그램은 DB2 Extender를 설치할 때 대상 디렉토리의 SAMPLES 서브디렉토리에 설치됩니다. 또한, DB2 Extender를 설치할 때 이미지, 오디오 및 비디오 파일이 대상 디렉토리의 SAMPLES 서브디렉토리에 설치됩니다. 설치 동안, 대상 디렉토리의 samples 서브디렉토리를 가리키도록 Extender 환경 변수가 설정됩니다.

샘플 프로그램

많은 파일이 DB2 Extender용 샘플 프로그램을 구성합니다. 파일은 다음과 같습니다.

파일	설명
enable.c	Audio, Image 및 Video Extender에 대해 데이터베이스를 사용 가능하게 하고 테이블을 작성하며 테이블과 그 컬럼을 사용 가능하게 합니다.
populate.c	데이터를 테이블에 가져옵니다.(프로그램은 C 형식입니다.)
Populate.java	데이터를 테이블에 가져옵니다.(프로그램은 Java 형식입니다.)
query.c	테이블의 데이터를 조회합니다.(프로그램은 C 형식입니다.)
Query.java	테이블의 데이터를 조회합니다.(프로그램은 Java 형식입니다.)
api.c	Extender API를 사용하여 데이터베이스를 조회합니다.
handle.c	UDF 내의 핸들 사용법 및 SELECT문 내에서 Where 절 비교를 하는 방법을 설명합니다.

qbcatdmo.c QBIC 카탈로그를 작성하고 이미지의 컬럼을 카탈로그 안으로 카탈로그화합니다.

qbicdemo.c QBIC 카탈로그를 조회합니다.

color.c qbicdemo.c에 대해 색상 테이블 선언을 합니다.

QbicQry.java QBIC 조회의 평균 색상과 히스토그램 색상 섹터를 제시합니다.

makesf.c makehtml.exe에서 사용할 컷 카탈로그 파일을 작성합니다.

makehtml.c 컷 카탈로그에 액세스하고 웹 브라우저에서 표시할 수 있도록 HTML 페이지를 작성합니다.

storybrd.java
makehtml.c에서 생성된 HTML 페이지에서 호출되는 컷들을 표시할 애플릿.

utility.c 유틸리티 루틴.

utility.h 유틸리티 루틴용 헤더 파일.

makefile.aix AIX에서 프로그램을 빌드하기 위한 makefile.

makefile.os2 OS/2에서 프로그램을 빌드하기 위한 makefile.

makefile.iva IBM VisualAge C++을 사용하여, Windows NT 4.0(이상)에서 프로그램을 빌드하기 위한 makefile.

makefile.mvc Microsoft Visual C++을 사용하여 Windows에서 프로그램을 빌드하기 위한 makefile.

makefile.sun Solaris에서 프로그램을 빌드하기 위한 makefile.

makefile.hp HP-UX에서 프로그램을 빌드하기 위한 makefile.

다음 샘플 프로그램용으로 실행 가능한 파일들이 제공됩니다. 샘플 프로그램은 보이는 순서대로 실행되도록 의도되었습니다.

1. Enable
2. Populate
3. Query
4. API

5. Handle
6. Qbcatdmo
7. Qbicdemo
8. QbicQry
9. Makesf
10. Makehtml

실행 프로그램 클래스 파일(Populate.class, Query.class, QbicQry.class, storybrd.class)이 샘플 Java 프로그램과 함께 제공됩니다.

샘플 프로그램을 실행하기에 앞서 반드시 서버에 데이터베이스를 만들어야 합니다. Extender 서비스 또한 반드시 서버 상에서 시작되어야 합니다. 샘플 프로그램을 실행하려면, 프로그램 이름을 입력하십시오. (프로그램의 실행 파일이 시작됩니다.) 데이터베이스 이름, 사용자 ID 및 암호에 대해 프롬프트가 표시됩니다. 데이터베이스를 작성한 사용자의 사용자 ID와 암호를 사용하십시오.

또한 샘플 프로그램용으로 자신의 실행 파일을 만들 수 있습니다. 그렇게 하려면 다음이 필요합니다.

1. 샘플 프로그램 파일을 쓰기 가능한 디렉토리에 복사하십시오.
2. DB2, Extender 및 컴파일러가 설치된 시스템 상의 위치를 지정하기 위해 makefile을 편집하십시오.
3. 파일을 실행 가능 프로그램으로 컴파일하기 위해 make 또는 nmake를 사용하십시오.

샘플 프로그램 설치 및 사용에 관한 자세한 정보는, 샘플 프로그램 디렉토리의 README.CNT 파일을 참조하십시오.

샘플 이미지, 오디오 및 비디오 파일

DB2 Extender에 제공되는 샘플 이미지, 오디오 및 비디오 파일은 다음과 같습니다.

- 이미지 파일
  - lizzi.bmp

  - sws_stri.bmp
  - nitecry.bmp
  - ranger_r.bmp
  - fuzzblue.bmp
- 오디오 파일
  - lizzi.wav
  - sws_stri.wav
  - nitecry.wav
  - ranger_r.wav
  - fuzzblue.wav
- 비디오 파일
  - nitecry.avi
  - sample.mpg

샘플 Net.Data 매크로 파일

DB2 Extender에는 extender.d2w라고 하는 Net.Data 매크로 파일이 제공됩니다. 웹 서버에서 실행할 때, 매크로 파일은 DB2 Extender UDF를 호출하는 SQL 문을 실행합니다. 매크로 파일은 웹 브라우저에서 표시하는 결과를 리턴합니다. 647 페이지의 그림30에 표시된 것처럼, 각각의 결과 페이지는 또한 결과를 생성하기 위해 실행되는 SQL문을 표시합니다. 648 페이지의 그림31은 샘플 Net.Data 매크로 파일의 내용을 표시합니다.

샘플 Net.Data 매크로 파일을 실행하려면, 웹 브라우저에서 URL http://*your server*/cgi-bin/db2www/extender.d2w/startHere를 입력하십시오.

여기서 *your server*는 웹 서버 이름입니다.

```
select cast(mmdbsys.thumbnail(covers) as blob(10000), cast(mmdbsys.thumbnail(video)
blob(3000)), mmdbsys.comment(music), artist, title, price, stock_no from sobay_catalog
```

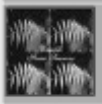
Cover	Video	Audio	Artist	Title	Price
		[Listen]	Lizzi	Decisions	25.00
		[Listen]	SWS Strings	Vivaldi: Four Seasons	25.50
		[Listen]	Nitecry	Run for Cover	15.00
		[Listen]	Ranger Rick	Handy Sue	12.25
		[Listen]	Fuzzy Blues	Aurora	22.00

그림 30. 샘플 Net.Data 매크로 파일을 수행하는 웹 응용프로그램. 각각의 결과 페이지는 결과를 생성하기 위해 실행되는 SQL문을 표시합니다.

```
%{ ------------------------------------------------------------------------ %}
%{ Copyright International Business Machines Corporation, 1998.            %}
%{ All rights reserved.                                                     %}
%{                                                                          %}
%{ Sample Net.Data macro which shows how to call image, audio, and video   %}
%{ extender UDFs.                                                          %}
%{                                                                          %}
%{ To run, put this macro in your MACRO_PATH root, make sure the tmplobs   %}
%{ directory exists under your web server's document root, and create      %}
%{ the database to be used when running the extender sample programs       %}
%{ 'enable' and 'populate'.  Run 'enable' and 'populate'.  If you name your %}
%{ database something other than 'testdb2', you'll need to change the     %}
%{ definition of DATABASE below.  The extender environment variable        %}
%{ DB2MMEXPORT needs to be set for the instance used by Net.Data to point  %}
%{ to the webserver's <document root>/tmplobs directory.  Then restart DB2 %}
%{ and the extenders to have the variable take effect.                     %}
%{ If you are not running Net.Data's Connection Manager, you'll need to    %}
%{ provide the LOGIN and PASSWORD to the database.  If these instructions  %}
%{ seem unfamiliar to you, you should read the Net.Data documentation at   %}
%{ http://www.software.ibm.com/data/netdata/docs (or the extender documen- %}
%{ tation on the extender sample programs).                                %}
%{                                                                          %}
%{ To disable the showing of SQL statements, change the value of SHOWSQL   %}
%{ below to "no".                                                          %}
%{ ------------------------------------------------------------------------ %}

%{ ------------------------------------------------------------------------ %}
%{ Definitions section                                                     %}
%{ ------------------------------------------------------------------------ %}
%define{
  DATABASE="testdb2"
  SHOWSQL="yes"
%}
```

그림 31. Net.Data 샘플 매크로 파일 (1/5)

```
%{ ------------------------------------------------------------------------ %}
%{ SQL functions                                                            %}
%{ ------------------------------------------------------------------------ %}
%function (DTW_SQL) startHereSQL(){
  select artist, title, stock_no, price from sobay_catalog

  %REPORT{
    <table border="2" bgcolor="#b1b1b1">
           <tr><th>Artist <th> Title <th> Stock<th> Number <th> Price </tr>
    %ROW{ <tr><td> $(V_artist) <td> $(V_title) <td>  $(V_stock_no) <td> $(V_price) <tr>
    %}
    </table>
   %}
%}

%function (DTW_SQL) addThumbsSQL(){
  select cast(mmdbsys.thumbnail(covers) as blob(10000)),
         cast(mmdbsys.thumbnail(video) as blob(3000)),
         mmdbsys.comment(music), artist, title, price, stock_no
  from sobay_catalog

  %REPORT{
   <table border="2" bgcolor="#b1b1b1">
                 <tr><th>Cover <th>Video <th>Audio <th>Artist <th>Title <th>Price </tr>
    %ROW{  <tr><td>< a href="showCover?stock_no=$(V_stock_no)"><img src="$V1"></a>
               <td><  a href="getVideo?stock_no=$(V_stock_no)"><img src="$V2"></a>
               <td><  a href="getAudio?stock_no=$(V_stock_no)&filename=$V3")>[Listen]</a>
               <td> $(V_artist) <td> $(V_title) <td> $(V_price) </tr>
    %}
    </table>
  %}
%}

%function (DTW_SQL) showCoverSQL(){
  select cast(mmdbsys.content(covers, 'GIF') as blob(150000)), mmdbsys.format(covers)
    from sobay_catalog
   where stock_no = '$(stock_no)'

  %REPORT{
    %ROW{  <img src="$(V1) "> <br><br><b>Original image format: $(V2)</b>%}
  %}
%}
```

그림 31. Net.Data 샘플 매크로 파일 (2/5)

```
%{ The following Content call depends on DB2MMEXPORT being set properly to
   point to the tmplobs directory under the web server's document root.  %}

%function (DTW_SQL) showVideoSQL(){
  select mmdbsys.comment(video), mmdbsys.content(video, mmdbsys.comment(video), 1),
         mmdbsys.format(video)
    from sobay_catalog
   where stock_no = '$(stock_no)'

  %REPORT{
    %ROW{ <a href="/tmplobs/$(V1)"><i><b> Play Video Clip></b></i></a>
       <br><br><b>Format: $(V3) <br>(Note: NT/Win95 may not come with
          a decompressor<br>for this video format.  OS/2 Warp does.)</br>
    %}
  %}
%}

%{ The following Content call depends on DB2MMEXPORT being set properly to
   point to the tmplobs directory under the web server's document root.  %}

%function (DTW_SQL) showAudioSQL(){
  select mmdbsys.comment(music), mmdbsys.content(music, mmdbsys.comment(music), 1),
         mmdbsys.format(music)
    from sobay_catalog
   where stock_no = '$(stock_no)'

  %REPORT{
      %ROW{ < a href="/tmplobs/$(V1) "<i><b>Play Audio Clip</b></i></a>
         <br><br><b>Format: $(V3)</b>
    %}
  %}
%}
```

그림 31. Net.Data 샘플 매크로 파일 (3/5)

```
%{ ------------------------------------------------------------------------- %}
%{ HTML sections                                                             %}
%{   E.g., http://<your server>/cgi-bin/db2www/extender.d2w/startHere        %}
%{ -------------------------------------------------------------------------%}
%{   E.g., http://
%{   E.g., http://
%HTML(startHere){
<html>
   <head><title>UDB Extenders Macro Sample: Simple Row Listing</title></head>
   <body bgcolor="#ffffff">
   <font color="#3300ff" size="3"><b>If no data appears below, you might need
    to run the UDB Extender sample programs <i>enable</i> and <i>populate</i>.
    This first HTML section of the extender.d2w macro simply retrieves all the
    traditional data for all the rows in the UDB Extenders' sample database.
    %if ( "$(SHOWSQL)" == "yes" || "$(SHOWSQL)" == "YES" )
    <br><br> By default, every page generated by this macro shows the SQL used
      to generate that page.  Here is the SQL statement for this page:
    %else
      <br>
    %endif
    </b></font>
     <br>@startHereSQL()
    <br><b>Click < a href="addThumbs"><i>here</i></a> to display thumbnails
    and links to image/audio/video data.</b>
   </body>
</html>
%}

%HTML(addThumbs){
<html>
   <head><title>UDB Extenders Macro Sample: Add Thumbnails</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>This page adds album cover thumbnails
    and links to display the multimedia content of the database.  To access
    the multimedia content:
    <ul>
      <li> Click on a thumbnail of a CD cover to view a full-size image
      <li> Click on a "video thumbnail" to view a video
      <li> Click on a "[Listen]" link to listen to an audio clip
    </ul>
    </b></font> @addThumbsSQL()
    <br><b>Click < a href="startHere"><i>here</i></a> to go back to the first page.</b>
  </body>
</html>
%}
```

그림 31. Net.Data 샘플 매크로 파일 (4/5)

```
%HTML(showCover){
<html>
   <head><title>UDB Extenders Macro Sample: Cover for item $(stock_no)</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>For this page, the macro gets a full-size cover
image, converting the image format to GIF so that a browser can show it:
    </b></font><br><br>
    <table width="400" border="2" bgcolor="#b1b1b1" cellpadding="5">
      <tr><td align=center> @showCoverSQL()
      <tr><td align=center> <b>Stock Number: $(stock_no)</b>
    </table>
    <br><b>Go <a href="addThumbs"><i>back</i></a>.</b>
 </body>
</html>
%}

%HTML(getVideo){
<html>
   <head><title>UDB Extenders Macro Sample: Video clip for item $(stock_no)</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>From this page, you can view a video clip:
    </b></font><br><br>
    <table width="400" border="2" bgcolor="#b1b1b1" cellpadding="5">
      <tr><td align=center> @showVideoSQL()
      <tr><td align=center> <b>Stock Number: $(stock_no)</b>
    </table>
    <br><b>Go <a href="addThumbs"><i>back</i></a>.</b>
 </body>
</html>
%}

%HTML(getAudio){
<html>
   <head><title>UDB Extenders Macro Sample: Audio clip for item $(stock_no)</title></head>
   <body bgcolor="#ffffff">
     <font color="#3300ff" size="3"><b>From this page, you can listen to an audio clip:
    </b></font><br><br>
    <table width="400" border="2" bgcolor="#b1b1b1" cellpadding="5">
      <tr><td align=center> @showAudioSQL()
      <tr><td align=center> <b>Stock Number: $(stock_no)</b>
    </table>
    <br><b>Go <a href="addThumbs"><i>back</i></a>.</b>
 <body>
</html>
%}
```

그림 31. Net.Data 샘플 매크로 파일 (5/5)

부록C. 주의사항

IBM은 이 책에서 논의된 제품, 서비스 또는 기능을 다른 나라에서는 제공하지 않을 수도 있습니다. 해당 지역에서 현재 사용 가능한 제품 및 서비스에 관해서는 해당 지역 IBM 영업 대표에 문의하십시오. IBM 제품, 프로그램 또는 서비스에 대한 어떠한 참조도 IBM 제품, 프로그램 또는 서비스만을 사용해야 함을 설명하거나 암시하지 않습니다. IBM의 지적 소유권을 침해하지 않는 기능상으로 동등한 제품, 프로그램 또는 서비스를 IBM 제품, 프로그램 또는 서비스 대신 사용할 수 있습니다. 단, IBM이 아닌 타사의 제품, 프로그램 또는 서비스의 조작에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에 나오는 특정 항목에 대한 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책의 제공으로 이러한 특허에 대한 사용권을 부여하는 것은 아닙니다. 사용권에 대한 문의는 아래 주소로 서면으로 하시기 바랍니다.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
지적 재산권부

2바이트(DBCS) 정보와 관련한 사용권에 대해서는, 해당 국가의 IBM 지적 재산권부로 문의하거나 다음 주소로 서면으로 문의하십시오.

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

다음 사항은 영국 또는 이러한 조항이 지역법에 위배되는 국가에서는 적용되지 않습니다. IBM은 이 책을 비침해성, 상품 가능성 또는 특정 목적에의 적합성 등(단, 이에 국한되지 않음), 명시적이거나 암시적인 어떠한 보증도 없이『있는 그대로』제공합니다. 특정 거래에 있어서 명시적이거나 암시적인 보증의 부인을 허용하지 않는 일부 국가에서는 이러한 사항이 적용되지 않을 수도 있습니다.

이 책에는 기술상으로 정확하지 않은 정보나 인쇄상의 오류가 있을 수 있습니다. 이 책의 내용은 주기적으로 변경되며, 이러한 변경사항은 이 책의 개정판에 통합됩니다. IBM은 이 책에 기술된 제품이나 프로그램을 아무런 통고 없이 언제라도 수정하거나 개선할 수 있습니다.

이 책에서 타사의 웹 사이트를 언급한 것은 단지 편의를 위해서일뿐이며 이들 웹 사이트를 추천하려는 의도는 아닙니다. 이들 웹 사이트의 데이터는 이 IBM 제품에 대한 데이터의 일부가 아니므로 이들 웹 사이트의 사용에 대한 책임은 사용자가 져야 합니다.

IBM은 독자가 제공한 정보를 적절한 방식으로 사용하거나 배포할 수 있으며, 제공한 독자는 이에 대해 책임을 지지 않습니다.

이 프로그램의 사용권을 보유한 고객이 (i) 별도로 작성된 프로그램과 기타 프로그램(이 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 공동 사용을 가능케 할 목적으로 이에 대한 정보를 필요로 하는 경우 다음 주소로 문의하시기 바랍니다.

135-270
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터

그러한 정보는 적절한 조항 및 조건 하에서 사용 가능하며, 경우에 따라서는 사용료를 지불해야 할 수도 있습니다.

이 책에 나와 있는 사용권 프로그램과 이러한 프로그램에 사용 가능한 모든 사용권 자료는 IBM이 IBM 고객 계약, IBM 프로그램 사용 계약 또는 그와 동등한 IBM과의 협약 하에 제공됩니다.

여기에 제시된 모든 성능 데이터는 통제된 환경에서 결정된 것입니다. 따라서, 다른 운영 환경에서 제시된 결과 값과 상당히 다를 수 있습니다. 몇몇 측정값은 개발 단계의 시스템에서 얻은 값일 수 있으며 따라서 일반적인 사용자 시스템에서 얻은 값과 다를 수 있습니다. 또한 몇몇 측정값은 보외법을 통해 측정된 값입니다. 실제 값과는 다를 수 있습니다. 이 책의 사용자는 사용자의 특정 환경에 맞게 적용 가능한 데이터를 검증해야 합니다.

타사 제품과 관련된 정보는 해당 제품의 공급자, 공개 발표 또는 기타 공개적으로 사용 가능한 소스에서 확보한 것입니다. IBM은 이들 제품을 검사하지 않았고 성능 상의 정확성, 호환성 또는 타사 제품과 관련된 기타 모든 주장을 확인할 수 없습니다. 타사 제품의 성능에 관한 문제는 해당 제품의 공급자에게 제기되어야 합니다.

IBM이 제시하는 방향 또는 의도에 관한 어떠한 언급도 특별한 통지 없이 변경될 수 있습니다.

이 정보에는 일상적인 업무 처리에 사용되는 데이터와 보고서의 예가 들어 있을 수 있습니다. 이러한 데이터와 보고서를 가능한 한 완전하게 설명하기 위해, 예에서는 개인, 회사, 브랜드 및 제품의 이름이 사용됩니다. 이 이름들은 모두 실제 이름이 아니며, 실제로 사용되는 이름과 주소와 유사한 이름이 있는 경우 이는 전적으로 우연에 의한 것입니다.

저작권 사용권:

이 정보에는 여러 운영 환경에서 다양한 프로그래밍 기법을 설명하기 위해 소스 언어로 작성된 샘플 응용프로그램이 들어 있을 수 있습니다. 이들 샘플 프로그램이 작성된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스에 맞는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로, IBM에 사용료를 지불하지 않고 원하는 형식으로 이들 샘플 프로그램을 복사, 수정 및 배포할 수 있습니다. 이들 예는 모든 조건에서 철저히 테스트된 것이 아닙니다. 따라서 IBM은 이들 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 암시할 수 없습니다.

이들 예제 프로그램 또는 모든 파생된 작업의 각각의 복사본이나 특정 부분은 다음과 같은 사용권 주의사항을 포함해야 합니다.

© (회사 이름) (연도). 이 코드 부분은 IBM Corp. 샘플 프로그램에서 발췌한 것입니다. © Copyright IBM Corp. _연도 입력_. All rights reserved.

등록상표

별표(*)로 표시된 다음의 용어는 미국 및/또는 기타 국가에서 사용되는 International Business Machines Corporation의 상표입니다.

ACF/VTAM
AISPO
AIX
AIX/6000
AIXwindows
AnyNet
APPN
AS/400
BookManager
CICS
C Set++
C/370
DATABASE 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extenders
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eNetwork
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
LAN DistanceMVS
MVS/ESA
MVS/XA
Net.Data
OS/2
OS/390
OS/400
PowerPC
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/DS
SQL/400
System/370
System/390
SystemView
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WIN-OS/2

다음 용어는 해당 회사의 상표 또는 등록 상표입니다.

Microsoft, Windows 및 Windows NT는 Microsoft Corporation의 상표 또는 등록 상표입니다.

Java 또는 모든 Java 관련 상표 및 로고 그리고 Solaris는 미국 및/또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.

Tivoli 및 NetView는 미국 및/또는 기타 국가에서 사용되는 Tivoli Systems Inc.의 상표입니다.

UNIX는 미국 및/또는 기타 국가에서 X/Open Company Limited를 통해 독점 인가된 등록 상표입니다.

두 개의 별표(**)가 붙은 기타 회사 이름, 제품 이름 또는 서비스 이름은 해당 회사의 상표이거나 서비스 표시입니다.

용어

가

객체 지향(object orientation). 일련의 조작과 데이터 값을 구성하는 오브젝트로 응용프로그램 내에서 제시될 수 있는 구체적 또는 추상적인 프로그래밍 접근 방법. 예를 들어, 문서는 문서 데이터와 파일링, 보내기, 프린트 등과 같이 문서에 대해 수행할 수 있는 조작으로 구성되는 문서 오브젝트에 의해 표현될 수 있습니다. 비디오 클립은 비디오 데이터와 비디오 클립의 재생 및 지정된 비디오 프레임을 발견하는 것과 같은 조작으로 구성되는 비디오 오브젝트에 의해 표현될 수 있습니다.

거친 정도(coarseness). 텍스처의 스케일을 측정하는 *텍스처*의 속성(페블(pebbles) 대 보울더 (boulders)).

관리 지원 테이블(administrative support table). 이미지, 오디오 및 비디오 오브젝트에 대한 사용자 요청을 처리하기 위해 DB2 Extender에서 사용되는 테이블 중 하나. 일부 관리 지원 테이블은 Extender용으로 사용 가능하게 된 사용자 테이블과 컬럼을 식별합니다. 기타 다른 관리 지원 테이블은 사용 가능한 컬럼 내의 오브젝트에 관한 속성 정보를 포함합니다. 또한 *메타데이터 테이블*이라고도 합니다.

구별 유형(distinct type). *사용자 정의 유형(user-defined type)*을 참조하십시오.

규모 확장(scaling). 저장 공간을 늘리고 성능을 향상시키기 위해 데이터베이스에 노드를 추가하는 것.

기가바이트(GB: gigabyte). 십억(10^9) 바이트. 메모리 용량을 언급할 때에는 1 073 741 824 바이트.

나

노드 그룹(nodegroup). 하나 이상의 데이터베이스 파티션으로 된 이름이 지정된 그룹.

노드(node). 데이터베이스 파티셔닝에서, 데이터베이스 파티션과 동의어.

다

다중 파티션 노드 그룹(multipartition nodegroup). 둘 이상의 *데이터베이스 파티션 서버*를 포함하는 노드 그룹.

대비(contrast). 패턴의 선명성을 나타내고 회색-레벨 히스토그램의 분산 함수인 *텍스처*의 속성.

대형 오브젝트(LOB: large object). 바이트 순서. 여기서 길이는 2GB까지 가능합니다. LOB은 세가지 유형이 가능합니다. *2진 대형 오브젝트*(BLOB), *문자 대형 오브젝트(CLOB)* 또는 *2바이트 문자 대형 오브젝트(DBCLOB)*.

데이터베이스 파티션 서버(database partition server). *데이터베이스 파티션*을 관리합니다. 데이터베이스 파티션 서버는 데이터베이스 관리 프로그램과 이 프로그램이 관리하는 데이터 및 시스템 자원 집합으로 구성됩니다. 일반적으로, 각 머신에는 하나의 데이터베이스 파티션 서버가 지정됩니다.

데이터베이스 파티션(database partition). 사용자 데이터, 색인, 구성 파일 및 트랜잭션 로그로 구성되는 데이터베이스의 부분. 간혹 노드나 데이터베이스 노드라고도 합니다.

디졸브(dissolve). 다음 비디오 프레임의 신호 강도를 증가시키면서 현 비디오 프레임 신호의 강도를 감소시키는 것.

마

메가바이트(MB: megabyte). 백만(10^6) 바이트. 메모리 용량을 언급할 때에는 1 048 576 바이트.

메타데이터 테이블(metadata table). *관리 지원 테이블(administrative support table)*을 참조하십시오.

문자 대형 오브젝트(CLOB: character large object). 단일 바이트 문자의 문자열. 여기서 문자열은 2GB까지 가능합니다. CLOB은 연관된 코드페이지를 가지고 있습니다. 단일 바이트 문자를 포함한 텍스트 오브젝트는 DB2 데이터베이스 내에 CLOB으로 저장됩니다.

바

방향성(directionality). 이미지가 선호하는 방향(grass 선호)을 가지고 있는지 또는 요철이 없는 오브젝트와 같은지(glass 선호)를 설명하는 *텍스처*의 속성.

분석(analyze). 이미지의 특성용 숫자 값을 계산하고 QBIC 카탈로그에 그 값을 추가하는 것.

비디오 색인(video index). 비디오 클립에서 Video Extender가 특정 *컷* 또는 프레임을 찾기 위해 사용하는 파일.

비디오 클립(video clip). 자료에서 필름이나 비디오 테입을 사용한 부분.

비디오(video). 볼 수 있는 레코드된 정보의 부분에 속하는 것.

사

사용자 정의 유형(UDT: user-defined type). 사용자에 의해 DB2에 정의된 데이터 유형. UDT는 한 LOB을 다른 것과 구별하기 위해 사용됩니다. 예를 들어, 한 UDT는 이미지 오브젝트용으로 생성되고 다른 것은 오디오 오브젝트용으로 작성될 수 있습니다. BLOB으로 저장되지만 이미지와 오디오 오브젝트는 BLOB에서 구별 유형으로 취급되며 서로 구분됩니다.

사용자 정의 함수(UDF: user-defined function). 사용자에 의해 DB2에 정의된 함수. 일단 정의되면 함수는 SQL 조회 및 비디오 오브젝트에서 사용 가능합니다. 예를 들어, UDF는 비디오의 압축 형식을 확보하기 위해서 또는 오디오의 샘플링율을 리턴하기 위해서 작성될 수 있습니다. 이것은 특정한 유형의 오브젝트의 동작을 정의하는 방법을 제공합니다.

색인 파일(index file). Video Extender에 의해 비디오 클립 내의 컷이나 개별 프레임을 찾기 위해 사용되는 색인 정보를 포함하는 파일.

스코어(score). 특성 값이 이미지 내용별 조회 내에서 지정된 것과 얼마나 유사한지를 반영하는 계산된 값. 숫자가 높을수록 더 유사합니다. 스코어는 이미지 내용별 조회의 결과를 정렬하는 데 사용됩니다.

스토리보드(storyboard). 비디오의 시각적 합계. Video Extender는 비디오 내의 컷의 대표인 비디오 프레임을 식별하고 저장하는 데 사용할 수 있는 특성을 포함합니다. 이러한 대표 프레임은 스토리보드를 빌드하는 데 사용될 수 있습니다.

썸네일(thumbnail). 모형 이미지.

아

오디오 클립(audio clip). 레코드된 오디오 자료의 부분.

오디오(audio). 들을 수 있는 레코드된 정보의 부분에 속하는 것.

오브젝트(object). 객체 지향 프로그래밍에서 데이터 및 그 데이터와 연관된 조작으로 구성되는 추상적 표현.

위치 색상(positional color). 지정된 영역의 이미지 내에 있는 픽셀의 *평균 색상* 값.

응용프로그램 프로그래밍 인터페이스(API: application programming interface).

(1) 운영 체제에서 제공되거나 별도로 주문 가능한 사용권 프로그램에 의해 제공되는 기능적 인터페이스. API는 고급 언어로 작성된 응용프로그램이 운영 체제나 사용권 프로그램의 특정 데이터나 기능을 사용할 수 있게 합니다.

(2) DB2에서, 오류 메시지 확보 API와 같은, 인터페이스 내의 함수.

(3) DB2 Extender는 사용자 정의 함수, 관리 조작, 표시 조작 및 비디오 장면 변경 검출 요청에 대한 API를 제공합니다.

이미지 내용별 조회(QBIC: Query by Image Content). *평균 색상*과 *텍스처*와 같은 시각적 특성으로 사용자가 이미지를 찾을 수 있도록 해 주는 Image Extender에 의해 제공되는 기능.

이미지(image). 그림의 전자적 표현.

인스턴스(instance). 논리적 DB2 Extender 서버 환경. 같은 워크스테이션에서 몇 개의 DB2 Extender 서버 인스턴스를 가질 수 있지만, 각각의 DB2 인스턴스에 대해 하나의 인스턴스만 허용됩니다. 인스턴스를 사용하여 다음을 수행할 수 있습니다.

제품 환경과 개발 환경을 구분합니다.

관련 정보를 특정 구성원 그룹으로 제한합니다.

자

장면 변경(scene change). 두 연속되는 프레임 사이에 중요한 차이가 있는 *비디오 클립*내의 한 지점. 예를 들어 카메라가 비디오를 레코드하는 동안 뷰 포인트를 바꿀 때 발생합니다.

조회 문자열(query string). QBIC 조회용 단수 또는 복수의 특성, 값 및 특성 가중치를 지정하는 문자열. 조회 문자열은 DB2 명령행에서 조회 내로 입력될 수 있습니다. 조회 오브젝트와 대조됩니다.

조회 오브젝트(query object). QBIC 조회용 단수 또는 복수의 특성, 값 및 특성 가중치를 지정하는 오브젝트. 오브젝트는 QBIC 조회 내에서 후속 사용을 위해 이름 지정되고 저장될 수 있습니다. 조회 문자열과 대조됩니다.

카

컷 카탈로그(shot catalog). 비디오 클립 내에서 컷용 시작 및 끝 프레임 번호와 같이 컷에 관한 데이터를 저장하는 데 사용되는 데이터베이스 테이블 또는 파일. 사용자는 SQL 조회를 통해 테이블의 뷰에 액세스하거나, 파일의 데이터에 액세스할 수 있습니다.

컷(shot). 두 장면 변경 사이의 프레임.

킬로바이트(KB: kilobyte). 천(10^3) 바이트. 메모리 용량을 언급할 때에는 1024바이트.

타

테라바이트(terabyte). 1조(10^{12}) 바이트. 10의 12승 바이트. 메모리 용량을 언급할 때에는, 1 099 511 627 776 바이트.

텍스처(texture). 이미지 내용별 조회에서 사용되는 특성의 하나. 조악성, 대비 또는 이미지의 방향성을 참조합니다.

트리거(trigger). 테이블이 변경될 때 일어나는 일련의 조치에 대한 정의. 트리거는 입력 데이터의 유효성을 확인하거나 새로 삽입된 행의 값을 자동으로 생성하거나 상호 참조 목적으로 다른 테이블을 읽거나 또는 감사 목적으로 다른 테이블에 쓰는 것과 같은 조치를 수행하기 위해 사용될 수 있습니다. 트리거는 종종 무결성 점검 및 업무 규칙 시행을 위해 사용됩니다.

특성(feature). *평균 색상*과 같은 이미지의 시각적 속성.

파

파일 참조 변수(file reference variable). 클라이언트 워크스테이션 상의 파일로/로부터 LOB을 옮기는 데 유용한 프로그래밍 변수.

파티션된 데이터베이스(partitioned database). 데이터베이스 파티션이 두 개 이상인 데이터베이스. 사용자 테이블의 데이터는 하나 이상의 데이터베이스 파티션에 위치될 수 있습니다. 테이블이 여러 파티션에 있을 때, 해당되는 행 중 일부는 하나의 파티션에 저장되고 나머지는 다른 파티션에 저장됩니다.

평균 색상(average color). 이미지의 픽셀 내에 포함된 색상 값의 평균으로 계산된 색상의 측정.

픽셀(pixel). 스크린이 표시할 수 있는 이미지의 가장 작은 구성 요소.

하

핸들(handle). 테이블에 있는 이미지, 오디오 또는 비디오를 표현하는 데 사용되는 Extender가 작성하는 문자열. 핸들은 사용자 테이블 및 *관리 지원 테이블* 내의 오브젝트용으로 저장됩니다. 이 방법으로, Extender는 관리 지원 테이블 내에 저장된 오브젝트에 관련된 정보와 사용자 테이블 내에 저장된 핸들을 링크할 수 있습니다.

호스트 변수(host variable). Embedded SQL문 내에서 참조될 수 있는 응용프로그램 내의 변수. 호스트 변수는 데이터베이스와 응용프로그램 작업 영역 간에 데이터를 전달하는 기본 메커니즘입니다.

환경 변수(environment variable). DB2 Extender의 운영 환경을 설정하고 환경 값의 기본값을 제공하기 위해 사용되는 변수.

히스토그램 색상(histogram color). 이미지 내의 구별되는 색상의 측정. *QBIC 카탈로그* 내에 분리되어 저장된 각 색상용 데이터.

숫자

2바이트 문자 대형 오브젝트(DBCLOB: double-byte character large object). 2바이트 문자의 문자열 또는 단일 바이트와 2바이트 문자의 조합, 여기서 문자열은 2GB까지 가능합니다. DBCLOB는 연관된 코드 페이지를 가지고 있습니다. 2바이트 문자를 포함한 텍스트 오브젝트는 DB2 데이터베이스 내에 DBCLOB으로 저장됩니다.

2진 대형 오브젝트(BLOB: binary large object). 길이가 2GB에 이르는 2진 문자열. 이미지, 오디오 및 비디오 오브젝트는 BLOB으로 DB2 내에 저장됩니다.

A

API. *응용프로그램 프로그래밍 인터페이스(API)*를 참조하십시오.

D

DB2 Extender. 일반적인 숫자 및 문자 데이터를 벗어나는 데이터 유형의 저장 및 검색에 사용하는 프로그램 중 하나나 프로그램 그룹(예: 이미지, 오디오 및 비디오 데이터와 복합 문서).

E

Extender. *DB2 Extender*를 참조하십시오.

L

LOB 위치 지정자(LOB locator). DB2 데이터베이스 내의 훨씬 큰 LOB을 참조하기 위해 프로그램 내에서 사용될 수 있는 호스트 변수 내에 저장된 작은 (4바이트) 값. LOB 위치 지정자를 사용하여 사용자는 LOB이 일반 호스트 변수에 저장된 것처럼 클라이언트 머신의 응용프로그램과 데이터베이스 서버 간에 LAB을 이동시킬 필요 없이 LOB을 조작할 수 있습니다.

Q

QBIC 카탈로그(QBIC catalog). 이미지의 시각적 특성에 관한 데이터를 보관하는 저장소.

U

UDF. *사용자 정의 함수(user-defined function)*를 참조하십시오.

UDT. *사용자 정의 유형(user-defined type)*을 참조하십시오.

색인

[아]

[자]

[차]

[카]

[타]

[파]

[하]

[숫자]

A

B

C

D

E

F

G

H

I

L

M

N

O

Q

R

S

T

U

V

W

IBM에 문의

기술적인 문제가 발생할 경우에는, DB2 고객 지원 부서에 문의하기 전에 *문제점 해결 안내서*에 제안된 조치를 검토하고 실행해 보십시오. 이것은 DB2 고객 지원 부서로 하여금 사용자를 보다 더 잘 지원할 수 있도록 사용자가 모을 수 있는 정보를 제공합니다.

DB2 Universal Database 제품에 대한 정보나 주문은 그 지역의 IBM 영업 대표나 공인 IBM 소프트웨어 재판매업자에게 문의하십시오.

미국 내에 거주하는 경우는 다음 번호 중 하나를 선택하여 전화할 수 있습니다.

- 고객 지원을 받으려면 1-800-237-5511
- 사용 가능한 서비스 옵션을 알려면 1-888-426-4343

제품 정보

미국 내에 거주하는 경우는 다음 번호 중 하나를 선택하여 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 Product and Service Technical Library에서는 자주 제기되는 질문, 수정 내용, 서적 및 최신 DB2 기술 정보에 액세스할 수 있습니다.

주: 이 정보는 영어로만 제공됩니다.

http://www.elink.ibmlink.ibm.com/pbl/pbl/

International Publications ordering 웹 사이트는 책을 주문하는 방법에 대한 정보를 제공합니다.

http://www.ibm.com/education/certify/

IBM 웹 사이트에서 기술 전문가 인증 프로그램은 DB2를 포함하여 다양한 IBM 제품에 대한 기술 전문가 인증 테스트 정보를 포함합니다.

ftp.software.ibm.com

anonymous로 로그온하십시오. /ps/products/db2 디렉토리에서, DB2와 많은 다른 제품에 관련한 데모, 수정 사항, 정보 및 도구 등을 찾을 수 있습니다.

comp.databases.ibm-db2, bit.listserv.db2-l

이 인터넷 뉴스그룹으로 사용자는 DB2 제품에 대한 자신의 사용 경험을 토론할 수 있습니다.

Compuserve에서: GO IBMDB2

이 명령을 입력하여 IBM DB2 Family 포럼에 액세스하십시오. 모든 DB2 제품이 이들 포럼을 통해 지원됩니다.

미국 이외의 지역에서 IBM에 연락하는 방법에 관한 정보는 *IBM Software Support Handbook*의 부록 A를 참조하십시오. 이 문서에 액세스하려면, 웹 페이지 http://www.ibm.com/support/로 가서 페이지 맨 아래에 있는 IBM Software Support Handbook 링크를 선택하십시오.

주: 일부 국가에서는 IBM 공인 딜러는 IBM 지원 센터 대신 해당 딜러 지원 부서에 연락해야 합니다.

IBM 한글 지원에 관한 설문

FAX : (02) 781-7778

보내 주시는 의견은 더 나은 고객 지원 체제를 위한 귀중한 자료가 됩니다.
독자 여러분의 좋은 의견을 기다립니다.

책 제목:IBM® DB2® Universal Database
Image, Audio, Video Extender
관리 및 프로그래밍 버전 7

책 번호:SA30-1043-00

성 명		직위/담당업무	
회 사 명		부 서 명	
주 소			
전화번호		팩스번호	
전자우편 주소			
사용중인 시스템	☐ 중대형 서버 ☐ UNIX 서버 ☐ PC 및 PC 서버		

1. IBM에서 제공하는 한글 책자와 영문 책자 중 어느 것을 더 좋아하십니까? 그 이유는 무엇입니까?
   ☐ 한글 책자 ☐ 영문 책자
   (이유:)

2. 본 책자와 해당 소프트웨어에서 사용된 한글 용어에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

3. 본 책자와 해당 소프트웨어에서 번역 품질에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

4. 본 책자의 인쇄 상태에 대한 귀하의 평가 점수는?
   ☐ 수 ☐ 우 ☐ 미 ☐ 양 ☐ 가

5. 한글 소프트웨어 및 책자가 지원되는 분야에 대해 귀하는 어떻게 생각하십니까?
   ☐ 한글 책자를 늘려야 함 ☐ 현재 수준으로 만족
   ☐ 그다지 필요성을 느끼지 않음

6. IBM은 인쇄물 형식(hardcopy)과 화면 형식(softcopy)의 두 종류로 책자를 제공합니다. 어느 형식을 더 좋아하십니까?
   ☐ 인쇄물 형식(hardcopy) ☐ 화면 형식(softcopy) ☐ 둘 다

☞ IBM 한글 지원 서비스에 대해 기타 제안사항이 있으시면 적어주십시오.

☺ 설문에 답해 주셔서 감사합니다.
귀하의 의견은 저희에게 매우 소중한 것이며, 고객 여러분들께 보다 좋은 제품을 제공해 드리기 위해 최선을 다하겠습니다.

SA30-1043-00

Spine information:

IBM® DB2® Universal Database

DB2 Image, Audio, Video Extender *버전 7*