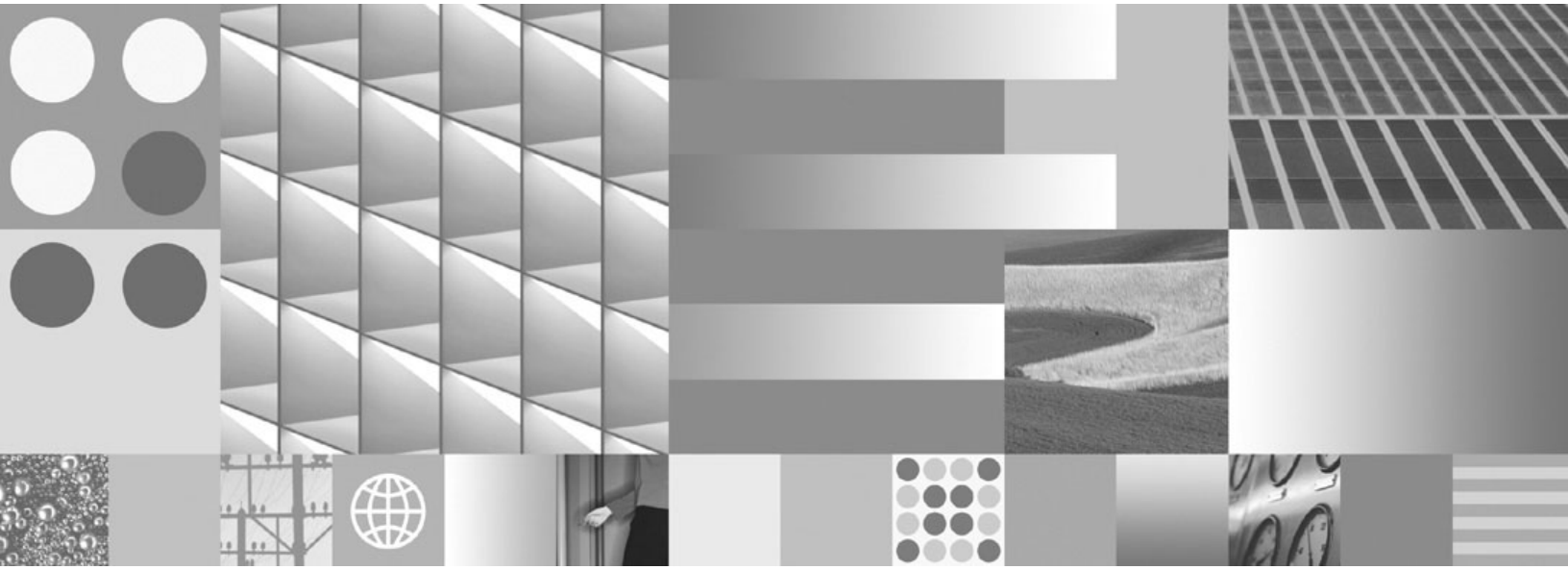
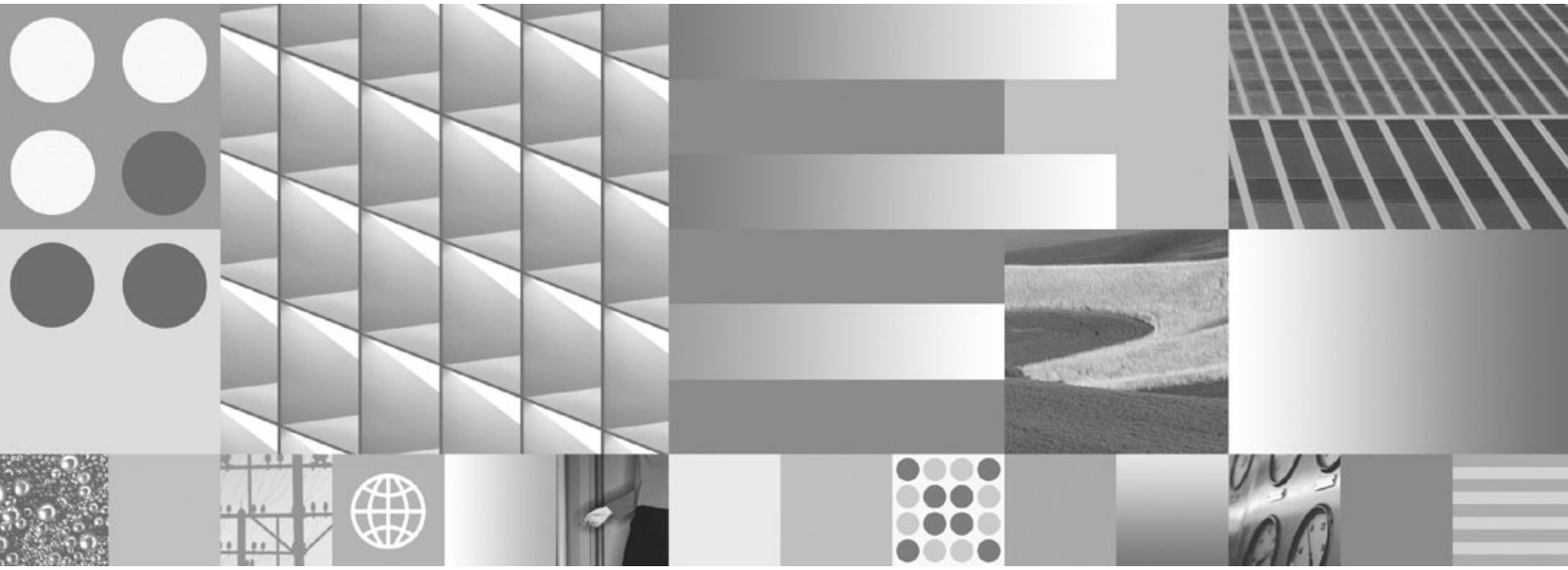
Linux, UNIX 및 Windows용
IBM DB2 9.7



관리 API 참조서

SA30-3958-00

Linux, UNIX 및 Windows용
IBM DB2 9.7



관리 API 참조서

SA30-3958-00

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 901 페이지의 부록 D『주의사항』의 정보를 읽으십시오.

개정판 주의사항

이 문서에는 IBM에서 소유하고 있는 정보가 있습니다. 이는 라이센스 계약에 따라 제공한 것이며 저작권의 보호를 받습니다. 이 책의 정보에는 제품 보증이 포함되지 않으며, 이 매뉴얼에서 제공된 어떠한 문장도 이와 같이 해석할 수 없습니다.

온라인으로 IBM 서적을 주문하거나 로컬 IBM 담당자를 통해 서적을 주문할 수 있습니다.

- 온라인으로 서적을 주문하려면 IBM Publications Center(www.ibm.com/shop/publications/order)로 이동하십시오.
- 로컬 IBM 담당자를 찾으려면 IBM Directory of Worldwide Contacts(www.ibm.com/planetwide)로 이동하십시오.

미국 또는 캐나다의 DB2 Marketing and Sales에서 DB2 서적을 주문하려면 1-800-IBM-4YOU (426-4968)로 전화하십시오.

IBM은 귀하가 IBM으로 보낸 정보를 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 사용하거나 배포할 수 있습니다.

목차

이 책에 대한 정보

이 책에서는 데이터베이스 관리 기능을 실행하는 API에 대한 사용 정보를 제공합니다. 다음 프로그래밍 언어로 작성된 응용프로그램에서 데이터베이스 관리 프로그램 API 호출을 사용하는 방법에 대한 자세한 정보를 제공합니다.

- C
- C++
- COBOL
- FORTRAN
- REXX

컴파일된 언어의 경우, 적절한 프리컴파일러를 사용하여 명령문을 처리해야 합니다. 프리컴파일러는 지원되는 모든 언어에서 제공됩니다.

이 책의 사용자

이 책의 사용자는 데이터베이스 관리 및 응용프로그램 프로그래밍에 대해 이해하고 다음에 대한 지식이 있다고 가정합니다.

- SQL
- C, C++, COBOL, FORTRAN 또는 REXX 프로그래밍 언어
- 응용프로그램 설계

이 책의 구성

이 책은 응용프로그램 개발 시 관리 API 사용에 필요한 참조 정보를 제공합니다.

이 책에서 설명하는 주요 주제 영역은 다음과 같습니다.

관리 API 및 데이터 구조 개요

- 제1장 『DB2® API』에는 관리 API, 내장 파일 및 샘플 프로그램을 설명한 테이블이 포함되어 있습니다.
- 제2장 『변경된 API 및 데이터 구조』에서는 변경된 데이터 구조 및 지원되거나 지원되지 않는 API를 나열한 테이블을 제공합니다.
- 제3장 『API 설명 구성 방법』에서는 API 설명을 구성하는 방법에 대해 설명하고 DB2 API 응용프로그램에 대한 내장 파일을 제공합니다.

API

- 제4장 『관리 API』는 DB2 관리 API를 알파벳 순으로 나열합니다.

- 제5장 『REXX에서 DB2 API 호출』에서는 REXX 응용프로그램에서 DB2 API를 호출하는 방법을 설명합니다.
- 제6장 『인다우트(Indoubt) 트랜잭션 관리 API』는 인다우트 트랜잭션에 대한 경험적 기능을 수행하는 도구 기록기에 제공된 API 세트에 대해 설명합니다.
- 제7장 『동시 액세스가 포함된 스레드 응용프로그램』에서는 스레드 응용프로그램에 사용할 수 있는 DB2 API에 대해 설명합니다.

플러그인 API

- 제8장 『데이터베이스 관리 사용자 정의를 위한 DB2 데이터베이스 시스템 플러그인』에서는 특정 데이터베이스 관리 기능을 사용자 정의하기 위해 사용자와 다른 업체가 사용할 수 있는 보안, 백업, 리스토어, 로그 아카이브 및 백업 이미지 플러그인 API에 대한 압축/압축 해제에 대해 설명합니다.

데이터 구조

- 제9장 『API에서 사용된 데이터 구조』에서는 API에서 사용된 데이터 구조에 대해 설명합니다.

부록

- 부록 A 『프리컴파일러 사용자 정의 API』는 다른 응용프로그램 개발 도구로 해당 제품 내에서 직접 DB2에 대한 프리컴파일러 지원을 구현할 수 있는 문서화된 API 세트 관련 정보를 얻을 수 있는 링크를 제공합니다.
- 부록 B 『DB2 로그 레코드』에서는 다양한 DB2 로그 레코드의 구조에 대해 설명합니다.

강조표시 규칙

이 책에서 사용되는 강조표시 규칙은 다음과 같습니다.

굵게	명령, 키워드 및 이름이 시스템에서 사전 정의된 기타 항목을 표시합니다.
기울임꼴	다음 중 하나를 표시합니다. • 사용자가 제공해야 되는 이름이나 값(변수) • 일반 강조 • 새 용어 소개 • 다른 정보 소스에 대한 참조

모노스페이스	다음 중 하나를 표시합니다. • 파일 및 디렉토리 • 명령 프롬프트나 창에 입력하도록 지시한 정보 • 특정 데이터 값의 예 • 시스템에서 표시할 수 있는 텍스트와 유사한 텍스트 예 • 시스템 메시지 예 • 프로그래밍 코드 샘플

제 1 부 DB2 API

다음 표에는 DB2 샘플이 포함된 DB2 API가 표시됩니다. 첫 번째 표에는 기능 범주, 각각의 내장 파일 및 해당 API를 보여주는 샘플 프로그램으로 그룹화된 DB2 API가 표시됩니다(내장 파일에 대한 자세한 정보는 테이블 뒤의 메모를 참조). 두 번째 표에는 C/C++ 샘플 프로그램이 표시되며 각 C/C++ 프로그램에 설명된 DB2 API가 표시됩니다. 세 번째 표에는 COBOL 샘플 프로그램 및 각 COBOL 프로그램에서 설명되는 DB2 API가 표시됩니다.

DB2 API, 내장 파일 및 샘플 프로그램
표 1

DB2 API를 사용하는 C/C++ 샘플 프로그램
11 페이지의 표 2

DB2 API를 사용하는 COBOL 샘플 프로그램
15 페이지의 표 3

표 1. DB2 API, 내장 파일 및 샘플 프로그램

API 유형	DB2 API	내장 파일	샘플 프로그램
데이터베이스 제어 API	89 페이지의 제 14 장 『db2DatabaseQuiesce - 데이터베이스 Quiesce』	db2ApiDf	n/a
데이터베이스 제어 API	97 페이지의 제 16 장 『db2DatabaseUnquiesce - 데이터베이스 Unquiesce』	db2ApiDf	n/a
데이터베이스 제어 API	93 페이지의 제 15 장 『db2DatabaseRestart - 데이터베이스 재시작』	db2ApiDf	C: dbconn.sqc C++: dbconn.sqC
데이터베이스 제어 API	493 페이지의 제 108 장『sqlecrea - 데이터베이스 작성』	sqlenv	C: dbcreate.c dbrecov.sqc dbsample.sqc C++: dbcreate.C dbrecov.sq COBOL: db_udcs.cbl dbconf.cbl ebcdicdb.cbl
데이터베이스 제어 API	491 페이지의 제 107 장『sqlecran - 데이터베이스 파티션 서버에 데이터베이스 작성』	sqlenv	n/a
데이터베이스 제어 API	513 페이지의 제 112 장『sqledrpd - 데이터베이스 삭제』	sqlenv	C: dbcreate.c C++: dbcreate.C COBOL: dbconf.cbl
데이터베이스 제어 API	511 페이지의 제 111 장『sqledpan - 데이터베이스 파티션 서버에서 데이터베이스 삭제』	sqlenv	n/a
데이터베이스 제어 API	db2DatabaseUpgrade - 이전 버전의 DB2 데이터베이스를 현재 릴리스로 업그레이드	db2ApiDf	C: dbupgrade.c C++: dbupgrade.C COBOL: dbupgrade.cbl
데이터베이스 제어 API	603 페이지의 제 148 장 『db2XaListIndTrans - 인다우트 트랜잭션 목록』	db2ApiDf	n/a

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
데이터베이스 제어 API	463 페이지의 제 101 장『sqle_activate_db - 데이터베이스 활성화』	sqlenv	n/a
데이터베이스 제어 API	467 페이지의 제 102 장『sqle_deactivate_db - 데이터베이스 비활성화』	sqlenv	n/a
데이터베이스 제어 API	461 페이지의 제 100 장『sqlcspqy - DRDA 인다우트(Indoubt) 트랜잭션 나열』	sqlxa	n/a
데이터베이스 제어 API	389 페이지의 제 74 장『db2SetWriteForDB - 데이터베이스에 대한 입출력 쓰기 일시중단 또는 다시 시작』	db2ApiDf	n/a
데이터베이스 제어 API	521 페이지의 제 116 장『sqlefrce - 사용자 및 응용프로그램에서 시스템 강제 중단』	sqlenv	C: dbconn.sqc dbsample.sqc instart.c C++: dbconn.sqC instart.C COBOL: dbstop.cbl
인스턴스 제어 API	213 페이지의 제 45 장『db2InstanceStart - 인스턴스 시작』	db2ApiDf	C: instart.c C++: instart.C
인스턴스 제어 API	219 페이지의 제 46 장『db2InstanceStop - 인스턴스 중지』	db2ApiDf	C: instart.c C++: instart.C
인스턴스 제어 API	209 페이지의 제 44 장『db2InstanceQuiesce - 인스턴스 Quiesce』	db2ApiDf	n/a
인스턴스 제어 API	223 페이지의 제 47 장『db2InstanceUnquiesce - 인스턴스 Unquiesce』	db2ApiDf	n/a
인스턴스 제어 API	479 페이지의 제 105 장『sqleatin - 인스턴스에 접속』	sqlenv	C: inattach.c utilapi.c C++: inattach.C utilapi.C COBOL: dbinst.cbl
인스턴스 제어 API	475 페이지의 제 104 장『sqleatcp - 인스턴스에 접속 및 암호 변경』	sqlenv	C: inattach.c C++: inattach.C COBOL: dbinst.cbl
인스턴스 제어 API	517 페이지의 제 114 장『sqledtin - 인스턴스에서 접속 해제』	sqlenv	C: inattach.c utilapi.c C++: inattach.C utilapi.C COBOL: dbinst.cbl
인스턴스 제어 API	537 페이지의 제 123 장『sqlegins - 현재 인스턴스 가져오기』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbinst.cbl
인스턴스 제어 API	419 페이지의 제 84 장『db2UtilityControl - 실행 중인 유틸리티의 우선순위 레벨 설정』	db2ApiDf	n/a
데이터베이스 관리 프로그램 및 데이터베이스 구성 API	71 페이지의 제 10 장『db2CfgGet - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 가져오기』	db2ApiDf	C: dbinfo.c dbrecov.sqc ininfo.c tscreate.sqc C++: dbinfo.C dbrecov.sqC ininfo.C tscreate.sqC
데이터베이스 관리 프로그램 및 데이터베이스 구성 API	75 페이지의 제 11 장『db2CfgSet - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 설정』	db2ApiDf	C: dbinfo.c dbrecov.sqc ininfo.c C++: dbinfo.C dbrecov.sqC ininfo.C
데이터베이스 관리 프로그램 및 데이터베이스 구성 API	51 페이지의 제 7 장『db2AutoConfig - 구성 어드바이저 액세스』	db2AuCfg	C: dbcfg.sqc C++: dbcfg.sqC
데이터베이스 관리 프로그램 및 데이터베이스 구성 API	57 페이지의 제 8 장『db2AutoConfigFreeMemory - db2AutoConfig API로 할당된 메모리 사용 가능화』	db2AuCfg	C: dbcfg.sqc C++: dbcfg.sqC

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
데이터베이스 모니터링 API	147 페이지의 제 33 장 『db2GetSnapshotSize - db2GetSnapshot API에 필요한 출력 버퍼 크기 계산』	db2ApiDf	n/a
데이터베이스 모니터링 API	41 페이지의 제 4 장 『db2AddSnapshotRequest - 스냅샷 요청 추가』	db2ApiDf	n/a
데이터베이스 모니터링 API	285 페이지의 제 58 장 『db2MonitorSwitches - 모니터 스위치 설정 가져오기 또는 갱신』	db2ApiDf	C: utilsnap.c C++: utilsnap.C
데이터베이스 모니터링 API	143 페이지의 제 32 장 『db2GetSnapshot - 데이터베이스 관리 프로그램 조작 상태에 대한 스냅샷 가져오기』	db2ApiDf	C: utilsnap.c C++: utilsnap.C
데이터베이스 모니터링 API	335 페이지의 제 68 장 『db2ResetMonitor - 데이터베이스 시스템 모니터 데이터 재설정』	db2ApiDf	n/a
데이터베이스 모니터링 API	81 페이지의 제 12 장 『db2ConvMonStream - 모니터 스트림을 버전 6 이전의 형식으로 변환』	db2ApiDf	n/a
데이터베이스 모니터링 API	199 페이지의 제 43 장 『db2Inspect - 구조적인 무결성을 위해 데이터베이스 검사』	db2ApiDf	n/a
데이터베이스 Health Monitor API	37 페이지의 제 2 장 『db2AddContact - 통지 메시지를 수신할 수 있는 문의처 추가』	db2ApiDf	n/a
데이터베이스 Health Monitor API	39 페이지의 제 3 장 『db2AddContactGroup - 통지 메시지를 수신할 수 있는 문의처 그룹 추가』	db2ApiDf	n/a
데이터베이스 Health Monitor API	107 페이지의 제 20 장 『db2DropContact - 통지 메시지를 수신할 수 있는 문의처 목록에서 문의처 제거』	db2ApiDf	n/a
데이터베이스 Health Monitor API	109 페이지의 제 21 장 『db2DropContactGroup - 통지 메시지를 수신할 수 있는 문의처 목록에서 문의처 그룹 제거』	db2ApiDf	n/a
데이터베이스 Health Monitor API	119 페이지의 제 23 장 『db2GetAlertCfg - Health 표시기의 경보 구성 설정 가져오기』	db2ApiDf	n/a
데이터베이스 Health Monitor API	125 페이지의 제 24 장 『db2GetAlertCfgFree - db2GetAlertCfg API로 할당된 메모리 사용 가능화』	db2ApiDf	n/a
데이터베이스 Health Monitor API	127 페이지의 제 25 장 『db2GetContactGroup - 통지 메시지를 수신할 수 있는 단일 문의처 그룹의 문의처 목록 가져오기』	db2ApiDf	n/a
데이터베이스 Health Monitor API	129 페이지의 제 26 장 『db2GetContactGroups - 통지 메시지를 수신할 수 있는 문의처 그룹 목록 가져오기』	db2ApiDf	n/a

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
데이터베이스 **Health Monitor API**	131 페이지의 제 27 장 『db2GetContacts - 통지 메시지를 수신할 수 있는 문의처 목록 가져오기』	db2ApiDf	n/a
데이터베이스 **Health Monitor API**	135 페이지의 제 29 장 『db2GetHealthNotificationList - Health 경보 통지를 수신할 수 있는 문의처 목록 가져오기』	db2ApiDf	n/a
데이터베이스 **Health Monitor API**	331 페이지의 제 67 장 『db2ResetAlertCfg - Health 표시기의 경보 구성 재설정』	db2ApiDf	n/a
데이터베이스 **Health Monitor API**	401 페이지의 제 79 장 『db2UpdateAlertCfg - Health 표시기의 경보 구성 설정 갱신』	db2ApiDf	n/a
데이터베이스 **Health Monitor API**	413 페이지의 제 81 장 『db2UpdateContact - 문의처 속성 갱신』	db2ApiDf	n/a
데이터베이스 **Health Monitor API**	415 페이지의 제 82 장 『db2UpdateContactGroup - 문의처 그룹의 속성 갱신』	db2ApiDf	n/a
데이터베이스 **Health Monitor API**	417 페이지의 제 83 장 『db2UpdateHealthNotificationList - Health 경보를 수신할 수 있는 문의처 목록 갱신』	db2ApiDf	n/a
데이터베이스 **Health Monitor API**	143 페이지의 제 32 장 『db2GetSnapshot - 데이터베이스 관리 프로그램 조작 상태에 대한 스냅샷 가져오기』	db2ApiDf	C: utilsnap.c C++: utilsnap.C
데이터베이스 **Health Monitor API**	147 페이지의 제 33 장 『db2GetSnapshotSize - db2GetSnapshot API에 필요한 출력 버퍼 크기 계산』	db2ApiDf	n/a
데이터베이스 **Health Monitor API**	137 페이지의 제 30 장 『db2GetRecommendations - 경보 상태의 Health 표시기를 해결하는 권장사항 가져오기』	db2ApiDf	n/a
데이터베이스 **Health Monitor API**	141 페이지의 제 31 장 『db2GetRecommendationsFree - db2GetRecommendations API로 할당된 메모리 사용 가능화』	db2ApiDf	n/a
데이터 이동 **API**	111 페이지의 제 22 장 『db2Export - 데이터베이스에서 데이터 익스포트』	sqlutil	C: tbmove.sqc C++: tbmove.sqC COBOL: expsamp.sqb impexp.sqb tload.sqb
데이터 이동 **API**	181 페이지의 제 42 장 『db2Import - 테이블, 계층 구조, 별칭 또는 뷰로 데이터 임포트』	db2ApiDf	C: dtformat.sqc tbmove.sqc C++: tbmove.sqC COBOL: expsamp.sqb impexp.sqb
데이터 이동 **API**	249 페이지의 제 56 장 『db2Load - 테이블에 데이터 로드』	db2ApiDf	C: dtformat.sqc tbload.sqc tbmove.sqc C++: tbmove.sqC
데이터 이동 **API**	275 페이지의 제 57 장 『db2LoadQuery - 로드 조작 상태 가져오기』	db2ApiDf	C: tbmove.sqc C++: tbmove.sqC COBOL: loadqry.sqb

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
복구 API	59 페이지의 제 9 장『db2Backup - 데이터베이스 또는 테이블 스페이스 백업』	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
복구 API	339 페이지의 제 69 장『db2Restore - 데이터베이스 또는 테이블 스페이스 리스토어』	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
복구 API	311 페이지의 제 65 장『db2Recover - 데이터베이스 리스토어 및 롤 포워드』	db2ApiDf	n/a
복구 API	357 페이지의 제 70 장『db2Rollforward - 데이터베이스 롤 포워드』	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
복구 API	169 페이지의 제 40 장『db2HistoryOpenScan - 실행기록 파일 스캔 시작』	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
복구 API	165 페이지의 제 39 장『db2HistoryGetEntry - 실행기록 파일의 다음 항목 가져오기』	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
복구 API	163 페이지의 제 38 장『db2HistoryCloseScan - 실행기록 파일 스캔 종료』	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
복구 API	289 페이지의 제 59 장『db2Prune - 사용 중인 로그 경로에서 실행기록 파일 항목 또는 로그 파일 삭제』	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
복구 API	175 페이지의 제 41 장『db2HistoryUpdate - 실행기록 파일 항목 갱신』	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
복구 API	47 페이지의 제 6 장『db2ArchiveLog - 사용 중인 로그 파일 아카이브』	db2ApiDf	n/a
고가용성 재해 복구 (HADR) API	153 페이지의 제 35 장『db2HADRStart - 고가용성 재해 복구(HADR) 조작 시작』	db2ApiDf	n/a
고가용성 재해 복구 (HADR) API	157 페이지의 제 36 장『db2HADRStop - 고가용성 재해 복구(HADR) 조작 중지』	db2ApiDf	n/a
고가용성 재해 복구 (HADR) API	159 페이지의 제 37 장『db2HADRTakeover - 데이터베이스를 고가용성 재해 복구(HADR) 기본 데이터베이스로 사용』	db2ApiDf	n/a
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	483 페이지의 제 106 장『sqlecadb - 시스템 데이터베이스 디렉토리에 데이터베이스 카탈로그』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	569 페이지의 제 136 장『sqleuncd - 시스템 데이터베이스 디렉토리에서 데이터베이스 카탈로그 해제』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	525 페이지의 제 117 장『sqlegdad - 데이터베이스 연결 서비스(DCS) 디렉토리에서 데이터베이스 카탈로그』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	529 페이지의 제 119 장『sqlegdel - 데이터베이스 연결 서비스(DCS) 디렉토리에서 데이터베이스 카탈로그 해제』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	507 페이지의 제 110 장 『sqledcgd - 시스템 또는 로컬 데이터베이스 디렉토리에서 데이터베이스 주석 변경』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dbcmt.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	105 페이지의 제 19 장 『db2DbDirOpenScan - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 시작』	db2ApiDf	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl dbcmt.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	101 페이지의 제 18 장 『db2DbDirGetNextEntry - 다음 시스템 또는 로컬 데이터베이스 디렉토리 항목 가져오기』	db2ApiDf	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl dbcmt.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	99 페이지의 제 17 장 『db2DbDirCloseScan - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 종료』	db2ApiDf	C: ininfo.c C++: ininfo.C COBOL: dbcat.cbl dbcmt.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	535 페이지의 제 122 장 『sqlegdsc - 데이터베이스 연결 서비스(DCS) 디렉토리 스캔 시작』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	533 페이지의 제 121 장 『sqlegdgt - 데이터베이스 연결 서비스(DCS) 디렉토리 항목 가져오기』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	527 페이지의 제 118 장 『sqlegdcl - 데이터베이스 연결 서비스(DCS) 디렉토리 스캔 종료』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	531 페이지의 제 120 장 『sqlegdge - 데이터베이스 연결 서비스(DCS) 디렉토리에서 특정 항목 가져오기』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: dcscat.cbl
데이터베이스 디렉토리 및 DCS 디렉토리 관리 API	409 페이지의 제 80 장 『db2UpdateAlternateServerForDB - 시스템 데이터베이스 디렉토리에서 데이터베이스 별명에 대한 대체 서버 갱신』	db2ApiDf	n/a
클라이언트/서버 관리 API	553 페이지의 제 130 장 『sqleqryc - 클라이언트 연결 설정 쿼리』	sqlenv	C: cli_info.c C++: cli_info.C COBOL: client.cbl
클라이언트/서버 관리 API	555 페이지의 제 131 장 『sqleqryi - 클라이언트 정보 쿼리』	sqlenv	C: cli_info.c C++: cli_info.C
클라이언트/서버 관리 API	561 페이지의 제 134 장 『sqlesetc - 클라이언트 연결 설정값 설정』	sqlenv	C: cli_info.c dbcfg.sqc dbmcon.sqc C++: cli_info.C dbcfg.sqC dbmcon.sqC COBOL: client.cbl
클라이언트/서버 관리 API	565 페이지의 제 135 장 『sqleseti - 클라이언트 정보 설정』	sqlenv	C: cli_info.c C++: cli_info.C
클라이언트/서버 관리 API	557 페이지의 제 132 장 『sqlesact - 어카운팅 문자열 설정』	sqlenv	COBOL: setact.cbl
클라이언트/서버 관리 API	85 페이지의 제 13 장 『db2DatabasePing - 네트워크 응답 시간을 테스트하기 위해 데이터베이스 Ping』	db2ApiDf	n/a
클라이언트/서버 관리 API	541 페이지의 제 125 장 『sqleisig - 신호 핸들러 설치』	sqlenv	COBOL: dbcmt.cbl

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
클라이언트/서버 관리 API	539 페이지의 제 124 장『sqleintr - 응용프로그램 요청 인터럽트』	sqlenv	n/a
LDAP(Lightweight Directory Access Protocol) 디렉토리 관리 API	233 페이지의 제 51 장『db2LdapRegister - LDAP 서버에 DB2 서버 등록』	db2ApiDf	n/a
LDAP(Lightweight Directory Access Protocol) 디렉토리 관리 API	243 페이지의 제 54 장『db2LdapUpdate - LDAP 서버에서 DB2 서버 속성 갱신』	db2ApiDf	n/a
LDAP(Lightweight Directory Access Protocol) 디렉토리 관리 API	231 페이지의 제 50 장『db2LdapDeregister - LDAP 서버에서 DB2 서버 및 카탈로그된 데이터베이스 등록 해제』	db2ApiDf	n/a
LDAP(Lightweight Directory Access Protocol) 디렉토리 관리 API	229 페이지의 제 49 장『db2LdapCatalogNode - LDAP 서버에서 노드 이름의 별명 제공』	db2ApiDf	n/a
LDAP(Lightweight Directory Access Protocol) 디렉토리 관리 API	241 페이지의 제 53 장『db2LdapUncatalogNode - LDAP 서버에서 노드 이름의 별명 삭제』	db2ApiDf	n/a
LDAP(Lightweight Directory Access Protocol) 디렉토리 관리 API	225 페이지의 제 48 장『db2LdapCatalogDatabase - LDAP 서버에 데이터베이스 등록』	db2ApiDf	n/a
LDAP(Lightweight Directory Access Protocol) 디렉토리 관리 API	239 페이지의 제 52 장『db2LdapUncatalogDatabase - LDAP 서버에서 데이터베이스 등록 해제』	db2ApiDf	n/a
LDAP(Lightweight Directory Access Protocol) 디렉토리 관리 API	247 페이지의 제 55 장『db2LdapUpdateAlternateServerForDB - LDAP 서버에서 데이터베이스의 대체 서버 갱신』	db2ApiDf	n/a
응용프로그램 프로그래밍 및 준비 API	425 페이지의 제 86 장『sqlaintp - 오류 메시지 가져오기』	sql	C: dbcfg.sqc utilapi.c C++: dbcfg.sqC utilapi.C COBOL: checkerr.cbl
응용프로그램 프로그래밍 및 준비 API	579 페이지의 제 141 장『sqlogstt - SQLSTATE 메시지 가져오기』	sql	C: utilapi.c C++: utilapi.C COBOL: checkerr.cbl
응용프로그램 프로그래밍 및 준비 API	541 페이지의 제 125 장『sqleisig - 신호 핸들러 설치』	sqlenv	COBOL: dbcmt.cbl
응용프로그램 프로그래밍 및 준비 API	539 페이지의 제 124 장『sqleintr - 응용프로그램 요청 인터럽트』	sqlenv	n/a
응용프로그램 프로그래밍 및 준비 API	427 페이지의 제 87 장『sqlaprep - 응용프로그램 프리컴파일』	sql	C: dbpkg.sqc C++: dbpkg.sqC
응용프로그램 프로그래밍 및 준비 API	421 페이지의 제 85 장『sqlabndx - 패키지 작성을 위해 응용프로그램 바인드』	sql	C: dbpkg.sqc dbsample.sqc C++: dbpkg.sqC

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
응용프로그램 프로그래밍 및 준비 API	431 페이지의 제 88 장『sqlarbnd - 패키지 리바인드』	sql	C: dbpkg.sqc C++: dbpkg.sqC COBOL: rebind.sqb
COBOL, FORTRAN 및 REXX 응용프로그램 특정 API	573 페이지의 제 138 장『sqlgaddr - 변수 주소 가져오기』	sqlutil	n/a
COBOL, FORTRAN 및 REXX 응용프로그램 특정 API	575 페이지의 제 139 장『sqlgdref - 주소 비참조』	sqlutil	n/a
COBOL, FORTRAN 및 REXX 응용프로그램 특정 API	577 페이지의 제 140 장『sqlgmcpy - 임의 메모리 영역에서 다른 영역으로 데이터 복사』	sqlutil	n/a
테이블 스페이스 및 테이블 관리 API	459 페이지의 제 99 장『sqlbtcq - 모든 테이블 스페이스 컨테이너에 대한 쿼리 데이터 가져오기』	sqlutil	C: dbrecov.sqc tsinfo.sqc C++: dbrecov.sqC tsinfo.sqC COBOL: tabscont.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	447 페이지의 제 95 장『sqlbotcq - 테이블 스페이스 컨테이너 쿼리 열기』	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabscont.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	439 페이지의 제 91 장『sqlbftcq - 테이블 스페이스 컨테이너의 행에 대한 쿼리 데이터 페치』	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabscont.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	435 페이지의 제 89 장『sqlbctcq - 테이블 스페이스 컨테이너 쿼리 닫기』	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabscont.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	455 페이지의 제 98 장『sqlbstsc - 테이블 스페이스 컨테이너 설정』	sqlutil	C: dbrecov.sqc C++: dbrecov.sqC COBOL: tabscont.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	445 페이지의 제 94 장『sqlbmtsq - 모든 테이블 스페이스에 대해 쿼리 데이터 가져오기』	sqlutil	C: dbrecov.sqc tsinfo.sqc C++: dbrecov.sqC tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	453 페이지의 제 97 장『sqlbstpq - 단일 테이블 스페이스에 대한 정보 가져오기』	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	449 페이지의 제 96 장『sqlbotsq - 테이블 스페이스 쿼리 열기』	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	441 페이지의 제 92 장『sqlbftpq - 테이블 스페이스의 행에 대해 쿼리 데이터 페치』	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	437 페이지의 제 90 장『sqlbctsq - 테이블 스페이스 쿼리 닫기』	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	443 페이지의 제 93 장『sqlbgtss - 테이블 스페이스 사용 통계 가져오기』	sqlutil	C: tsinfo.sqc C++: tsinfo.sqC COBOL: tabspace.sqb tspace.sqb
테이블 스페이스 및 테이블 관리 API	591 페이지의 제 145 장『sqluvqdp - 테이블에 대한 테이블 스페이스 Quiesce』	sqlutil	C: tbmove.sqc C++: tbmove.sqC COBOL: tload.sqb
테이블 스페이스 및 테이블 관리 API	371 페이지의 제 71 장『db2Runstats - 테이블 및 인덱스의 통계 갱신』	db2ApiDf	C: tbreorg.sqc C++: tbreorg.sqC COBOL: dbstat.sqb
테이블 스페이스 및 테이블 관리 API	319 페이지의 제 66 장『db2Reorg - 인덱스 또는 테이블 재구성』	db2ApiDf	C: tbreorg.sqc C++: tbreorg.sqC COBOL: dbstat.sqb

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
테이블 스페이스 및 테이블 관리 API	519 페이지의 제 115 장『sqlefmem - sqlbtcq 및 sqlbmtsq API로 할당된 메모리 사용 가능화』	sqlenv	C: dbrecov.sqc tsinfo.sqc C++: dbrecov.sqC tsinfo.sqC COBOL: tabscont.sqb tabspace.sqb tspace.sqb
노드 디렉토리 관리 API	503 페이지의 제 109 장『sqlectnd - 노드 디렉토리의 항목 카탈로그』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
노드 디렉토리 관리 API	571 페이지의 제 137 장『sqleuncn - 노드 디렉토리에서 항목 카탈로그 해제』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
노드 디렉토리 관리 API	551 페이지의 제 129 장『sqlenops - 노드 디렉토리 스캔 시작』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
노드 디렉토리 관리 API	547 페이지의 제 128 장『sqlengne - 다음 노드 디렉토리 항목 가져오기』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
노드 디렉토리 관리 API	545 페이지의 제 127 장『sqlencls - 노드 디렉토리 스캔 종료』	sqlenv	C: ininfo.c C++: ininfo.C COBOL: nodecat.cbl
노드 디렉토리 관리 API	409 페이지의 제 80 장『db2UpdateAlternateServerForDB - 시스템 데이터베이스 디렉토리에서 데이터베이스 별명에 대한 대체 서버 갱신』	db2ApiDf	n/a
Satellite 동기화 API	151 페이지의 제 34 장『db2GetSyncSession - Satellite 동기화 세션 ID 가져오기』	db2ApiDf	n/a
Satellite 동기화 API	293 페이지의 제 60 장『db2QuerySatelliteProgress - Satellite 동기화 세션 상태 가져오기』	db2ApiDf	n/a
Satellite 동기화 API	387 페이지의 제 73 장『db2SetSyncSession - Satellite 동기화 세션 설정』	db2ApiDf	n/a
Satellite 동기화 API	395 페이지의 제 76 장『db2SyncSatellite - Satellite 동기화 시작』	db2ApiDf	n/a
Satellite 동기화 API	397 페이지의 제 77 장『db2SyncSatelliteStop - Satellite 동기화 일시정지』	db2ApiDf	n/a
Satellite 동기화 API	399 페이지의 제 78 장『db2SyncSatelliteTest - Satellite 동기화 가능 여부 테스트』	db2ApiDf	n/a
로그 파일 읽기 API	295 페이지의 제 61 장『db2ReadLog - 로그 레코드 읽기』	db2ApiDf	C: dbrecov.sqc C++: dbrecov.sqC
로그 파일 읽기 API	301 페이지의 제 62 장『db2ReadLogNoConn - 데이터베이스에 연결하지 않고 데이터베이스 로그 읽기』	db2ApiDf	n/a
로그 파일 읽기 API	305 페이지의 제 63 장『db2ReadLogNoConnInit - 데이터베이스에 연결하지 않고 데이터베이스 로그 읽기 초기화』	db2ApiDf	n/a

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
로그 파일 읽기 API	309 페이지의 제 64 장 『db2ReadLogNoConnTerm - 데이터베이스에 연결하지 않고 데이터베이스 로그 읽기 종료』	db2ApiDf	n/a
인다우트(Indoubt) 트랜잭션 관리 API	603 페이지의 제 148 장 『db2XaListIndTrans - 인다우트 트랜잭션 목록』	db2ApiDf	n/a
인다우트(Indoubt) 트랜잭션 관리 API	609 페이지의 제 149 장 『sqlxhfrg - 트랜잭션 상태 무시』	sqlxa	n/a
인다우트(Indoubt) 트랜잭션 관리 API	611 페이지의 제 150 장 『sqlxphcm - 인다우트 트랜잭션 커미트』	sqlxa	n/a
인다우트(Indoubt) 트랜잭션 관리 API	613 페이지의 제 151 장 『sqlxphrl - 인다우트(Indoubt) 트랜잭션 롤백』	sqlxa	n/a
인다우트(Indoubt) 트랜잭션 관리 API	461 페이지의 제 100 장 『sqlcspqy - DRDA 인다우트(Indoubt) 트랜잭션 나열』	sqlxa	n/a
데이터베이스 동시 액세스 확보 API	617 페이지의 제 152 장 『sqleAttachToCtx - 컨텍스트에 접속』	sql	C: dbthrds.sqc C++: dbthrds.sqC
데이터베이스 동시 액세스 확보 API	619 페이지의 제 153 장 『sqleBeginCtx - 응용프로그램 컨텍스트에 접속 및 작성』	sql	C: dbthrds.sqc C++: dbthrds.sqC
데이터베이스 동시 액세스 확보 API	621 페이지의 제 154 장 『sqleDetachFromCtx - 컨텍스트에서 접속 해제』	sql	C: dbthrds.sqc C++: dbthrds.sqC
데이터베이스 동시 액세스 확보 API	623 페이지의 제 155 장 『sqleEndCtx - 응용프로그램 컨텍스트에서 접속 해제하여 관련된 메모리 사용 가능』	sql	n/a
데이터베이스 동시 액세스 확보 API	625 페이지의 제 156 장 『sqleGetCurrentCtx - 현재 컨텍스트 가져오기』	sql	n/a
데이터베이스 동시 액세스 확보 API	627 페이지의 제 157 장 『sqleInterruptCtx - 컨텍스트 인터럽트』	sql	n/a
데이터베이스 동시 액세스 확보 API	629 페이지의 제 158 장 『sqleSetTypeCtx - 응용프로그램 컨텍스트 유형 설정』	sql	C: dbthrds.sqc C++: dbthrds.sqC
데이터베이스 파티션 관리 API	471 페이지의 제 103 장 『sqleaddn - 파티션된 데이터베이스 환경에 데이터베이스 파티션 추가』	sqlenv	n/a
데이터베이스 파티션 관리 API	515 페이지의 제 113 장 『sqledrpn - 데이터베이스 파티션 서버 삭제 가능 여부 점검』	sqlenv	n/a
데이터베이스 파티션 관리 API	491 페이지의 제 107 장 『sqlecran - 데이터베이스 파티션 서버에 데이터베이스 작성』	sqlenv	n/a
데이터베이스 파티션 관리 API	511 페이지의 제 111 장 『sqledpan - 데이터베이스 파티션 서버에서 데이터베이스 삭제』	sqlenv	n/a
데이터베이스 파티션 관리 API	559 페이지의 제 133 장 『sqlesdeg - 최대 런타임 파티션 내 병렬 처리 레벨 또는 SQL문의 등급 설정』	sqlenv	C: ininfo.c C++: ininfo.C

표 1. DB2 API, 내장 파일 및 샘플 프로그램 (계속)

API 유형	DB2 API	내장 파일	샘플 프로그램
데이터베이스 파티션 관리 API	589 페이지의 제 144 장『sqlugtpi - 테이블 분산 정보 가져오기』	sqlutil	n/a
데이터베이스 파티션 관리 API	585 페이지의 제 143 장『sqlugrpn - 행에 대해 데이터베이스 파티션 서버 번호 가져오기』	sqlutil	n/a
기타 API	45 페이지의 제 5 장『db2AdminMsgWrite - 관리 복제 함수에 대한 로그 메시지 쓰기』	db2ApiDf	n/a
기타 API	601 페이지의 제 147 장『db2XaGetInfo - 자원 관리 프로그램에 대한 정보 가져오기』	sqlxa	n/a

주: 내장 파일 확장자는 프로그래밍 언어에 따라 달라집니다. C/C++ 내장 파일의 파일 확장자는 `.h`이며 Cobol 내장 파일의 확장자는 `.cbl`입니다. 내장 파일은 다음 디렉토리에 있습니다.

C/C++(UNIX®):

```
sqllib/include
```

C/C++(Windows®):

```
sqllib¥include
```

COBOL(UNIX):

```
sqllib/include/cobol_a

sqllib/include/cobol_i

sqllib/include/cobol_mf
```

COBOL(Windows):

```
sqllib¥include¥cobol_a

sqllib¥include¥cobol_i

sqllib¥include¥cobol_mf
```

표 2. DB2 API를 사용하는 C/C++ 샘플 프로그램

샘플 프로그램	포함된 API
cli_info.c, cli_info.C	• sqlesetc API - 클라이언트 연결 설정값 설정 • sqleseti API - 클라이언트 정보 설정 • sqleqryc API - 클라이언트 연결 설정값 쿼리 • sqleqryi API - 클라이언트 정보 쿼리
dbcfg.sqc, dbcfg.sqC	• db2AutoConfig API - 구성 어드바이저에 액세스 • db2AutoConfigFreeMemory API - db2AutoConfig API로 할당된 메모리 사용 가능화 • sqlesetc API - 클라이언트 연결 설정값 설정 • sqlaintp API - 오류 메시지 가져오기
dbconn.sqc, dbconn.sqC	• db2DatabaseRestart API - 데이터베이스 재시작 • sqlefrce API - 사용자 및 응용프로그램에서 시스템 강제 중단

표 2. DB2 API를 사용하는 C/C++ 샘플 프로그램 (계속)

샘플 프로그램	포함된 API
dbcreate.c, dbcreate.C	• sqlecrea API - 데이터베이스 작성 • sqledrpd API - 데이터베이스 삭제
dbinfo.c, dbinfo.C	• db2CfgGet API - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 가져오기 • db2CfgSet API - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 설정
dbmcon.sqc, dbmcon.sqC	• sqlesetc API - 클라이언트 연결 설정값 설정
dbmigrat.c, dbmigrat.C	• sqlemgdb API - 이전 버전의 DB2 데이터베이스를 현재 버전으로 이주
dbpkg.sqc, dbpkg.sqC	• sqlaprep API - 프리컴파일된 응용프로그램 • sqlabndx API - 패키지 작성을 위해 응용프로그램 바인드 • sqlarbnd API - 패키지 리바인드
dbrecov.sqc, dbrecov.sqC	• db2HistoryCloseScan API - 실행기록 파일 스캔 종료 • db2HistoryGetEntry API - 실행기록 파일의 다음 항목 가져오기 • db2HistoryOpenScan API - 실행기록 파일 스캔 시작 • db2HistoryUpdate API - 실행기록 파일 항목 갱신 • db2Prune API - 사용 중인 로그 경로에서 실행기록 파일 항목 또는 로그 파일 삭제 • db2CfgGet API - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 가져오기 • db2CfgSet API - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 설정 • sqlbmtsq API - 모든 테이블 스페이스에 대해 쿼리 데이터 가져오기 • sqlbstsc API - 테이블 스페이스 컨테이너 설정 • sqlbtcq API - 모든 테이블 스페이스 컨테이너에 대한 쿼리 데이터 가져오기 • sqlecrea API - 데이터베이스 작성 • sqledrpd API - 데이터베이스 삭제 • sqlefmem API - sqlbtcq 및 sqlbmtsq API로 할당된 메모리 사용 가능화 • db2Backup API - 데이터베이스 또는 테이블 스페이스 백업 • db2Restore API - 데이터베이스 또는 테이블 스페이스 리스토어 • db2ReadLog API - 비동기 읽기 로그 • db2ReadLogNoConn API - 데이터베이스에 연결하지 않고 로그 읽기 • db2Rollforward API - 데이터베이스 롤 포워드

표 2. DB2 API를 사용하는 C/C++ 샘플 프로그램 (계속)

샘플 프로그램	포함된 API
dbsample.sqc	• db2DatabaseRestart API - 데이터베이스 재시작 • sqlecrea API - 데이터베이스 작성 • sqlefrce API - 사용자 및 응용프로그램에서 시스템 강제 중단 • sqlabndx API - 패키지 작성을 위해 응용프로그램 바인드
dbthrds.sqc, dbthrds.sqC	• sqleAttachToCtx API - 컨텍스트에 접속 • sqleBeginCtx API - 응용프로그램 컨텍스트에 접속 및 작성 • sqleDetachFromCtx API - 컨텍스트에서 접속 해제 • sqleSetTypeCtx API - 응용프로그램 컨텍스트 유형 설정
dtformat.sqc	• db2Load API - 테이블에 데이터 로드 • db2Import API - 테이블, 계층 구조, 별칭 또는 뷰에 데이터 임포트
inattach.c, inattach.C	• sqleatcp API - 인스턴스에 접속 및 암호 변경 • sqleatin API - 인스턴스에 접속 • sqledtin API - 인스턴스에서 접속 해제

표 2. DB2 API를 사용하는 C/C++ 샘플 프로그램 (계속)

샘플 프로그램	포함된 API
ininfo.c, ininfo.C	• db2CfgGet API - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 가져오기 • db2CfgSet API - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 설정 • sqlegins API - 현재 인스턴스 가져오기 • sqlectnd API - 노드 디렉토리에서 항목 카탈로그 • sqlenops API - 노드 디렉토리 스캔 시작 • sqlengne API - 다음 노드 디렉토리 항목 가져오기 • sqlencls API - 노드 디렉토리 스캔 종료 • sqleuncn API - 노드 디렉토리에서 항목 카탈로그 해제 • sqlecadb API - 시스템 데이터베이스 디렉토리에서 데이터베이스 카탈로그 • db2DbDirOpenScan API - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 시작 • db2DbDirGetNextEntry API - 다음 시스템 또는 로컬 데이터베이스 디렉토리 항목 가져오기 • sqledcgd API - 시스템 또는 로컬 데이터베이스 디렉토리에서 데이터베이스 주석 변경 • db2DbDirCloseScan API - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 종료 • sqleuncd API - 시스템 데이터베이스 디렉토리에서 데이터베이스 카탈로그 해제 • sqlegdad API - 데이터베이스 연결 서비스(DCS) 디렉토리에서 데이터베이스 카탈로그 • sqlegdsc API - 데이터베이스 연결 서비스(DCS) 디렉토리 스캔 시작 • sqlegdge API - 데이터베이스 연결 서비스(DCS) 디렉토리에서 특정 항목 가져오기 • sqlegdgt API - 데이터베이스 연결 서비스(DCS) 디렉토리 항목 가져오기 • sqlegdcl API - 데이터베이스 연결 서비스(DCS) 디렉토리 스캔 종료 • sqlegdel API - 데이터베이스 연결 서비스(DCS) 디렉토리에서 데이터베이스 카탈로그 해제 • sqlesdeg API - 최대 런타임 파티션 내 병렬 처리 레벨 또는 SQL문 등급 설정
instart.c, instart.C	• sqlefrce API - 사용자 및 응용프로그램에서 시스템 강제 중단 • db2InstanceStart API - 인스턴스 시작 • db2InstanceStop API - 인스턴스 중지

표 2. DB2 API를 사용하는 C/C++ 샘플 프로그램 (계속)

샘플 프로그램	포함된 API
tbmove.sqc, tbmove.sqC	• db2Export API - 데이터베이스에서 데이터 익스포트 • db2Import API - 테이블, 계층 구조, 별칭 또는 뷰에 데이터 임포트 • sqluvqdp API - 테이블의 테이블 스페이스 Quiesce • db2Load API - 테이블에 데이터 로드 • db2LoadQuery API - 로드 조작 상태 가져오기
tbreorg.sqc, tbreorg.sqC	• db2Reorg API - 인덱스 또는 테이블 재구성 • db2Runstats API - 테이블 및 관련 인덱스 특성에 대한 통계 갱신
tscreate.sqc, tscreate.sqC	• db2CfgGet API - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 가져오기
tsinfo.sqc, tsinfo.sqC	• sqlbstpq API - 단일 테이블 스페이스에 대한 정보 가져오기 • sqlbgtss API - 테이블 스페이스 사용 통계 가져오기 • sqlbmtsq API - 모든 테이블 스페이스에 대해 쿼리 데이터 가져오기 • sqlefmem API - sqlbtcq 및 sqlbmtsq API로 할당된 메모리 사용 가능화 • sqlbotsq API - 테이블 스페이스 쿼리 열기 • sqlbftpq API - 테이블 스페이스의 행에 대한 쿼리 데이터 페치 • sqlbctsq API - 테이블 스페이스 쿼리 닫기 • sqlbtcq API - 모든 테이블 스페이스 컨테이너에 대한 쿼리 데이터 가져오기 • sqlbotcq API - 테이블 스페이스 컨테이너 쿼리 열기 • sqlbftcq API - 테이블 스페이스 컨테이너의 행에 대한 쿼리 데이터 페치 • sqlbctcq API - 테이블 스페이스 컨테이너 쿼리 닫기
utilapi.c, utilapi.C	• sqlaintp API - 오류 메시지 가져오기 • sqlogstt API - SQLSTATE 메시지 가져오기 • sqleatin API - 인스턴스에 접속 • sqledtin API - 인스턴스에서 접속 해제
utilsnap.c, utilsnap.C	• db2GetSnapshot API - 데이터베이스 관리 프로그램 조작 상태에 대한 스냅샷 가져오기 • db2MonitorSwitches API - 모니터 스위치 설정 가져오기 또는 갱신

표 3. DB2 API를 사용하는 COBOL 샘플 프로그램

샘플 프로그램	포함된 API
checkerr.cbl	• sqlaintp API - 오류 메시지 가져오기 • sqlogstt API - SQLSTATE 메시지 가져오기
client.cbl	• sqleqryc API - 클라이언트 연결 설정값 쿼리 • sqlesetc API - 클라이언트 연결 설정값 설정

표 3. DB2 API를 사용하는 COBOL 샘플 프로그램 (계속)

샘플 프로그램	포함된 API
db_udcs.cbl	• sqleatin API - 인스턴스에 접속 • sqlecrea API - 데이터베이스 작성 • sqledrpd API - 데이터베이스 삭제
dbcat.cbl	• sqlecadb API - 시스템 데이터베이스 디렉토리에서 데이터베이스 카탈로그 • db2DbDirCloseScan API - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 종료 • db2DbDirGetNextEntry API - 다음 시스템 또는 로컬 데이터베이스 디렉토리 항목 가져오기 • db2DbDirOpenScan API - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 시작 • sqleuncd API - 시스템 데이터베이스 디렉토리에서 데이터베이스 카탈로그 해제
dbcmt.cbl	• sqledcgd API - 시스템 또는 로컬 데이터베이스 디렉토리에서 데이터베이스 주석 변경 • db2DbDirCloseScan API - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 종료 • db2DbDirGetNextEntry API - 다음 시스템 또는 로컬 데이터베이스 디렉토리 항목 가져오기 • db2DbDirOpenScan API - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 시작 • sqleisig API - 신호 핸들러 설치
dbinst.cbl	• sqleatcp API - 인스턴스에 접속 및 암호 변경 • sqleatin API - 인스턴스에 접속 • sqledtin API - 인스턴스에서 접속 해제 • sqlegins API - 현재 인스턴스 가져오기
dbstat.sqb	• db2Reorg API - 인덱스 또는 테이블 재구성 • db2Runstats API - 테이블 및 관련 인덱스 특성에 대한 통계 갱신
dcscat.cbl	• sqlegdad API - 데이터베이스 연결 서비스(DCS) 디렉토리에서 데이터베이스 카탈로그 • sqlegdcl API - 데이터베이스 연결 서비스(DCS) 디렉토리 스캔 종료 • sqlegdel API - 데이터베이스 연결 서비스(DCS) 디렉토리에서 데이터베이스 카탈로그 해제 • sqlegdge API - 데이터베이스 연결 서비스(DCS) 디렉토리에서 특정 항목 가져오기 • sqlegdgt API - 데이터베이스 연결 서비스(DCS) 디렉토리 항목 가져오기 • sqlegdsc API - 데이터베이스 연결 서비스(DCS) 디렉토리 스캔 시작

표 3. DB2 API를 사용하는 COBOL 샘플 프로그램 (계속)

샘플 프로그램	포함된 **API**
ebcdicdb.cbl	• sqleatin API - 인스턴스에 접속 • sqlecrea API - 데이터베이스 작성 • sqledrpd API - 데이터베이스 삭제
expsamp.sqb	• db2Export API - 데이터베이스에서 데이터 익스포트 • db2Import API - 테이블, 계층 구조, 별칭 또는 뷰에 데이터 임포트
impexp.sqb	• db2Export API - 데이터베이스에서 데이터 익스포트 • db2Import API - 테이블, 계층 구조, 별칭 또는 뷰에 데이터 임포트
loadqry.sqb	• db2LoadQuery API - 로드 조작 상태 가져오기
migrate.cbl	• sqlemgdb API - 이전 버전의 DB2 데이터베이스를 현재 버전으로 이주
nodecat.cbl	• sqlectnd API - 노드 디렉토리에서 항목 카탈로그 • sqlencls API - 노드 디렉토리 스캔 종료 • sqlengne API - 다음 노드 디렉토리 항목 가져오기 • sqlenops API - 노드 디렉토리 스캔 시작 • sqleuncn API - 노드 디렉토리에서 항목 카탈로그 해제
rebind.sqb	• sqlarbnd API - 패키지 리바인드
tabscont.sqb	• sqlbctcq API - 테이블 스페이스 컨테이너 쿼리 닫기 • sqlbftcq API - 테이블 스페이스 컨테이너의 행에 대한 쿼리 데이터 페치 • sqlbotcq API - 테이블 스페이스 컨테이너 쿼리 열기 • sqlbtcq API - 모든 테이블 스페이스 컨테이너에 대한 쿼리 데이터 가져오기 • sqlefmem API - sqlbtcq 및 sqlbmtsq API로 할당된 메모리 사용 가능화
tabspace.sqb	• sqlbctsq API - 테이블 스페이스 쿼리 닫기 • sqlbftpq API - 테이블 스페이스의 행에 대한 쿼리 데이터 페치 • sqlbgtss API - 테이블 스페이스 사용 통계 가져오기 • sqlbmtsq API - 모든 테이블 스페이스에 대해 쿼리 데이터 가져오기 • sqlbotsq API - 테이블 스페이스 쿼리 열기 • sqlbstpq API - 단일 테이블 스페이스에 대한 정보 가져오기 • sqlefmem API - sqlbtcq 및 sqlbmtsq API로 할당된 메모리 사용 가능화
tload.sqb	• db2Export API - 데이터베이스에서 데이터 익스포트 • sqluvqdp API - 테이블의 테이블 스페이스 Quiesce

표 3. DB2 API를 사용하는 COBOL 샘플 프로그램 (계속)

샘플 프로그램	포함된 API
tspace.sqb	• sqlbctcq API - 테이블 스페이스 컨테이너 쿼리 닫기 • sqlbctsq API - 테이블 스페이스 쿼리 닫기 • sqlbftcq API - 테이블 스페이스 컨테이너의 행에 대한 쿼리 데이터 페치 • sqlbftpq API - 테이블 스페이스의 행에 대한 쿼리 데이터 페치 • sqlbgtss API - 테이블 스페이스 사용 통계 가져오기 • sqlbmtsq API - 모든 테이블 스페이스에 대해 쿼리 데이터 가져오기 • sqlbotcq API - 테이블 스페이스 컨테이너 쿼리 열기 • sqlbotsq API - 테이블 스페이스 쿼리 열기 • sqlbstpq API - 단일 테이블 스페이스에 대한 정보 가져오기 • sqlbstsc API - 테이블 스페이스 컨테이너 설정 • sqlbtcq API - 모든 테이블 스페이스 컨테이너에 대한 쿼리 데이터 가져오기 • sqlefmem API - sqlbtcq 및 sqlbmtsq API로 할당된 메모리 사용 가능화
setact.cbl	• sqlesact API - 어카운팅 문자열 설정

제 2 부 API 및 데이터 구조 변경

표 4. 이전 레벨 지원 API 및 데이터 구조

API 또는 데이터 구조(버전)	설명 이름	새 API 또는 데이터 구조(버전)
sqlbftsq (V2)	테이블 스페이스 쿼리 페치	sqlbftpq (V5)
sqlbstsq (V2)	단일 테이블 스페이스 쿼리	sqlbstpq (V5)
sqlbtsq (V2)	테이블 스페이스 쿼리	sqlbmtsq (V5)
sqlectdd (V2)	데이터베이스 카탈로그	sqlecadb (V5)
sqledosd (V8.1)	데이터베이스 디렉토리 스캔 열기	db2DbDirOpenScan (V8.2)
sqledgne (V8.1)	다음 데이터베이스 디렉토리 항목 가져오기	db2DbDirGetNextEntry (V8.2)
sqledcls (V8.1)	데이터베이스 디렉토리 스캔 닫기	db2DbDirCloseScan (V8.2)
sqlepstart (V5)	데이터베이스 관리 프로그램 시작	db2InstanceStart (V8)
sqlepstp (V5)	데이터베이스 관리 프로그램 중지	db2InstanceStop (V8)
sqlepstr (V2)	데이터베이스 관리 프로그램 시작(DB2 Parallel Edition 버전 1.2)	db2InstanceStart (V8)
sqlestar (V2)	데이터베이스 관리 프로그램 시작(DB2 버전 2)	db2InstanceStart (V8)
sqlestop (V2)	데이터베이스 관리 프로그램 중지	db2InstanceStop (V8)
sqlerstd (V5)	데이터베이스 재시작	db2DatabaseRestart (V6)
sqlfddb (V7)	데이터베이스 구성 디폴트 가져오기	db2CfgGet (V8)
sqlfdsys (V7)	데이터베이스 관리 프로그램 구성 디폴트 가져오기	db2CfgGet (V8)
sqlfrdb (V7)	데이터베이스 구성 재설정	db2CfgSet (V8)
sqlfrsys (V7)	데이터베이스 관리 프로그램 구성 재설정	db2CfgSet (V8)
sqlfudb (V7)	데이터베이스 구성 갱신	db2CfgSet (V8)
sqlfusys (V7)	데이터베이스 관리 프로그램 구성 갱신	db2CfgSet (V8)
sqlfxdb (V7)	데이터베이스 구성 가져오기	db2CfgGet (V8)
sqlfxsys (V7)	데이터베이스 구성 가져오기	db2CfgGet (V8)
sqlmon (V6)	모니터 스위치 가져오기/갱신	db2MonitorSwitches (V7)
sqlmonss (V5)	스냅샷 가져오기	db2GetSnapshot (V6)
sqlmonsz (V6)	sqlmonss() 출력 버퍼에 필요한 크기 계산	db2GetSnapshotSize (V7)
sqlmrset (V6)	모니터 재설정	db2ResetMonitor (V7)
sqlubkp (V5)	데이터베이스 백업	db2Backup (V8)
sqlubkup (V2)	데이터베이스 백업	db2Backup (V8)
sqluexpr	익스포트	db2Export (V8)
sqlugrpi (V2)	행 파티셔닝 정보 가져오기(DB2 Parallel Edition 버전 1.x)	sqlugrpn (V5)
sqluhcls (V5)	복구 실행기록 파일 스캔 닫기	db2HistoryCloseScan (V6)
sqluhget (V5)	실행기록 파일에서 DDL 정보 검색	db2HistoryGetEntry (V6)
sqluhgne (V5)	다음 복구 실행기록 파일 항목 가져오기	db2HistoryGetEntry (V6)

표 4. 이전 레벨 지원 API 및 데이터 구조 (계속)

API 또는 데이터 구조(버전)	설명 이름	새 API 또는 데이터 구조(버전)
sqluhops (V5)	복구 실행기록 파일 스캔 열기	db2HistoryOpenScan (V6)
sqluhprn (V5)	복구 실행기록 파일 프룬(prune)	db2Prune (V6)
sqluhupd (V5)	복구 실행기록 파일 갱신	db2HistoryUpdate (V6)
sqluimpr	임포트	db2Import (V8)
sqluload (V7)	로드	db2Load (V8)
sqluqry (V5)	쿼리 로드	db2LoadQuery (V6)
sqlureot (V7)	테이블 재구성	db2Reorg (V8)
sqlurestore (V7)	데이터베이스 리스토어	db2Restore (V8)
sqlurlog (V7)	비동기 읽기 로그	db2ReadLog (V8)
sqluroll (V7)	데이터베이스 롤 포워드	db2Rollforward (V8)
sqlursto (V2)	데이터베이스 리스토어	sqlurst (V5)
sqlustat (V7)	Runstats	db2Runstats (V8)
sqlxhcom (V2)	인다우트 트랜잭션 커미트	sqlxphcm (V5)
sqlxhqry (V2)	인다우트 트랜잭션 목록	sqlxphqr (V5)
sqlxhrol (V2)	인다우트(Indoubt) 트랜잭션 롤백	sqlxphrl (V5)
SQLB-TBSQRY-DATA (V2)	테이블 스페이스 데이터 구조	SQLB-TBSPQRY-DATA (V5)
SQLE-START-OPTIONS (V7)	데이터베이스 관리 프로그램 데이터 구조 시작	db2StartOptionsStruct (V8)
SQLEDBSTOPOPT (V7)	데이터베이스 관리 프로그램 데이터 구조 시작	db2StopOptionsStruct (V8)
SQLEDBSTRTOPT (V2)	데이터베이스 관리 프로그램 데이터 구조 시작(DB2 Parallel Edition 버전 1.2)	db2StartOptionsStruct (V8)
SQLEDINFO (v8.1)	다음 데이터베이스 디렉토리 항목 데이터 구조 가져오기	db2DbDirInfo (V8.2)
SQLUEXPT-OUT	출력 익스포트 구조	db2ExportOut (V8.2)
SQLUHINFO and SQLUHADM (V5)	실행기록 파일 데이터 구조	db2HistData (V6)
SQLUIMPT-IN	입력 임포트 구조	db2ImportIn (V8.2)
SQLUIMPT-OUT	출력 임포트 구조	db2ImportOut (V8.2)
SQLULOAD-IN (V7)	입력 로드 구조	db2LoadIn (V8)
SQLULOAD-OUT (V7)	출력 로드 구조	db2LoadOut (V8)
db2DbDirInfo (V8.2)	다음 데이터베이스 디렉토리 항목 데이터 구조 가져오기	db2DbDirInfoV9 (V9.1)
db2DbDirNextEntryStruct (V8.2)	다음 데이터베이스 디렉토리 항목 데이터 구조 가져오기	db2DbDirNextEntryStructV9 (V9.1)
db2gDbDirNextEntryStruct (V8.2)	다음 데이터베이스 디렉토리 항목 데이터 구조 가져오기	db2gDbDirNextEntryStrV9 (V9.1)

표 5. 이전 레벨을 지원하지 않는 API 및 데이터 구조

이름	설명 이름	교체 API, 데이터 구조 또는 테이블 함수
sqlufrol/sqlgfrol	데이터베이스 롤 포워드(DB2 버전 1.1)	db2Rollforward
sqluprfw	데이터베이스 롤 포워드(DB2 Parallel Edition 버전 1.x)	db2Rollforward
sqlurfwd/sqlgrfwd	데이터베이스 롤 포워드(DB2 버전 1.2)	db2Rollforward
sqlurllf/sqlgrfwd	데이터베이스 롤 포워드(DB2 버전 2)	db2Rollforward
sqlxphqr	인다우트(Indoubt) 트랜잭션 목록	db2XaListIndTrans
SQLXA-RECOVER	트랜잭션 API 구조	db2XaRecoverStruct
sqluadau	현재 사용자 권한 가져오기	AUTH_LIST_ AUTHORITIES_ FOR_AUTHID 테이블 함수
SQL-AUTHORIZATIONS	권한 부여 구조	불필요

제 3 부 API 및 응용프로그램 동작에 영향을 주는 로그 시퀀스 번호 변경사항

로그 시퀀스 번호(LSN) *크기*가 6바이트에서 8바이트로 증가되었습니다. 새 데이터 유형인 db2LSN이 새 LSN *크기* 지원을 위해 작성되었습니다.

LSN 필드를 포함하는 입력 및 출력 구조가 포함된 많은 API가 새 db2LSN 데이터 유형을 사용하도록 갱신되었습니다. 갱신된 API는 아래에서 설명됩니다. 최신 버전의 API가 사용되도록 하려면 새 DB2 버전 번호 상수를 버전 번호 API 입력 매개변수를 통해 API에 전달해야 합니다.

LSN *크기*가 늘고 새 버전의 API가 작성됨으로 인해 영향을 받는 API가 현재 클라이언트 서버 구성 및 사용 중인 응용프로그램 레벨에 따라 변경됩니다. 특정 시나리오에서는 API 기능 확장에 관련하여 주의해야 하는 제한사항 및 조건이 있습니다.

주: 아래 표시된 이전 API의 경우 이 API에는 새 버전이 포함되지 않기 때문에 해당 구조에서는 db2LSN 데이터 유형을 사용하지 않지만, 사용 중인 클라이언트 서버 구성이 여전히 해당 동작에 영향을 줍니다.

db2ReadLog API

db2ReadLog API의 동작은 클라이언트 서버 구성에 따라 변경됩니다.

다음은 가능한 구성입니다.

- 최신 클라이언트 및 최신 서버: 기존 응용프로그램이 이전 버전의 db2ReadLog API를 사용하는 경우 오류 메시지 SQL2032N이 리턴됩니다. 이전 버전의 db2ReadLog API 호출은 *지원되지 않습니다*. 이런 응용프로그램은 최신 버전의 db2ReadLog API를 사용하도록 갱신해야 합니다. 최신 버전의 db2ReadLog API를 사용하도록 개발된 응용프로그램을 사용 중인 경우에는 API 함수에 제한사항이 없습니다.

  주: 클라이언트와 서버가 다른 엔디안 플랫폼에 있는 경우 db2ReadLogInfoStruct 출력 구조의 모든 원시 데이터 유형 필드는 LSN 필드를 *포함하여* 바이트 반전됩니다.
- 이전 클라이언트 및 최신 서버: 이전 구성을 사용하는 경우 이전 레벨의 db2ReadLog API 호출은 지원되지 않으며 오류 메시지 SQL2032N이 리턴됩니다. 계속하려면 클라이언트는 업그레이드되어야 하며 모든 기존 응용프로그램은 최신 버전의 db2ReadLog API를 사용하도록 갱신되어야 합니다.
- 최신 클라이언트 및 이전 서버: 새로 개발된 응용프로그램에서 최신 버전의 db2ReadLog API가 호출되거나, 또는 기존 응용프로그램에서 이전 버전의

db2ReadLog API가 호출되는지에 상관 없이 API 함수에는 제한사항이 없습니다. 그러나 두 경우 모두 로그 버퍼에서 리턴된 로그 레코드는 이전 LSN 형식으로 이전 로그 레코드를 표시합니다.

주: 클라이언트와 서버가 다른 엔디안 플랫폼에 있는 경우 db2ReadLogInfoStruct 출력 구조의 모든 원시 데이터 유형 필드는 LSN 필드를 *제외하여* 바이트 반전됩니다.

db2ReadLogNoConn API

db2ReadLogNoConn API의 동작은 클라이언트 서버 구성에 따라 변경됩니다.

다음은 가능한 구성입니다.

- 최신 클라이언트 및 최신 서버: 기존 응용프로그램이 이전 버전의 db2ReadLogNoConn API를 사용하는 경우 오류 메시지 SQL2032N이 리턴됩니다. 이전 버전의 db2ReadLogNoConn API 호출은 *지원되지 않습니다*. 이런 응용프로그램은 최신 버전의 db2ReadLogNoConn API를 사용하도록 갱신해야 합니다. 최신 버전의 db2ReadLogNoConn API를 사용하도록 개발된 응용프로그램을 사용 중인 경우에는 API 함수에 제한사항이 없습니다.

db2HistoryGetEntry API

db2HistoryGetEntry API의 동작은 클라이언트 서버 구성에 따라 변경됩니다.

다음은 가능한 구성입니다.

- 최신 클라이언트 및 최신 서버: 기존 응용프로그램이 이전 버전의 db2HistoryGetEntry API를 사용하고 실행기록 파일 항목의 LSN이 이전 레벨 API 출력 구조의 『oLastLSN』 필드에서 포함할 수 있는 것보다 큰 경우 오류 메시지 SQL2032N이 리턴됩니다. 이 동작을 수정하려면 이런 응용프로그램이 최신 버전의 db2HistoryGetEntry API를 사용하도록 갱신해야 합니다. 최신 버전의 db2HistoryGetEntry API를 사용하도록 개발된 응용프로그램을 사용 중인 경우에는 API 함수에 제한사항이 없습니다.
- 이전 클라이언트 및 최신 서버: 이전 버전의 db2HistoryGetEntry API를 사용하고 실행기록 파일 항목의 LSN이 이전 레벨 API 출력 구조의 『oLastLSN』 필드에서 포함할 수 있는 것보다 큰 경우 오류 메시지 SQL2032N이 리턴됩니다. 이 문제를 해결하려면 클라이언트를 업그레이드하고 갱신된 기존 응용프로그램을 갱신해야 합니다.
- 최신 클라이언트 및 이전 서버: 새로 개발된 응용프로그램에서 최신 버전의 db2HistoryGetEntry API가 호출되거나, 또는 기존 응용프로그램에서 이전 버전의 db2HistoryGetEntry API를 호출하는 경우 API 함수에는 제한사항이 없습니다.

db2Prune API

db2Prune API의 동작은 클라이언트 서버 구성에 따라 변경됩니다.

다음은 가능한 구성입니다.

- 최신 클라이언트 및 최신 서버: 최신 버전의 db2Prune API가 포함된 새 응용프로그램을 사용하거나 이전 버전의 db2Prune API를 사용하는 기존 응용프로그램을 사용하는 경우에는 API 함수에는 제한사항이 없습니다.
- 이전 클라이언트 및 최신 서버: 기존 응용프로그램을 사용하고 이전 버전의 db2Prune API를 호출하는 경우 API 함수에는 제한사항이 없습니다.
- 최신 클라이언트 및 이전 서버: LSN 문자열이 입력으로 포함된 최신 버전의 db2Prune API를 사용하는 경우에 LSN 문자열이 0xFFFF FFFF FFFF 값을 초과하는 LSN을 나타내거나 LSN 문자열 길이가 12자 미만인 경우 오류 메시지 SQL2032N이 리턴됩니다. 이 오류를 우회하려면 최소한 클라이언트 레벨에 일치하도록 서버를 업그레이드해야 합니다. 기존 응용프로그램을 사용하여 이전 버전의 db2Prune API를 호출하는 경우 LSN 변경사항에서 소개한 대로 API 함수에는 제한사항이 없습니다.

sqlbftpq API, sqlbstpq API 및 sqlbmtsq API

sqlbftpq API, sqlbstpq API 및 sqlbmtsq API 동작은 클라이언트 서버 구성에 따라 변경됩니다.

다음은 가능한 구성입니다.

- 최신 클라이언트 및 최신 서버: 이 API 각각에 대해 기존 응용프로그램을 사용하여 이전 버전의 API를 호출하는 경우 리턴된 LSN 크기가 이전 SQLB_TBSPQRY_DATA 구조의『lifeLSN』필드가 포함할 수 있는 크기보다 큰 경우 오류 메시지 SQL2032N이 리턴됩니다. 이전 버전의 이 API를 호출할 때 sqlbmtsq API에 대해 추가로 고려해야 하는 사항이 있습니다. SQLB_TBSPQRY_DATA 구조의『lifeLSN』필드는 구조의 다른 모든 필드가 유효한 값을 포함하는 경우에도 유효하지 않은 값을 포함합니다. 최신 버전의 API를 사용하도록 응용프로그램을 갱신하여 이 문제를 해결할 수 있습니다. 이러한 API의 최신 버전을 사용하는 경우에는 API 함수에 제한사항이 없습니다.
- 이전 클라이언트 및 최신 서버: 이 구성의 경우, sqlbmtsq API에 대해 명시한 제한사항을 제외하고는 위에서 설명한 대로 세 API 중 하나의 이전 버전을 실행해도 동작은 클라이언트 및 서버 모두 최신 릴리스를 사용하는 구성의 API 동작을 미러링하게 됩니다. 이 제한사항을 해결하려면 클라이언트는 최소한 서버의 릴리스 레벨로 업그레이드해야 하며 사용 중인 응용프로그램은 최신 버전의 API를 사용하도록 갱신해야 합니다.
- 최신 클라이언트 및 이전 서버: 기존 응용프로그램을 사용하여 이전 버전의 sqlbmtsq API를 호출하는 경우 SQLB_TBSPQRY_DATA 구조의『lifeLSN』필드는 유효하지 않은 LSN 값과 같이 리턴됩니다. 구조의 다른 모든 필드는 계속 유효합니다. 이

동작을 정정하려면 사용 중인 응용프로그램이 최신 버전의 sqlbmtsq API를 통합하도록 갱신해야 합니다. 다른 API의 경우 사용 중인 API 버전에 상관 없이 API 함수에 제한사항이 없습니다.

sqlurlog API (이전 레벨의 API)

sqlurlog API의 동작은 클라이언트 서버 구성에 따라 변경됩니다.

다음은 가능한 구성입니다.

- 최신 클라이언트 및 최신 서버: 이 클라이언트 서버 구성에서 sqlurlog API를 사용하면 LSN이 리턴하는 값이 SQLU_RLOG_INFO 출력 구조의 LSN 필드에 포함할 수 있는 값보다 큰 경우 오류 메시지 SQL2650N이 리턴됩니다. sqlurlog API를 사용하는 모든 응용프로그램이 해당 후임자인 db2ReadLog API를 사용하도록 수정하는 방법이 이 문제를 예방할 수 있는 유일한 방법입니다.

  주: 이 구성에서 sqlurlog API를 사용하면 로그 버퍼에서 리턴되는 로그 레코드가 새 로그 레코드가 되며 새 db2LSN 형식의 db2LSN을 포함합니다.
- 이전 클라이언트 및 최신 서버: db2ReadLog API 및 db2ReadLogNoConn API와 유사하게 이전 레벨 클라이언트에서 현재 레벨의 서버에 대해 발행된 sqlurlog API 호출은 『지원되지 않습니다』. sqlurlog API를 사용하려고 하면 오류 메시지 SQL1198N이 리턴됩니다. 이 문제를 방지하려면 서버 레벨에 맞게 클라이언트를 업그레이드해야 합니다.
- 최신 클라이언트 및 이전 서버: 이 클라이언트 서버 구성으로 sqlurlog API를 사용하면 API 함수에는 제한사항이 없습니다.

sqlbftsq API, sqlbstsq API 및 sqlbtsq API(이전 레벨의 API)

sqlbftsq API, sqlbstsq API 및 sqlbtsq API 동작은 클라이언트 서버 구성에 따라 변경됩니다.

다음은 가능한 구성입니다.

- 최신 클라이언트 및 최신 서버: 이 유형의 클라이언트 서버 구성에서 이 세 개의 API 중 하나를 사용하면 이전 SQLB_TBSPQRY_DATA 구조의 『lifeLSN』 필드가 유효하지 않은 LSN 값으로 채워집니다. 그렇게 되면 SQLB_TBSPQRY_DATA 구조의 『lifeLSN』 필드가 큰 LSN 값을 포함할 수 없게 됩니다. 이 문제를 방지하려면 후임자가 각각 sqlbftpq API, sqlbstpq API 및 sqlbmtsq API인 sqlbftsq API, sqlbstsq API 및 sqlbtsq API의 후임자를 사용해야 합니다.
- 이전 클라이언트 및 최신 서버: 이전 구성에서 설명한 API 동작이 이 구성에도 적용됩니다.
- 최신 클라이언트 및 이전 서버: 이 클라이언트 서버 구성에서 sqlbftsq API, sqlbstsq API 및 sqlbtsq API를 사용하는 경우에는 API 함수에 제한사항이 없습니다.

제 4 부 API 설명 구성 방법

각 API의 간단한 설명에는 다음과 같은 서브섹션 중 일부 또는 전체가 표시됩니다.

범위

인스턴스 내의 API 조작 범위. 단일 파티션 데이터베이스 환경에서 범위는 해당하는 단일 데이터베이스 파티션만 해당합니다. 파티션된 데이터베이스 환경에서 범위는 노드 구성 파일(db2nodes.cfg) 또는 API가 호출된 데이터베이스 파티션에 정의된 모든 논리적 데이터베이스 파티션 서버 콜렉션일 수 있습니다.

권한 부여

API를 제대로 호출하는 데 필요한 권한

필수 연결

데이터베이스, 인스턴스, 없음 또는 연결 설정 중 하나입니다. 성공적으로 조작하기 위해 함수에 데이터베이스를 연결, 인스턴스에 접속 또는 연결이 필요하지 않은지를 나타냅니다.

*없음*은 API가 제대로 작동하기 위해 데이터베이스를 연결할 필요가 없음을 나타냅니다. *연결 설정*은 API가 호출될 때 API가 데이터베이스에 연결되는 것을 나타냅니다.

특정 API를 호출하기 전에 데이터베이스에 대한 명시적 연결 또는 인스턴스 접속이 필요할 수도 있습니다. 데이터베이스에 연결하거나 인스턴스에 접속해야 하는 API는 로컬 또는 리모트로 실행할 수 있습니다. 둘 다 필요하지 않은 API는 리모트로 실행할 수 없습니다. 클라이언트에서 호출할 때 클라이언트 환경에만 영향을 줍니다.

API 내장 파일

API 프로토타입 및 필요한 모든 사전 정의된 상수 및 매개변수를 포함하는 내장 파일의 이름

주: 내장 파일 확장자는 사용한 프로그래밍 언어에 따라 달라집니다. C/C++ 내장 파일의 파일 확장자는 .h이며 Cobol 내장 파일의 확장자는 .cbl입니다. 내장 파일은 다음 디렉토리에 있습니다.

C/C++(UNIX):

```
sqllib/include
```

C/C++(Windows):

```
sqllib¥include
```

COBOL(UNIX):

```
sqllib/include/cobol_a

sqllib/include/cobol_i

sqllib/include/cobol_mf
```

COBOL(Windows):

```
sqllib¥include¥cobol_a

sqllib¥include¥cobol_i

sqllib¥include¥cobol_mf
```

C API 구문

API 호출의 C 구문

버전 6 이후에 새 표준이 DB2 관리 API에 적용되었습니다. 새 API 정의 구현은 단계별로 수행되고 있습니다. 다음은 변경사항에 대한 간략한 설명입니다.

- 새 API 이름에는 의미 있는 대소문자 혼용 문자열이 뒤에 오는 접두부가 포함됩니다(예: db2LoadQuery). 관련 API는 논리적으로 그룹화할 수 있는 이름이 포함됩니다. 예를 들면, 다음과 같습니다.

  ```
  db2HistoryCloseScan
  db2HistoryGetEntry
  db2HistoryOpenScan
  db2HistoryUpdate
  ```

- 일반 API의 이름에는 C API 이름에 일치하는 문자열이 뒤에 오는 접두부 ″db2g″가 포함됩니다. 일반 API에 사용되는 데이터 구조 이름에도 접두부 ″db2g″가 포함됩니다.
- 함수(*versionNumber*)의 첫 번째 매개변수는 코드가 컴파일되는 버전, 릴리스 또는 PTF 레벨을 나타냅니다. 이 버전 번호를 사용하여 두 번째 매개변수로 전달되는 구조 레벨을 지정합니다.
- 함수의 두 번째 매개변수는 API의 기본 인터페이스 구조의 void 포인터입니다. 구조의 각 요소는 자동 유형(예: db2Long32) 또는 포인터입니다. 각 매개변수 이름은 다음과 같은 이름 지정 규칙을 사용합니다.

  ```
  piCamelCase  - 입력 데이터의 포인터
  poCamelCase  - 출력 데이터의 포인터
  pioCamelCase - 입력 또는 출력 데이터의 포인터
  iCamelCase   - 입력 데이터
  ioCamelCase  - 입/출력 데이터
  oCamelCase   - 출력 데이터
  ```

- 세 번째 매개변수는 SQLCA의 포인터로 필수입니다.

일반 API 구문

COBOL 및 FORTRAN 프로그래밍 언어의 API 호출 구문

경고: API에 전달되는 모든 문자열에 대해 한 개의 추가 바이트를 제공합니다. 이에 실패하면 예기치 않은 오류가 발생합니다. 이 추가 바이트는 데이터베이스 관리 프로그램에 의해 수정됩니다.

API 매개변수

각 API 매개변수 및 해당 값의 설명. 사전 정의된 값은 해당 기호와 같이 표시됩니다. 기호의 실제 값은 해당 언어 내장 파일에서 확보할 수 있습니다. COBOL 프로그래머는 모든 기호에서 밑줄(_)을 하이픈(-)으로 대체해야 합니다. 각 호스트 언어에서 매개변수 데이터 유형에 대한 자세한 정보는 샘플 프로그램을 참조하십시오.

주: 데이터베이스 관리 프로그램 API를 호출하는 응용프로그램은 리턴 코드 및 SQLCA 구조를 검사하여 오류 조건을 적절하게 점검해야 합니다. 성공하는 경우 대부분의 데이터베이스 관리 프로그램 API는 영(0) 리턴 코드를 리턴합니다. 일반적으로 0이 아닌 리턴 코드는 보조 오류 처리 메커니즘인 SQLCA 구조가 손상되었을 수 있음을 나타냅니다. 이 경우 호출된 API가 실행되지 않았습니다. 손상된 SQLCA 구조의 가능한 원인은 구조에 대해 유효하지 않은 주소의 전달입니다.

대부분의 데이터베이스 관리 프로그램 API 호출 후에 갱신되는 SQLCA 구조의 SQLCODE 및 SQLSTATE 필드에 오류 정보가 리턴됩니다. 데이터베이스 관리 프로그램 API를 호출하는 소스 파일은 하나 이상의 SQLCA 구조를 제공할 수 있으며 임의의 이름이 지정됩니다. 실행이 성공하면 SQLCODE 값이 영(0)입니다(가능한 SQLWARN 경고 조건 포함). 양수 값인 경우에는 명령문이 제대로 실행되었지만 호스트 변수가 절단되는 경고가 포함되어 있습니다. 음수 값은 오류 조건 발생을 나타냅니다.

추가 필드인 SQLSTATE에는 기타 IBM® 데이터베이스 제품 및 SQL92 호환 데이터베이스 관리 프로그램에서 일관성이 있는 표준화된 오류 코드가 포함됩니다. SQLSTATEs는 다수의 데이터베이스 관리 프로그램에서 공통적으로 사용되기 때문에 이식성에 대해 고려하는 경우에는 SQLSTATEs를 사용하십시오.

SQLWARN 필드에는 SQLCODE가 영(0)이라도 경고 표시기 배열이 포함됩니다.

REXX API 구문

해당하는 경우, API 호출의 REXX 구문.

SQLDB2 인터페이스는 REXX에서 API의 호출을 지원합니다. SQLDB2 인터페이스는 SQLCA 이외의 다른 출력이 없는 지원되지 않는 새 API 또는 이전의 API에 대해 REXX에서 지원되도록 작성되었습니다. SQLDB2 인터페이스를 통해 명령을 호출하는

것은 토큰 `call db2`가 `CALL SQLDB2`로 교체되는 점만 제외한다면 명령행 처리기(CLP)에서 명령을 호출하는 것과 구문상 동일합니다. REXX에서 `CALL SQLDB2`를 사용하면 CLP를 직접 호출하는 경우 보다 다음과 같은 장점이 있습니다.

- 복합 REXX 변수 SQLCA가 설정됩니다.
- 디폴트로 모든 CLP 출력 메시지가 해제됩니다.

REXX API 매개변수

각 REXX API 매개변수 및 해당 값이 있는 경우의 설명

제 1 장 DB2 API 응용프로그램에 대한 내장 파일

C, C++, COBOL 및 FORTRAN 응용프로그램에서 DB2 API를 호출하는 데 사용하는 내장 파일은 아래에서 설명됩니다.

C 및 C++ 내장 파일

DB2APIDF(db2ApiDf.h)

이 파일은 이름이 'db2'로 시작하는 거의 대부분의 DB2 API에 대해 구조, 상수 및 프로토타입을 정의합니다.

DB2AUCFG(db2AuCfg.h)

이 파일은 DB2 API, db2AutoConfig 및 db2AutoConfigFreeMemory에 대해 구조, 상수 및 프로토타입을 정의합니다.

DB2SECPLUGIN(db2secPlugin.h)

이 파일은 인증 및 그룹 멤버쉽 찾아보기에 사용되는 사용자 정의 보안 플러그인을 개발하는 데 사용되는 API에 대해 구조, 상수 및 프로토타입을 정의합니다.

SQL(sql.h)

이 파일은 바인더, 프리컴파일러 및 오류 메시지 검색 API에 대한 언어 고유의 프로토타입을 포함합니다. 시스템 상수도 정의합니다.

SQLAPREP(sqlaprep.h)

이 파일에는 자체 프리컴파일러를 작성하는 데 필요한 정의가 포함됩니다.

SQLENV(sqlenv.h)

이 파일은 데이터베이스 환경 API에 대한 언어 고유의 호출 및 해당 인터페이스에 대한 구조, 상수 및 리턴 코드를 정의합니다.

SQLMON(sqlmon.h)

이 파일은 데이터베이스 시스템 모니터 API에 대한 언어 고유의 호출 및 해당 인터페이스에 대한 구조, 상수 및 리턴 코드를 정의합니다.

SQLUTIL(sqlutil.h)

이 파일은 유틸리티 API에 대한 언어 고유의 호출 및 해당 인터페이스에 필요한 구조, 상수 및 코드를 정의합니다.

SQLUVEND(sqluvend.h)

이 파일은 스토리지 관리 벤더에서 사용하는 API에 대한 구조, 상수 및 프로토타입을 정의합니다.

SQLXA(sqlxa.h)

이 파일에는 X/Open XA 인터페이스를 사용하는 응용프로그램에서 사용되는 함수 프로토타입 및 상수가 포함됩니다.

COBOL 내장 파일

SQL(sql.cbl)

이 파일은 바인더, 프리컴파일러 및 오류 메시지 검색 API에 대한 언어 고유의 프로토타입을 포함합니다. 시스템 상수도 정의합니다.

SQLAPREP(sqlaprep.cbl)

이 파일에는 자체 프리컴파일러를 작성하는 데 필요한 정의가 포함됩니다.

SQLENV(sqlenv.cbl)

이 파일은 데이터베이스 환경 API에 대한 언어 고유의 호출 및 해당 인터페이스에 대한 구조, 상수 및 리턴 코드를 정의합니다.

SQLMON(sqlmon.cbl)

이 파일은 데이터베이스 시스템 모니터 API에 대한 언어 고유의 호출 및 해당 인터페이스에 대한 구조, 상수 및 리턴 코드를 정의합니다.

SQLMONCT(sqlmonct.cbl)

이 파일에는 데이터베이스 시스템 모니터 API 호출에 필요한 상수 정의 및 로컬 데이터 구조 정의가 포함됩니다.

SQLUTIL(sqlutil.cbl)

이 파일은 유틸리티 API에 대한 언어 고유의 호출 및 해당 인터페이스에 필요한 구조, 상수 및 코드를 정의합니다.

FORTRAN 내장 파일

SQL(sql.f)

이 파일은 바인더, 프리컴파일러 및 오류 메시지 검색 API에 대한 언어 고유의 프로토타입을 포함합니다. 시스템 상수도 정의합니다.

SQLAPREP(sqlaprep.f)

이 파일에는 자체 프리컴파일러를 작성하는 데 필요한 정의가 포함됩니다.

SQLENV(sqlenv.f)

이 파일은 데이터베이스 환경 API에 대한 언어 고유의 호출 및 해당 인터페이스에 대한 구조, 상수 및 리턴 코드를 정의합니다.

SQLMON(sqlmon.f)

이 파일은 데이터베이스 시스템 모니터 API에 대한 언어 고유의 호출 및 해당 인터페이스에 대한 구조, 상수 및 리턴 코드를 정의합니다.

SQLUTIL(sqlutil.f)

이 파일은 유틸리티 API에 대한 언어 고유의 호출 및 해당 인터페이스에 필요한 구조, 상수 및 코드를 정의합니다.

제 5 부 관리 API

제 2 장 db2AddContact - 통지 메시지를 수신할 수 있는 문의처 추가

문의처를 문의처 목록에 추가합니다. 문의처는 통지 메시지를 수신할 수 있는 사용자입니다. 문의처는 시스템에 로컬로 또는 전역 목록으로 정의할 수 있습니다. DB2 관리 서버(DAS) 구성 매개변수인 contact_host의 설정으로 목록이 로컬 또는 전역인지 판별합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2AddContact (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2AddContactData
{
   char *piUserid;
   char *piPassword;
   char *piName;
   db2Uint32 iType;
   char *piAddress;
   db2Uint32 iMaxPageLength;
   char *piDescription;
} db2AddContactData;
```

db2AddContact API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2AddContactData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2AddContactData 데이터 구조 매개변수

piUserid

입력. 사용자 이름

piPassword

입력. piUserid 매개변수로 지정한 사용자 ID의 암호

piName

입력. 담당자

iType 입력. 문의처 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2CONTACT_EMAIL
- DB2CONTACT_PAGE

piAddress

입력. iType 매개변수의 전자 우편 또는 호출기 주소.

iMaxPageLength

입력. iType이 DB2CONTACT_PAGE로 설정된 경우 메시지의 최대 길이

piDescription

입력. 문의처에 대해 사용자가 제공하는 설명

사용 시 참고사항

이 API는 UNIX 및 Linux®에서는 지원되지 않습니다. 그러나 SQL 인터페이스를 통해 동일한 기능에 액세스할 수 있습니다.

제 3 장 db2AddContactGroup - 통지 메시지를 수신할 수 있는 문의처 그룹 추가

문의처 그룹의 목록에 새 문의처 그룹을 추가합니다. 문의처 그룹에는 통지 메시지를 수신할 수 있는 사용자 목록이 포함됩니다. 문의처 그룹은 시스템에 로컬로 또는 전역 목록으로 정의할 수 있습니다. DB2 관리 서버(DAS) 구성 매개변수 contact_host의 설정으로 목록이 로컬 또는 전역인지 판별합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2AddContactGroup (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2AddContactGroupData
{
   char *piUserid;
   char *piPassword;
   char *piGroupName;
   char *piDescription;
   db2Uint32 iNumContacts;
   struct db2ContactTypeData *piContacts;
} db2AddContactGroupData;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

db2AddContactGroup API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2AddContactGroupData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2AddContactGroupData 데이터 구조 매개변수

piUserid

입력. 사용자 이름

piPassword

입력. piUserid의 암호

piGroupName

입력. 검색할 그룹 이름

piDescription

입력. 그룹 설명

iNumContacts

입력. piContacts 수.

piContacts

db2ContactTypeData 구조의 포인터

db2ContactTypeData 데이터 구조 매개변수

contactType

문의처 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName

contactType을 DB2CONTACT_SINGLE로 설정한 경우 문의처 그룹 이름 또는 담당자

사용 시 참고사항

이 API는 UNIX 및 Linux에서는 지원되지 않습니다. 그러나 SQL 인터페이스를 통해 동일한 기능에 액세스할 수 있습니다.

제 4 장 db2AddSnapshotRequest - 스냅샷 요청 추가

이 API는 db2GetSnapshotSize 및 db2GetSnapshot에 대한 스냅샷 요청 스트림을 준비합니다.

범위

db2GetSnapshotSize 및 db2GetSnapshot API에 대한 스냅샷 요청 스트림을 준비합니다. 출력(db2AddSnapshotRequest API에서 생성된 스냅샷 요청)이 db2GetSnapshotSize 및 db2GetSnapshot API에 전달됩니다. 스냅샷 요청에는 스냅샷 요청 유형 및 ID 정보가 포함됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2AddSnapshotRequest (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2AddSnapshotRqstData
{
   void *pioRequestData;
   db2Uint32 iRequestType;
   db2int32 iRequestFlags;
   db2Uint32 iQualType;
   void *piQualData;
} db2AddSnapshotRqstData;

SQL_API_RC SQL_API_FN
  db2gAddSnapshotRequest (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gAddSnapshotRqstData
{
   void *pioRequestData;
```

```
   db2Uint32 iRequestType;
   db2int32 iRequestFlags;
   db2Uint32 iQualType;
   void *piQualData;
   db2Uint32 iQualDataLen;
} db2gAddSnapshotRqstData;
```

db2AddSnapshotRequest API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다. 위에 설명된 대로 db2AddSnapshotData 구조를 사용하려면 db2Versio910을 지정하십시오. 이 구조의 다른 버전을 사용하려면 include 디렉토리의 db2ApiDf 헤더 파일에서 지원되는 버전의 전체 목록을 점검하십시오. 지정한 버전 번호에 해당하는 db2AddSnapshotRequestData 구조의 버전을 사용하고 있는지 확인하십시오.

pParmStruct

입력 및/또는 출력. db2AddSnapshotRequestData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2AddSnapshotRqstData 데이터 구조 매개변수

pioRequestData

입/출력(I/O). db2AddSnapshotRequest API로 구문화되는 요청 데이터. 초기에 이 매개변수는 NULL로 설정됩니다. pioRequestData에 필요한 메모리는 db2AddSnapshotRequest API가 할당합니다. 사용이 종료되면 pioRequestData를 사용 가능하게 해야 합니다(예: db2GetSnapshot API 호출 이후).

iRequestType

입력. 스냅샷 요청 유형(예: SQLMA_DB2).

iRequestFlags

입력. 비트맵된 조치 플래그이며 값은 SQLM_INSTREAM_ADD_REQUEST, SQLM_INSTREAM_ADD_QUAL 또는 SQLM_INSTREAM_ADD_REQQUAL입니다. 호출자가 iRequestFlags를 설정하지 않은 경우,

- iRequestType이 설정되면 iRequestFlags 비트 SQLM_INSTREAM_ADD_REQUEST가 API로 켜집니다.
- piQualifierData 포인터가 널이 아닌 경우 SQLM_INSTREAM_ADD_QUAL이 API로 켜집니다.

API 호출 시에 iRequestType, iQualifierType, iRequestFlags 및 piQualifierData가 0으로 재설정됩니다.

iQualType

입력. 스냅샷 요청에 첨부되는 유형(예: SQLM_INSTREAM_ELM_DBNAME).

piQualData

입력. 규정자를 설명하는 데이터. 널(null)로 끝나는 문자열의 포인터입니다.

db2gAddSnapshotRqstData 데이터 구조 특정 매개변수

iQualDataLen

입력. piQualData 매개변수의 규정자 데이터 길이

제 5 장 db2AdminMsgWrite - 관리 복제 함수에 대한 로그 메시지 쓰기

사용자 및 복제 프로그램이 db2diag 로그 파일 및 관리 통지 로그에 정보를 쓸 수 있는 메커니즘을 제공합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2AdminMsgWrite (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2AdminMsgWriteStruct
{
   db2Uint32 iMsgType;
   db2Uint32 iComponent;
   db2Uint32 iFunction;
   db2Uint32 iProbeID;
   char *piData_title;
   void *piData;
   db2Uint32 iDataLen;
   db2Uint32 iError_type;
} db2AdminMsgWriteStruct;
```

db2AdminMsgWrite API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2AdminMsgWriteStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2AdminMsgWriteStruct 데이터 구조 매개변수

iMsgType

입력. 로그되는 데이터의 유형을 지정합니다. 유효한 값은 2진 데이터의 경우 BINARY_MSG이고 문자열 데이터의 경우 STRING_MSG입니다.

iComponent

입력. 0을 지정합니다.

iFunction

입력. 0을 지정합니다.

iProbeID

입력. 숫자 프로브 포인트를 지정합니다. 숫자 프로브 포인트는 메시지를 보고한 소스 코드에서 포인트를 찾기 위해 사용되는 고유한 내부 ID입니다.

piData_title

입력. 로그할 데이터를 설명하는 제목 문자열에 대한 포인터입니다. 제목이 필요없는 경우에는 NULL로 설정할 수 있습니다.

piData

입력. 로그할 데이터에 대한 포인터입니다. 데이터 로깅이 필요없는 경우에는 NULL로 설정할 수 있습니다.

iDataLen

입력. iMsgType이 BINARY_MSG인 경우 로깅에 사용할 2진 데이터의 바이트 수입니다. iMsgType이 STRING_MSG인 경우 사용되지 않습니다.

iError_type

입력. 가능한 값은 다음과 같습니다.

- DB2LOG_SEVERE_ERROR: (1) 심각한 오류가 발생했습니다.
- DB2LOG_ERROR: (2) 오류가 발생했습니다.
- DB2LOG_WARNING: (3) 경고가 발생했습니다.
- DB2LOG_INFORMATION: (4) 정보용

사용 시 참고사항

이 API는 지정된 오류 유형이 **notifylevel** 데이터베이스 관리 프로그램 구성 매개변수의 값 이하인 경우에만 관리 통지 로그에 로그합니다. 지정된 오류 유형이 **diaglevel** 데이터베이스 관리 프로그램 구성 매개변수의 값 이하인 경우에만 db2diag 로그 파일에 로그합니다. 그러나 관리 통지 로그에 쓴 모든 정보는 **diaglevel** 데이터베이스 관리 프로그램 구성 매개변수가 0으로 설정되지 않으면 db2diag 로그 파일에서 중복됩니다.

제 6 장 db2ArchiveLog - 사용 중인 로그 파일 아카이브

복구 가능한 데이터베이스에 대해 사용 중인 로그 파일을 닫고 절단합니다. userexit가 사용 가능한 경우 아카이브 요청도 발행합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- dbadm

필수 연결

이 API는 지정한 데이터베이스에 자동으로 연결됩니다. 지정한 데이터베이스에 이미 연결되어 있는 경우에는 API가 오류를 리턴합니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2ArchiveLog (
       db2Uint32 versionNumber,
       void * pDB2ArchiveLogStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ArchiveLogStruct
{
   char *piDatabaseAlias;
   char *piUserName;
   char *piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint32 iOptions;
} db2ArchiveLogStruct;

SQL_API_RC SQL_API_FN
  db2gArchiveLog (
       db2Uint32 versionNumber,
       void * pDB2ArchiveLogStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gArchiveLogStruct
{
```

```
   db2Uint32 iAliasLen;
   db2Uint32 iUserNameLen;
   db2Uint32 iPasswordLen;
   char *piDatabaseAlias;
   char *piUserName;
   char *piPassword;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint32 iOptions;
} db2gArchiveLogStruct;
```

db2ArchiveLog API 매개변수

versionNumber

입력. 두 번째 매개변수인 pDB2ArchiveLogStruct로 전달된 변수의 버전 및 릴리스 레벨을 지정합니다.

pDB2ArchiveLogStruct

입력. db2ArchiveLogStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ArchiveLogStruct 데이터 구조 매개변수

piDatabaseAlias

입력. 사용 중인 로그가 아카이브되는 데이터베이스의 데이터베이스 별명(시스템 데이터베이스 디렉토리에서 카탈로그)이 포함된 문자열

piUserName

입력. 연결 시도 시에 사용되는 사용자 이름이 포함된 문자열

piPassword

입력. 연결 시도 시에 사용되는 암호가 포함된 문자열

iAllNodeFlag

파티션된 데이터베이스 환경에만 적용됩니다. 입력. 조작이 db2nodes.cfg 파일에 나열된 모든 노드에 적용되어야 하는지를 나타내는 플래그. 가능한 값은 다음과 같습니다.

DB2ARCHIVELOG_NODE_LIST

piNodeList로 전달된 노드 목록의 노드에 적용됩니다.

DB2ARCHIVELOG_ALL_NODES

모든 노드에 적용됩니다. piNodeList는 NULL이어야 합니다. 이것은 디폴트값입니다.

DB2ARCHIVELOG_ALL_EXCEPT

piNodeList로 전달된 노드 목록에 있는 노드를 제외한 모든 노드에 적용됩니다.

iNumNodes

파티션된 데이터베이스 환경 전용. 입력. piNodeList 배열의 노드 수를 지정합니다.

piNodeList

파티션된 데이터베이스 환경 전용. 입력. 아카이브 로그 조작에 적용되는 노드 수 배열의 포인터

iOptions

입력. 나중에 사용하기 위해 예약됨

db2gArchiveLogStruct 데이터 구조 특정 매개변수

iAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수

iUserNameLen

입력. 사용자 이름의 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수. 사용자 이름을 사용하지 않는 경우에는 영(0)으로 설정하십시오.

iPasswordLen

입력. 암호 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수. 암호를 사용하지 않는 경우에는 영(0)으로 설정하십시오.

제 7 장 db2AutoConfig - 구성 어드바이저 액세스

응용프로그램이 제어 센터에서 구성 어드바이저에 액세스할 수 있습니다. 이 어드바이저에 대한 세부사항 정보는 제어 센터에서 온라인 도움말 기능을 통해 제공됩니다.

범위

파티션된 데이터베이스 환경에서 모든 데이터베이스 파티션에서 디폴트로 데이터베이스 권장사항이 적용됩니다. db2AutoConfigInterface 데이터 구조의 iApply 매개변수에 대한 DB2_SG_APPLY_ON_ONE_NODE 플래그는 변경사항을 코디네이터 파티션으로만 강제로 제한합니다. 버퍼 풀 변경은 시스템 카탈로그에 항상 적용되기 때문에 모든 데이터베이스 파티션에 영향을 줍니다(DB2_SG_APPLY_ON_ONE_NODE는 버퍼 풀 권장사항에 종속되지 않음).

권한 부여

sysadm

필수 연결

데이터베이스

API 내장 파일

db2AuCfg.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
db2AutoConfig(
  db2Uint32 db2VersionNumber,
  void * pAutoConfigInterface,
  struct sqlca * pSqlca);

typedef struct {
  db2int32 iProductID;
  char iProductVersion[DB2_SG_PROD_VERSION_SIZE+1];
  char iDbAlias[SQL_ALIAS_SZ+1];
  db2int32 iApply;
  db2AutoConfigInput iParams;
  db2AutoConfigOutput oResult;
} db2AutoConfigInterface;

typedef struct {
  db2int32 token;
  db2int32 value;
} db2AutoConfigElement;

typedef struct {
```

```
  db2Uint32 numElements;
  db2AutoConfigElement *  pElements;
} db2AutoConfigArray;
typedef db2AutoConfigArray  db2AutoConfigInput;
typedef db2AutoConfigArray  db2AutoConfigDiags;

typedef struct {
  db2Uint32 numElements;
  struct db2CfgParam * pConfigs;
  void * pDataArea;
} db2ConfigValues;

typedef struct {
  char * pName;
  db2int32 value;
} db2AutoConfigNameElement;

typedef struct {
  db2Uint32 numElements;
  db2AutoConfigNameElement * pElements;
} db2AutoConfigNameArray;
typedef db2AutoConfigNameArray db2BpValues;

typedef struct {
  db2ConfigValues oOldDbValues;
  db2ConfigValues oOldDbmValues;
  db2ConfigValues oNewDbValues;
  db2ConfigValues oNewDbmValues;
  db2AutoConfigDiags oDiagnostics;
  db2BpValues oOldBpValues;
  db2BpValues oNewBpValues;
} db2AutoConfigOutput;
```

db2AutoConfig API 매개변수

db2VersionNumber

입력. 두 번째 매개변수인 pAutoConfigInterface로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pAutoConfigInterface

입력. db2AutoConfigInterface 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2AutoConfigInterface 데이터 구조 매개변수

iProductID

입력. 고유 제품 ID를 지정합니다. iProductID 매개변수의 유효한 값(include 디렉토리에 있는 db2AuCfg.h에 정의)은 다음과 같습니다.

- DB2_SG_PID_DEFAULT
- DB2_SG_PID_WEBSPHERE_COMMERCE_SUITE
- DB2_SG_PID_SAP

- DB2_SG_PID_WEBSPHERE_ADVANCED_SERVER
- DB2_SG_PID_SIEBEL
- DB2_SG_PID_PS_EPM
- DB2_SG_PID_PS_ONLINE
- DB2_SG_PID_PS_BATCH
- DB2_SG_PID_PS
- DB2_SG_PID_LOTUS_DOMINO
- DB2_SG_PID_CONTENT_MANAGER

iProductVersion

입력. 제품 버전을 지정하는 16바이트 문자열.

iDbAlias

입력. 데이터베이스 별명을 지정하는 문자열.

iApply

입력. 자동으로 구성을 갱신합니다. iApply 매개변수의 유효한 값(include 디렉토리에 있는 db2AuCfg.h에 정의)은 다음과 같습니다.

DB2_SG_NOT_APPLY

권장사항을 적용하지 않음

DB2_SG_APPLY

모든 권장사항을 적용

DB2_SG_APPLY_DB

데이터베이스(및 버퍼 풀) 권장사항만 적용

DB2_SG_APPLY_ON_ONE_NODE

현재 데이터베이스 파티션에서만 데이터베이스 권장사항 (DB2_SG_APPLY 및 DB2_SG_APPLY_DB에서만 유효)을 적용합니다. 디폴트로 데이터베이스 권장사항은 모든 데이터베이스 파티션에 적용됩니다.

iParams

입력. 매개변수를 어드바이저로 전달합니다.

oResult

출력. 어드바이저의 모든 결과를 포함합니다.

db2AutoConfigElement 데이터 구조 매개변수

token

입력 또는 출력. 입력 매개변수 및 출력 진단의 구성 값을 지정합니다.

값 입력 또는 출력. 토큰으로 지정한 데이터를 보유합니다.

db2AutoConfigArray 데이터 구조 매개변수

numElements

입력 또는 출력. 배열 요소 수

pElements

입력 또는 출력. 요소 배열의 포인터

db2ConfigValues 데이터 구조 매개변수

numElements

입력 또는 출력. 배열 요소 수

pConfigs

입력 또는 출력. db2CfgParam 구조 배열의 포인터

pDataArea

입력 또는 출력. 구성 값을 포함하는 데이터 영역의 포인터

db2AutoConfigNameElement 데이터 구조 매개변수

pName

출력. 출력 버퍼 풀 이름

값 입력 또는 출력. 이름에 지정된 버퍼 풀 크기(페이지 단위)를 보유합니다.

db2AutoConfigNameArray 데이터 구조 매개변수

numElements

입력 또는 출력. 배열 요소 수

pElements

입력 또는 출력. 요소 배열의 포인터

db2AutoConfigOutput 데이터 구조 매개변수

oOldDbValues

출력. 데이터베이스 구성 또는 모든 구성을 갱신하도록 iApply 값을 설정하면 이 값은 어드바이저 사용 전의 데이터베이스 구성 값을 나타냅니다. 그 이외의 경우에 이 값은 현재 값입니다.

oOldDbmValues

출력. 모든 구성을 갱신하도록 iApply 값을 설정하면 이 값은 어드바이저 사용 전의 데이터베이스 관리 프로그램 구성 값을 나타냅니다. 그 이외의 경우에 이 값은 현재 값입니다.

oNewDbValues

출력. 데이터베이스 구성 또는 모든 구성을 갱신하도록 iApply 값을 설정하면 이 값은 현재 데이터베이스 구성 값을 나타냅니다. 그 이외의 경우에는 어드바이저에 대한 권장 값입니다.

oNewDbmValues

출력. 모든 구성을 갱신하도록 iApply 값을 설정하면 이 값은 현재 현재 데이터베이스 관리 프로그램 구성 값을 나타냅니다. 그 이외의 경우에는 어드바이저에 대한 권장 값입니다.

oDiagnostics

출력. 어드바이저 진단을 포함합니다.

oOldBpValues

출력. 데이터베이스 구성 또는 모든 구성을 갱신하도록 iApply 값을 설정하면 이 값은 어드바이저 사용 전의 버퍼 풀 *크기*(페이지 단위)를 나타냅니다. 그 이외의 경우에 이 값은 현재 값입니다.

oNewBpValues

출력. 데이터베이스 구성 또는 모든 구성을 갱신하도록 iApply 값을 설정하면 이 값은 현재 버퍼 풀 *크기*(페이지 단위)를 나타냅니다. 그 이외의 경우에는 어드바이저에 대한 권장 값입니다.

사용 시 참고사항

db2AutoConfig API로 할당된 메모리를 사용 가능하게 하려면 db2AutoConfigFreeMemory API를 호출하십시오.

maxagents 및 maxcagents 구성 매개변수는 더 이상 사용되지 않으며 db2AutoConfig API의 동작은 API에 전달된 db2VersionNumber에 따라 달라집니다. 버전이 DB2 v9.5 이상인 경우 maxagents가 리턴되지 않지만 이전 버전에서는 리턴됩니다. 이 구성 매개변수는 추후 릴리스에서 완전히 제거될 수 있습니다.

제 8 장 db2AutoConfigFreeMemory - db2AutoConfig API로 할당된 메모리 사용 가능화

db2AutoConfig API로 할당된 메모리를 사용 가능화합니다.

권한 부여

sysadm

필수 연결

데이터베이스

API 내장 파일

```
db2AuCfg.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
db2AutoConfigFreeMemory(
  db2Uint32 db2VersionNumber,
  void * pAutoConfigInterface,
  struct sqlca * pSqlca);
```

db2AutoConfigFreeMemory API 매개변수

db2VersionNumber

입력. 두 번째 매개변수인 pAutoConfigInterface로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pAutoConfigInterface

입력. db2AutoConfigInterface 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

제 9 장 db2Backup - 데이터베이스 또는 테이블 스페이스 백업

데이터베이스 또는 테이블 스페이스의 백업 사본을 작성합니다.

범위

파티션된 데이터베이스 환경에서 디폴트로 이 API는 실행되는 데이터베이스 파티션에만 영향을 줍니다.

파티션된 백업을 수행하는 옵션을 지정하면 카탈로그 노드에서 명령을 호출할 수 있습니다. 옵션이 모든 데이터베이스 파티션 서버를 백업하도록 지정한 경우 db2nodes.cfg 파일에 나열된 모든 데이터베이스 파티션 서버에 영향을 줍니다. 그 외의 경우에는 API에 지정된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*

필수 연결

데이터베이스. 이 API는 지정한 데이터베이스에 자동으로 연결됩니다.

연결은 백업이 완료되면 종료됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Backup (
       db2Uint32 versionNumber,
       void * pDB2BackupStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2BackupStruct
{
  char *piDBAlias;
  char oApplicationId[SQLU_APPLID_LEN+1];
  char oTimestamp[SQLU_TIME_STAMP_LEN+1];
  struct db2TablespaceStruct *piTablespaceList;
  struct db2MediaListStruct *piMediaList;
  char *piUsername;
```

```
  char *piPassword;
  void *piVendorOptions;
  db2Uint32 iVendorOptionsSize;
  db2Uint32 oBackupSize;
  db2Uint32 iCallerAction;
  db2Uint32 iBufferSize;
  db2Uint32 iNumBuffers;
  db2Uint32 iParallelism;
  db2Uint32 iOptions;
  db2Uint32 iUtilImpactPriority;
  char *piComprLibrary;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  db2NodeType *piNodeList;
  db2int32 iNumMPPOutputStructs;
  struct db2BackupMPPOutputStruct *poMPPOutputStruct;
} db2BackupStruct;

typedef SQL_STRUCTURE db2TablespaceStruct
{
  char                        **tablespaces;
  db2Uint32 numTablespaces;
} db2TablespaceStruct;

typedef SQL_STRUCTURE db2MediaListStruct
{
  char                        **locations;
  db2Uint32 numLocations;
  char locationType;
} db2MediaListStruct;

typedef SQL_STRUCTURE db2BackupMPPOutputStruct
{
  db2NodeType nodeNumber;
  db2Uint64 backupSize;
  struct sqlca  sqlca;
} db2BackupMPPOutputStruct;

SQL_API_RC SQL_API_FN
  db2gBackup (
       db2Uint32 versionNumber,
       void * pDB2gBackupStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gBackupStruct
{
  char *piDBAlias;
  db2Uint32 iDBAliasLen;
  char *poApplicationId;
  db2Uint32 iApplicationIdLen;
  char *poTimestamp;
  db2Uint32 iTimestampLen;
  struct db2gTablespaceStruct *piTablespaceList;
  struct db2gMediaListStruct *piMediaList;
  char *piUsername;
  db2Uint32 iUsernameLen;
```

```
  char *piPassword;
  db2Uint32 iPasswordLen;
  void *piVendorOptions;
  db2Uint32 iVendorOptionsSize;
  db2Uint32 oBackupSize;
  db2Uint32 iCallerAction;
  db2Uint32 iBufferSize;
  db2Uint32 iNumBuffers;
  db2Uint32 iParallelism;
  db2Uint32 iOptions;
  db2Uint32 iUtilImpactPriority;
  char *piComprLibrary;
  db2Uint32 iComprLibraryLen;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  db2NodeType *piNodeList;
  db2int32 iNumMPPOutputStructs;
  struct db2gBackupMPPOutputStruct *poMPPOutputStruct;
} db2gBackupStruct;

typedef SQL_STRUCTURE db2gTablespaceStruct
{
  struct db2Char *tablespaces;
  db2Uint32 numTablespaces;
} db2gTablespaceStruct;

typedef SQL_STRUCTURE db2gMediaListStruct
{
  struct db2Char *locations;
  db2Uint32 numLocations;
  char locationType;
} db2gMediaListStruct;

typedef SQL_STRUCTURE db2gBackupMPPOutputStruct
{
  db2NodeType nodeNumber;
  db2Uint64 backupSize;
  struct sqlca  sqlca;
} db2gBackupMPPOutputStruct;

typedef SQL_STRUCTURE db2Char
{
   char *pioData;
   db2Uint32 iLength;
   db2Uint32 oLength;
} db2Char;
```

db2Backup API 매개변수

versionNumber

입력. 두 번째 매개변수인 **pDB2BackupStruct**로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2BackupStruct

입력. db2BackupStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2BackupStruct 데이터 구조 매개변수

piDBAlias

입력. 백업할 데이터베이스의 데이터베이스 별명(시스템 데이터베이스 디렉토리에서 카탈로그)이 포함된 문자열

oApplicationId

출력. API는 에이전트 서비스 응용프로그램을 식별하는 문자열을 리턴합니다. 데이터베이스 모니터를 사용하는 백업 조작 진행 상태에 대한 정보를 가져오는 데 사용할 수 있습니다.

oTimestamp

출력. API가 백업 이미지의 시간소인을 리턴합니다.

piTablespaceList

입력. 백업할 테이블 스페이스 목록. 테이블 스페이스 레벨 백업에만 필요합니다. 데이터베이스 레벨 백업에서는 NULL이어야 합니다. db2TablespaceStruct 구조를 참조하십시오.

piMediaList

입력. 이 구조를 사용하여 호출자는 백업 조작의 대상을 지정할 수 있습니다. 자세한 정보는 아래 db2MediaListStruct 구조를 참조하십시오.

piUsername

입력. 연결 시도 시에 사용되는 사용자 이름이 포함된 문자열. NULL일 수 있습니다.

piPassword

입력. 사용자 이름에 사용되는 암호가 포함된 문자열. NULL일 수 있습니다.

piVendorOptions

입력. 응용프로그램의 정보를 벤더 함수에 전달하는 데 사용됩니다. 이 데이터 구조는 플랫해야 합니다. 즉, 간접 레벨이 지원되지 않습니다. 이 데이터에 대해서는 바이트 리버설이 수행되지 않고 코드 페이지가 점검되지 않습니다.

iVendorOptionsSize

입력. 65535 바이트를 초과할 수 없는 **piVendorOptions** 필드 길이

oBackupSize

출력. 백업 이미지 크기(MB).

iCallerAction

입력. 수행할 조치를 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2BACKUP_BACKUP

백업을 시작합니다.

DB2BACKUP_NOINTERRUPT

백업을 시작합니다. 백업이 자동으로 실행되며 일반적으로 사용자가 개입해야 하는 시나리오가 호출자에게 우선 리턴되지 않고 시도되거나 오류를 생성하도록 지정합니다. 예를 들어, 이 호출자 조치를 사용하면 예를 들어, 백업에 필요한 모든 미디어가 마운트된 것을 알고 있는 경우 유틸리티 프롬프트가 필요하지 않습니다.

DB2BACKUP_CONTINUE

유틸리티에서 요청된 일부 조치(예: 새 테이프 마운트)를 수행한 후에 백업이 계속됩니다.

DB2BACKUP_TERMINATE

유틸리티에서 요청된 일부 조치 수행이 실패한 후에는 백업을 종료하십시오.

DB2BACKUP_DEVICE_TERMINATE

백업에서 사용되는 디바이스 목록에서 특정 디바이스를 제거하십시오. 특정 미디어가 가득 차면 백업이 호출자에게 경고를 리턴합니다(나머지 디바이스를 사용하여 처리는 계속됨). 이 호출자 조치로 백업을 다시 호출하여 경고를 생성한 디바이스를 사용 중인 디바이스 목록에서 제거하십시오.

DB2BACKUP_PARM_CHK

백업을 수행하지 않고 매개변수의 유효성을 확인하는 데 사용됩니다. 이 옵션은 호출이 리턴된 후에도 데이터베이스 연결을 종료하지 않습니다. 이 호출이 성공적으로 리턴되면 조치를 계속 진행하기 위해 사용자가 SQLUB_CONTINUE와 함께 호출을 발행합니다.

DB2BACKUP_PARM_CHK_ONLY

백업을 수행하지 않고 매개변수의 유효성을 확인하는 데 사용됩니다. 이 호출이 리턴되기 전에 이 호출로 작성된 데이터베이스 연결이 종료되며 연속된 호출은 필요하지 않습니다.

iBufferSize

입력. 4KB 할당 단위(페이지)로 버퍼 *크기*를 백업하십시오. 최소 단위는 8입니다.

iNumBuffers

입력. 사용할 백업 버퍼 수를 지정합니다. 최소는 2이며 최대는 메모리에 의한 한계입니다.

iParallelism

입력. 병렬 처리 수준(버퍼 조작 수). 최소는 1이며 최대는 1024입니다.

iOptions

입력. 백업 등록 정보의 비트맵. **iOptions** 값을 작성하기 위해 비트 방향의 OR 연산자를 사용하여 옵션이 조합됩니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2BACKUP_OFFLINE

오프라인으로 데이터베이스에 대한 독점 연결이 가능합니다.

DB2BACKUP_ONLINE

온라인을 사용하면 백업 조작이 수행되는 동안 다른 응용프로그램이 데이터베이스에 액세스할 수 있습니다.

주: 사용자가 SMS LOB 데이터에 잠금이 있는 경우 온라인 백업 조작은 중지된 것으로 표시될 수도 있습니다.

DB2BACKUP_DB

전체 데이터베이스 백업.

DB2BACKUP_TABLESPACE

테이블 스페이스 레벨 백업. 테이블 스페이스 레벨 백업의 경우 **piTablespaceList** 매개변수에 테이블 스페이스 목록을 제공하십시오.

DB2BACKUP_INCREMENTAL

누적(증분) 백업 이미지를 지정합니다. 증분 백업 이미지는 가장 최근에 완료된 전체 백업 조작 이후 변경된 모든 데이터베이스 데이터의 사본입니다.

DB2BACKUP_DELTA

누적되지 않는(델타) 백업 이미지를 지정합니다. 델타 백업 이미지는 가장 최근에 완료된 모든 유형의 백업 조작 이후 변경된 모든 데이터베이스 데이터의 사본입니다.

DB2BACKUP_COMPRESS

백업이 압축되도록 지정합니다.

DB2BACKUP_INCLUDE_COMPR_LIB

백업 압축에 사용되는 라이브러리가 백업 이미지에 포함되도록 지정합니다.

DB2BACKUP_EXCLUDE_COMPR_LIB

백업 압축에 사용되는 라이브러리가 백업 이미지에 포함되지 않도록 지정합니다.

DB2BACKUP_INCLUDE_LOGS

백업 이미지가 이 이미지를 어떤 일관된 특정 시점으로 리스토어하거나 롤 포워드해야 하는 로그 파일 범위도 포함하도록 지정합니다. 이 옵션은 오프라인 백업이나 다중 파티션 백업에는 유효하지 않습니다.

DB2BACKUP_EXCLUDE_LOGS

백업 이미지가 로그 파일을 포함해서는 안된다는 것을 지정합니다.

주: 오프라인 백업 조작을 수행할 때 이 옵션 지정 여부와 상관 없이 로그가 제외되며 INCLUDE가 디폴트인 스냅샷 백업만 예외입니다.

DB2BACKUP_MPP

파티션된 데이터베이스에 적합한 방식으로 백업을 수행하십시오.

iUtilImpactPriority

입력. 백업 중에 사용되는 우선순위 값을 지정합니다.

- 이 값이 0이 아닌 경우에는 유틸리티는 조절 모드로 실행됩니다. 그 외의 경우에는 유틸리티가 조절되지 않은 채로 실행됩니다.
- 여러 개의 유틸리티가 동시에 실행되는 경우 이 매개변수를 사용하여 조절된 태스크 사이의 관련 우선순위를 판별합니다. 예를 들어 두 개의 동시 백업의 경우 하나는 우선순위가 2이고 다른 하나는 4입니다. 둘 다 조절되지만 우선순위가 4인 백업은 더 많은 자원을 할당합니다. 우선순위를 2와 4로 설정하는 것은 이를 5와 10 또는 30과 60으로 설정하는 것과 다르지 않습니다. 우선순위 값은 완전히 상대적입니다.

piComprLibrary

입력. 백업 이미지 압축 수행에 사용되는 외부 라이브러리 이름을 나타냅니다. 이름은 서버의 파일을 나타내는 완전한 경로여야 합니다. 값이 NULL 포인터 또는 빈 문자열의 포인터인 경우 DB2는 압축에 디폴트 라이브러리를 사용합니다. 지정한 라이브러리를 찾을 수 없는 경우 백업이 실패합니다.

piComprOptions

입력. 압축 라이브러리의 초기화 루틴으로 패스될 2진 데이터 블록을 설명합니다. DB2는 이 문자열을 클라이언트에서 서버로 직접 전달하여 바이트 리버설이나 코드 페이지 변환에 관련된 모든 문제는 압축 라이브러리가 처리하도록 합니다. 데이터 블록의 첫 번째 문자가 '@'인 경우 데이터의 나머지는 서버에 있는 파일 이름으로 DB2가 해석합니다. 그런 다음 DB2가 **piComprOptions** 및 **iComprOptionsSize**의 컨텐츠를 이 파일의 컨텐츠와 크기로 교체하고 대신 이 새 값을 초기화 루틴에 전달합니다.

iComprOptionsSize

입력. **piComprOptions**로 전달된 데이터 블록의 크기를 나타내는 4바이트의 부호없는 정수. **piComprOptions**가 NULL 포인터인 경우에만 **iComprOptionsSize**가 0이 됩니다.

iAllNodeFlag

입력. 파티션된 데이터베이스 환경에만 적용됩니다. 백업 조작이 db2nodes.cfg에 정의된 일부 또는 전체 데이터베이스 파티션 서버에 적용되는지를 나타냅니다. 가능한 값은 다음과 같습니다.

DB2_NODE_LIST

piNodeList로 전달된 목록에 있는 데이터베이스 파티션 서버에 적용하십시오.

DB2_ALL_NODES

모든 데이터베이스 파티션 서버에 적용됩니다. **piNodeList**가 NULL이어야 합니다. 이것은 디폴트값입니다.

DB2_ALL_EXCEPT

piNodeList에 전달된 목록에 있는 데이터베이스 파티션 서버를 제외한 모든 데이터베이스 파티션 서버에 적용하십시오.

iNumNodes

입력. **piNodeList** 배열의 데이터베이스 파티션 서버의 수를 지정합니다.

piNodeList

입력. 백업을 수행하는 데이터베이스 파티션 서버 수 배열의 포인터

iNumMPPOutputStructs

입력. **piMPPOutputStruct** 배열의 요소 수를 지정합니다. 이는 이 백업 조작에 사용된 데이터베이스 파티션 서버 수와 동일하거나 커야 합니다.

piMPPOutputStruct

출력. 특정 데이터베이스 파티션 서버의 출력 매개변수를 지정하는 db2BackupMPPOutputStruct 구조 배열의 포인터

db2TablespaceStruct 데이터 구조 특정 매개변수

tablespaces

입력. 백업되는 테이블 스페이스 목록의 포인터. C의 경우 목록은 널(null)로 끝나는 문자열입니다. 일반적으로 db2Char 구조의 목록입니다.

numTablespaces

입력. **tablespaces** 매개변수의 항목 수입니다.

db2MediaListStruct 데이터 구조 매개변수

locations

입력. 미디어 위치 목록의 포인터. C의 경우 목록은 널(null)로 끝나는 문자열입니다. 일반적으로 db2Char 구조의 목록입니다.

numLocations

입력. 위치 매개변수의 항목 수

locationType

입력. 미디어 유형을 나타내는 문자. 유효한 값(include 디렉토리에 있는 `sqlutil` 헤더 파일에 정의)은 다음과 같습니다.

SQLU_LOCAL_MEDIA: 'L'
로컬 디바이스(테이프, 디스크, 디스켓 또는 Named Pipes).

SQLU_XBSA_MEDIA: 'X'
XBSA 인터페이스.

SQLU_TSM_MEDIA: 'A'
Tivoli® Storage Manager.

SQLU_OTHER_MEDIA: 'O'
벤더 라이브러리.

SQLU_SNAPSHOT_MEDIA: 'F'
수행할 스냅샷 백업을 지정합니다.

SQLU_SNAPSHOT_MEDIA는 다음과 같이 사용할 수 없습니다.

- DB2BACKUP_COMPRESS
- DB2BACKUP_TABLESPACE
- DB2BACKUP_INCREMENTAL
- **iNumBuffers**
- **iBufferSize**
- **iParallelism**
- **piComprOptions**
- **iUtilImpactPriority**
- 이 구조의 **numLocations** 필드는 스냅샷 리스토어의 경우 1이어야 합니다.

스냅샷 백업의 디폴트 동작은 모든 컨테이너, 로컬 볼륨 디렉토리, 데이터베이스 경로(DBPATH) 및 1차 로그 및 미러 로그 경로를 포함하는 데이터베이스를 구성하는 모든 경로의 FULL DATABASE OFFLINE 백업입니다(INCLUDE LOGS는 EXCLUDE LOGS를 명시적으로 지시한 경우를 제외하고는 모든 스냅샷 백업의 디폴트임).

IBM Data Server로의 통합은 다음 스토리지 하드웨어에 대한 DB2 ACS API 드라이버입니다.

- IBM TotalStorage® SAN Volume Controller
- IBM Enterprise Storage Server® Model 800
- IBM System Storage™ DS6000™
- IBM System Storage DS8000®
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS

db2BackupMPPOutputStruct 및 db2gBackupMPPOutputStruct 데이터 구조 매개변수

nodeNumber

옵션이 적용되는 데이터베이스 파티션 서버

backupSize

지정한 데이터베이스 파티션의 백업 *크기*(KB).

sqlca 지정한 데이터베이스 파티션의 sqlca

db2gBackupStruct 데이터 구조 특정 매개변수

iDBAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수

iApplicationIdLen

입력. **poApplicationId** 버퍼 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수. SQLU_APPLID_LEN+1(`sqlutil.h`에 정의)와 동일해야 합니다.

iTimestampLen

입력. **poTimestamp** 버퍼 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수. SQLU_TIME_STAMP_LEN+1(`sqlutil.h`에 정의)와 동일해야 합니다.

iUsernameLen

입력. 사용자 이름의 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수. 사용자 이름을 제공하지 않는 경우에는 영(0)으로 설정하십시오.

iPasswordLen

입력. 암호 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수. 암호를 제공하지 않은 경우에는 영(0)으로 설정하십시오.

iComprLibraryLen

입력. **piComprLibrary**에 지정된 라이브러리 이름 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수. 라이브러리 이름을 지정하지 않은 경우 영(0)으로 설정하십시오.

db2Char 데이터 구조 매개변수

pioData

문자 데이터 버퍼의 포인터. NULL인 경우 데이터가 리턴되지 않습니다.

iLength

입력. **pioData** 버퍼 *크기*

oLength

출력. **pioData** 버퍼에 있는 데이터의 유효한 문자 수

사용 시 참고사항

versionNumber db2Version950 또는 이상으로만 스냅샷 백업을 수행할 수 있습니다. 미디어 유형 SQLU_SNAPSHOT_MEDIA를 db2Version950보다 낮은 **versionNumber**로 지정하면 DB2 데이터베이스가 오류를 리턴합니다.

이 함수는 모든 레이블 기반 액세스 제어(LBAC) 규칙을 제외합니다. 보호된 데이터를 포함한 모든 데이터를 백업합니다. 또한 백업 자체의 데이터는 LBAC에서 보호되지 않습니다. 백업 및 이를 리스토어하는 위치를 포함하는 모든 사용자는 데이터에 액세스할 수 있습니다.

정기적으로 데이터베이스를 백업하는 경우 매우 큰 데이터베이스 백업 이미지, 많은 데이터베이스 로그 및 로드 사본 이미지가 누적되기 때문에 많은 양의 디스크 스페이스를 사용할 수도 있습니다. 이 복구 오브젝트를 관리하는 방법에 대한 정보는 『복구 오브젝트 관리』를 참조하십시오.

파티션된 데이터베이스 환경에서 단일 시스템 뷰(SSV) 백업을 위한 사용법 참고

- SSV 백업을 수행하려면 **iOptions** DB2BACKUP_MPP 및 DB2BACKUP_DB 또는 DB2BACKUP_TABLESPACE 중 하나를 지정하십시오.
- **versionNumber** db2Version950 또는 이상으로만 SSV 백업을 수행할 수 있습니다. db2Version950 이하의 **versionNumber**를 **iOptions** DB2BACKUP_MPP에 지정하면 DB2 데이터베이스가 오류를 리턴합니다. SSV 백업에 관련된 기타 옵션을 db2Version950 이하의 **versionNumber**와 같이 지정하는 경우 DB2 데이터베이스가 이 옵션을 무시합니다.
- db2BackupStruct에 직접 지정한 **piMediaList**의 값은 모든 노드에서 디폴트값으로 사용됩니다.
- db2BackupStruct에 리턴된 **oBackupSize** 값은 모든 노드에서의 백업 크기 합계입니다. db2BackupMPPOutputStruct에 리턴된 **backupSize**의 값은 지정한 데이터베이스 파티션의 백업 크기입니다.
- **iAllNodeFlag**, **iNumNodes** 및 **piNodeList**는 **iAllNodeFlag**에 대해 CAT_NODE_ONLY 값이 없는 점을 제외하고는 db2RollforwardStruct에서 유사하게 이름이 지정된 것과 동일하게 작동합니다.
- DB2BACKUP_BACKUP 호출자 조치로 수행된 SSV 백업은 DB2BACKUP_NOINTERRUPT 호출자 조치가 지정된 것처럼 수행됩니다.
- ***poMPPOutputStruct**는 최소한 백업에 사용되는 데이터베이스 파티션과 동일한 수의 요소를 포함하는 호출자가 할당한 메모리를 지시합니다.

제 10 장 db2CfgGet - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 가져오기

특정 데이터베이스 구성 파일 또는 데이터베이스 관리 프로그램 구성 파일의 개별 항목 값을 리턴합니다.

범위

특정 데이터베이스 구성 파일에 대한 정보는 실행되는 데이터베이스 파티션에 대해서만 리턴됩니다.

권한 부여

없음

필수 연결

특정 데이터베이스 구성 파일의 구성 매개변수에 대한 현재 온라인 값을 가져오려면 데이터베이스에 연결해야 합니다. 데이터베이스 관리 프로그램의 구성 매개변수에 대한 현재 온라인 값을 가져오려면 인스턴스에 접속해야 합니다. 이 외의 경우에는 데이터베이스 연결이나 인스턴스에 접속할 필요가 없습니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2CfgGet (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2Cfg
{
   db2Uint32 numItems;
   struct db2CfgParam *paramArray;
   db2Uint32 flags;
   char *dbname;
} db2Cfg;

typedef SQL_STRUCTURE db2CfgParam
{
   db2Uint32 token;
   char *ptrvalue;
   db2Uint32 flags;
} db2CfgParam;
```

```
SQL_API_RC SQL_API_FN
  db2gCfgGet (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gCfg
{
   db2Uint32 numItems;
   struct db2gCfgParam *paramArray;
   db2Uint32 flags;
   db2Uint32 dbname_len;
   char *dbname;
} db2gCfg;

typedef SQL_STRUCTURE db2gCfgParam
{
   db2Uint32 token;
   db2Uint32 ptrvalue_len;
   char *ptrvalue;
   db2Uint32 flags;
} db2gCfgParam;
```

db2CfgGet API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2Cfg 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2Cfg 데이터 구조 매개변수

numItems

입력. paramArray 배열의 구성 매개변수 수. paramArray의 최대 요소 수를 지정하려면 이 값을 db2CfgMaxParam으로 설정하십시오.

paramArray

입력. db2CfgParam 구조의 포인터

flags 입력. 수행할 조치 유형을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

db2CfgDatabase

데이터베이스 구성의 값을 리턴하도록 지정합니다.

db2CfgDatabaseManager

데이터베이스 관리 프로그램 구성 파일의 값을 리턴하도록 지정합니다.

db2CfgImmediate

메모리에 저장된 구성 매개변수의 현재 값을 리턴합니다.

db2CfgDelayed

디스크의 구성 매개변수 값을 가져옵니다. 다음 데이터베이스 연결이나 인스턴스 접속 시까지는 이 값이 메모리에서 현재 값이 되지 않습니다.

db2CfgGetDefaults

구성 매개변수의 디폴트값을 리턴합니다.

db2CfgReset

디폴트값으로 재설정합니다.

dbname

입력. 데이터베이스 이름

db2CfgParam 데이터 구조 매개변수

token 입력. 구성 매개변수 ID

db2CfgParam 토큰 요소의 유효한 항목 및 데이터 유형이 "구성 매개변수 요약"에 나열됩니다.

ptrvalue

출력. 구성 매개변수 값

flags 출력. 요청에서 각 매개변수에 대한 특정 정보를 제공합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

db2CfgParamAutomatic

검색된 매개변수에 자동 값이 있는지를 표시합니다. 지정된 구성 매개변수가 자동으로 설정되었는지 판별하려면 db2ApiDf.h에 정의된 플래그 및 db2CfgParamAutomatic 키워드로 리턴된 값에 대해 부울 AND 연산을 수행하십시오.

db2CfgParamComputed

검색된 매개변수에 계산된 값이 있는지를 표시합니다. 지정된 구성 매개변수가 계산됨으로 설정되었는지 판별하려면 db2ApiDf.h에 정의된 플래그 및 db2CfgParamComputed 키워드로 리턴된 값에 대해 부울 AND 연산을 수행하십시오.

부울 AND 연산이 위의 두 키워드에 대해 거짓인 경우에는 검색된 매개변수 값이 수동으로 설정되었음을 나타냅니다.

db2gCfg 데이터 구조 특정 매개변수

dbname_len

입력. dbname 매개변수의 길이(바이트)

db2gCfgParam 데이터 구조 특정 매개변수

ptrvalue_len

입력. ptrvalue 매개변수의 길이(바이트)

사용 시 참고사항

구성 매개변수 maxagents 및 maxcagents는 더 이상 사용되지 않습니다. 이 구성 매개변수는 추후 릴리스에서 완전히 제거될 수 있습니다.

db2CfgGet API는 SQLF_KTN_MAXAGENTS 및 SQLF_KTN_MAXCAGENTS에 대한 요청은 허용하지만 서버가 DB2 v9.5인 경우에는 0이 리턴됩니다.

제 11 장 db2CfgSet - 데이터베이스 관리 프로그램 또는 데이터베이스 구성 매개변수 설정

특정 데이터베이스 구성 파일 또는 데이터베이스 관리 프로그램 구성 파일의 각 항목을 수정합니다. 데이터베이스 구성 파일은 데이터베이스가 작성된 모든 노드에 상주합니다.

범위

데이터베이스 구성 파일의 수정은 디폴트로 모든 데이터베이스 파티션에 영향을 줍니다.

권한 부여

데이터베이스 구성 파일의 수정은 다음 중 하나입니다.

- sysadm
- sysctrl
- sysmaint

데이터베이스 관리 프로그램 구성 파일의 수정은 다음과 같습니다.

- sysadm

필수 연결

특정 데이터베이스의 구성 매개변수를 온라인으로 수정하려면 데이터베이스에 연결해야 합니다. 데이터베이스 관리 프로그램의 구성 매개변수를 온라인으로 수정하는 경우 인스턴스에 접속할 필요는 없습니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
   db2CfgSet (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2Cfg
{
   db2Uint32 numItems;
   struct db2CfgParam *paramArray;
   db2Uint32 flags;
   char *dbname;
   SQL_PDB_NODE_TYPE dbpartitionnum;
}  db2Cfg;
```

```
typedef SQL_STRUCTURE db2CfgParam
{
   db2Uint32 token;
   char *ptrvalue;
   db2Uint32 flags;
}  db2CfgParam;

SQL_API_RC SQL_API_FN
   db2gCfgSet (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gCfg
{
   db2Uint32 numItems;
   struct db2gCfgParam *paramArray;
   db2Uint32 flags;
   db2Uint32 dbname_len;
   char *dbname;
}  db2gCfg;

typedef SQL_STRUCTURE db2gCfgParam
{
   db2Uint32 token;
   db2Uint32 ptrvalue_len;
   char *ptrvalue;
   db2Uint32 flags;
}  db2gCfgParam;
```

db2CfgSet API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2Cfg 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2Cfg 데이터 구조 매개변수

numItems

입력. paramArray 배열의 구성 매개변수 수. paramArray의 최대 요소 수를 지정하려면 이 값을 db2CfgMaxParam으로 설정하십시오.

paramArray

입력. db2CfgParam 구조의 포인터

flags 입력. 수행할 조치 유형을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

db2CfgDatabase

데이터베이스 구성의 값을 리턴하도록 지정합니다.

db2CfgDatabaseManager

데이터베이스 관리 프로그램 구성 파일의 값을 리턴하도록 지정합니다.

db2CfgImmediate

메모리에 저장된 구성 매개변수의 현재 값을 리턴합니다.

db2CfgDelayed

디스크의 구성 매개변수 값을 가져옵니다. 다음 데이터베이스 연결이나 인스턴스 접속 시까지는 이 값이 메모리에서 현재 값이 되지 않습니다.

db2CfgGetDefaults

구성 매개변수의 디폴트값을 리턴합니다.

db2CfgReset

디폴트값으로 재설정합니다.

db2CfgSingleDbpartition

특정 데이터베이스 파티션에서 데이터베이스 구성을 갱신하거나 재설정하려면 이 플래그를 설정하고 dbpartitionnum에 대한 값을 제공하십시오.

dbname

입력. 데이터베이스 이름

dbpartitionnum

입력. 이 API가 구성 값을 설정하는 데이터베이스 파티션을 지정합니다.

db2CfgParam 데이터 구조 매개변수

token 입력. 구성 매개변수 ID. db2CfgParam 토큰 요소의 유효한 항목 및 데이터 유형이 ″구성 매개변수 요약″에 나열됩니다.

ptrvalue

출력. 구성 매개변수 값

flags 입력. 요청에서 각 매개변수에 대한 특정 정보를 제공합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

db2CfgParamAutomatic

검색된 매개변수에 자동 값이 있는지를 표시합니다. 지정된 구성 매개변수가 자동으로 설정되었는지 판별하려면 db2ApiDf.h에 정의된 플래그 및 db2CfgParamAutomatic 키워드로 리턴된 값에 대해 부울 AND 연산을 수행하십시오.

db2CfgParamComputed

검색된 매개변수에 계산된 값이 있는지를 표시합니다. 지정된 구성 매

개변수가 계산됨으로 설정되었는지 판별하려면 db2ApiDf.h에 정의된 플래그 및 db2CfgParamComputed 키워드로 리턴된 값에 대해 부울 AND 연산을 수행하십시오.

db2CfgParamManual

매개변수의 자동 또는 계산됨을 설정 해제하는 데 사용하며 매개변수를 현재 값으로 설정합니다. **ptrvalue** 필드는 무시되며 NULL로 설정할 수 있습니다.

db2gCfg 데이터 구조 특정 매개변수

dbname_len

입력. dbname 매개변수의 길이(바이트)

db2gCfgParam 데이터 구조 특정 매개변수

ptrvalue_len

입력. ptrvalue 매개변수의 길이(바이트)

사용 시 참고사항

구성 매개변수 maxagents 및 maxcagents는 더 이상 사용되지 않습니다. 이 구성 매개변수는 추후 릴리스에서 완전히 제거될 수 있습니다.

db2CfgSet API는 maxagents 및 maxcagents 구성 매개변수 갱신을 허용하지만 DB2는 이 갱신사항을 무시합니다.

사용 샘플

사례 1: MAXAPPLS 매개변수가 dbpartitionnum 30에서 50으로 설정됩니다.

사례 2: MAXAPPLS 매개변수가 모든 dbpartitionnum에서 50으로 설정됩니다.

```
int updateDbConfig()
{
    struct sqlca sqlca = {0};
    db2Cfg cfgStruct = {0};
    db2CfgParam cfgParameters[2];
    char *dbAlias = "SAMPLE";

    /* initialize cfgParameters */
    cfgParameters[0].flags = 0;
    cfgParameters[0].token = SQLF_DBTN_TSM_OWNER;
    cfgParameters[0].ptrvalue = (char *)malloc(sizeof(char) * 65);
    cfgParameters[1].flags = 0;
    cfgParameters[1].token = SQLF_DBTN_MAXAPPLS;
    cfgParameters[1].ptrvalue = (char *)malloc(sizeof(sqluint16));

    /* set two DB Config. fields  */
    strcpy(cfgParameters[0].ptrvalue, "tsm_owner");
    *(sqluint16 *)(cfgParameters[1].ptrvalue) = 50;
```

```
    /* initialize cfgStruct to update db cfg on single partition*/
    cfgStruct.numItems = 2;
    cfgStruct.paramArray = cfgParameters;
    cfgStruct.flags = db2CfgDatabase | db2CfgImmediate;
    cfgStruct.flags |= db2CfgSingleDbpartition;
    cfgStruct.dbname = dbAlias;
    cfgStruct.dbpartitionnum = 30;

    /* CASE 1: update database configuration */
    db2CfgSet(db2Version950, (void *)&cfgStruct, &sqlca);

    /* set cfgStruct to update db cfg on all db partitions */
    cfgStruct.flags &= ~db2CfgSingleDbpartition;

    /* CASE 2: update database configuration */
    db2CfgSet(db2Version950, (void *)&cfgStruct, &sqlca);
    ..............
}
```

제 12 장 db2ConvMonStream - 모니터 스트림을 버전 6 이전의 형식으로 변환

단일 논리적 데이터 요소의 새로 추가된 자체 기술적 형식(예: SQLM_ELM_DB2)을 해당하는 버전 6 이전의 외부 모니터 구조(예: sqlm_db2)로 변환합니다. 버전 5 이후의 스트림을 사용하여 API 호출을 업그레이드하는 경우 새 스트림 형식을 사용하여 모니터 데이터를 트래버스해야 합니다(예: SQLM_ELM_DB2 요소를 찾아야 함). 그러면 스트림의 이 분할 영역은 변환 API로 전달되어 관련된 버전 6 이전의 데이터를 가져올 수 있습니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2ConvMonStream (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ConvMonStreamData
{
   void *poTarget;
   struct sqlm_header_info *piSource;
   db2Uint32 iTargetType;
   db2Uint32 iTargetSize;
   db2Uint32 iSourceType;
} db2ConvMonStreamData;

SQL_API_RC SQL_API_FN
  db2gConvMonStream (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

db2ConvMonStream API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2ConvMonStreamData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ConvMonStreamData 데이터 구조 매개변수

poTarget

출력. 목표 모니터 출력 구조의 포인터(예: sqlm_db2). 출력 유형 및 해당 입력 유형 목록이 아래 제공됩니다.

piSource

입력. 변환 중인 논리 데이터 요소의 포인터(예: SQLM_ELM_DB2). 출력 유형 및 해당 입력 유형 목록이 아래 제공됩니다.

iTargetType

입력. 수행 중인 변환 유형. SQLM_DB2_SS 인스턴스에 대해 sqlmon.h에 v5 유형의 값을 지정합니다.

iTargetSize

입력. 이 매개변수는 일반적으로 poTarget으로 지시되는 구조 *크기*로 설정할 수 있습니다. 그러나 일반적으로 구조 끝에서 오프셋 값으로 참조되는 요소의 경우(예: sqlm_stmt의 명령문 텍스트) sqlm_stmt 정적 *크기* 요소 뿐만 아니라 추출되는 가장 큰 *크기*의 명령문도 포함할 수 있는 *크기*로 버퍼를 지정하십시오. 즉, SQL_MAX_STMT_SIZ 더하기 sizeof(sqlm_stmt).

iSourceType

입력. 소스 스트림 유형. 유효한 값은 SQLM_STREAM_SNAPSHOT (스냅샷 스트림) 또는 SQLM_STREAM_EVMON(이벤트 모니터 스트림)입니다.

사용 시 참고사항

다음은 지원되는 변환 가능한 데이터 요소 목록입니다.

표 6. 지원되는 변환 가능한 데이터 요소: 스냅샷 변수

스냅샷 변수 데이터 스트림 유형	구조
SQLM_ELM_APPL	sqlm_appl
SQLM_ELM_APPL_INFO	sqlm_applinfo
SQLM_ELM_DB2	sqlm_db2
SQLM_ELM_FCM	sqlm_fcm

표 6. 지원되는 변환 가능한 데이터 요소: 스냅샷 변수 (계속)

스냅샷 변수 데이터 스트림 유형	구조
SQLM_ELM_FCM_NODE	sqlm_fcm_node
SQLM_ELM_DBASE	sqlm_dbase
SQLM_ELM_TABLE_LIST	sqlm_table_header
SQLM_ELM_TABLE	sqlm_table
SQLM_ELM_DB_LOCK_LIST	sqlm_dbase_lock
SQLM_ELM_APPL_LOCK_LIST	sqlm_appl_lock
SQLM_ELM_LOCK	sqlm_lock
SQLM_ELM_STMT	sqlm_stmt
SQLM_ELM_SUBSECTION	sqlm_subsection
SQLM_ELM_TABLESPACE_LIST	sqlm_tablespace_header
SQLM_ELM_TABLESPACE	sqlm_tablespace
SQLM_ELM_ROLLFORWARD	sqlm_rollfwd_info
SQLM_ELM_BUFFERPOOL	sqlm_bufferpool
SQLM_ELM_LOCK_WAIT	sqlm_lockwait
SQLM_ELM_DCS_APPL	sqlm_dcs_appl, sqlm_dcs_applid_info, sqlm_dcs_appl_snap_stats, sqlm_xid, sqlm_tpmon
SQLM_ELM_DCS_DBASE	sqlm_dcs_dbase
SQLM_ELM_DCS_APPL_INFO	sqlm_dcs_applid_info
SQLM_ELM_DCS_STMT	sqlm_dcs_stmt
SQLM_ELM_COLLECTED	sqlm_collected

표 7. 지원되는 변환 가능한 데이터 요소: 이벤트 모니터 변수

이벤트 모니터 변수 데이터 스트림 유형	구조
SQLM_ELM_EVENT_DB	sqlm_db_event
SQLM_ELM_EVENT_CONN	sqlm_conn_event
SQLM_ELM_EVENT_TABLE	sqlm_table_event
SQLM_ELM_EVENT_STMT	sqlm_stmt_event
SQLM_ELM_EVENT_XACT	sqlm_xaction_event
SQLM_ELM_EVENT_DEADLOCK	sqlm_deadlock_event
SQLM_ELM_EVENT_DLCONN	sqlm_dlconn_event
SQLM_ELM_EVENT_TABLESPACE	sqlm_tablespace_event
SQLM_ELM_EVENT_DBHEADER	sqlm_dbheader_event
SQLM_ELM_EVENT_START	sqlm_evmon_start_event
SQLM_ELM_EVENT_CONNHEADER	sqlm_connheader_event
SQLM_ELM_EVENT_OVERFLOW	sqlm_overflow_event
SQLM_ELM_EVENT_BUFFERPOOL	sqlm_bufferpool_event
SQLM_ELM_EVENT_SUBSECTION	sqlm_subsection_event
SQLM_ELM_EVENT_LOG_HEADER	sqlm_event_log_header

sqlm_rollfwd_ts_info 구조는 변환되지 않습니다. 스트림에서 직접 액세스할 수 있는 테이블 스페이스 이름만 포함합니다. sqlm_agent 구조도 변환되지 않습니다. 이는 스트림에서 직접 액세스할 수 있는 에이전트의 pid만 포함합니다.

제 13 장 db2DatabasePing - 네트워크 응답 시간을 테스트하기 위해 데이터베이스 Ping

클라이언트와 데이터베이스 서버 사이의 기본 연결에 대한 네트워크 응답 시간을 테스트합니다. 호스트 데이터베이스 서버가 직접 또는 게이트웨이를 통해 DB2® Connect™에 액세스할 때 응용프로그램이 이 API를 사용합니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2DatabasePing (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DatabasePingStruct
{
   char iDbAlias[SQL_ALIAS_SZ + 1];
   db2int32 RequestPacketSz;
   db2int32 ResponsePacketSz;
   db2Uint16 iNumIterations;
   db2Uint32 *poElapsedTime;
} db2DatabasePingStruct;

SQL_API_RC SQL_API_FN
  db2gDatabasePing (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDatabasePingStruct
{
   db2Uint16 iDbAliasLength;
   char iDbAlias[SQL_ALIAS_SZ + 1];
   db2int32 RequestPacketSz;
   db2int32 ResponsePacketSz;
   db2Uint16 iNumIterations;
   db2Uint32 *poElapsedTime;
} db2gDatabasePingStruct;
```

db2DatabasePing API 매개변수

versionNumber

입력. 응용프로그램이 사용 중인 DB2 데이터베이스 또는 DB2 Connect 제품의 버전 및 릴리스를 지정합니다.

pParmStruct

입력. db2DatabasePingStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2DatabasePingStruct 데이터 구조 매개변수

iDbAlias

입력. 데이터베이스 별명 이름. 나중에 사용할 수 있도록 예약되어 있습니다. 값을 입력해도 무시됩니다.

RequestPacketSz

입력. 서버에 송신하는 패킷 *크기*(바이트). *크기*는 0 - 32767(포함)이어야 합니다. 이 매개변수는 Linux, UNIX 및 Windows용 DB2® Universal Database™(UDB) 버전 8 이상 또는 z/OS®용 DB2 UDB 버전 8 이상을 실행 중인 서버에서만 유효합니다.

ResponsePacketSz

입력. 클라이언트로 다시 리턴되는 패킷 *크기*(바이트). *크기*는 0 - 32767(포함)이어야 합니다. 이 매개변수는 Linux, UNIX 및 Windows용 DB2 UDB 버전 8 이상 또는 z/OS용 DB2 UDB 버전 8 이상을 실행 중인 서버에서만 유효합니다.

iNumIterations

입력. 테스트 요청 반복 수. 값은 1 - 32767(포함)이어야 합니다.

poElapsedTime

출력. 요소 수가 iNumIterations와 동일한 32비트 정수 배열의 포인터. 배열의 각 요소에는 한 테스트 요청 반복에 대한 경과 시간(마이크로초)이 포함됩니다.

주: 응용프로그램은 이 API를 호출하기 전에 이 배열에 대한 메모리를 할당해야 합니다.

db2gDatabasePingStruct 데이터 구조 특정 매개변수

iDbAliasLength

입력. 데이터베이스 별명 이름 길이. 나중에 사용하기 위해 예약됨

사용 시 참고사항

이 API를 DB2 호스트 데이터베이스 서버에 연결된 DB2 Connect 버전 8을 통해 DB2 UDB 버전 7 클라이언트에서 사용하는 경우 작동하지 않습니다.

제 14 장 db2DatabaseQuiesce - 데이터베이스 Quiesce

모든 사용자가 데이터베이스를 강제로 종료하고 즉시 활성 트랜잭션을 모두 롤백하거나 지정된 시간(분) 내에 현재 작업 단위(UOW)가 완료되도록 대기하며(지정된 시간 내에 완료되지 않으면 조작이 실패) 데이터베이스를 Quiesce 모드로 설정합니다. 이 API는 데이터베이스에 대한 독점 액세스를 제공합니다. 이 Quiesce 상태에서 시스템 관리는 적절한 권한이 있는 사용자가 데이터베이스에서 수행할 수 있습니다. 관리가 완료되면 db2DatabaseUnquiesce API를 사용하여 데이터베이스를 Unquiesce할 수 있습니다. db2DatabaseUnquiesce API를 사용하면 다른 사용자가 데이터베이스를 종료하고 다른 데이터베이스를 시작하지 않고도 데이터베이스에 연결할 수 있습니다. 이 모드에서는 QUIESCE CONNECT 권한 및 sysadm, sysmaint 또는 sysctrl이 있는 사용자나 그룹만 데이터베이스 및 해당 오브젝트에 액세스합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- dbadm

필수 연결

데이터베이스

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2DatabaseQuiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DbQuiesceStruct
{
  char *piDatabaseName;
  db2Uint32 iImmediate;
  db2Uint32 iForce;
  db2Uint32 iTimeout;
} db2DbQuiesceStruct;

SQL_API_RC SQL_API_FN
  db2gDatabaseQuiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
```

```
            struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDbQuiesceStruct
{
  db2Uint32 iDatabaseNameLen;
  char *piDatabaseName;
  db2Uint32 iImmediate;
  db2Uint32 iForce;
  db2Uint32 iTimeout;
} db2gDbQuiesceStruct;
```

db2DatabaseQuiesce API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2DbQuiesceStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2DbQuiesceStruct 데이터 구조 매개변수

piDatabaseName

입력. 데이터베이스 이름

iImmediate

입력. 가능한 값은 다음과 같습니다.

TRUE=1

응용프로그램을 즉시 강제로 실행합니다.

FALSE=0

지연된 강제 실행. 응용프로그램은 iTimeout 매개변수로 지정한 시간(분) 동안 대기하여 현재 작업 단위(UOW)가 완료되도록 한 다음 종료됩니다. iTimeout 매개변수로 지정한 시간(분) 동안 이 지연된 강제 실행을 완료할 수 없으면 Quiesce 조작이 실패합니다.

iForce 입력. 나중에 사용하기 위해 예약됨

iTimeout

입력. 응용프로그램이 현재 작업 단위(UOW)를 커미트할 때까지 대기하는 시간(분)을 지정합니다. iTimeout을 지정하지 않은 경우 단일 파티션 데이터베이스 환경에서 디폴트값은 10분입니다. 파티션된 데이터베이스 환경에서는 start_stop_time 데이터베이스 관리 프로그램 구성 매개변수로 지정한 값이 사용됩니다.

db2gDbQuiesceStruct 데이터 구조 특정 매개변수

iDatabaseNameLen

입력. piDatabaseName 길이를 바이트 단위로 지정합니다.

제 15 장 db2DatabaseRestart - 데이터베이스 재시작

비정상적으로 종료되어 불일치 상태인 데이터베이스를 재시작합니다. 이 API가 제대로 완료되면 사용자에게 CONNECT 특권이 있는 경우 응용프로그램이 계속 데이터베이스에 연결되어 있습니다.

범위

이 API는 실행된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

없음

필수 연결

이 API는 데이터베이스 연결을 작성합니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2DatabaseRestart (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2RestartDbStruct
{
   char *piDatabaseName;
   char *piUserId;
   char *piPassword;
   char *piTablespaceNames;
   db2int32 iOption;
} db2RestartDbStruct;

SQL_API_RC SQL_API_FN
  db2gDatabaseRestart (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2gRestartDbStruct
{
   db2Uint32 iDatabaseNameLen;
   db2Uint32 iUserIdLen;
   db2Uint32 iPasswordLen;
   db2Uint32 iTablespaceNamesLen;
```

```
   char *piDatabaseName;
   char *piUserId;
   char *piPassword;
   char *piTablespaceNames;
} db2gRestartDbStruct;
```

db2DatabaseRestart API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParamStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParamStruct

입력. db2RestartDbStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2RestartDbStruct 데이터 구조 매개변수

piDatabaseName

입력. 재시작할 데이터베이스 별명이 포함된 문자열의 포인터

piUserId

입력. 응용프로그램의 사용자 이름이 포함된 문자열의 포인터 NULL도 가능합니다.

piPassword

입력. 지정한 사용자 이름이 있는 경우 해당 암호가 포함된 문자열의 포인터(있는 경우). NULL도 가능합니다.

piTablespaceNames

입력. 재시작 조작 중에 삭제되는 테이블 스페이스 이름 목록이 포함된 문자열의 포인터. NULL도 가능합니다.

iOption

입력. 가능한 값은 다음과 같습니다.

DB2_DB_SUSPEND_NONE

일반 응급 복구를 수행합니다.

DB2_DB_RESUME_WRITE

입출력 쓰기가 일시중단된 데이터베이스의 응급 복구 수행에 필요합니다.

db2gRestartDbStruct 데이터 구조 특정 매개변수

iDatabaseNameLen

입력. piDatabaseName 매개변수 길이(바이트)

iUserIdLen

입력. piUserId 매개변수 길이(바이트)

iPasswordLen

입력. piPassword 매개변수 길이(바이트)

iTablespaceNamesLen

입력. piTablespaceNames 매개변수 길이(바이트)

사용 시 참고사항

데이터베이스 연결 시도 중에 데이터베이스를 재시작하라는 오류 메시지가 리턴되면 이 API를 호출하십시오. 이전 세션에서 이 데이터베이스가 비정상적으로 종료된 경우에만 (전원 장애 등으로) 이 조치가 발생합니다.

이 API가 제대로 완료되면 사용자에게 CONNECT 특권이 있는 경우 데이터베이스의 공유 연결이 유지되고 인다우트(Indoubt) 트랜잭션이 있는 경우 SQL 경고가 발행됩니다. 이 경우, 데이터베이스는 여전히 사용 가능하지만 데이터베이스의 최종 연결이 삭제되기 전에 인다우트(Indoubt) 트랜잭션이 해결되지 않으면 데이터베이스를 다시 사용하기 전에 API의 다른 호출이 완료되어야 합니다.

순환 로그의 경우 테이블 스페이스에 입출력 오류, 파일 시스템 언마운트 등과 같은 문제가 있는 경우 데이터베이스 재시작 조작이 실패합니다. 이러한 테이블 스페이스 손실이 문제가 되지 않는 경우 해당 이름을 명시적으로 지정할 수 있습니다. 그러면 해당하는 테이블 스페이스는 삭제 보류 상태가 되고 조작은 제대로 완료될 수 있습니다.

REXX API 구문

```
RESTART DATABASE database_alias [USER username USING password]
```

REXX API 매개변수

database_alias

재시작하려는 데이터베이스 별명

username

데이터베이스가 재시작되는 사용자 이름

password

사용자 이름을 인증하는 데 사용되는 암호

제 16 장 db2DatabaseUnquiesce - 데이터베이스 Unquiesce

유지보수 또는 기타 이유로 Quiesce 상태에 있는 데이터베이스로 사용자 액세스를 리스토어합니다. 데이터베이스를 종료한 다음 재시작하지 않고도 사용자 액세스를 리스토어합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- dbadm

필수 연결

데이터베이스

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2DatabaseUnquiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DbUnquiesceStruct
{
             char *piDatabaseName;
} db2DbUnquiesceStruct;

SQL_API_RC SQL_API_FN
  db2gDatabaseUnquiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDbUnquiesceStruct
{
             db2Uint32 iDatabaseNameLen;
             char *piDatabaseName;
} db2gDbUnquiesceStruct;
```

db2DatabaseUnquiesce API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2DbUnquiesceStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2DbUnquiesceStruct 데이터 구조 매개변수

piDatabaseName

입력. 데이터베이스 이름

db2gDbUnquiesceStruct 데이터 구조 특정 매개변수

iDatabaseNameLen

입력. piDatabaseName 길이를 바이트 단위로 지정합니다.

제 17 장 db2DbDirCloseScan - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 종료

db2DbDirOpenScan으로 할당된 자원을 사용 가능하게 합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2DbDirCloseScan (
        db2Uint32 versionNumber,
        void * pParmStruct,
        struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DbDirCloseScanStruct
{
   db2Uint16 iHandle;
} db2DbDirCloseScanStruct;

SQL_API_RC SQL_API_FN
  db2gDbDirCloseScan (
        db2Uint32 versionNumber,
        void * pParmStruct,
        struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDbDirCloseScanStruct
{
   db2Uint16 iHandle;
} db2gDbDirCloseScanStruct;
```

db2DbDirCloseScan API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2DbDirCloseScanStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2DbDirCloseScanStruct 데이터 구조 매개변수

iHandle

입력. 관련 db2DbDirOpenScan API에서 리턴된 ID

제 18 장 db2DbDirGetNextEntry - 다음 시스템 또는 로컬 데이터베이스 디렉토리 항목 가져오기

db2DbDirOpenScan에서 리턴된 시스템 데이터베이스 디렉토리 또는 로컬 데이터베이스 디렉토리 사본의 다음 항목이 리턴됩니다. 이 API에 대한 후속 호출은 추가 항목을 리턴합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2DbDirGetNextEntry (
        db2Uint32 versionNumber,
        void * pParmStruct,
        struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DbDirNextEntryStructV9
{
   db2Uint16 iHandle;
   struct db2DbDirInfoV9 *poDbDirEntry;
} db2DbDirNextEntryStructV9;

SQL_STRUCTURE db2DbDirInfoV9
{
   _SQLOLDCHAR alias[SQL_ALIAS_SZ];
   _SQLOLDCHAR dbname[SQL_DBNAME_SZ];
   _SQLOLDCHAR drive[SQL_DB_PATH_SZ];
   _SQLOLDCHAR intname[SQL_INAME_SZ];
   _SQLOLDCHAR nodename[SQL_NNAME_SZ];
   _SQLOLDCHAR dbtype[SQL_DBTYP_SZ];
   _SQLOLDCHAR comment[SQL_CMT_SZ];
   short com_codepage;
   _SQLOLDCHAR type;
   unsigned short authentication;
   char glbdbname[SQL_DIR_NAME_SZ];
   _SQLOLDCHAR dceprincipal[SQL_DCEPRIN_SZ];
   short cat_nodenum;
   short nodenum;
   _SQLOLDCHAR althostname[SQL_HOSTNAME_SZ];
   _SQLOLDCHAR altportnumber[SQL_SERVICE_NAME_SZ];
```

```
};

SQL_API_RC SQL_API_FN
  db2gDbDirGetNextEntry (
        db2Uint32 versionNumber,
        void * pParmStruct,
        struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDbDirNextEntryStrV9
{
   db2Uint16 iHandle;
   struct db2DbDirInfoV9 *poDbDirEntry;
} db2gDbDirNextEntryStrV9;
```

db2DbDirGetNextEntry API 매개변수

versionNumber

입력. 두 번째 매개변수인 **pParmStruct**로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2DbDirGetNextEntryStructV9 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2DbDirNextEntryStructV9 데이터 구조 매개변수

iHandle

입력. 관련 db2DbDirOpenScan API에서 리턴된 ID

poDbDirEntry

출력. db2DbDirInfoV9 구조의 포인터. 디렉토리 데이터의 스페이스는 API가 할당하며 해당 스페이스의 포인터가 호출자에게 리턴됩니다.

db2DbDirInfoV9 데이터 구조 매개변수

alias 대체 데이터베이스 이름

dbname

데이터베이스 이름

drive 데이터베이스가 있는 로컬 데이터베이스 디렉토리 경로 이름. 이 필드는 시스템 데이터베이스 디렉토리가 스캔에 사용되는 경우에만 리턴됩니다.

주: Windows에서 이 매개변수는 CHAR(12)입니다.

intname

데이터베이스 서브디렉토리를 식별하는 토큰. 이 필드는 로컬 데이터베이스 디렉토리가 스캔에 사용되는 경우에만 리턴됩니다.

nodename

데이터베이스가 위치한 노드 이름. 이 필드는 카탈로그된 데이터베이스가 리모트 데이터베이스인 경우에만 리턴됩니다.

dbtype

데이터베이스 관리 프로그램 릴리스 정보.

comment

데이터베이스에 연관된 주석

com_codepage

주석의 코드 페이지. 사용되지 않음

유형 항목 유형. 가능한 값은 다음과 같습니다.

SQL_INDIRECT

현재 인스턴스로 작성된 데이터베이스(**DB2INSTANCE** 환경 변수 값으로 정의)

SQL_REMOTE

데이터베이스가 다른 인스턴스에 상주합니다.

SQL_HOME

데이터베이스가 이 볼륨에 상주합니다(항상 로컬 데이터베이스 디렉토리의 HOME)

SQL_DCE

데이터베이스가 DCE 디렉토리에 상주합니다.

authentication

인증 유형. 가능한 값은 다음과 같습니다.

SQL_AUTHENTICATION_SERVER

사용자 이름 및 암호 인증이 서버에서 수행됩니다.

SQL_AUTHENTICATION_CLIENT

사용자 이름 및 암호 인증이 클라이언트에서 수행됩니다.

SQL_AUTHENTICATION_DCE

DCE 보안 서비스를 사용하여 인증이 수행됩니다.

SQL_AUTHENTICATION_KERBEROS

Kerberos 보안 메커니즘을 사용하여 인증이 수행됩니다.

SQL_AUTHENTICATION_NOT_SPECIFIED

DB2에서는 더 이상 인증이 데이터베이스 디렉토리에서 필요하지 않습니다. 이전의(DB2 V2 이하) 서버가 아닌 다른 서버에 연결된 경우에 이 값을 지정하십시오.

SQL_AUTHENTICATION_SVR_ENCRYPT
목표 데이터베이스를 포함하는 노드에서 인증이 수행되고 인증 암호가 암호화되도록 지정합니다.

SQL_AUTHENTICATION_DATAENC
목표 데이터베이스가 들어있는 노드에서 인증이 발생하고 연결이 데이터 암호화를 사용해야 하도록 지정합니다.

SQL_AUTHENTICATION_GSSPLUGIN
인증에서 외부 GSS API 기반 플러그인 보안 메커니즘을 사용하도록 지정합니다.

glbdbname
항목이 SQL_DCE 유형인 경우 전역(DCE) 디렉토리에서 목표 데이터베이스의 전역 이름

dceprincipal
인증의 유형이 DCE 또는 Kerberos인 경우 핵심부 이름

cat_nodenum
카탈로그 노드 번호

nodenum
노드 번호

althostname
장애 복구 시에 데이터베이스가 다시 연결되는 대체 서버의 호스트 이름 또는 IP 주소

altportnumber
장애 복구 시에 데이터베이스가 다시 연결되는 대체 서버의 포트 번호

사용 시 참고사항

디렉토리 항목 정보 버퍼의 모든 필드 오른쪽은 공백으로 채워집니다.

후속 db2DbDirGetNextEntry는 현재 항목 다음에 항목을 확보합니다.

스캔할 항목이 더 이상 없는 경우에 db2DbDirGetNextEntry가 호출되면 SQLCA에 SQL1014N이 설정됩니다.

스캔 수와 항목 계수가 동일할 때까지 db2DbDirOpenScan API가 리턴하는 계수 값은 한 번에 하나씩 db2DbDirGetNextEntry 호출을 발행하여 전체 디렉토리를 스캔하는 데 사용할 수 있습니다.

제 19 장 db2DbDirOpenScan - 시스템 또는 로컬 데이터베이스 디렉토리 스캔 시작

시스템 데이터베이스 디렉토리 또는 로컬 데이터베이스 디렉토리의 사본을 메모리에 저장하고 다수의 항목을 리턴합니다. 이 사본은 디렉토리를 연 시점의 디렉토리 스냅샷을 나타냅니다. 이 사본은 디렉토리 자체가 이후에 변경되어도 갱신되지 않습니다.

db2DbDirGetNextEntry API를 사용하여 데이터베이스 항목에 대한 정보를 평가하면서 데이터베이스 디렉토리를 통해 진행하십시오. db2DbDirCloseScan API를 사용하여 스캔을 닫으십시오. 그러면 메모리에서 디렉토리 사본이 제거됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2DbDirOpenScan (
        db2Uint32 versionNumber,
        void * pParmStruct,
        struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DbDirOpenScanStruct
{
   char *piPath;
   db2Uint16 oHandle;
   db2Uint16 oNumEntries;
} db2DbDirOpenScanStruct;

SQL_API_RC SQL_API_FN
  db2gDbDirOpenScan (
        db2Uint32 versionNumber,
        void * pParmStruct,
        struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gDbDirOpenScanStruct
{
   db2Uint32 iPath_len;
```

```
   char *piPath;
   db2Uint16 oHandle;
   db2Uint16 oNumEntries;
} db2gDbDirOpenScanStruct;
```

db2DbDirOpenScan API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2DbDirOpenScanStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2DbDirOpenScanStruct 데이터 구조 매개변수

piPath

입력. 로컬 데이터베이스 디렉토리가 있는 경로 이름. 지정한 경로가 NULL 포인터인 경우 시스템 데이터베이스 디렉토리가 사용됩니다.

oHandle

출력. 리턴된 ID의 2바이트 영역. 이 ID는 데이터베이스 항목을 스캔하려면 db2DbDirGetNextEntry API에 전달하고 자원을 릴리스하려면 db2DbDirCloseScan API에 전달해야 합니다.

oNumEntries

출력. 디렉토리 항목 수가 리턴되는 2바이트 영역

db2gDbDirOpenScanStruct 데이터 구조 특정 매개변수

iPath_len

입력. piPath 매개변수의 길이(바이트)

사용 시 참고사항

이 API로 할당되는 스토리지는 db2DbDirCloseScan API로 사용 가능해집니다.

다중 db2DbDirOpenScan API는 동일한 디렉토리에 대해 발행할 수 있습니다. 그러나 결과는 동일하지 않을 수 있습니다. 디렉토리는 열 때마다 달라질 수 있습니다.

프로세스별로 최대 8개의 데이터베이스 디렉토리가 열려 있을 수 있습니다.

제 20 장 db2DropContact - 통지 메시지를 수신할 수 있는 문의처 목록에서 문의처 제거

문의처 목록에서 문의처를 제거합니다. 문의처는 통지 메시지를 수신할 수 있는 사용자입니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2DropContact (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DropContactData
{
   char *piUserid;
   char *piPassword;
   char *piName;
} db2DropContactData;
```

db2DropContact API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2DropContactData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2DropContactData 데이터 구조 매개변수

piUserid

입력. 사용자 이름

piPassword

입력. piUserid의 암호

piName

입력. 삭제할 문의처 이름

사용 시 참고사항

이 API는 UNIX 및 Linux에서는 지원되지 않습니다. 그러나 SQL 인터페이스를 통해 동일한 기능에 액세스할 수 있습니다.

제 21 장 db2DropContactGroup - 통지 메시지를 수신할 수 있는 문의처 목록에서 문의처 그룹 제거

문의처 목록에서 문의처 그룹을 제거합니다. 문의처 그룹에는 통지 메시지를 수신할 수 있는 사용자 목록이 포함됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2DropContactGroup (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2DropContactData
{
   char *piUserid;
   char *piPassword;
   char *piName;
} db2DropContactData;
```

db2DropContactGroup API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2DropContactData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2DropContactData 데이터 구조 매개변수

piUserid

입력. 사용자 이름

piPassword

입력. piUserid의 암호

piName

입력. 삭제할 문의처 이름

사용 시 참고사항

이 API는 UNIX 및 Linux에서는 지원되지 않습니다. 그러나 SQL 인터페이스를 통해 동일한 기능에 액세스할 수 있습니다.

제 22 장 db2Export - 데이터베이스에서 데이터 익스포트

데이터베이스의 데이터를 여러 외부 파일 형식 중 하나로 익스포트합니다. 사용자는 SQL SELECT문을 제공하거나 유형이 지정된 테이블에 대한 계층 정보를 제공하여 익스포트할 데이터를 지정합니다.

권한 부여

다음 중 하나가 필요합니다.

- *DATAACESS* 권한
- 각 파티션 테이블이나 뷰의 CONTROL 또는 SELECT 특권

레이블 기반 액세스 제어(LBAC)가 이 함수에 강제로 사용됩니다. LBAC로 데이터가 보호되는 경우 익스포트되는 데이터는 호출자의 LBAC 증명서로 제한됩니다.

필수 연결

데이터베이스. 내재된 연결이 사용 가능한 경우 디폴트 데이터베이스에 연결됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Export (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ExportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqlu_media_list *piLobFileList;
   struct sqldcol *piDataDescriptor;
   struct sqllob *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ExportOut *poExportInfoOut;
   struct db2ExportIn *piExportInfoIn;
   struct sqlu_media_list *piXmlPathList;
   struct sqlu_media_list *piXmlFileList;
} db2ExportStruct;

typedef SQL_STRUCTURE db2ExportIn
{
```

```
   db2Uint16 *piXmlSaveSchema;
} db2ExportIn;

typedef SQL_STRUCTURE db2ExportOut
{
   db2Uint64 oRowsExported;
} db2ExportOut;

SQL_API_RC SQL_API_FN
  db2gExport (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gExportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqlu_media_list *piLobFileList;
   struct sqldcol *piDataDescriptor;
   struct sqllob *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ExportOut *poExportInfoOut;
   db2Uint16 iDataFileNameLen;
   db2Uint16 iFileTypeLen;
   db2Uint16 iMsgFileNameLen;
   struct db2ExportIn *piExportInfoIn;
   struct sqlu_media_list *piXmlPathList;
   struct sqlu_media_list *piXmlFileList;
} db2gExportStruct;
```

db2Export API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2ExportStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ExportStruct 데이터 구조 매개변수

piDataFileName

입력. 데이터가 익스포트되는 외부 파일의 경로 및 이름이 포함된 문자열

piLobPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 LOB 파일이 저장되는 클라이언트 경로를 나열하는

sqlu_media_list 구조의 포인터. 익스포트되는 LOB는 sqlu_media_entry 구조에 나열되는 모든 경로에서 균등하게 분산됩니다.

piLobFileList

입력. 해당 media_type 필드가 SQLU_CLIENT_LOCATION으로 설정되고 해당 sqlu_location_entry 구조에는 기본 파일 이름이 포함되는 sqlu_media_list 구조의 포인터.

목록의 첫 번째 이름을 사용하여 이름 스페이스가 모두 사용된 경우 API는 두 번째 이름을 사용하며 계속 이와 같이 합니다. 익스포트 조작 중에 LOB 파일을 작성할 경우, 파일 이름은 이 목록에서 현재 기본 이름을 현재 경로(piLobPathList에서)에 추가하고 3자의 시퀀스 번호를 .lob 확장에 추가하여 작성됩니다. 예를 들어 현재 LOB 경로가 /u/foo/lob/path 디렉토리이고 현재 LOB 파일 이름은 bar이며 LOBSINSEPFILES 파일 유형 수정자가 설정된 경우 작성된 LOB 파일은 /u/foo/LOB/path/bar.001.lob, /u/foo/LOB/path/bar.002.lob 등이 됩니다. LOBSINSEPFILES 파일 유형 수정자가 설정되지 않은 경우 모든 LOB 문서는 병합되어 한 개의 /u/foo/lob/path/bar.001.lob 파일에 포함됩니다.

piDataDescriptor

입력. 출력 파일의 컬럼 이름을 지정하는 sqldcol 구조의 포인터. dcolmeth 필드 값으로 이 매개변수에서 제공되는 정보의 나머지를 익스포트 유틸리티가 해석하는 방법을 판별합니다. 이 매개변수의 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQL_METH_N

이름. 출력 파일에 사용할 컬럼 이름을 지정합니다.

SQL_METH_D

디폴트. 테이블의 기존 컬럼 이름이 출력 파일에 사용됩니다. 이 경우 컬럼 수 및 컬럼 스펙 배열 모두 무시됩니다. 컬럼 이름은 piActionString에 지정된 SELECT 문 출력에서 파생됩니다.

piActionString

입력. 유효한 동적 SQL SELECT 문을 포함하는 sqllob 구조의 포인터. 구조에는 뒤에 SELECT 문을 구성하는 문자가 오는 4바이트 길이의 필드입니다. SELECT 문은 데이터베이스에서 추출되어 외부 파일로 쓰여지는 데이터를 지정합니다.

외부 파일의 컬럼(piDataDescriptor의) 및 SELECT 문의 데이터베이스 컬럼은 해당하는 각 목록/구조 위치에 따라 일치됩니다. 데이터베이스에서 선택한 데이터의 첫 번째 컬럼은 외부 파일의 첫 번째 컬럼이 되고 해당 컬럼 이름은 외부 컬럼 배열의 첫 번째 요소에서 사용됩니다.

piFileType

입력. 외부 파일 내의 데이터 형식을 나타내는 문자열. 지원되는 외부 파일 형식(sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQL_DEL

컬럼 식별자가 있는 ASCII, dBase, BASIC 및 IBM Personal Decision Series 프로그램과 교환에 사용되는 다수의 데이터베이스 관리 프로그램 및 파일 관리자

SQL_WSF

Lotus® Symphony 및 1-2-3® 프로그램과의 교환을 위한 워크시트 형식(WSF). 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

SQL_IXF

통합 교환 형식의 PC 버전으로 테이블에서 데이터를 익스포트할 때 자주 사용되는 방법입니다. 이 파일 형식으로 익스포트된 데이터는 이후에 동일한 테이블이나 다른 데이터베이스 관리 프로그램 테이블로 임포트 또는 로드할 수 있습니다.

piFileTypeMod

입력. 하나 이상의 처리 옵션을 지정하는 문자 배열이 뒤에 오는 2바이트 길이의 필드를 포함하는 sqldcol 구조의 포인터. 이 포인터가 NULL이거나 지시된 구조에 영(0) 문자가 있는 경우 이 조치는 디폴트 스펙 선택으로 해석됩니다.

지원되는 모든 파일 유형에 모든 옵션을 사용할 수 있는 것은 아닙니다. 아래의 관련 링크인 "익스포트 유틸리티를 위한 파일 유형 수정자"를 참조하십시오.

piMsgFileName

입력. 유틸리티에서 리턴된 오류, 경고 및 정보 메시지 목적지를 포함하는 문자열. 운영 체제 파일 또는 표준 디바이스의 경로 및 이름일 수 있습니다. 파일이 이미 있는 경우 정보가 추가됩니다. 없는 경우에는 파일이 작성됩니다.

iCallerAction

입력. 호출자가 요청한 조치. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_INITIAL

초기 호출. 이 값을 API의 첫 번째 호출에 사용해야 합니다. 초기 호출이나 이후에 임의의 후속 호출이 리턴되고 요청된 익스포트 조작을 완료하기 전에 일부 조치를 수행하는 응용프로그램을 호출해야 하는 경우 호출자 조치는 다음 중 하나로 설정해야 합니다.

SQLU_CONTINUE

처리를 계속합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테

이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 완료되고 유틸리티는 초기 요청 처리를 계속하도록 지정합니다.

SQLU_TERMINATE

처리를 종료합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 수행되지 않고 유틸리티가 초기 요청 처리를 종료하도록 지정합니다.

poExportInfoOut

db2ExportOut 구조의 포인터

piExportInfoIn

입력. db2ExportIn 구조의 포인터

piXmlPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 XML 파일이 저장되는 클라이언트 경로를 나열하는 sqlu_media_list 구조의 포인터. 익스포트되는 XML은 sqlu_media_entry 구조에 나열되는 모든 경로에서 균등하게 분산됩니다.

piXmlFileList

입력. 해당 media_type 필드가 SQLU_CLIENT_LOCATION으로 설정되고 해당 sqlu_location_entry 구조에는 기본 파일 이름이 포함되는 sqlu_media_list 구조의 포인터.

목록의 첫 번째 이름을 사용하여 이름 스페이스가 모두 사용된 경우 API는 두 번째 이름을 사용하며 계속 이와 같이 합니다. 익스포트 조작 중에 XML 파일을 작성할 경우, 파일 이름은 이 목록에서 현재 기본 이름을 현재 경로(piXmlFileList에서)에 추가하고 3자의 시퀀스 번호를 .xml 확장에 추가하여 작성됩니다. 예를 들어 현재 XML 경로가 /u/foo/xml/path 디렉토리이고 현재 XML 파일 이름은 bar이며 XMLINSEPFILES 파일 유형 수정자가 설정된 경우 작성된 XML 파일은 /u/foo/xml/path/bar.001.xml, /u/foo/xml/path/bar.002.xml 등이 됩니다. XMLINSEPFILES 파일 유형 수정자가 설정되지 않은 경우 모든 XML 문서는 병합되어 한 개의 /u/foo/xml/path/bar.001.xml 파일에 포함됩니다.

db2ExportIn 데이터 구조 매개변수

piXmlSaveSchema

입력. 익스포트된 각 XML 문서가 익스포트된 데이터 파일에 저장되었는지 확인하는 데 사용되는 XML 스키마의 SQL ID를 표시합니다. 가능한 값은 참 및 거짓입니다.

db2ExportOut 데이터 구조 매개변수

oRowsExported

출력. 목표 파일에 익스포트된 레코드 수를 리턴합니다.

db2gExportStruct 데이터 구조 특정 매개변수

iDataFileNameLen

입력. 데이터 파일 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

iFileTypeLen

입력. 파일 유형의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

iMsgFileNameLen

입력. 메시지 파일 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

익스포트 조작을 시작하기 전에 다음 중 하나의 방법을 사용하여 모든 테이블 조작을 완료하고 모든 잠금을 해제해야 합니다.

- WITH HOLD 절로 정의된 모든 열려 있는 커서를 닫고 COMMIT 문을 실행하여 데이터 변경을 커미트하십시오.
- ROLLBACK 문을 실행하여 데이터 변경을 롤백하십시오.

SELECT문에서 테이블 별명을 사용할 수 있습니다.

메시지 파일에 넣어진 메시지는 메시지 검색 서비스에서 리턴된 정보를 포함합니다. 각 메시지는 새 행에서 시작합니다.

익스포트 유틸리티에서 경고가 작성되면 메시지가 메시지 파일에 기록되거나 해당 파일이 없는 경우 표준 출력으로 기록됩니다.

외부 컬럼 이름 배열 piDataDescriptor의 컬럼 수(sqldcol 구조의 dcolnum 필드)가 SELECT 문으로 생성된 컬럼 수와 동일하지 않는 경우 경고 메시지가 발행됩니다. 이 경우, 외부 파일에 기록되는 컬럼 수는 두 수 중 작은 수입니다. 초과 데이터베이스 컬럼이나 외부 컬럼 이름은 출력 파일 생성에 사용되지 않습니다.

db2uexpm.bnd 모듈이나 다른 모든 제공된 .bnd 파일은 수동으로 바인드되고 바인더의 형식 옵션은 사용하면 안됩니다.

DB2 Connect를 사용하여 z/OS 및 OS/390®용 DB2, VM 및 VSE용 DB2 및 System i®용 DB2와 같은 DRDA® 서버에서 테이블을 익스포트하는 데 사용할 수 있습니다. PC/IXF 익스포트만 지원됩니다.

데이터베이스 간에 데이터를 이동하려면 PC/IXF 임포트를 사용해야 합니다. 행 구분자가 있는 문자 데이터를 컬럼 식별자가 있는 ASCII(DEL) 파일로 익스포트하며 텍스트 전송 프로그램으로 이를 처리할 경우 행 구분자가 있는 필드는 줄어들거나 확장됩니다.

익스포트 유틸리티는 AIX® 시스템에서 호출될 때 다중 파트 PC/IXF 파일을 작성하지 않습니다.

단일 데이터베이스 테이블의 컨텐츠가 SELECT * FROM tablename으로 시작되는 piActionString 매개변수 및 디폴트 이름을 지정하는 piDataDescriptor 매개변수가 포함된 PC/IXF 파일로 익스포트되는 경우 테이블의 인덱스 정의가 PC/IXF 파일에 포함됩니다. piActionString의 SELECT 절에 조인이 포함된 경우 인덱스가 뷰에 대해서는 저장되지 않습니다. piActionString 매개변수의 WHERE 절, GROUP BY 절 또는 HAVING 절을 사용하여 인덱스는 저장됩니다. 이 모든 경우에 유형이 지정된 테이블에서 익스포트할 때 전체 계층 구조를 익스포트해야 합니다.

제공된 SELECT 문이 SELECT * FROM tablename 양식인 경우 익스포트 유틸리티는 테이블의 NOT NULL WITH DEFAULT 속성을 IXF 파일에 저장합니다.

유형이 지정된 테이블을 익스포트할 때 subselect문은 단지 목표 테이블 이름과 WHERE 절을 지정하여 표현할 수 있습니다. 계층 구조를 익스포트할 때는 Fullselect 및 select-statement를 지정할 수 없습니다.

IXF 이외의 파일 형식의 경우 DB2가 계층 구조를 트래버스하는 방법 및 익스포트할 서브테이블을 트래버스 순서 목록에서 알 수 있기 때문에 이를 지정하는 것이 좋습니다. 이 목록을 지정하지 않으면 해당 계층 구조에 있는 모든 테이블이 익스포트되고 디폴트 순서는 OUTER 순서가 됩니다. 대안은 OUTER 기능이 제공하는 순서인 디폴트 순서를 사용하는 방법입니다.

주: 임포트 조작 동안 동일한 트래버스 순서를 사용하십시오. 로드 유틸리티는 계층 구조 또는 하위 계층 구조를 지원하지 않습니다.

REXX™ API 구문

```
EXPORT :stmt TO datafile OF filetype
[MODIFIED BY :filetmod] [USING :dcoldata]
MESSAGES msgfile [ROWS EXPORTED :number]


CONTINUE EXPORT


STOP EXPORT
```

REXX API 매개변수

stmt 유효한 동적 SQL SELECT 문이 포함된 REXX 호스트 변수. 명령문은 데이터베이스에서 추출하는 데이터를 지정합니다.

datafile

데이터가 익스포트되는 파일 이름

filetype

익스포트 파일의 데이터 형식. 지원되는 파일 형식은 다음과 같습니다.

DEL 컬럼 식별자가 있는 ASCII

WSF 워크시트 형식. 이 파일 형식에 대한 지원은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다. 지원이 제거되기 전에 WSF 파일 대신 지원되는 파일 형식을 사용하여 시작하는 것이 좋습니다.

IXF 통합 교환 형식의 PC 버전

filetmod

추가 처리 옵션이 포함된 호스트 변수

dcoldata

익스포트 파일에서 사용되는 컬럼 이름이 포함된 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름입니다.

XXX.0

컬럼 수(나머지 변수의 요소 수)

XXX.1

첫 번째 컬럼 이름

XXX.2

두 번째 컬럼 이름

XXX.3

기타.

이 매개변수가 NULL이거나 dcoldata의 값을 지정하지 않은 경우 유틸리티는 데이터베이스 테이블의 컬럼 이름을 사용합니다.

msgfile

오류 및 경고 메시지를 보낸 파일, 경로 또는 디바이스 이름

number

익스포트한 행 번호가 포함된 호스트 변수

제 23 장 db2GetAlertCfg - Health 표시기의 경보 구성 설정 가져오기

Health 표시기의 경보 구성 설정을 리턴합니다.

중요사항: Health Monitor가 버전 9.7에서 사용되지 않으므로
이 명령 또는 API는 사용되지 않으며 이후 릴리스에서 제거될 수 있습니다. 자세한 정보는 *버전 9.7의 새로운 내용* 책에 있는 『Health Monitor는 사용되지 않음』 주제를 참조하십시오.

권한 부여

없음

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetAlertCfg (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetAlertCfgData
{
   db2Uint32 iObjType;
   char *piObjName;
   db2Uint32 iDefault;
   char *piDbName;
   db2Uint32 ioNumIndicators;
   struct db2GetAlertCfgInd *pioIndicators;
} db2GetAlertCfgData;

typedef SQL_STRUCTURE db2GetAlertCfgInd
{
   db2Uint32 ioIndicatorID;
   sqlint64 oAlarm;
   sqlint64 oWarning;
   db2Uint32 oSensitivity;
   char *poFormula;
   db2Uint32 oActionEnabled;
   db2Uint32 oCheckThresholds;
   db2Uint32 oNumTaskActions;
```

```
   struct db2AlertTaskAction *poTaskActions;
   db2Uint32 oNumScriptActions;
   struct db2AlertScriptAction *poScriptActions;
   db2Uint32 oDefault;
} db2GetAlertCfgInd;

typedef SQL_STRUCTURE db2AlertTaskAction
{
   char *pTaskName;
   db2Uint32 condition;
   char *pUserID;
   char *pPassword;
   char *pHostName;
} db2AlertTaskAction;

typedef SQL_STRUCTURE db2AlertScriptAction
{
   db2Uint32 scriptType;
   db2Uint32 condition;
   char *pPathName;
   char *pWorkingDir;
   char *pCmdLineParms;
   char stmtTermChar;
   char *pUserID;
   char *pPassword;
   char *pHostName;
} db2AlertScriptAction;
```

db2GetAlertCfg API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2GetAlertCfgData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2GetAlertCfgData 데이터 구조 매개변수

iObjType

입력. 구성이 요청되는 오브젝트 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2ALERTCFG_OBJTYPE_DBM
- DB2ALERTCFG_OBJTYPE_DATABASES
- DB2ALERTCFG_OBJTYPE_TABLESPACES
- DB2ALERTCFG_OBJTYPE_TS_CONTAINERS
- DB2ALERTCFG_OBJTYPE_DATABASE
- DB2ALERTCFG_OBJTYPE_TABLESPACE

- DB2ALERTCFG_OBJTYPE_TS_CONTAINER

piObjName

입력. 오브젝트 유형 iObjType이 DB2ALERTCFG_OBJTYPE_TABLESPACE 또는 DB2ALERTCFG_OBJTYPE_TS_CONTAINER로 설정된 경우 테이블 스페이스 또는 테이블 스페이스 컨테이너의 이름.

iDefault

입력. 디폴트 설치 구성 값이 검색되는 것을 나타냅니다.

piDbname

입력. 오브젝트 유형 iObjType이 DB2ALERTCFG_OBJTYPE_TS_CONTAINER, DB2ALERTCFG_OBJTYPE_TABLESPACE 및 DB2ALERTCFG_OBJTYPE_DATABASE인 경우 구성이 요청된 데이터베이스의 별명 이름

ioNumIndicators

이 매개변수는 입력이나 출력 매개변수로 사용할 수 있습니다.

입력: Health 표시기의 서브세트 설정 요청 시에 체출된 pioIndicators 수를 나타냅니다.

출력: API에서 리턴된 총 Health 표시기 수를 나타냅니다.

pioIndicators

db2GetAlertCfgInd 구조의 포인터 NULL로 설정된 경우 해당 오브젝트의 모든 Health 표시기가 리턴됩니다.

db2GetAlertCfgInd 데이터 구조 매개변수

ioIndicatorID

Health 표시기(sqlmon.h에 정의)

oAlarm

출력. Health 표시기 알람 임계값 설정. 이 설정은 임계값 기준 Health 표시기에만 유효합니다.

oWarning

출력. Health 표시기 경고 임계값 설정. 이 설정은 임계값 기준 Health 표시기에만 유효합니다.

oSensitivity

출력. Health 표시기의 값이 관련된 알람이나 경고 조건이 등록되기 전까지 임계값 범위에서 유지되어야 하는 기간

poFormula

출력. Health 표시기 값 계산에 사용된 공식의 문자열 표현

oActionEnabled

출력. 참인 경우 poTaskActions 또는 poScriptActions에 정의된 모든 경보 조치는 임계값이 초과되면 호출됩니다. 거짓인 경우 정의된 조치가 전혀 호출되지 않습니다.

oCheckThresholds

출력. 참인 경우 임계값 초과 또는 상태 변경이 평가됩니다. 임계값 초과나 상태가 평가되지 않으면 경보가 발행되지 않고 경보 조치는 oActionEnabled가 참인지에 상관 없이 호출되지 않습니다.

oNumTaskActions

출력. pTaskAction 배열의 태스크 경보 조치 수

poTaskActions

db2AlertTaskAction 구조의 포인터

oNumScriptActions

출력. poScriptActions 배열의 스크립트 조치 수

poScriptActions

db2AlertScriptAction 구조의 포인터

oDefault

출력. 현재 설정이 디폴트에서 상속되었는지를 나타냅니다. 현재 설정이 디폴트에서 상속되었음을 나타내려면 참으로 설정하고 그 외의 경우에는 거짓으로 설정하십시오.

db2AlertTaskAction 데이터 구조 매개변수

pTaskname

태스크 이름

condition

조치를 실행하는 조건

pUserID

스크립트가 실행되는 사용자 어카운트

pPassword

사용자 어카운트 pUserId의 암호

pHostName

스크립트를 실행하는 호스트 이름. 태스크와 스크립트 모두에 적용됩니다.

Script 스크립트가 상주하고 실행되는 호스트 이름

Task 스케줄러가 있는 호스트 이름

db2AlertScriptAction 데이터 구조 매개변수

scriptType

스크립트 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2ALERTCFG_SCRIPTTYPE_DB2CMD
- DB2ALERTCFG_SCRIPTTYPE_OS

condition

조치 실행 조건. 임계값 기본 Health 표시기의 유효한 값은 다음과 같습니다.

- DB2ALERTCFG_CONDITION_ALL
- DB2ALERTCFG_CONDITION_WARNING
- DB2ALERTCFG_CONDITION_ALARM

상태 기반 Health 표시기의 경우 sqlmon에 정의된 숫자 값을 사용하십시오.

pPathname

스크립트의 절대 경로 이름

pWorkingDir

스크립트가 실행되는 디렉토리의 절대 경로 이름

pCmdLineParms

스크립트가 호출될 때 스크립트에 전달되는 명령 매개변수.
DB2ALERTCFG_SCRIPTTYPE_OS에만 선택적

stmtTermChar

명령문을 종료하기 위해 스크립트에서 사용되는 문자.
DB2ALERTCFG_SCRIPTTYPE_DB2CMD에만 선택적

pUserID

스크립트가 실행되는 사용자 어카운트

pPassword

사용자 어카운트 pUserId의 암호

pHostName

스크립트를 실행하는 호스트 이름. 태스크와 스크립트 모두에 적용됩니다.

Script 스크립트가 상주하고 실행되는 호스트 이름

Task 스케줄러가 있는 호스트 이름

사용 시 참고사항

pioIndicators가 계속 NULL인 경우 해당 오브젝트의 모든 Health 표시기가 리턴됩니다. 이 매개변수는 구성이 필요한 Health 표시기로 설정되는 ioIndicatorID가 포함된

db2GetAlertCfgInd 구조 배열로 설정할 수 있습니다. 이렇게 사용되는 경우 ioNumIndicators는 입력 배열 길이로 설정하고 db2GetAlertCfgInd의 모든 기타 필드는 0이나 NULL로 설정하십시오.

이 포인터의 모든 메모리는 엔진으로 할당되고 db2GetAlertCfg API가 오류 없이 리턴될 때마다 db2GetAlertCfgFree API를 호출하여 사용 가능해야 합니다. db2GetAlertCfgFree API에 대한 정보는 include 디렉토리의 db2ApiDf.h를 참조하십시오.

제 24 장 db2GetAlertCfgFree - db2GetAlertCfg API로 할당된 메모리 사용 가능화

db2GetAlertCfg API로 할당된 메모리를 사용 가능화합니다.

중요사항: Health Monitor가 버전 9.7에서 사용되지 않으므로
이 명령 또는 API는 사용되지 않으며 이후 릴리스에서 제거될 수 있습니다. 자세한 정보는 *버전 9.7의 새로운 내용* 책에 있는 『Health Monitor는 사용되지 않음』 주제를 참조하십시오.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetAlertCfgFree (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

db2GetAlertCfgFree API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2GetAlertCfgData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

제 25 장 db2GetContactGroup - 통지 메시지를 수신할 수 있는 단일 문의처 그룹의 문의처 목록 가져오기

단일 문의처 그룹에 포함된 문의처를 리턴합니다. 문의처는 통지 메시지를 수신할 수 있는 사용자입니다. 문의처는 시스템에 로컬로 또는 전역 목록으로 정의할 수 있습니다. DB2 관리 서버(DAS) 구성 매개변수 contact_host의 설정으로 목록이 로컬 또는 전역인지 판별합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetContactGroup (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ContactGroupData
{
   char *pGroupName;
   char *pDescription;
   db2Uint32 numContacts;
   struct db2ContactTypeData *pContacts;
} db2ContactGroupData;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

db2GetContactGroup API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2ContactGroupData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ContactGroupData 데이터 구조 매개변수

pGroupName

입력. 검색할 그룹 이름

pDescription

그룹 설명

numContacts

pContacts 수.

pContacts

db2ContactTypeData 구조의 포인터. 사용자는 각각의 최대 *크기*를 포함하여 pGroupName, pDescription, pContacts, pContacts.pName 필드를 사전에 할당해야 합니다. numContacts로 리턴되는 pContacts의 필수 길이가 포함되도록 numContacts=0 및 pContacts=NULL과 같이 db2GetContactGroup을 호출하십시오.

db2ContactTypeData 데이터 구조 매개변수

contactType

문의처 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName

contactType을 DB2CONTACT_SINGLE로 설정한 경우 문의처 그룹 이름 또는 담당자

사용 시 참고사항

이 API는 UNIX 및 Linux에서는 지원되지 않습니다. 그러나 SQL 인터페이스를 통해 동일한 기능에 액세스할 수 있습니다.

제 26 장 db2GetContactGroups - 통지 메시지를 수신할 수 있는 문의처 그룹 목록 가져오기

문의처 그룹 목록을 리턴합니다. 문의처는 통지 메시지를 수신할 수 있는 사용자입니다. 문의처 그룹은 시스템에 로컬로 또는 전역 목록으로 정의할 수 있습니다. DB2 관리 서버(DAS) 구성 매개변수 contact_host의 설정으로 목록이 로컬 또는 전역인지 판별합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetContactGroups (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetContactGroupsData
{
   db2Uint32 ioNumGroups;
   struct db2ContactGroupDesc *poGroups;
} db2GetContactGroupsData;

typedef SQL_STRUCTURE db2ContactGroupDesc
{
   char *poName;
   char *poDescription;
} db2ContactGroupDesc;
```

db2GetContactGroups API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2GetContactGroupsData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2GetContactGroupsData 데이터 구조 매개변수

ioNumGroups

그룹 수. oNumGroups = 0 및 poGroups = NULL인 경우 poGroups에 필요한 db2ContactGroupDesc 구조 수를 포함합니다.

poGroups

출력. db2ContactGroupDesc 구조의 포인터

db2ContactGroupDesc 데이터 구조 매개변수

poName

출력. 그룹 이름. 이 매개변수는 각 최대 *크기*를 포함하여 호출자가 미리 할당해야 합니다.

poDescription

출력. 그룹 설명. 이 매개변수는 각 최대 *크기*를 포함하여 호출자가 미리 할당해야 합니다.

사용 시 참고사항

이 API는 UNIX 및 Linux에서는 지원되지 않습니다. 그러나 SQL 인터페이스를 통해 동일한 기능에 액세스할 수 있습니다.

제 27 장 db2GetContacts - 통지 메시지를 수신할 수 있는 문의처 목록 가져오기

문의처 목록을 리턴합니다. 문의처는 통지 메시지를 수신할 수 있는 사용자입니다. 문의처는 시스템에 로컬로 또는 전역 목록으로 정의할 수 있습니다. DB2 관리 서버(DAS) 구성 매개변수 contact_host의 설정으로 목록이 로컬 또는 전역인지 판별합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetContacts (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetContactsData
{
   db2Uint32 ioNumContacts;
   struct db2ContactData *poContacts;
} db2GetContactsData;

typedef SQL_STRUCTURE db2ContactData
{
   char *pName;
   db2Uint32 type;
   char *pAddress;
   db2Uint32 maxPageLength;
   char *pDescription;
} db2ContactData;
```

db2GetContacts API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2GetContactsData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2GetContactsData 데이터 구조 매개변수

ioNumContacts

poContacts 수.

poContacts

출력. db2ContactData 구조의 포인터. 사용자는 각각의 최대 *크기*를 포함하여 poContacts, pocontacts.pAddress, pocontacts.pDescription, pocontacts.pName 필드를 사전에 할당해야 합니다. numContacts로 리턴되는 poContacts의 필수 길이가 포함되도록 numContacts=0 및 pContacts=NULL과 같이 db2GetContacts를 호출하십시오.

db2ContactData 데이터 구조 매개변수

pName

담당자

type 문의처 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2CONTACT_EMAIL
- DB2CONTACT_PAGE

pAddress

type 매개변수의 주소

maxPageLength

type이 DB2CONTACT_PAGE로 설정된 경우 메시지의 최대 길이

pDescription

문의처에 대해 사용자가 제공하는 설명

사용 시 참고사항

이 API는 UNIX 및 Linux에서는 지원되지 않습니다. 그러나 SQL 인터페이스를 통해 동일한 기능에 액세스할 수 있습니다.

제 28 장 db2GetDistMap - 분산 맵 가져오기

응용프로그램에서 테이블의 분산 정보를 가져올 수 있습니다. 분산 정보에는 분산 맵과 분산 키의 컬럼 정의가 포함됩니다.

이 API가 리턴한 정보는 sqlugrpn API로 전달되어 임의 행에 대한 데이터베이스 파티션 번호 및 테이블의 데이터베이스 파티션 서버 번호를 판별합니다.

이 API를 사용하려면 분산 정보가 요청된 테이블이 포함된 데이터베이스에 응용프로그램이 연결되어야 합니다.

범위

이 API는 db2nodes.cfg 파일에 정의된 모든 데이터베이스 파티션 서버에서 실행할 수 있습니다.

권한 부여

참조 중인 테이블의 경우 다음 권한이나 특권 중 하나 이상을 가지고 있어야 합니다.

- DATAACCESS 권한
- CONTROL 특권
- SELECT 특권

필수 연결

데이터베이스

API 내장 파일

sqlutil.h

API 및 데이터 구조 구문

```
db2GetDistMap(db2Uint32    versionNumber,   // DB2 version number
           void         *pParmStruct,    // In/Out parameters
           struct sqlca *pSqlca);        // Sqlca

where
SQL_STRUCTURE db2DistMapStruct
{
   unsigned char     *tname;                           /* Fully qualified table name          */
   unsigned short    pmaplen;                          /* Length of distribution map          */
   SQL_PDB_NODE_TYPE pmap[SQL_PDB_MAP_SIZE];           /* Distribution map                    */
   unsigned short    sqld;                             /* # of used SQLPARTKEY elements       */
   struct sqlpartkey sqlpartkey[SQL_MAX_NUM_PART_KEYS]; /* Distribution keys                  */};
```

db2GetDistMap API 매개변수

tname 테이블의 완전한 이름

pmaplen
분산 맵 길이

pmap 분산 맵 이름

sqld sqlpartkey 구조에서 사용되는 요소 수

sqlpartkey
테이블에 사용되는 분산 키

제 29 장 db2GetHealthNotificationList - Health 경보 통지를 수신할 수 있는 문의처 목록 가져오기

인스턴스 Health에 대해 통지 받는 문의처 및/또는 문의처 그룹 목록을 리턴합니다. 문의처 목록은 인스턴스 또는 해당 데이터베이스 오브젝트에 대해 비정상 Health 조건이 표시되면 통지를 수신하는 대상의 전자 우편 주소 또는 호출기 인터넷 주소로 구성됩니다.

권한 부여

없음

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetHealthNotificationList (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetHealthNotificationListData
{
   db2Uint32 ioNumContacts;
   struct db2ContactTypeData *poContacts;
} db2GetHealthNotificationListData;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

db2GetHealthNotificationList API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2GetHealthNotificationListData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2GetHealthNotificationListData 데이터 구조 매개변수

ioNumContacts

문의처 수. NULL poContact로 API가 호출되면 호출 성공을 위해 수행되도록 사용자가 할당해야 하는 문의처 수로 ioNumContacts가 설정됩니다.

poContacts

출력. db2ContactTypeData 구조의 포인터

db2ContactTypeData 데이터 구조 매개변수

contactType

문의처 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName

contactType을 DB2CONTACT_SINGLE로 설정한 경우 문의처 그룹 이름 또는 담당자

제 30 장 db2GetRecommendations - 경보 상태의 Health 표시기를 해결하는 권장사항 가져오기

특정 오브젝트에서 경보 상태의 Health 표시기를 해결하는 권장사항 세트를 검색합니다. 권장사항은 XML 문서로 리턴됩니다.

중요사항: Health Monitor가 버전 9.7에서 사용되지 않으므로
이 명령 또는 API는 사용되지 않으며 이후 릴리스에서 제거될 수 있습니다. 자세한 정보는 *버전 9.7의 새로운 내용* 책에 있는 『Health Monitor는 사용되지 않음』 주제를 참조하십시오.

권한 부여

없음

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetRecommendations (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetRecommendationsData
{
   db2Uint32 iSchemaVersion;
   db2Uint32 iNodeNumber;
   db2Uint32 iIndicatorID;
   db2Uint32 iObjType;
   char *piObjName;
   char *piDbName;
   char *poRecommendation;
} db2GetRecommendationsData;
```

db2GetRecommendations API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2GetRecommendationsData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2GetRecommendationsData 데이터 구조 매개변수

iSchemaVersion

입력. XML 문서를 나타내는 데 사용되는 스키마의 버전 ID. 권장사항 문서에는 해당 스키마 버전에 대해 정의된 요소나 속성만 포함됩니다. 이 매개변수는 DB2HEALTH_RECSCHEMA_VERSION8_2로 설정하십시오.

iNodeNumber

입력. Health 표시기(HI)가 경보 상태를 입력한 데이터베이스 파티션 번호를 지정합니다. 상수 SQLM_ALL_NODES를 사용하여 모든 데이터베이스 파티션에서 지정된 HI의 지정된 오브젝트에 대해 권장사항을 검색하십시오. HI에 다른 데이터베이스 파티션과 동일한 권장사항이 있는 경우 해당 권장사항은 한 개의 권장사항 세트로 그룹화되고 여기서 문제점은 다른 데이터베이스 파티션의 HI 그룹이 되며 권장사항은 이 HI 모두에 적용됩니다. 현재 데이터베이스 파티션에서 권장사항을 검색하려면 상수 값 SQLM_CURRENT_NODE를 사용하십시오. 독립형 인스턴스의 경우 SQLM_CURRENT_NODE를 사용해야 합니다.

iIndicatorID

입력. 경보 상태로 입력되는 권장사항이 요청된 Health 표시기. 값은 include 디렉토리의 헤더 파일 sqlmon.h에서 구체화됩니다.

iObjType

입력. Health 표시기(iIndicatorID로 식별)가 경보 상태를 입력한 오브젝트 유형을 지정합니다. 유효한 값은 다음과 같습니다.

- DB2HEALTH_OBJTYPE_DBM
- DB2HEALTH_OBJTYPE_DATABASE
- DB2HEALTH_OBJTYPE_TABLESPACE
- DB2HEALTH_OBJTYPE_TS_CONTAINER

piObjName

입력. 오브젝트 유형 매개변수 iObjType이 DB2HEALTH_OBJTYPE_TABLESPACE 또는 DB2HEALTH_OBJTYPE_TS_CONTAINER로 설정된 경우, 테이블 스페이스 또는 테이블 스페이스 컨테이너의 이름. 필요하지 않은 경우에는 NULL을 지정하십시오. 테이블 스페이스 컨테이너의 경우 오브젝트 이름이 <테이블 스페이스 이름>.<컨테이너 이름>으로 지정됩니다.

piDbname

입력. 오브젝트 유형 매개변수 iObjType이 DB2HEALTH_OBJTYPE_TS_CONTAINER, DB2HEALTH_OBJTYPE_TABLESPACE 또는 DB2HEALTH_OBJTYPE_DATABASE인 경우 HI가 경보 상태를 입력한 데이터베이스의 별명 이름. 그렇지 않으면, NULL을 지정하십시오.

poRecommendation

출력. sqllib/misc/DB2RecommendationSchema.xsd에서 제공되는 스키마에 따라 XML 문서로 형식화되는 권장사항 텍스트가 포함된 메모리의 버퍼 주소로 설정되는 문자 포인터. XML 문서는 UTF-8로 인코딩되며 문서의 텍스트는 호출자의 로케일입니다.

DB2_HEALTH 노드의 xml:lang 속성이 해당하는 클라이언트 언어로 설정됩니다. API는 트러스트된 소스로 고려되고 XML 문서의 유효성은 확인되지 않습니다. XML은 출력 데이터 구조화에 사용됩니다. 이 포인터의 모든 메모리는 엔진으로 할당되고 db2GetRecommendations가 오류를 리턴하지 않을 때마다 db2GetRecommendationsFree 호출로 사용 가능해져야 합니다.

사용 시 참고사항

- 특정 DB2 오브젝트에서 Health 경보를 해결하기 위해 권장사항 세트를 검색하는 이 API를 호출하십시오. 입력 Health 표시기가 식별된 오브젝트에서 경보 상태가 아닌 경우에는 오류가 리턴됩니다.
- 권장사항은 XML 문서로 리턴되고 경보 해결을 위해 실행할 수 있는 조치 및 스크립트에 대한 정보가 포함됩니다. 이 API가 리턴하는 모든 스크립트는 Health 표시기가 경보 상태를 입력한 인스턴스에서 실행해야 합니다. 리턴된 권장사항 XML 문서 구조 및 컨텐츠에 대한 정보는 sqllib/misc/DB2RecommendationSchema.xsd에서 스키마를 참조하십시오.
- 엔진에서 할당하고 이 함수로 리턴된 모든 메모리(권장사항 문서)는 db2GetRecommendations가 오류를 리턴하지 않을 때마다 db2GetRecommendationsFree 호출로 사용 가능해져야 합니다.

제 31 장 db2GetRecommendationsFree - db2GetRecommendations API로 할당된 메모리 사용 가능화

db2GetRecommendations API로 할당된 메모리를 사용 가능화합니다.

중요사항: Health Monitor가 버전 9.7에서 사용되지 않으므로
이 명령 또는 API는 사용되지 않으며 이후 릴리스에서 제거될 수 있습니다. 자세한 정보는 *버전 9.7의 새로운 내용* 책에 있는 『Health Monitor는 사용되지 않음』 주제를 참조하십시오.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetRecommendationsFree (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

db2GetRecommendationsFree API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2GetRecommendationsData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

제 32 장 db2GetSnapshot - 데이터베이스 관리 프로그램 조작 상태에 대한 스냅샷 가져오기

데이터베이스 관리 프로그램 모니터 정보를 수집하여 이를 사용자 할당 데이터 버퍼에 리턴합니다. 리턴된 정보는 API가 호출된 시점에서의 데이터베이스 관리 프로그램 조작 상태 스냅샷을 나타냅니다.

범위

이 API는 인스턴스의 데이터베이스 파티션 서버에 대한 정보 또는 인스턴스의 모든 데이터베이스 파티션에 대한 정보를 리턴할 수 있습니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

리모트 인스턴스(또는 다른 로컬 인스턴스)에서 스냅샷을 확보하려면 우선 해당 인스턴스에 접속해야 합니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetSnapshot (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetSnapshotData
{
   void *piSqlmaData;
   struct sqlm_collected *poCollectedData;
   void *poBuffer;
   db2Uint32 iVersion;
   db2Uint32 iBufferSize;
```

```
   db2Uint32 iStoreResult;
   db2int32 iNodeNumber;
   db2Uint32 *poOutputFormat;
   db2Uint32 iSnapshotClass;
} db2GetSnapshotData;

SQL_API_RC SQL_API_FN
  db2gGetSnapshot (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gGetSnapshotData
{
   void *piSqlmaData;
   struct sqlm_collected *poCollectedData;
   void *poBuffer;
   db2Uint32 iVersion;
   db2Uint32 iBufferSize;
   db2Uint32 iStoreResult;
   db2int32 iNodeNumber;
   db2Uint32 *poOutputFormat;
   db2Uint32 iSnapshotClass;
} db2gGetSnapshotData;
```

API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다. 위에서 설명한 대로 구조를 사용하려면 db2Version810 이상을 지정하십시오. 이 구조의 다른 버전을 사용하려면 include 디렉토리의 db2ApiDf.h 헤더 파일에서 지원되는 버전의 전체 목록을 점검하십시오. 지정한 버전 번호에 해당하는 db2GetSnapshotData 구조의 버전을 사용하고 있는지 확인하십시오.

pParmStruct

입/출력(I/O). db2GetSnapshotData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2GetSnapshotData 데이터 구조 매개변수

piSqlmaData

입력. 사용자 할당 sqlma(모니터 영역) 구조 또는 요청 데이터 구조의 포인터로, ″poRequestData″로 구문화되고 db2AddSnapshotRequest API로 리턴됩니다. 구조는 수집하려는 스냅샷 데이터 유형을 지정합니다. sqlma 구조의 포인터를 사용하는 경우에는 db2GetSnapshot API의 versionNumber 매개변수로 전달된 버전은 db2Version900보다 작아야 합니다(예: db2Version810, db2Version822). db2AddSnapshotRequest API가 poRequestData 매개변수로 리턴한 요청 데이터 구조의 포인터가 사용되고 db2Version900 값은 db2GetSnapshot API의 versionNumber 매개변수로 전달되어야 합니다.

poCollectedData

출력. 데이터베이스 모니터가 요약 통계를 전달하는 sqlm_collected 구조 및 버퍼 영역에서 리턴되는 각 유형의 데이터 구조 번호의 포인터.

주: 이 구조는 버전 6 이전의 데이터 스트림에만 사용됩니다. 그러나 이전의 리모트 서버에 대한 스냅샷 호출이 있는 경우 결과가 처리될 수 있도록 이 구조를 전달해야 합니다. 따라서 이 매개변수는 항상 전달되도록 권장됩니다.

poBuffer

출력. 스냅샷 정보가 리턴되는 사용자 정의 데이터 영역의 포인터

iVersion

입력. 수집할 데이터베이스 모니터 데이터의 버전 ID. 데이터베이스 모니터는 요청한 버전에서 사용할 수 있는 데이터만 리턴합니다. 이 매개변수를 다음 상수 중 하나로 설정하십시오.

- SQLM_DBMON_VERSION1
- SQLM_DBMON_VERSION2
- SQLM_DBMON_VERSION5
- SQLM_DBMON_VERSION5_2
- SQLM_DBMON_VERSION6
- SQLM_DBMON_VERSION7
- SQLM_DBMON_VERSION8
- SQLM_DBMON_VERSION9
- SQLM_DBMON_VERSION9_5
- SQLM_DBMON_VERSION9_7(재구성을 통한 MDC Extent 재개에 관한 정보에 필요함)

주: 상수 SQLM_DBMON_VERSION5_2 및 이전 버전은 더 이상 사용되지 않으며 DB2의 추후 릴리스에서 제거될 수 있습니다.

iBufferSize

입력. 데이터 버퍼 길이. db2GetSnapshotSize API를 사용하여 이 버퍼 *크기*를 추정하십시오. 버퍼의 *크기*가 충분하지 않은 경우 지정된 버퍼에 적합한 정보와 함께 경고가 리턴됩니다. 버퍼의 *크기*를 조정하고 API를 다시 호출해야 합니다.

iStoreResult

입력. SQL을 통해 확인할 수 있도록 스냅샷 결과가 DB2 서버에 저장되는지에 따라 상수 값인 참 또는 거짓으로 설정되는 표시기. 스냅샷이 데이터베이스 연결에서 작성되고 sqlma의 스냅샷 유형 중 하나가 SQLMA_DYNAMIC_SQL인 경우 이 매개변수는 참으로만 설정해야 합니다.

iNodeNumber

입력. 요청을 보내는 노드. 이 값을 기본으로 요청은 현재 노드, 모든 노드 또는 사용자 지정 노드에 대해 처리됩니다. 가능한 값은 다음과 같습니다.

- SQLM_CURRENT_NODE
- SQLM_ALL_NODES. iVersion 매개변수가 SQLM_DBMON_VERSION7 이상으로 설정된 경우에만 허용됩니다.
- 노드 값

주: 독립형 인스턴스의 경우 SQLM_CURRENT_NODE 값을 사용해야 합니다.

poOutputFormat

서버에서 리턴되는 스트림 형식. 다음 중 하나입니다.

- SQLM_STREAM_STATIC_FORMAT
- SQLM_STREAM_DYNAMIC_FORMAT

iSnapshotClass

입력. 스냅샷의 클래스 규정자. 유효한 값(sqlmon 헤더 파일에 정의되어 include 디렉토리에 있음)은 다음과 같습니다.

- SQLM_CLASS_DEFAULT - 표준 스냅샷
- SQLM_CLASS_HEALTH - Health 스냅샷
- SQLM_CLASS_HEALTH_WITH_DETAIL - 추가적인 세부사항이 포함된 Health 스냅샷

주: Health Monitor가 버전 9.7에서 사용되지 않으므로 SQLM_CLASS_HEALTH 및 SQLM_CLASS_HEALTH_WITH_DETAIL이 사용되지 않고 추후 릴리스에서 제거될 수 있습니다. 자세한 정보는 *버전 9.7의 새로운 기능* 책의 『Health Monitor가 사용되지 않음』 주제를 참조하십시오.

사용 시 참고사항

다른 인스턴스에 있는 데이터베이스 별명을 지정하는 경우에는 오류 메시지가 리턴됩니다.

전체 콜렉션 정보가 포함된 Health 스냅샷을 검색하려면 SQLMA 데이터 구조에 AGENT_ID 필드를 사용하십시오.

제 33 장 db2GetSnapshotSize - db2GetSnapshot API에 필요한 출력 버퍼 크기 계산

db2GetSnapshot API에서 필요한 버퍼 크기를 계산하십시오.

범위

이 API는 인스턴스의 데이터베이스 파티션 서버 또는 인스턴스의 모든 데이터베이스 파티션에 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

리모트 인스턴스(또는 다른 로컬 인스턴스)에서 정보를 확보하려면 우선 해당 인스턴스에 접속해야 합니다. 접속되지 않은 경우 DB2INSTANCE 환경 변수로 지정한 노드에 내재된 인스턴스 접속이 작성됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetSnapshotSize (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2GetSnapshotSizeData
{
   void *piSqlmaData;
   sqluint32 *poBufferSize;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
   db2Uint32 iSnapshotClass;
} db2GetSnapshotSizeData;
```

```
SQL_API_RC SQL_API_FN
  db2gGetSnapshotSize (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gGetSnapshotSizeData
{
   void *piSqlmaData;
   sqluint32 *poBufferSize;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
   db2Uint32 iSnapshotClass;
} db2gGetSnapshotSizeData;
```

db2GetSnapshotSize API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다. 위에서 설명한 대로 구조를 사용하려면 db2Version810 이상을 지정하십시오. 이 구조의 다른 버전을 사용하려면 include 디렉토리의 db2ApiDf.h 헤더 파일에서 지원되는 버전의 전체 목록을 점검하십시오. 지정한 버전 번호에 해당하는 db2GetSnapshotSizeStruct 구조의 버전을 사용하고 있는지 확인하십시오.

pParmStruct

입력. db2GetSnapshotSizeStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2GetSnapshotSizeData 데이터 구조 매개변수

piSqlmaData

입력. 사용자 할당 sqlma(모니터 영역) 구조 또는 요청 데이터 구조의 포인터로, ″poRequestData″로 구문화되고 db2AddSnapshotRequest API로 리턴됩니다. 구조는 수집하려는 스냅샷 데이터 유형을 지정합니다. sqlma 구조의 포인터를 사용하는 경우에는 db2GetSnapshotSize API의 versionNumber 매개변수로 전달된 버전은 db2Version900보다 작아야 합니다(예: db2Version810, db2Version822). db2AddSnapshotRequest API가 poRequestData 매개변수로 리턴한 요청 데이터 구조의 포인터가 사용되고 db2Version900 값은 db2GetSnapshotSize API의 versionNumber 매개변수로 전달되어야 합니다.

poBufferSize

출력. GET SNAPSHOT API에서 필요한 리턴된 계산된 버퍼 *크기*의 포인터

iVersion

입력. 수집할 데이터베이스 모니터 데이터의 버전 ID. 데이터베이스 모니터는 요청한 버전에서 사용할 수 있는 데이터만 리턴합니다. 이 매개변수를 다음 기호 상수 중 하나로 설정하십시오.

- SQLM_DBMON_VERSION1
- SQLM_DBMON_VERSION2
- SQLM_DBMON_VERSION5
- SQLM_DBMON_VERSION5_2
- SQLM_DBMON_VERSION6
- SQLM_DBMON_VERSION7
- SQLM_DBMON_VERSION8
- SQLM_DBMON_VERSION9
- SQLM_DBMON_VERSION9_5

주: 상수 SQLM_DBMON_VERSION5_2 및 그 이전 버전은 더 이상 사용되지 않으며 DB2의 추후 릴리스에서 제거될 수 있습니다.

iNodeNumber

입력. 요청을 보내는 데이터베이스 파티션 서버. 요청은 이 값을 사용하여 현재 데이터베이스 파티션 서버, 모든 데이터베이스 파티션 서버 또는 사용자가 지정한 데이터베이스 파티션 서버에 대해 처리됩니다. 가능한 값은 다음과 같습니다.

- SQLM_CURRENT_NODE
- SQLM_ALL_NODES. iVersion이 SQLM_DBMON_VERSION7 이상으로 설정된 경우에만 허용됩니다.
- 노드 값

독립형 인스턴스의 경우 SQLM_CURRENT_NODE 값을 사용해야 합니다.

iSnapshotClass

입력. 스냅샷의 클래스 규정자. 유효한 값(include 디렉토리에 있는 sqlmon 헤더 파일에 정의)은 다음과 같습니다.

- SQLM_CLASS_DEFAULT - 표준 스냅샷
- SQLM_CLASS_HEALTH - Health 스냅샷
- SQLM_CLASS_HEALTH_WITH_DETAIL - 추가적인 세부사항이 포함된 Health 스냅샷

사용 시 참고사항

이 함수는 큰 오버헤드를 생성합니다. 각 db2GetSnapshot API에 대해 동적으로 메모리를 할당하고 사용 가능하게 하는 데는 비용이 많이 소요됩니다. db2GetSnapshot을 반복적으로 호출하는 경우, 예를 들어 특정 기간 동안 데이터를 샘플링하는 경우 db2GetSnapshotSize를 호출하는 것보다 고정된 *크기*의 버퍼를 할당하는 것이 좋습니다.

데이터베이스 시스템 모니터가 사용 중인 데이터베이스나 응용프로그램을 찾지 못하면 버퍼 *크기*를 0으로 리턴합니다(예를 들어, 사용 중이지 않은 데이터베이스에 관련된 잠금 정보가 요청되는 경우). db2GetSnapshot을 호출하기 전에 이 API가 리턴한 계산된 버퍼 *크기*가 0이 아닌지 확인하십시오. 출력이 포함되는 버퍼 스페이스가 충분하지 않아서 db2GetSnapshot가 오류를 리턴하는 경우 이 API를 다시 호출하여 필요한 *크기*를 새로 판별하십시오.

제 34 장 db2GetSyncSession - Satellite 동기화 세션 ID 가져오기

Satellite의 현재 동기화 세션 ID를 가져옵니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2GetSyncSession (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2GetSyncSessionStruct
{
   char *poSyncSessionID;
} db2GetSyncSessionStruct;
```

db2GetSyncSession API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2GetSyncSessionStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2GetSyncSessionStruct 데이터 구조 매개변수

poSyncSessionID

출력. 현재 Satellite가 사용 중인 동기화 세션 ID를 지정합니다.

제 35 장 db2HADRStart - 고가용성 재해 복구(HADR) 조작 시작

데이터베이스에서 HADR 조작을 시작합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

인스턴스. API는 데이터베이스에 연결되어 있지 않은 경우 연결하며 API가 완료되면 연결이 끊어집니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2HADRStart (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRStartStruct
{
   char *piDbAlias;
   char *piUserName;
   char *piPassword;
   db2Uint32 iDbRole;
   db2Uint16 iByForce;
} db2HADRStartStruct;

SQL_API_RC SQL_API_FN
  db2gHADRStart (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRStartStruct
{
   char *piDbAlias;
   db2Uint32 iAliasLen;
   char *piUserName;
   db2Uint32 iUserNameLen;
   char *piPassword;
```

```
   db2Uint32 iPasswordLen;
   db2Uint32 iDbRole;
   db2Uint16 iByForce;
} db2gHADRStartStruct;
```

db2HADRStart API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2HADRStartStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2HADRStartStruct 데이터 구조 매개변수

piDbAlias

입력. 데이터베이스 별명의 포인터

piUserName

입력. 명령이 실행되는 사용자 이름의 포인터

piPassword

입력. 암호가 포함된 문자열의 포인터

iDbRole

입력. 지정한 데이터베이스에서 HADR 데이터베이스 역할이 시작되어야 하는지를 지정합니다. 가능한 값은 다음과 같습니다.

DB2HADR_DB_ROLE_PRIMARY

데이터베이스에서 HADR 조작을 기본 역할로 시작합니다.

DB2HADR_DB_ROLE_STANDBY

데이터베이스에서 HADR 조작을 대기 역할로 시작합니다.

iByForce

입력. iDbRole 매개변수가 DB2HADR_DB_ROLE_STANDBY로 설정된 경우에는 이 인수가 무시됩니다. 가능한 값은 다음과 같습니다.

DB2HADR_NO_FORCE

지정된 시간 제한 내에 대기 데이터베이스가 기본 데이터베이스에 연결된 경우에만 기본 데이터베이스에서 HADR이 시작되도록 지정합니다.

DB2HADR_FORCE

대기 데이터베이스가 기본 데이터베이스에 연결되도록 대기하지 않고 HADR이 강제로 시작되도록 지정합니다.

db2gHADRStartStruct 데이터 구조 특정 매개변수

iAliasLen

입력. 데이터베이스 별명 길이를 바이트 단위로 지정합니다.

iUserNameLen

입력. 사용자 이름의 길이를 바이트 단위로 지정합니다.

iPasswordLen

입력. 암호 길이를 바이트 단위로 지정합니다.

제 36 장 db2HADRStop - 고가용성 재해 복구(HADR) 조작 중지

데이터베이스에서 HADR 조작을 중지합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

인스턴스. API는 데이터베이스에 연결되어 있지 않은 경우 연결하며 API가 완료되면 연결이 끊어집니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2HADRStop (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRStopStruct
{
   char *piDbAlias;
   char *piUserName;
   char *piPassword;
} db2HADRStopStruct;

SQL_API_RC SQL_API_FN
  db2gHADRStop (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRStopStruct
{
   char *piDbAlias;
   db2Uint32 iAliasLen;
   char *piUserName;
   db2Uint32 iUserNameLen;
   char *piPassword;
   db2Uint32 iPasswordLen;
} db2gHADRStopStruct;
```

db2HADRStop API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2HADRStopStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2HADRStopStruct 데이터 구조 매개변수

piDbAlias

입력. 데이터베이스 별명의 포인터

piUserName

입력. 명령이 실행되는 사용자 이름의 포인터

piPassword

입력. 암호가 포함된 문자열의 포인터

db2gHADRStopStruct 데이터 구조 특정 매개변수

iAliasLen

입력. 데이터베이스 별명 길이를 바이트 단위로 지정합니다.

iUserNameLen

입력. 사용자 이름의 길이를 바이트 단위로 지정합니다.

iPasswordLen

입력. 암호 길이를 바이트 단위로 지정합니다.

제 37 장 db2HADRTakeover - 데이터베이스를 고가용성 재해 복구 (HADR) 기본 데이터베이스로 사용

대기 데이터베이스를 기본 데이터베이스로 사용되도록 지시합니다. 이 API는 대기 데이터베이스에 대해서만 호출할 수 있습니다.

권한 부여

다음 중 하나가 필요합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*

필수 연결

인스턴스. API는 데이터베이스에 연결되어 있지 않은 경우 연결하며 API가 완료되면 연결이 끊어집니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2HADRTakeover (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HADRTakeoverStruct
{
   char *piDbAlias;
   char *piUserName;
   char *piPassword;
   db2Uint16 iByForce;
} db2HADRTakeoverStruct;

SQL_API_RC SQL_API_FN
  db2gHADRTakeover (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHADRTakeoverStruct
{
   char *piDbAlias;
   db2Uint32 iAliasLen;
   char *piUserName;
```

```
   db2Uint32 iUserNameLen;
   char *piPassword;
   db2Uint32 iPasswordLen;
   db2Uint16 iByForce;
} db2gHADRTakeoverStruct;
```

db2HADRTakeover API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2HADRTakeoverStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2HADRTakeoverStruct 데이터 구조 매개변수

piDbAlias

입력. 데이터베이스 별명의 포인터

piUserName

입력. 명령이 실행되는 사용자 이름의 포인터

piPassword

입력. 암호가 포함된 문자열의 포인터

iByForce

입력. 가능한 값은 다음과 같습니다.

DB2HADR_NO_FORCE

서로 통신이 연결된 두 시스템이 피어 상태인 경우에만 인계가 발생하도록 지정합니다. 그러면 HADR 기본과 HADR 대기 데이터베이스 사이의 역할 전환을 수행할 수 있습니다.

DB2HADR_FORCE

대기 데이터베이스가 원래 기본 데이터베이스가 종료되었는지 확인을 대기하지 않고 기본 데이터베이스 역할을 인계합니다. 대기 데이터베이스가 리모트 만회 보류 또는 피어 상태인 경우 강제로 인계를 실행해야 합니다.

DB2HADR_FORCE_PEERWINDOW

이 옵션을 지정하면 명령이 성공하고 피어 창 기간이 종료되기 전에 기본 데이터베이스가 중단되는 경우 커미트된 모든 트랜잭션이 손실되지 않습니다(데이터베이스 구성 매개변수 **HADR_PEER_WINDOW**를 0이 아닌 값으로 설정). 피어 창이 만기되기 전에 기본 데이터베이스를

종료하지 않으면 *브레인 분할*이 초래됩니다. HADR 쌍이 피어가 아니거나 연결이 끊어진 피어 상태(피어 창 만기)에서 실행되면 오류가 리턴됩니다.

주: DB2HADR_FORCE_PEERWINDOW 매개변수를 사용한 인계 조작은 기본 데이터베이스 클럭과 대기 데이터베이스 클럭이 서로 5초 이내로 동기화되지 않은 경우에 제대로 작동하지 않을 수도 있습니다. 즉, 조작이 실패해야 하는 경우에도 성공할 수 있고 그 반대로 성공해야 하는 경우에 실패할 수도 있습니다. 동일한 소스로 클럭의 동기화가 유지되도록 시간 동기화 서비스(예: NTP)를 사용해야 합니다.

```
/* Values for iByForce                                          */
#define DB2HADR_NO_FORCE           0    /* Do not perform START or   */
                                        /* TAKEOVER HADR operation   */
                                        /* by force                  */
#define DB2HADR_FORCE              1    /* Do perform START or       */
                                        /* TAKEOVER HADR operation   */
                                        /* by force                  */
#define DB2HADR_FORCE_PEERWINDOW   2    /* Perform TAKEOVER HADR     */
                                        /* operation by force inside */
                                        /* the Peer Window only      */
```

db2gHADRTakeoverStruct 데이터 구조 특정 매개변수

iAliasLen

입력. 데이터베이스 별명 길이를 바이트 단위로 지정합니다.

iUserNameLen

입력. 사용자 이름의 길이를 바이트 단위로 지정합니다.

iPasswordLen

입력. 암호 길이를 바이트 단위로 지정합니다.

제 38 장 db2HistoryCloseScan - 실행기록 파일 스캔 종료

실행기록 파일 스캔을 종료하고 스캔에 필요한 DB2 자원을 다시 사용하게 합니다. 이 API는 thedb2HistoryOpenScan API를 제대로 호출한 후에 수행해야 합니다.

권한 부여

없음

필수 연결

인스턴스. 이 API를 호출하기 전에 sqleatin API를 호출할 필요는 없습니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2HistoryCloseScan (
       db2Uint32 versionNumber,
       void * piHandle,
       struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  db2gHistoryCloseScan (
       db2Uint32 versionNumber,
       void * piHandle,
       struct sqlca * pSqlca);
```

db2HistoryCloseScan API 매개변수

versionNumber

입력. 두 번째 매개변수인 piHandle의 버전 및 릴리스 레벨을 지정합니다.

piHandle

입력. db2HistoryOpenScan API가 리턴한 스캔 액세스 핸들 포인터를 지정합니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

실행기록 파일 API 사용에 대한 자세한 설명은 db2HistoryOpenScan API를 참조하십시오.

REXX API 구문

```
CLOSE RECOVERY HISTORY FILE :scanid
```

REXX API 매개변수

scanid OPEN RECOVERY HISTORY FILE SCAN에서 리턴된 스캔 ID가 포함된 호스트 변수.

제 39 장 db2HistoryGetEntry - 실행기록 파일의 다음 항목 가져오기

실행기록 파일에서 다음 항목을 가져옵니다. 이 API는 db2HistoryOpenScan API를 제대로 호출한 후에 수행해야 합니다.

권한 부여

없음

필수 연결

인스턴스. 이 API를 호출하기 전에 sqleatin을 호출할 필요는 없습니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2HistoryGetEntry (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HistoryGetEntryStruct
{
   struct db2HistoryData *pioHistData;
   db2Uint16 iHandle;
   db2Uint16 iCallerAction;
} db2HistoryGetEntryStruct;

SQL_API_RC SQL_API_FN
  db2gHistoryGetEntry (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);
```

db2HistoryGetEntry API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2HistoryGetEntryStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2HistoryGetEntryStruct 데이터 구조 매개변수

pioHistData

입력. db2HistData 구조의 포인터

iHandle

입력. db2HistoryOpenScan API가 리턴한 스캔 액세스 핸들이 포함됩니다.

iCallerAction

입력. 수행할 조치 유형을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2HISTORY_GET_ENTRY

명령 데이터를 포함하지 않은 상태로 다음 항목을 가져옵니다.

DB2HISTORY_GET_DDL

이전 페치의 명령 데이터만 가져옵니다.

DB2HISTORY_GET_ALL

모든 데이터를 포함하여 다음 항목을 가져옵니다.

사용 시 참고사항

리턴된 레코드는 db2HistoryOpenScan API 호출에 지정된 값을 사용하여 선택됩니다.

실행기록 파일 API 사용에 대한 자세한 설명은 db2HistoryOpenScan API를 참조하십시오.

REXX API 구문

```
GET RECOVERY HISTORY FILE ENTRY :scanid [USING :value]
```

REXX API 매개변수

scanid OPEN RECOVERY HISTORY FILE SCAN에서 리턴된 스캔 ID가 포함된 호스트 변수

값 실행기록 파일 항목 정보가 리턴되는 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0

변수의 첫 번째 레벨 요소 수(항상 15)

XXX.1

테이블 스페이스 요소 수

XXX.2

사용된 테이블 스페이스 요소 수

XXX.3

OPERATION(수행된 조작 유형)

XXX.4

OBJECT(조작의 세분화도)

XXX.5

OBJECT_PART(시간소인 및 시퀀스 번호)

XXX.6

OPTYPE(조작의 규정자)

XXX.7

DEVICE_TYPE(사용된 디바이스 유형)

XXX.8

FIRST_LOG(가장 빠른 로그 ID)

XXX.9

LAST_LOG(현재 로그 ID)

XXX.10

BACKUP_ID(백업의 ID)

XXX.11

SCHEMA(테이블 이름의 규정자)

XXX.12

TABLE_NAME(로드된 테이블 이름)

XXX.13.0

NUM_OF_TABLESPACES(백업이나 리스토어에 연관된 테이블 스페이스 수)

XXX.13.1

백업/리스토어된 첫 번째 테이블 스페이스 이름

XXX.13.2

백업/리스토어된 두 번째 테이블 스페이스 이름

XXX.13.3

기타

XXX.14

LOCATION(백업 또는 사본이 저장된 위치)

XXX.15

COMMENT(항목을 설명하는 텍스트)

제 40 장 db2HistoryOpenScan - 실행기록 파일 스캔 시작

이 API는 실행기록 파일 스캔을 시작합니다.

권한 부여

없음

필수 연결

인스턴스. 데이터베이스가 리모트로 카탈로그된 경우 이 API를 호출하기 전에 sqleatin API를 호출하십시오.

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2HistoryOpenScan (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HistoryOpenStruct
{
   char *piDatabaseAlias;
   char *piTimestamp;
   char *piObjectName;
   db2Uint32 oNumRows;
   db2Uint32 oMaxTbspaces;
   db2Uint16 iCallerAction;
   db2Uint16 oHandle;
} db2HistoryOpenStruct;

SQL_API_RC SQL_API_FN
  db2gHistoryOpenScan (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gHistoryOpenStruct
{
   char *piDatabaseAlias;
   char *piTimestamp;
   char *piObjectName;
   db2Uint32 iAliasLen;
   db2Uint32 iTimestampLen;
   db2Uint32 iObjectNameLen;
   db2Uint32 oNumRows;
```

```
   db2Uint32 oMaxTbspaces;
   db2Uint16 iCallerAction;
   db2Uint16 oHandle;
} db2gHistoryOpenStruct;
```

db2HistoryOpenScan API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력 또는 출력. db2HistoryOpenStruct 데이터 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2HistoryOpenStruct 데이터 구조 매개변수

piDatabaseAlias

입력. 데이터베이스 별명이 포함된 문자열의 포인터

piTimestamp

입력. 레코드 선택에 사용되는 시간소인을 지정하는 문자열의 포인터. 시간소인이 이 값과 동등하거나 이후인 레코드가 선택됩니다. 이 매개변수를 NULL로 설정하거나 영(0)으로 지정하면 시간소인을 사용하는 항목 필터링이 사용되지 않습니다.

piObjectName

입력. 레코드 선택에 사용되는 오브젝트 이름을 지정하는 문자열의 포인터. 오브젝트는 테이블 또는 테이블 스페이스일 수 있습니다. 테이블인 경우에는 완전한 테이블 이름을 입력해야 합니다. 이 매개변수를 NULL로 설정하거나 영(0)으로 지정하면 오브젝트 이름을 사용하는 항목 필터링이 사용되지 않습니다.

oNumRows

출력. API 호출에서 리턴될 때 이 매개변수에는 일치하는 실행기록 파일 항목 수가 포함됩니다.

oMaxTbspaces

출력. 임의의 실행기록 항목이 저장되는 테이블 스페이스 이름의 최대 수

iCallerAction

입력. 수행할 조치 유형을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2HISTORY_LIST_HISTORY

현재 실행기록 파일에 로그된 모든 이벤트가 나열됩니다.

DB2HISTORY_LIST_BACKUP
백업 및 리스토어 조작이 나열됩니다.

DB2HISTORY_LIST_ROLLFORWARD
롤 포워드 조작을 나열합니다.

DB2HISTORY_LIST_DROPPED_TABLE
삭제된 테이블 레코드가 나열됩니다. 항목에 연관된 DDL 필드가 리턴되지 않습니다. 항목의 DDL 정보를 검색하려면 항목을 페치한 후에 바로 호출자 조치인 DB2HISTORY_GET_DDL로 db2HistoryGetEntry를 호출해야 합니다.

DB2HISTORY_LIST_LOAD
로드 조작을 나열합니다.

DB2HISTORY_LIST_CRT_TABLESPACE
테이블 스페이스 작성 및 삭제 조작이 나열됩니다.

DB2HISTORY_LIST_REN_TABLESPACE
테이블 스페이스 이름 바꾸기 조작이 나열됩니다.

DB2HISTORY_LIST_ALT_TABLESPACE
테이블 스페이스 수정 조작이 나열됩니다. 항목에 연관된 DDL 필드가 리턴되지 않습니다. 항목의 DDL 정보를 검색하려면 항목을 페치한 후에 바로 호출자 조치인 DB2HISTORY_GET_DDL로 db2HistoryGetEntry를 호출해야 합니다.

DB2HISTORY_LIST_REORG
REORGANIZE TABLE 함수가 나열됩니다. 이 값은 현재 지원되지 않습니다.

oHandle
출력. API에서 리턴될 때 이 매개변수에는 스캔 액세스 핸들이 포함됩니다. 그리고 db2HistoryGetEntry 및 db2HistoryCloseScan API에서 연이어 사용됩니다.

db2gHistoryOpenStruct 데이터 구조 특정 매개변수

iAliasLen
입력. 데이터베이스 별명 문자열의 길이를 바이트 단위로 지정합니다.

iTimestampLen
입력. 시간소인 문자열의 길이를 바이트 단위로 지정합니다.

iObjectNameLen
입력. 오브젝트 이름 문자열의 길이를 바이트 단위로 지정합니다.

사용 시 참고사항

시간소인, 오브젝트 이름 및 호출자 조치의 조합을 사용하여 레코드를 필터링할 수 있습니다. 지정된 모든 필터를 패스하는 레코드만 리턴됩니다.

오브젝트 이름의 필터링 효과는 다음과 같이 지정된 값에 종속됩니다.

- 테이블을 지정하면 로드 조작의 레코드가 리턴됩니다. 이는 해당 레코드가 실행기록 파일에서 테이블에 대한 유일한 정보이기 때문입니다.
- 테이블 스페이스를 지정하면 테이블 스페이스의 백업, 리스토어 및 로드 조작 레코드가 리턴됩니다.

주: 테이블 레코드를 리턴하려면 schema.tablename으로 지정해야 합니다. 테이블 이름을 지정하면 테이블 스페이스 레코드만 리턴됩니다.

프로세스별로 최대 8개의 실행기록 파일 스캔만 허용됩니다.

실행기록 파일의 모든 항목을 나열하기 위해 일반 응용프로그램은 다음 단계를 수행합니다.

1. oNumRows 매개변수 값을 리턴하는 db2HistoryOpenScan API를 호출합니다.
2. n oTablespace 필드의 스페이스가 포함된 db2HistData 구조를 할당합니다(여기서 n은 임의 번호임)
3. db2HistoryData 구조의 iNumTablespaces 필드를 n으로 설정합니다.
4. 루프에서 다음을 수행하십시오.
   - db2HistoryGetEntry API를 호출하여 실행기록 파일에서 페치하십시오.
   - db2HistoryGetEntry API가 SQLCODE 값으로 SQL_RC_OK를 리턴하면 db2HistoryData 구조의 oNumTablespaces 필드를 사용하여 리턴된 테이블 스페이스 항목 수를 판별하십시오.
   - db2HistoryGetEntry API가 SQLCODE 값으로 SQLUH_SQLUHINFO_VARS_WARNING을 리턴하면 DB2에서 리턴하려는 모든 테이블 스페이스에 대해 충분한 스페이스가 할당되지 않은 것입니다. 사용 가능한 스페이스를 늘려서 oDB2UsedTablespace 테이블 스페이스에 필요한 공간이 충분한 db2HistoryData 구조를 다시 할당하고 iDB2NumTablespace를 oDB2UsedTablespace로 설정하십시오.
   - db2HistoryGetEntry API가 SQLCODE 값으로 SQLE_RC_NOMORE를 리턴하면 모든 실행기록 파일 항목이 검색된 것입니다.
   - 다른 모든 SQLCODE는 문제점을 나타냅니다.
5. 모든 정보가 페치되면 db2HistoryCloseScan API를 호출하여 db2HistoryOpenScan 호출로 할당된 자원을 사용 가능하도록 하십시오.

매크로 SQLUHINFOSIZE(n)(sqlutil 헤더 파일에 정의)는 n oTablespace 항목을 위한 스페이스가 포함된 db2HistoryData 구조에 필요한 메모리 양을 판별하는 데 사용됩니다.

REXX API 구문

```
OPEN [BACKUP] RECOVERY HISTORY FILE FOR database_alias
[OBJECT objname] [TIMESTAMP :timestamp]
USING :value
```

REXX API 매개변수

database_alias

나열된 해당 실행기록 파일을 포함하는 데이터베이스 별명

objname

레코드 선택에 사용되는 오브젝트 이름을 지정합니다. 오브젝트는 테이블 또는 테이블 스페이스일 수 있습니다. 테이블인 경우에는 완전한 테이블 이름을 입력해야 합니다. 이 매개변수를 NULL로 설정하면 objname을 사용하는 항목 필터링이 사용되지 않습니다.

시간소인

레코드 선택에 사용되는 시간소인을 지정합니다. 시간소인이 이 값과 동등하거나 이후인 레코드가 선택됩니다. 이 매개변수를 NULL로 설정하면 시간소인을 사용하는 항목 필터링이 사용되지 않습니다.

값 실행기록 파일 정보가 리턴되는 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0

변수의 요소 수(항상 2)

XXX.1

이후 스캔 액세스를 위한 ID(핸들)

XXX.2

일치하는 실행기록 파일 항목 수

제 41 장 db2HistoryUpdate - 실행기록 파일 항목 갱신

실행기록 파일 항목에서 위치, 디바이스 유형 또는 주석을 갱신합니다.

권한 부여

다음 중 하나가 필요합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

필수 연결

데이터베이스. 디폴트 데이터베이스 이외의 데이터베이스에 대해 실행기록 파일의 항목을 갱신하려면 이 API를 호출하기 전에 데이터베이스에 연결해야 합니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2HistoryUpdate (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2HistoryUpdateStruct
{
   char *piNewLocation;
   char *piNewDeviceType;
   char *piNewComment;
   char *piNewStatus;
   db2HistoryEID iEID;
} db2HistoryUpdateStruct;

typedef SQL_STRUCTURE db2HistoryEID
{
   SQL_PDB_NODE_TYPE ioNode;
   db2Uint32 ioHID;
} db2HistoryEID;

SQL_API_RC SQL_API_FN
  db2gHistoryUpdate (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);
```

```
typedef SQL_STRUCTURE db2gHistoryUpdateStruct
{
   char *piNewLocation;
   char *piNewDeviceType;
   char *piNewComment;
   char *piNewStatus;
   db2Uint32 iNewLocationLen;
   db2Uint32 iNewDeviceLen;
   db2Uint32 iNewCommentLen;
   db2Uint32 iNewStatusLen;
   db2HistoryEID iEID;
} db2gHistoryUpdateStruct;
```

db2HistoryUpdate API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2HistoryUpdateStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2HistoryUpdateStruct 데이터 구조 매개변수

piNewLocation

입력. 백업, 리스토어 또는 사본 이미지 로드의 새 위치를 지정하는 문자열의 포인터. 이 매개변수를 NULL로 설정하거나 영(0)으로 지시하는 경우에는 값이 변경되지 않습니다.

piNewDeviceType

입력. 백업 저장, 리스토어 또는 사본 이미지 로드를 위해 새 디바이스 유형을 지정하는 문자열의 포인터. 이 매개변수를 NULL로 설정하거나 영(0)으로 지시하는 경우에는 값이 변경되지 않습니다. 유효한 디바이스 유형은 다음과 같습니다.

D 디스크

K 디스켓

T 테이프

F 스냅샷 백업

A Tivoli Storage Manager

U User exit

P 파이프

N 널(NULL) 디바이스

X XBSA

Q SQL문

O 기타

piNewComment

입력. 항목을 설명하는 새 주석을 지정하는 문자열의 포인터. 이 매개변수를 NULL로 설정하거나 영(0)으로 지시하는 경우에는 주석이 변경되지 않습니다.

piNewStatus

입력. 항목에 대한 새 상태 유형을 지정하는 문자열의 포인터. 이 매개변수를 NULL로 설정하거나 영(0)으로 지시하는 경우에는 상태가 변경되지 않습니다. 가능한 값은 다음과 같습니다.

A 사용 중. 백업 이미지는 사용 중인 로그 체인에 있습니다. 대부분의 항목이 사용 중입니다.

I 비활동. 현재 로그 체인이라고도 하는 현재 로그 시퀀스에 더 이상 대응하지 않는 백업 이미지가 비활성으로 플래그됩니다.

E 만기됨. NUM_DB_BACKUPS를 초과한 활동 이미지가 있으므로 더 이상 필요하지 않은 백업 이미지는 만기된 것으로 플래그됩니다.

D 삭제됨. 더 이상 복구에 사용할 수 없는 백업 이미지는 삭제된 것으로 표시되어야 합니다.

X Do_not_delete. do not delete로 표시되는 복구 실행기록은 PRUNE HISTORY가 포함된 ADMIN_CMD 프로시저를 실행하는 PRUNE HISTORY 명령 호출, db2Prune API 호출 또는 자동화된 복구 실행기록 파일 프룬(prune)으로도 프룬되거나 삭제되지 않습니다. do_not_delete 상태를 사용하여 키 복구 파일 항목이 프룬되는 것을 방지하고 이에 연관된 복구 오브젝트가 삭제되는 것을 방지할 수 있습니다.

iEID 입력. 실행기록 파일에서 특정 항목을 갱신하는 데 사용할 수 있는 고유 ID

db2HistoryEID 데이터 구조 매개변수

ioNode

이 매개변수는 입력이나 출력 매개변수로 사용할 수 있습니다.

노드 번호를 표시합니다.

ioHID 이 매개변수는 입력이나 출력 매개변수로 사용할 수 있습니다.

로컬 실행기록 파일 항목 ID를 표시합니다.

db2gHistoryUpdateStruct 데이터 구조 특정 매개변수

iNewLocationLen

입력. piNewLocation 매개변수 길이를 바이트 단위로 지정합니다.

iNewDeviceLen

입력. piNewDeviceType 매개변수 길이를 바이트 단위로 지정합니다.

iNewCommentLen

입력. piNewComment 매개변수 길이를 바이트 단위로 지정합니다.

iNewStatusLen

입력. piNewStatus 매개변수 길이를 바이트 단위로 지정합니다.

사용 시 참고사항

이는 갱신 함수로 변경 이전의 모든 정보가 교체되며 다시 작성할 수 없습니다. 이 변경은 로그되지 않습니다.

데이터베이스 실행기록 파일의 1차 목적은 정보를 기록하는 것이지만 실행기록에 있는 데이터는 자동 리스토어 연산에서 직접 사용됩니다. AUTOMATIC 옵션을 지정하여 리스토어하는 동안 리스토어 유틸리티는 백업 이미지의 실행기록과 해당 위치를 참조 및 사용하여 자동 리스토어 요청을 이행합니다. 자동 리스토어 기능을 사용하며 백업 이미지가 작성된 이후 그 위치가 변경된 경우, 현재 위치를 반영하도록 해당 이미지에 대한 데이터베이스 실행기록 레코드를 갱신할 것을 권장합니다. 데이터베이스 실행기록에 있는 백업 이미지 위치를 갱신하지 않으면 자동 리스토어로 백업 이미지를 찾을 수 없으므로 직접 리스토어 명령을 사용할 수 있습니다.

REXX API 구문

```
UPDATE RECOVERY HISTORY USING :value
```

REXX API 매개변수

값 실행기록 파일 항목의 새 위치가 들어 있는 정보를 포함하는 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0

변수의 요소 수(1 - 4 사이어야 함)

XXX.1

OBJECT_PART(001 - 999 사이의 시퀀스 번호가 포함된 시간소인)

XXX.2

백업 또는 사본 이미지의 새 위치(이 매개변수는 선택적)

XXX.3

백업 또는 사본 이미지를 저장하는 데 사용되는 새 디바이스(이 매개변수는 선택적)

XXX.4

새 주석(이 매개변수는 선택적)

제 42 장 db2Import - 테이블, 계층 구조, 별칭 또는 뷰로 데이터 임포트

지원되는 파일 형식의 외부 파일 데이터를 테이블, 계층 구조, 뷰 또는 별칭에 삽입합니다. 로드 유틸리티가 이 기능보다 빠릅니다. 그러나 로드 유틸리티는 계층 구조 레벨에서 데이터 로드 또는 별칭으로 로드는 지원하지 않습니다.

권한 부여

- INSERT 옵션을 사용하여 IMPORT하려면 다음 중 하나가 필요합니다.
  - dataaccess
  - 각 파티셔닝 테이블, 뷰 또는 별칭의 CONTROL 특권
  - 각 참여 중인 테이블 또는 뷰에 대한 INSERT 및 SELECT 특권
- INSERT_UPDATE 옵션을 사용하여 기존 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - dataaccess
  - 테이블, 뷰 또는 별칭의 CONTROL 특권
  - 각 참여 중인 테이블 또는 뷰에 대한 INSERT, SELECT, UPDATE 및 DELETE 특권
- REPLACE 또는 REPLACE_CREATE 옵션을 사용하여 기존 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - dataaccess
  - 테이블 또는 뷰에 대한 CONTROL 특권
  - 테이블 또는 뷰에 대한 INSERT, SELECT 및 DELETE 특권
- CREATE 또는 REPLACE_CREATE 옵션을 사용하여 새 테이블로 IMPORT하려면 다음 중 하나가 필요합니다.
  - dbadm
  - 데이터베이스에 대한 CREATETAB 권한 및 테이블 스페이스에 대한 USE 특권은 다음 경우 중 하나입니다.
    - 테이블의 내재적 또는 명시적 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
    - 테이블의 스키마 이름이 기존의 스키마를 참조할 경우, 스키마에 대한 CREATEIN 특권
- CREATE 또는 REPLACE_CREATE 옵션을 사용하여 존재하지 않는 테이블이나 계층 구조로 IMPORT하려면 다음 중 하나가 필요합니다.

  - dbadm
  - 데이터베이스에 대한 CREATETAB 권한 및 다음 중 하나가 있어야 합니다.
    - 테이블의 스키마 이름이 존재하지 않을 경우, 데이터베이스에 대한 IMPLICIT_SCHEMA 권한
    - 테이블의 스키마가 존재할 경우, 스키마에 대한 CREATEIN 특권
    - 전체 계층 구조에 대한 REPLACE_CREATE 옵션이 사용될 경우, 계층 구조의 모든 서브테이블에 대한 CONTROL 특권
- REPLACE 옵션을 사용하여 기존 계층 구조로 IMPORT하려면 다음 중 하나가 필요합니다.
  - dataaccess
  - 계층 구조에 있는 모든 서브테이블에 대한 CONTROL 특권

필수 연결

데이터베이스. 내재된 연결이 사용 가능한 경우 디폴트 데이터베이스에 연결됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Import (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ImportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
   db2int16 iCallerAction;
   struct db2ImportIn *piImportInfoIn;
   struct db2ImportOut *poImportInfoOut;
   db2int32 *piNullIndicators;
   struct sqllob *piLongActionString;
} db2ImportStruct;

typedef SQL_STRUCTURE db2ImportIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   db2Uint64 iSkipcount;
   db2int32 *piCommitcount;
   db2Uint32 iWarningcount;
```

```
   db2Uint16 iNoTimeout;
   db2Uint16 iAccessLevel;
   db2Uint16 *piXmlParse;
   struct db2DMUXmlValidate *piXmlValidate;
} db2ImportIn;

typedef SQL_STRUCTURE db2ImportOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsInserted;
   db2Uint64 oRowsUpdated;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsCommitted;
} db2ImportOut;

typedef SQL_STRUCTURE db2DMUXmlMapSchema
{
   struct db2Char                       iMapFromSchema;
   struct db2Char                       iMapToSchema;
} db2DMUXmlMapSchema;

typedef SQL_STRUCTURE db2DMUXmlValidateXds
{
   struct db2Char *piDefaultSchema;
   db2Uint32 iNumIgnoreSchemas;
   struct db2Char *piIgnoreSchemas;
   db2Uint32 iNumMapSchemas;
   struct db2DMUXmlMapSchema *piMapSchemas;
} db2DMUXmlValidateXds;

typedef SQL_STRUCTURE db2DMUXmlValidateSchema
{
   struct db2Char *piSchema;
} db2DMUXmlValidateSchema;

typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16 iUsing;
   struct db2DMUXmlValidateXds *piXdsArgs;
   struct db2DMUXmlValidateSchema *piSchemaArgs;
} db2DMUXmlValidate;

SQL_API_RC SQL_API_FN
  db2gImport (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gImportStruct
{
   char *piDataFileName;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piMsgFileName;
```

```
   db2int16 iCallerAction;
   struct db2gImportIn *piImportInfoIn;
   struct dbg2ImportOut *poImportInfoOut;
   db2int32 *piNullIndicators;
   db2Uint16 iDataFileNameLen;
   db2Uint16 iFileTypeLen;
   db2Uint16 iMsgFileNameLen;
   struct sqllob *piLongActionString;
} db2gImportStruct;

typedef SQL_STRUCTURE db2gImportIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   db2Uint64 iSkipcount;
   db2int32 *piCommitcount;
   db2Uint32 iWarningcount;
   db2Uint16 iNoTimeout;
   db2Uint16 iAccessLevel;
   db2Uint16 *piXmlParse;
   struct db2DMUXmlValidate *piXmlValidate;
} db2gImportIn;

typedef SQL_STRUCTURE db2gImportOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsInserted;
   db2Uint64 oRowsUpdated;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsCommitted;
} db2gImportOut;
```

db2Import API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입/출력(I/O). db2ImportStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ImportStruct 데이터 구조 매개변수

piDataFileName

입력. 데이터가 임포트되는 외부 입력 파일의 경로 및 이름이 포함된 문자열

piLobPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당

sqlu_media_entry 구조는 LOB 파일이 있는 클라이언트 경로를 나열하는 sqlu_media_list의 포인터. 별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

piDataDescriptor

입력. 외부 파일에서 임포트하도록 선택한 컬럼에 대한 정보가 포함된 sqldcol 구조의 포인터. dcolmeth 필드 값으로 이 매개변수에서 제공되는 나머지 정보가 임포트 유틸리티로 해석되는 방법을 판별합니다. 이 매개변수의 유효한 값은 다음과 같습니다.

SQL_METH_N

이름. 외부 입력 파일 컬럼 선택은 컬럼 이름으로 수행됩니다.

SQL_METH_P

위치. 외부 입력 파일 컬럼 선택은 컬럼 위치로 수행됩니다.

SQL_METH_L

위치. 외부 입력 파일 컬럼 선택은 컬럼 위치로 수행됩니다. 데이터베이스 관리 프로그램은 다음 조건 중 하나로 인해 유효하지 않게 되는 위치 쌍이 있는 임포트 호출을 거부합니다.

- 시작 또는 종료 위치가 1에서 가장 큰 부호있는 2바이트 정수 사이가 아닙니다.
- 종료 위치가 시작 위치보다 작습니다.
- 입력 쌍으로 정의된 입력 컬럼 너비가 목표 컬럼의 유형 및 길이와 호환되지 않습니다.

두 위치의 위치 쌍이 0인 경우 NULL로 채워지는 널(NULL) 입력 가능 컬럼임을 나타냅니다.

SQL_METH_D

디폴트. piDataDescriptor가 NULL이거나 SQL_METH_D로 설정된 경우 외부 입력 파일의 디폴트 컬럼 선택이 수행됩니다. 이 경우 컬럼 수 및 컬럼 스펙 배열 모두 무시됩니다. DEL, IXF 또는 WSF 파일의 경우 외부 입력 파일 데이터의 첫 번째 n 컬럼은 기본 순서대로 사용되며 여기서 n은 데이터가 임포트되는 데이터베이스 컬럼 수입니다.

piActionString

사용되지 않습니다. piLongActionString으로 교체됩니다.

piLongActionString

입력. 테이블에 영향을 주는 조치를 지정하는 문자 배열이 뒤에 오는 4바이트 길이의 필드를 포함하는 *sqllob* 구조의 포인터.

문자 배열은 다음과 같은 양식입니다.

```
{INSERT | INSERT_UPDATE | REPLACE | CREATE | REPLACE_CREATE}
INTO {tname[(tcolumn-list)] |
[{ALL TABLES | (tname[(tcolumn-list)][, tname[(tcolumn-list)]])}]
[IN] HIERARCHY {STARTING tname | (tname[, tname])}
[UNDER sub-table-name | AS ROOT TABLE]}
```

INSERT

기존 테이블 데이터를 변경하지 않고 임포트된 데이터를 테이블에 추가합니다.

INSERT_UPDATE

임포트된 행의 기본 키가 테이블에 없는 경우 해당 행을 추가하고 해당 기본 키가 있는 경우 이를 갱신에 사용합니다. 이 옵션은 목표 테이블에 기본 키가 있고 임포트 중인 목표 컬럼의 지정된(또는 내재된) 목록에 기본 키의 모든 컬럼이 포함된 경우에만 유효합니다. 이 옵션은 뷰에는 적용할 수 없습니다.

REPLACE

테이블 오브젝트를 잘라내어 테이블에서 모든 기존 데이터를 삭제한 후 임포트된 데이터를 삽입합니다. 테이블 정의 및 인덱스 정의는 변경되지 않습니다. (indexixf가 FileTypeMod에 있고 FileType이 SQL_IXF인 경우 인덱스가 삭제되어 교체됩니다.) 테이블이 이미 정의되어 있지 않은 경우에는 오류가 리턴됩니다.

주: 기존 데이터가 삭제된 후에 오류가 발생하면 해당 데이터는 손실됩니다.
별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

CREATE

주: CREATE 매개변수는 더 이상 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 추가적인 세부사항에 대해서는 『IMPORT 명령 옵션 CREATE 및 REPLACE_CREATE가 사용되지 않음』을 참조하십시오.

지정한 테이블이 정의되지 않은 경우에는 테이블 정의 및 지정된 PC/IXF 파일의 정보를 사용하는 행 컨텐츠를 작성합니다. DB2에서 이전에 파일을 익스포트하지 않은 경우에는 인덱스도 작성됩니다. 지정한 테이블이 이미 정의되어 있는 경우에는 오류가 리턴됩니다. 이 옵션은 PC/IXF 파일 형식에서만 유효합니다. 별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

REPLACE_CREATE

주: REPLACE_CREATE 매개변수는 더 이상 사용되지 않으며 추후 릴리스에서 제거될 수 있습니다. 추가적인 세부사항에 대해서는 『IMPORT 명령 옵션 CREATE 및 REPLACE_CREATE가 사용되지 않음』을 참조하십시오.

지정한 테이블이 정의된 경우 PC/IXF 파일의 PC/IXF 행 정보를 사용하여 테이블 컨텐츠를 교체합니다. 테이블이 아직 정의되어 있지 않은 경우에는 테이블 정의 및 행 컨텐츠가 지정한 PC/IXF 파일의 정보를 사용하여 작성됩니다. DB2에서 이미 PC/IXF 파일을 익스포트한 경우 인덱스도 작성됩니다. 이 옵션은 PC/IXF 파일 형식에서만 유효합니다.

주: 기존 데이터가 삭제된 후에 오류가 발생하면 해당 데이터는 손실됩니다.
별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

tname 데이터가 삽입되는 테이블, 유형이 지정된 테이블, 뷰 또는 오브젝트 뷰의 이름. 규정되거나 규정되지 않은 이름을 지정해야 하는 경우 REPLACE, INSERT_UPDATE 또는 INSERT의 별명은 지정할 수 있습니다(DB2 제품의 이전 버전이 설치된 서버는 제외). 뷰인 경우 읽기 전용 뷰는 불가능합니다.

tcolumn-list
데이터가 삽입되는 테이블 또는 뷰 컬럼 이름 목록. 컬럼 이름은 쉼표로 구분해야 합니다. 컬럼 이름을 지정하지 않으면 CREATE TABLE 또는 ALTER TABLE 문의 컬럼 이름이 사용됩니다. 유형이 지정된 테이블에 대해 컬럼 목록을 지정하지 않은 경우 데이터가 각 서브테이블의 모든 컬럼에 삽입됩니다.

sub-table-name
CREATE 옵션에 하나 이상의 서브테이블을 작성하는 경우 상위 테이블을 지정합니다.

ALL TABLES
계층 구조 전용 내재적 키워드. 계층 구조를 임포트할 때 traversal-order-list로 지정된 모든 테이블을 임포트하는 것이 디폴트입니다.

HIERARCHY
계층 구조 데이터가 임포트되도록 지정합니다.

STARTING
계층 구조의 키워드 전용. 지정한 서브테이블 이름을 시작으로 디폴트 순서가 사용되도록 지정합니다.

UNDER

계층 구조 및 CREATE 키워드 전용. 지정한 서브테이블에 새 계층 구조, 하위 계층 구조 또는 서브테이블이 작성되도록 지정합니다.

AS ROOT TABLE

계층 구조 및 CREATE 키워드 전용. 새 계층 구조, 하위 계층 구조 또는 서브테이블이 독립형 계층 구조로 작성되도록 지정합니다.

tname 및 tcolumn-list 매개변수는 SQL INSERT 문의 tablename 및 colname에 대응하며 동일한 제한사항이 적용됩니다.

tcolumn-list의 컬럼 및 외부 컬럼(지정 또는 내재)은 목록이나 구조의 위치에 따라 일치됩니다(sqldcol 구조에 지정된 첫 번째 데이터가 tcolumn-list의 첫 번째 요소에 따라 테이블이나 뷰 필드에 삽입됨).

동일하지 않은 컬럼 수가 지정된 경우 실제로 처리된 컬럼 수는 두 수 중 작은 수입니다. 이 경우 오류가 발생하거나(일부 널(null) 값을 허용하지 않는 테이블 필드에 값이 없기 때문에) 또는 정보 메시지(일부 외부 파일 컬럼이 무시되기 때문에)가 표시될 수 있습니다.

별칭으로 임포트할 때 이 매개변수는 유효하지 않습니다.

piFileType

입력. 외부 파일 내의 데이터 형식을 나타내는 문자열. 지원되는 외부 파일 형식은 다음과 같습니다.

SQL_ASC

컬럼 식별자가 없는 ASCII

SQL_DEL

컬럼 식별자가 있는 ASCII, dBase, BASIC 및 IBM Personal Decision Series 프로그램과 교환에 사용되는 다수의 데이터베이스 관리 프로그램 및 파일 관리자

SQL_IXF

통합 교환 형식의 PC 버전으로 테이블에서 데이터를 익스포트할 때 이후에 동일한 테이블이나 다른 데이터베이스 관리 프로그램 테이블에 임포트할 수 있기 때문에 자주 사용되는 방법입니다.

SQL_WSF

Lotus Symphony 및 1-2-3 프로그램과의 교환을 위한 워크시트 형식. 별칭으로 임포트할 때 WSF 파일 유형은 지원되지 않습니다.

piFileTypeMod

입력. 하나 이상의 처리 옵션을 지정하는 문자 배열이 뒤에 오는 2바이트 길이의 필드를 포함하는 구조의 포인터. 이 포인터가 NULL이거나 지시된 구조에 영(0) 문자가 있는 경우 이 조치는 디폴트 스펙 선택으로 해석됩니다.

지원되는 모든 파일 유형에 모든 옵션을 사용할 수 있는 것은 아닙니다. 관련 링크인 "임포트 유틸리티의 파일 유형 수정자"를 참조하십시오.

piMsgFileName

입력. 유틸리티에서 리턴된 오류, 경고 및 정보 메시지 목적지를 포함하는 문자열. 운영 체제 파일 또는 표준 디바이스의 경로 및 이름일 수 있습니다. 파일이 이미 있는 경우 추가됩니다. 없는 경우에는 파일이 작성됩니다.

iCallerAction

입력. 호출자가 요청한 조치. 가능한 값은 다음과 같습니다.

SQLU_INITIAL

초기 호출. 이 값을 API의 첫 번째 호출에 사용해야 합니다. 초기 호출이나 이후에 임의의 후속 호출이 리턴되고 요청된 임포트 조작을 완료하기 전에 일부 조치를 수행하는 응용프로그램을 호출해야 하는 경우 호출자 조치는 다음 중 하나로 설정해야 합니다.

SQLU_CONTINUE

처리를 계속합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 완료되고 유틸리티는 초기 요청 처리를 계속하도록 지정합니다.

SQLU_TERMINATE

처리를 종료합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 수행되지 않고 유틸리티가 초기 요청 처리를 종료하도록 지정합니다.

piImportInfoIn

입력. db2ImportIn 구조의 포인터

poImportInfoOut

출력. db2ImportOut 구조의 포인터

piNullIndicators

입력. ASC 파일에만 해당. 컬럼 데이터가 널(NULL) 입력 가능한지 여부를 표시하는 정수 배열. 이 배열의 요소 수는 입력 파일의 컬럼 수와 일치해야 합니다. 이 배열 요소와 데이터 파일에서 임포트 중인 컬럼은 1대1 대응 방식입니다. 따라서 요소 수는 piDataDescriptor 매개변수의 dcolnum 필드와 동일해야 합니다. 배열의 각 요소에는 널(NULL) 표시기 필드로 사용되는 데이터 파일의 컬럼을 식별하는 번호 또는 테이블 컬럼이 널(NULL) 입력 가능한 것을 표시하는 영(0)이 포함됩니다. 요소가 영(0)이 아닌 경우 데이터 파일의 식별된 컬럼에는 Y 또는 N이 포함됩니다. Y는 테이블 컬럼 데이터가 NULL임을 표시하고 N은 테이블 컬럼 데이터가 NULL이 아닌 것을 표시합니다.

piXmlPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 XML 파일이 있는 클라이언트 경로를 나열하는 sqlu_media_list의 포인터

db2ImportIn 데이터 구조 매개변수

iRowcount

입력. 로드할 실제 레코드 수. 사용자가 파일의 첫 번째 iRowcount 행만 로드하도록 허용합니다. iRowcount가 0인 경우 임포트는 파일에서 모든 행을 처리하려고 합니다.

iRestartcount

입력. 레코드의 삽입 또는 갱신을 시작하기 전에 건너뛰어야 하는 레코드 수. 기능상으로는 iSkipcount 매개변수와 동등합니다. iRestartcount 및 iSkipcount 매개변수는 상호 독점적입니다.

iSkipcount

입력. 레코드의 삽입 또는 갱신을 시작하기 전에 건너뛰어야 하는 레코드 수. 기능상으로는 iRestartcount와 동등합니다.

piCommitcount

입력. 데이터베이스에 레코드를 커미트하기 전에 임포트하려는 레코드 수. piCommitcount 레코드가 임포트될 때마다 커미트가 수행됩니다. NULL 값은 디폴트 커미트 계수 값을 지정하고 오프라인 임포트의 경우 영(0)이며 온라인 임포트의 경우 AUTOMATIC입니다. Commitcount AUTOMATIC은 DB2IMPORT_COMMIT_AUTO 값을 전달하여 지정합니다.

iWarningcount

입력. iWarningcount 경고 후에 임포트 조작을 중지합니다. 경고가 예상되지는 않지만 올바른 파일 및 테이블이 사용되고 있는지에 대한 검증을 원할 경우 이 매개변수를 설정하십시오. 임포트 파일 또는 목표 테이블이 올바르지 않게 지정될 경우 임포트 유틸리티는 임포트하려는 각 행에 대해 경고를 생성하므로 임포트를 실패하게 됩니다.

iWarningcount가 0이거나 이 옵션을 지정하지 않은 경우 임포트 조작은 발행된 경고 수에 상관 없이 계속됩니다.

iNoTimeout

입력. 임포트 유틸리티가 잠금 대기 중 시간종료되지 않도록 지정합니다. 이 옵션은 locktimeout 데이터베이스 구성 매개변수를 대체합니다. 다른 응용프로그램에는 영향을 주지 않습니다.가능한 값은 다음과 같습니다.

DB2IMPORT_LOCKTIMEOUT

locktimeout 구성 매개변수 값이 사용되도록 표시합니다.

DB2IMPORT_NO_LOCKTIMEOUT
시간종료가 없음을 표시합니다.

iAccessLevel
입력. 액세스 등급을 지정합니다. 가능한 값은 다음과 같습니다.

- SQLU_ALLOW_NO_ACCESS
임포트 유틸리티가 테이블을 배타적으로 잠그도록 지정합니다.

- SQLU_ALLOW_WRITE_ACCESS
임포트가 진행 중인 동안에도 판독기 및 기록기가 테이블의 데이터에 액세스할 수 있도록 지정합니다.

첫 번째 행이 삽입될 때 목표 테이블에 대해 의도를 가진 독점(IX) 잠금을 획득합니다. 이렇게 하면 동시 판독기 및 기록기가 테이블 데이터에 액세스할 수 있습니다. 온라인 모드는 REPLACE, CREATE 또는 REPLACE_CREATE 임포트 옵션과 호환 가능하지 않습니다. 온라인 모드는 버퍼링된 삽입과 함께 지원되지 않습니다. 임포트 조작은 테이블 잠금에 대한 잠금 에스컬레이션을 방지하고 사용 중인 로그 스페이스 외부에서 실행되지 않도록 하기 위해 삽입된 데이터를 주기적으로 커미트합니다. piCommitCount가 사용되지 않는 경우에도 이 커미트가 수행됩니다. 각 커미트를 수행하는 동안 임포트는 해당 IX 테이블 잠금을 유실하므로 커미트 후 이를 다시 획득하려고 시도합니다. 별칭으로 임포트할 때 이 매개변수가 필요하며 유효한 숫자를 사용하여 piCommitCount 매개변수를 지정해야 합니다(AUTOMATIC은 유효한 옵션으로 간주하지 않음).

piXmlParse
입력. XML 문서에 대해 수행해야 하는 구문 분석 유형. include 디렉토리의 db2ApiDf 헤더 파일에 있는 유효한 값은 다음과 같습니다.

DB2DMU_XMLPARSE_PRESERVE_WS
공백은 유지해야 합니다.

DB2DMU_XMLPARSE_STRIP_WS
공백은 스트립되어야 합니다.

piXmlValidate
입력. db2DMUXmlValidate 구조의 포인터. XML 스키마 유효성 확인이 XML 문서에 대해 수행되어야 하는 것을 나타냅니다.

db2ImportOut 데이터 구조 매개변수

oRowsRead
출력. 임포트 중 파일에서 읽은 레코드 수

oRowsSkipped
출력. 삽입 또는 갱신이 시작되기 전에 건너뛴 레코드 수

oRowsInserted

출력. 목표 테이블로 삽입된 행 수

oRowsUpdated

출력. 임포트된 레코드(테이블에 기본 키 값이 이미 존재하는 레코드)의 정보로 갱신된 목표 테이블의 행 수

oRowsRejected

출력. 임포트할 수 없는 레코드 수

oRowsCommitted

출력. 데이터베이스에 임포트되고 커미트된 레코드 수

db2DMUXmlMapSchema 데이터 구조 매개변수

iMapFromSchema

입력. 맵핑 원본의 XML 스키마에 대한 SQL ID

iMapToSchema

입력. 맵핑 대상의 XML 스키마에 대한 SQL ID

db2DMUXmlValidateXds 데이터 구조 매개변수

piDefaultSchema

입력. XDS에 SCH 속성이 포함되지 않은 경우 유효성 확인에 사용해야 하는 XML 스키마의 SQL ID

iNumIgnoreSchemas

입력. XDS에서 SCH 속성으로 참조되는 경우 XML 스키마 유효성 확인에서 무시되는 XML 스키마 수

piIgnoreSchemas

입력. XDS에서 SCH 속성으로 참조되는 경우 XML 스키마 유효성 확인에서 무시되는 XML 스키마 목록

iNumMapSchemas

입력. XML 스키마 유효성 확인 중에 맵핑되는 XML 스키마 수. 스키마 맵 쌍의 첫 번째 스키마는 XDS의 SCH 속성으로 참조됩니다. 쌍에서 두 번째 스키마는 스키마 유효성 확인을 수행하는 데 사용되어야 하는 스키마를 나타냅니다.

piMapSchemas

입력. XML 스키마 쌍 목록으로 여기서 각 쌍은 임의의 스키마에서 다른 스키마로의 맵핑을 나타냅니다. 쌍에서 첫 번째 스키마는 XDS에 있는 SCH 속성에 의해 참조되는 스키마를 나타냅니다. 쌍에서 두 번째 스키마는 스키마 유효성 확인을 수행하는 데 사용되어야 하는 스키마를 나타냅니다.

db2DMUXmlValidateSchema 데이터 구조 매개변수

piSchema

입력. 사용하려는 XML 스키마의 SQL ID

db2DMUXmlValidate 데이터 구조 매개변수

iUsing 입력. XML 스키마 유효성 확인 수행에 사용하려는 스펙. include 디렉토리의 db2ApiDf 헤더 파일에 있는 유효한 값은 다음과 같습니다.

- **DB2DMU_XMLVAL_XDS**

  XDS에 따라 유효성 확인이 수행되어야 합니다. 이는 CLP ″XMLVALIDATE USING XDS″ 절에 대응합니다.

- **DB2DMU_XMLVAL_SCHEMA**

  지정한 스키마에 따라 유효성 확인이 수행되어야 합니다. 이는 CLP ″XMLVALIDATE USING SCHEMA″ 절에 대응합니다.

- **DB2DMU_XMLVAL_SCHEMALOC_HINTS**

  XML 문서에 있는 schemaLocation 힌트에 따라 유효성 확인이 수행되어야 합니다. 이는 ″XMLVALIDATE USING SCHEMALOCATION HINTS″ 절에 대응합니다.

piXdsArgs

입력. CLP ″XMLVALIDATE USING XDS″ 절에 대응하는 인수를 나타내는 db2DMUXmlValidateXds 구조의 포인터.

이 매개변수는 동일한 구조의 iUsing 매개변수가 DB2DMU_XMLVAL_XDS로 설정된 경우에만 적용됩니다.

piSchemaArgs

입력. CLP ″XMLVALIDATE USING SCHEMA″ 절에 대응하는 인수를 나타내는 db2DMUXmlValidateSchema 구조의 포인터.

이 매개변수는 동일한 구조의 iUsing 매개변수가 DB2DMU_XMLVAL_SCHEMA로 설정된 경우에만 적용됩니다.

db2gImportStruct 데이터 구조 특정 매개변수

iDataFileNameLen

입력. piDataFileName 매개변수 길이를 바이트 단위로 지정합니다.

iFileTypeLen

입력. piFileType 매개변수 길이를 바이트 단위로 지정합니다.

iMsgFileNameLen

입력. piMsgFileName 매개변수 길이를 바이트 단위로 지정합니다.

사용 시 참고사항

임포트 조작을 시작하기 전에 다음 중 하나의 방법을 사용하여 모든 테이블 조작을 완료하고 모든 잠금을 해제해야 합니다.

- WITH HOLD 절로 정의된 모든 열려 있는 커서를 닫고 COMMIT 문을 실행하여 데이터 변경을 커미트하십시오.
- ROLLBACK 문을 실행하여 데이터 변경을 롤백하십시오.

임포트 유틸리티는 SQL INSERT문을 사용하여 목표 테이블에 행을 추가합니다.

이 유틸리티는 입력 파일에 있는 각 데이터 행마다 하나의 INSERT문을 발행합니다. INSERT문을 실패하면 다음 두 가지 조치 중 하나가 발생합니다.

- 후속 INSERT문이 성공할 수 있을 것 같으면 메시지 파일에 경고 메시지를 작성하고 처리를 계속합니다.
- 후속 INSERT문이 실패할 것 같으며 데이터베이스 손상이 생길 가능성이 있으면 메시지 파일에 오류 메시지를 작성하고 처리를 정지합니다.

이 유틸리티는 REPLACE 또는 REPLACE_CREATE 조작 중 이전 행이 삭제된 후 자동 COMMIT를 수행합니다. 따라서 테이블 오브젝트가 절단된 후에 시스템이 실패하거나 응용프로그램이 데이터베이스 관리 프로그램을 인터럽트하면 이전 데이터 모두가 유실됩니다. 이들 옵션을 사용하기 전에 이전 데이터가 더 이상 필요하지 않은지 확인하십시오.

CREATE, REPLACE 또는 REPLACE_CREATE 조작 중 로그가 가득차게 되면 이 유틸리티는 삽입된 레코드에 대해 자동 COMMIT를 수행합니다. 자동 COMMIT 후에 시스템이 실패하거나 응용프로그램이 데이터베이스 관리 프로그램을 인터럽트하면 부분적인 데이터가 포함된 테이블이 데이터베이스에 남게 됩니다. REPLACE 또는 REPLACE_CREATE 옵션을 사용하여 전체 임포트 조작을 재실행하거나 성공적으로 임포트된 행 수로 설정된 iRestartcount 매개변수를 사용하여 INSERT를 사용하십시오.

디폴트로 INSERT 또는 INSERT_UPDATE 옵션에 대해서는 자동 COMMIT가 수행되지 않습니다. 그러나 *piCommitcount 매개변수가 영(0)이 아닌 경우에는 수행됩니다. 전체 로그가 ROLLBACK됩니다.

임포트 유틸리티가 COMMIT를 수행할 때마다 메시지 파일에 두 개의 메시지가 작성되는데, 하나는 커미트되는 레코드 수를 나타내고 다른 하나는 COMMIT 완료 후 작성됩니다. 실패 후 임포트 조작을 재시작할 때 마지막 완료된 COMMIT에서 판별된 대로 생략할 레코드 수를 지정하십시오.

임포트 유틸리티는 사소한 비호환성 문제점이 있는 입력 데이터는 승인합니다(예를 들어, 문자 데이터가 채우기 또는 절단을 사용하여 임포트되고 숫자 데이터가 다른 숫자 데이터 유형으로 임포트될 수 있습니다). 그러나 주요한 비호환성 문제점이 있는 데이터는 승인되지 않습니다.

오브젝트 테이블이 자체 테이블 외에 하위 테이블이 있는 경우 이 오브젝트 테이블을 REPLACE 또는 REPLACE_CREATE 할 수 없으며 기본 테이블에 하위 테이블(자체 테이블을 포함하여)이 있는 경우 해당 오브젝트 뷰를 REPLACE 또는 REPLACE_CREATE 할 수 없습니다. 이러한 테이블 또는 뷰를 바꾸려면 다음을 수행하십시오.

1. 해당 테이블이 상위 테이블인 모든 외부 키를 삭제하십시오.
2. 임포트 유틸리티를 실행하십시오.
3. 테이블을 변경하여 외부 키를 재작성하십시오.

외부 키를 재작성하는 동안 오류가 발생하면 데이터를 수정하여 참조 무결성을 유지보수하십시오.

PC/IXF 파일에서 테이블을 작성할 때 참조 제한조건 및 외부 키 정의는 보존되지 않습니다. (기본 키 정의는 데이터가 이전에 SELECT *를 사용하여 익스포트된 경우에 보존됩니다.)

리모트 데이터베이스로 임포트하려면 입력 데이터 파일의 사본, 출력 메시지 파일 및 잠재적인 데이터베이스 크기 확장에 필요한 디스크 스페이스가 충분해야 합니다.

리모트 데이터베이스에 대해 임포트 조작이 수행되고 출력 메시지 파일이 매우 긴 경우(60KB 초과), 클라이언트의 사용자에게 리턴되는 메시지 파일은 임포트 조작 중간부터 메시지가 누락될 수 있습니다. 메시지 정보의 처음 30KB와 메시지 정보의 마지막 30KB는 항상 유지됩니다.

piDataDescriptor에 대해 디폴트가 아닌 값 또는 piLongActionString의 테이블 컬럼에 대한 내재적인 목록을 지정하면 리모트 데이터베이스로의 임포트가 느려집니다.

ASC, DEL 또는 WSF 파일 형식의 데이터를 임포트하려면 먼저 데이터베이스 테이블 또는 계층 구조가 존재해야 합니다. 그러나 테이블이 이미 존재하지 않을 경우 IMPORT CREATE 또는 IMPORT REPLACE_CREATE가 PC/IXF 파일에서 데이터를 임포트할 때 테이블을 작성합니다. 유형이 지정된 테이블에서 IMPORT CREATE는 자료형 계층 구조와 테이블 계층 구조를 작성할 수 있습니다.

데이터베이스 간에 데이터를 이동하려면(계층 데이터를 포함하여) PC/IXF 임포트를 사용해야 합니다. 행 구분자가 있는 문자 데이터를 컬럼 식별자가 있는 ASCII(DEL) 파일로 익스포트하며 텍스트 전송 프로그램으로 이를 처리할 경우 행 구분자가 있는 필드는 줄어들거나 확장됩니다.

ASC 및 DEL 파일의 데이터는 임포트를 수행하는 클라이언트 응용프로그램의 코드 페이지에 있는 것으로 간주됩니다. 서로 다른 코드 페이지의 데이터를 임포트할 때는 서로 다른 코드 페이지에 사용할 수 있는 PC/IXF 파일을 사용할 것을 권장합니다. PC/IXF 파일과 임포트 유틸리티가 동일한 코드 페이지에 있을 경우 일반 응용프로그램에 대해서와 같은 처리가 발생합니다. 두 코드 페이지가 서로 다르고 FORCEIN 옵션이 지정된 경우, 임포트 유틸리티는 PC/IXF 파일의 데이터가 임포트를 수행 중인 응용프로그램과 동일한 코드 페이지를 가지고 있다고 간주합니다. 이는 두 코드 페이지에 대한 변환 테이블이 있는 경우에도 마찬가지입니다. 두 코드 페이지가 서로 다르고, FORCEIN 옵션이 지정되지 않았으며 변환 테이블이 있는 경우, PC/IXF 파일의 모든 데이터는 파일 코드 페이지에서 응용프로그램 코드 페이지로 변환됩니다. 두 코드 페이지가 서로 다르고 FORCEIN 옵션이 지정되지 않았으며 변환 테이블이 없는 경우 임포트 조작은 실패합니다. 이는 AIX용 DB2 클라이언트에서 PC/IXF 파일에만 적용됩니다.

1012 컬럼의 제한에 근접한 8KB 페이지에 있는 테이블 오브젝트의 경우, PC/IXF 데이터 파일을 임포트하면 SQL 문의 최대 *크기*를 초과했기 때문에 DB2가 오류를 리턴할 수 있습니다. 이러한 상황은 컬럼 유형이 CHAR, VARCHAR 또는 CLOB인 경우에만 발생할 수 있습니다. DEL 또는 ASC 파일을 임포트하는 데는 이러한 제한사항이 적용되지 않습니다.

DB2 Connect를 사용하여 데이터를 OS/390용 DB2, VM 및 VSE용 DB2 및 OS/400® 용 DB2와 같은 DRDA 서버로 임포트할 수 있습니다. PC/IXF 임포트(INSERT 옵션)만 지원됩니다. commitcnt 매개변수가 아닌 restartcnt 매개변수도 지원됩니다.

유형이 지정된 테이블에서 CREATE 옵션을 사용할 때 PC/IXF 파일에 정의된 모든 서브테이블을 작성하십시오. 서브테이블 정의는 변경할 수 없습니다. 유형이 지정된 테이블에서 CREATE 이외의 옵션을 사용할 때, 트래버스 순서 목록을 통해 해당 옵션이 트래버스 순서를 지정하므로 트래버스 순서 목록은 익스포트 조작시 사용된 옵션과 일치해야 합니다. PC/IXF 파일 형식의 경우, 옵션은 목표 서브테이블 이름만 지정해야 하며 파일에 저장된 트래버스 순서를 사용해야 합니다.임포트 유틸리티를 사용하여 이전에 PC/IXF 파일로 익스포트된 테이블을 복구할 수 있습니다. 이 테이블은 익스포트될 때의 상태로 리턴합니다.

데이터는 시스템 테이블, 선언된 임시 테이블, 작성된 임시 테이블 또는 요약 테이블로 임포트할 수 없습니다.

뷰는 임포트 유틸리티를 통해 작성될 수 없습니다.

Windows 운영 체제에서,

- 논리적으로 분리된 PC/IXF 파일 임포트는 지원되지 않습니다.
- 잘못된 형식의 PC/IXF 또는 WSF 파일 임포트는 지원되지 않습니다.

페더레이티드 고려사항

db2Import API 및 INSERT, UPDATE 또는 INSERT_UPDATE 매개변수를 사용하는 경우 참여 중인 별칭에 CONTROL 특권이 있는지 확인해야 합니다. 임포트 조작을 수행할 때 사용할 별칭이 이미 존재하는지 확인해야 합니다.

제 43 장 db2Inspect - 구조적인 무결성을 위해 데이터베이스 검사

구조적인 무결성을 위해 데이터베이스를 검사하고 페이지 일관성을 위해 데이터베이스 페이지를 점검합니다.

범위

단일 파티션된 데이터베이스 환경에서 범위에는 단일 데이터베이스 파티션만 포함됩니다. 이는 파티션된 데이터베이스 환경에서 db2nodes.cfg에 정의된 모든 논리적 데이터베이스 파티션 콜렉션입니다. 파티션된 테이블의 경우, 데이터베이스 및 테이블 스페이스 레벨 검사 범위에는 각 데이터 파티션 및 파티션되지 않은 인덱스가 포함됩니다. 파티션된 테이블의 테이블 레벨 검사에서는 한 개의 데이터 파티션이나 인덱스가 아니라 테이블의 모든 데이터 파티션 및 인덱스가 점검됩니다.

권한 부여

다음 중 하나가 필요합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*
- 테이블에 대한 CONTROL 특권

필수 연결

데이터베이스

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Inspect (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2InspectStruct
{
   char *piTablespaceName;
   char *piTableName;
   char *piSchemaName;
   char *piResultsName;
   char *piDataFileName;
```

```
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint32 iAction;
   db2int32 iTablespaceID;
   db2int32 iObjectID;
   db2Uint32 iFirstPage;
   db2Uint32 iNumberOfPages;
   db2Uint32 iFormatType;
   db2Uint32 iOptions;
   db2Uint32 iBeginCheckOption;
   db2int32 iLimitErrorReported;
   db2Uint16 iObjectErrorState;
   db2Uint16 iCatalogToTablespace;
   db2Uint16 iKeepResultfile;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
   db2Uint16 iLevelObjectData;
   db2Uint16 iLevelObjectIndex;
   db2Uint16 iLevelObjectLong;
   db2Uint16 iLevelObjectLOB;
   db2Uint16 iLevelObjectBlkMap;
   db2Uint16 iLevelExtentMap;
   db2Uint16 iLevelObjectXML;
   db2Uint32 iLevelCrossObject;
} db2InspectStruct;

SQL_API_RC SQL_API_FN
  db2gInspect (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInspectStruct
{
   char *piTablespaceName;
   char *piTableName;
   char *piSchemaName;
   char *piResultsName;
   char *piDataFileName;
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint32 iResultsNameLength;
   db2Uint32 iDataFileNameLength;
   db2Uint32 iTablespaceNameLength;
   db2Uint32 iTableNameLength;
   db2Uint32 iSchemaNameLength;
   db2Uint32 iAction;
   db2int32 iTablespaceID;
   db2int32 iObjectID;
   db2Uint32 iFirstPage;
   db2Uint32 iNumberOfPages;
   db2Uint32 iFormatType;
   db2Uint32 iOptions;
   db2Uint32 iBeginCheckOption;
   db2int32 iLimitErrorReported;
   db2Uint16 iObjectErrorState;
   db2Uint16 iCatalogToTablespace;
   db2Uint16 iKeepResultfile;
   db2Uint16 iAllNodeFlag;
   db2Uint16 iNumNodes;
```

```
   db2Uint16 iLevelObjectData;
   db2Uint16 iLevelObjectIndex;
   db2Uint16 iLevelObjectLong;
   db2Uint16 iLevelObjectLOB;
   db2Uint16 iLevelObjectBlkMap;
   db2Uint16 iLevelExtentMap;
   db2Uint16 iLevelObjectXML;
   db2Uint32 iLevelCrossObject;
} db2gInspectStruct;
```

db2Inspect API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2InspectStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2InspectStruct 데이터 구조 매개변수

piTablespaceName

입력. 테이블 스페이스 이름이 포함된 문자열. 테이블 스페이스는 테이블 스페이스의 조작을 식별해야 합니다. 포인터가 NULL인 경우 입력으로 테이블 스페이스 ID 값이 사용됩니다.

piTableName

입력. 테이블 이름이 포함된 문자열. 테이블은 테이블이나 테이블 오브젝트의 조작을 식별해야 합니다. 포인터가 NULL인 경우 테이블 스페이스 ID 및 테이블 오브젝트 값이 입력으로 사용됩니다.

piSchemaName

입력. 스키마 이름이 포함된 문자열

piResultsName

입력. 결과 출력 파일의 이름이 포함된 문자열. 이 입력은 제공해야 합니다. 파일은 진단 데이터 디렉토리 경로에서 기록됩니다.

piDataFileName

입력. 나중에 사용하기 위해 예약됩니다. NULL로 설정해야 합니다.

piNodeList

입력. 조작을 수행하는 데이터베이스 파티션 번호 배열의 포인터

iAction

입력. 검사 조치를 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2INSPECT_ACT_CHECK_DB
전체 데이터베이스를 검사하십시오.

DB2INSPECT_ACT_CHECK_TABSPACE
테이블 스페이스를 검사하십시오.

DB2INSPECT_ACT_CHECK_TABLE
테이블을 검사하십시오.

DB2INSPECT_ACT_FORMAT_XML
XML 오브젝트 페이지를 형식화하십시오.

DB2INSPECT_ACT_ROWCMPEST_TBL
테이블의 행 압축 효과를 계산하십시오.

iTablespaceID
입력. 테이블 스페이스 ID를 지정합니다. 테이블 스페이스를 식별해야 하는 경우 테이블 스페이스 이름의 포인터가 NULL인 경우 테이블 스페이스 ID 값이 입력으로 사용됩니다.

iObjectID
입력. 오브젝트 ID를 지정합니다. 테이블을 식별해야 하는 경우 테이블 이름의 포인터가 NULL인 경우 오브젝트 ID 값이 입력으로 사용됩니다.

iBeginCheckOption
입력. 조작 시작 위치를 표시하기 위해 데이터베이스를 점검하거나 테이블 스페이스 조작을 점검하는 옵션. 일반적으로 시작하려면 영(0)으로 설정해야 합니다. 값은 다음과 같습니다.

DB2INSPECT_BEGIN_TSPID
테이블 스페이스 ID 필드로 지정한 테이블 스페이스에서 시작하려면 데이터베이스 점검에 이 값을 사용하십시오. 이를 위해서는 테이블 스페이스 ID를 설정해야 합니다.

DB2INSPECT_BEGIN_TSPID_OBJID
테이블 스페이스 ID 및 오브젝트 ID 필드로 지정한 테이블에서 시작하려면 데이터베이스 점검에 이 값을 사용하십시오. 이를 위해서는 테이블 스페이스 ID를 설정해야 합니다. 이 옵션을 사용하려면 테이블 스페이스 ID와 오브젝트 ID를 설정해야 합니다.

DB2INSPECT_BEGIN_OBJID
오브젝트 ID 필드로 지정한 테이블에서 시작하려면 테이블 스페이스 점검에 이 값을 사용하십시오. 이를 위해서는 오브젝트 ID를 설정해야 합니다.

iLimitErrorReported

입력. 오브젝트에 대해 오류가 발생한 페이지 수 보고 제한을 지정합니다. 제한 값으로 사용하려는 수를 지정하거나 다음 값 중 하나를 지정하십시오.

DB2INSPECT_LIMIT_ERROR_DEFAULT

보고할 오류 발생 최대 페이지 수가 오브젝트의 Extent 크기가 되도록 지정하려면 이 값을 사용하십시오.

DB2INSPECT_LIMIT_ERROR_ALL

이 값을 사용하여 오류가 발생한 모든 페이지를 보고하십시오.

DB2INSPECT_LVL_XOBJ_INXDAT_RID를 iLevelCrossObject 필드, 지정한 제한 값 또는 위의 DEFAULT 또는 ALL 값에 사용하는 경우 오류가 발생한 페이지 수 대신 데이터의 온라인 인덱스 일관성 검사 중에 보고되는 오류 수 제한을 표시하십시오.

iObjectErrorState

입력. 오류 상태의 오브젝트 스캔 여부를 지정합니다. 가능한 값은 다음과 같습니다.

DB2INSPECT_ERROR_STATE_NORMAL

일반 상태의 오브젝트만 처리하십시오.

DB2INSPECT_ERROR_STATE_ALL

오류 상태의 오브젝트를 포함하여 모든 오브젝트를 처리하십시오.

DB2INSPECT_LVL_XOBJ_INXDAT_RID를 iLevelCrossObject 필드에 사용하는 경우 인덱스나 데이터 오브젝트가 오류 상태인 동안 DB2INSPECT_ERROR_STATE_ALL이 이 필드에 지정된 경우 무시되며 데이터의 온라인 인덱스 일관성 검사가 수행되지 않습니다.

iKeepResultfile

입력. 결과 파일 보존을 지정합니다. 가능한 값은 다음과 같습니다.

DB2INSPECT_RESFILE_CLEANUP

오류가 보고되는 경우 결과 출력 파일이 보유됩니다. 그 외의 경우에는 결과 파일은 조작이 종료되면 제거됩니다.

DB2INSPECT_RESFILE_KEEP_ALWAYS

결과 출력 파일이 보유됩니다.

iAllNodeFlag

입력. 조작이 db2nodes.cfg에 정의된 모든 노드에 적용되는지를 표시합니다. 가능한 값은 다음과 같습니다.

DB2_NODE_LIST

pNodeList로 전달된 노드 목록의 모든 노드에 적용됩니다.

DB2_ALL_NODES

모든 노드에 적용됩니다. pNodeList는 NULL이어야 합니다. 이것은 디폴트값입니다.

DB2_ALL_EXCEPT

pNodeList로 전달된 노드 목록의 노드를 제외한 모든 노드에 적용됩니다.

iNumNodes

입력. pNodeList 배열의 노드 수를 지정합니다.

iLevelObjectData

입력. 데이터 오브젝트의 처리 레벨을 지정합니다. 가능한 값은 다음과 같습니다.

DB2INSPECT_LEVEL_NORMAL

레벨이 일반입니다.

DB2INSPECT_LEVEL_LOW

레벨이 낮음입니다.

DB2INSPECT_LEVEL_NONE

레벨이 없음입니다.

iLevelObjectIndex

입력. 인덱스 오브젝트의 처리 레벨을 지정합니다. 가능한 값은 다음과 같습니다.

DB2INSPECT_LEVEL_NORMAL

레벨이 일반입니다.

DB2INSPECT_LEVEL_LOW

레벨이 낮음입니다.

DB2INSPECT_LEVEL_NONE

레벨이 없음입니다.

iLevelObjectLong

입력. Long 오브젝트의 처리 레벨을 지정합니다. 가능한 값은 다음과 같습니다.

DB2INSPECT_LEVEL_NORMAL

레벨이 일반입니다.

DB2INSPECT_LEVEL_LOW

레벨이 낮음입니다.

DB2INSPECT_LEVEL_NONE

레벨이 없음입니다.

iLevelObjectLOB

입력. LOB 오브젝트의 처리 레벨을 지정합니다. 가능한 값은 다음과 같습니다.

DB2INSPECT_LEVEL_NORMAL
레벨이 일반입니다.

DB2INSPECT_LEVEL_LOW
레벨이 낮음입니다.

DB2INSPECT_LEVEL_NONE
레벨이 없음입니다.

iLevelObjectBlkMap

입력. 블록 맵 오브젝트의 처리 레벨을 지정합니다. 가능한 값은 다음과 같습니다.

DB2INSPECT_LEVEL_NORMAL
레벨이 일반입니다.

DB2INSPECT_LEVEL_LOW
레벨이 낮음입니다.

DB2INSPECT_LEVEL_NONE
레벨이 없음입니다.

iLevelExtentMap

입력. Extent Map의 처리 레벨을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2INSPECT_LEVEL_NORMAL
레벨이 일반입니다.

DB2INSPECT_LEVEL_LOW
레벨이 낮음입니다.

DB2INSPECT_LEVEL_NONE
레벨이 없음입니다.

iLevelObjectXML

입력. XML 오브젝트의 처리 레벨을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2INSPECT_LEVEL_NORMAL
레벨이 일반입니다.

DB2INSPECT_LEVEL_LOW
레벨이 낮음입니다.

DB2INSPECT_LEVEL_NONE
레벨이 없음입니다.

iLevelCrossObject

오브젝트 간 일관성 검사에 사용되는 비트 기반 필드. 가능한 값은 다음과 같습니다.

DB2INSPECT_LVL_XOBJ_NONE

온라인 인덱스 데이터 일관성 검사는 수행되지 않습니다(0x00000000).

DB2INSPECT_LVL_XOBJ_INXDAT_RID

INDEXDATA 검사가 RID 인덱스(0x00000001)에서 사용되며 IS 테이블 잠금을 사용하여 수행되어 판독기 및 기록기 모두에 허용됩니다.

db2gInspectStruct 데이터 구조 특정 매개변수

iResultsNameLength

입력. 결과 파일 이름의 문자열 길이

iDataFileNameLength

입력. 데이터 출력 파일 이름의 문자열 길이

iTablespaceNameLength

입력. 테이블 스페이스 이름의 문자열 길이

iTableNameLength

입력. 테이블 이름의 문자열 길이

iSchemaNameLength

입력. 스키마 이름의 문자열 길이

사용 시 참고사항

온라인 검사 처리에서는 분리 레벨의 언커미트 읽기를 사용하여 데이터베이스 오브젝트에 액세스합니다. 검사 처리 중에 커미트 처리가 완료됩니다. 검사 조작을 시작하기 전에 각각 COMMIT 또는 ROLLBACK 문을 실행하여 커미트 또는 롤백 변경으로 작업 단위(UOW)를 종료하는 것이 바람직합니다.

검사 점검 처리 중에 형식화되지 않은 검사 데이터 결과가 결과 파일로 작성됩니다. 파일은 진단 데이터 디렉토리 경로에서 기록됩니다. 점검 처리 중에 오류가 발견되지 않은 경우 결과 출력 파일은 점검 조작 종료 시에 지워집니다. 점검 처리 중에 오류가 발견된 경우 결과 출력 파일은 검사 조작 종료 시에도 지워지지 않습니다. 검사 세부사항을 확인하려면 검사 결과 출력 파일을 db2inspf 유틸리티로 형식화하십시오.

파티션된 데이터베이스 환경에서 결과 출력 파일의 확장자는 데이터베이스 파티션 번호에 해당합니다. 파일은 데이터베이스 관리 프로그램 진단 데이터 디렉토리 경로에 있습니다.

고유한 결과 출력 파일 이름을 지정해야 합니다. 결과 출력 파일이 이미 있는 경우 조작이 처리되지 않습니다.

db2Inspect API를 호출하는 경우 db2InspectStruct에서 적절한 값으로 iLevelCrossObject를 지정해야 합니다. DB2INSPECT_LVL_XOBJ_NONE을 사용하는 경우 온라인 인덱스 데이터 일관성 검사가 수행되지 않습니다. 온라인 인덱스 데이터 일관성 검사를 사용하려면 iLevelCrossObject 필드에 DB2INSPECT_LVL_XOBJ_INXDAT_RID를 지정해야 합니다.

테이블 스페이스 처리를 통해 해당 테이블 스페이스에 있는 오브젝트만 처리됩니다. 인덱스 오브젝트가 검사할 테이블 스페이스에 있는 동안 데이터 오브젝트가 다른 테이블 스페이스에 상주할 수 있고 여전히 검사를 통해 얻을 수 있는 장점이 있는 경우의 인덱스 데이터 일관성 검사는 예외입니다. 파티션된 테이블의 경우 각 인덱스는 다른 테이블 스페이스에 상주할 수 있습니다. 지정한 테이블 스페이스에 있는 인덱스만 데이터의 인덱스 검사에 활용할 수 있습니다.

제 44 장 db2InstanceQuiesce - 인스턴스 Quiesce

모든 사용자가 인스턴스를 종료하고 즉시 모든 활성 트랜잭션을 롤백한 다음 데이터베이스를 Quiesce 모드로 설정합니다. 이 API는 인스턴스에 대한 독점 액세스를 제공합니다. 이 Quiesce 상태 기간 동안에 시스템 관리는 인스턴스에서 수행할 수 있습니다. 관리가 완료되면 db2DatabaseUnquiesce를 사용하여 데이터베이스를 다시 Unquiesce 할 수 있습니다. 이 API를 사용하면 데이터베이스를 종료한 후에 다른 데이터베이스를 시작하지 않고도 다른 사용자가 데이터베이스에 연결할 수 있습니다.

이 모드에서는 QUIESCE CONNECT 권한 및 sysadm, sysmaint 또는 sysctrl 권한이 있는 사용자나 그룹만 데이터베이스 및 해당 오브젝트에 액세스합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2InstanceQuiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2InsQuiesceStruct
{
             char *piInstanceName;
             char *piUserId;
             char *piGroupId;
             db2Uint32 iImmediate;
             db2Uint32 iForce;
             db2Uint32 iTimeout;
} db2InsQuiesceStruct;

SQL_API_RC SQL_API_FN
  db2gInstanceQuiesce (
             db2Uint32 versionNumber,
             void * pParmStruct,
```

```
            struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInsQuiesceStruct
{
            db2Uint32 iInstanceNameLen;
            char *piInstanceName;
            db2Uint32 iUserIdLen;
            char *piUserId;
            db2Uint32 iGroupIdLen;
            char *piGroupId;
            db2Uint32 iImmediate;
            db2Uint32 iForce;
            db2Uint32 iTimeout;
} db2gInsQuiesceStruct;
```

db2InstanceQuiesce API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2InsQuiesceStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2InsQuiesceStruct 데이터 구조 매개변수

piInstanceName

입력. 인스턴스 이름

piUserId

입력. Quiesce 상태인 인스턴스에 액세스할 수 있는 사용자 이름

piGroupId

입력. Quiesce 상태인 인스턴스에 액세스할 수 있는 그룹 이름

iImmediate

입력. 가능한 값은 다음과 같습니다.

TRUE=1

응용프로그램을 즉시 강제로 실행합니다.

FALSE=0

지연된 강제 실행. 응용프로그램은 iTimeout 매개변수로 지정한 시간(분) 동안 대기하여 현재 작업 단위(UOW)가 완료되도록 한 다음 종료됩니다. iTimeout 매개변수로 지정한 시간(분) 동안 이 지연된 강제 실행을 완료할 수 없으면 Quiesce 조작이 실패합니다.

iForce 입력. 나중에 사용하기 위해 예약됨

iTimeout

입력. 응용프로그램이 현재 작업 단위(UOW)를 커미트할 때까지 대기하는 시간(분)을 지정합니다. iTimeout을 지정하지 않은 경우 단일 파티션 데이터베이스 환경에서 디폴트값은 10분입니다. 파티션된 데이터베이스 환경에서는 start_stop_time 데이터베이스 관리 프로그램 구성 매개변수로 지정한 값이 사용됩니다.

db2gInsQuiesceStruct 데이터 구조 특정 매개변수

iInstanceNameLen

입력. piInstanceName 길이를 바이트 단위로 지정합니다.

iUserIdLen

입력. piUserID 길이를 바이트 단위로 지정합니다.

iGroupIdLen

입력. piGroupId 길이를 바이트 단위로 지정합니다.

제 45 장 db2InstanceStart - 인스턴스 시작

로컬 또는 리모트 인스턴스를 시작합니다.

범위

단일 파티션 데이터베이스 환경에서 범위는 해당하는 단일 데이터베이스 파티션만 해당합니다. 파티션된 데이터베이스 환경에서 노드 구성 파일인 db2nodes.cfg에 정의된 모든 논리적 데이터베이스 파티션 서버의 콜렉션입니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

없음

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2InstanceStart (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2InstanceStartStruct
{
             db2int8 iIsRemote;
             char *piRemoteInstName;
             db2DasCommData * piCommData;
             db2StartOptionsStruct * piStartOpts;
} db2InstanceStartStruct;

typedef SQL_STRUCTURE db2DasCommData
{
             db2int8 iCommParam;
             char *piNodeOrHostName;
             char *piUserId;
             char *piUserPw;
} db2DasCommData;
```

```
typedef SQL_STRUCTURE db2StartOptionsStruct
{
             db2Uint32 iIsProfile;
             char *piProfile;
             db2Uint32 iIsNodeNum;
             db2NodeType iNodeNum;
             db2Uint32 iOption;
             db2Uint32 iIsHostName;
             char *piHostName;
             db2Uint32 iIsPort;
             db2PortType iPort;
             db2Uint32 iIsNetName;
             char *piNetName;
             db2Uint32 iTblspaceType;
             db2NodeType iTblspaceNode;
             db2Uint32 iIsComputer;
             char *piComputer;
             char *piUserName;
             char *piPassword;
             db2QuiesceStartStruct iQuiesceOpts;
} db2StartOptionsStruct;

typedef SQL_STRUCTURE db2QuiesceStartStruct
{
             db2int8 iIsQRequested;
             char *piQUsrName;
             char *piQGrpName;
             db2int8 iIsQUsrGrpDef;
} db2QuiesceStartStruct;

SQL_API_RC SQL_API_FN
  db2gInstanceStart (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInstanceStStruct
{
             db2int8 iIsRemote;
             db2Uint32 iRemoteInstLen;
             char *piRemoteInstName;
             db2gDasCommData * piCommData;
             db2gStartOptionsStruct * piStartOpts;
} db2gInstanceStStruct;

typedef SQL_STRUCTURE db2gDasCommData
{
             db2int8 iCommParam;
             db2Uint32 iNodeOrHostNameLen;
             char *piNodeOrHostName;
             db2Uint32 iUserIdLen;
             char *piUserId;
             db2Uint32 iUserPwLen;
             char *piUserPw;
} db2gDasCommData;

typedef SQL_STRUCTURE db2gStartOptionsStruct
{
```

```
            db2Uint32 iIsProfile;
            char *piProfile;
            db2Uint32 iIsNodeNum;
            db2NodeType iNodeNum;
            db2Uint32 iOption;
            db2Uint32 iIsHostName;
            char *piHostName;
            db2Uint32 iIsPort;
            db2PortType iPort;
            db2Uint32 iIsNetName;
            char *piNetName;
            db2Uint32 iTblspaceType;
            db2NodeType iTblspaceNode;
            db2Uint32 iIsComputer;
            char *piComputer;
            char *piUserName;
            char *piPassword;
            db2gQuiesceStartStruct iQuiesceOpts;
} db2gStartOptionsStruct;

typedef SQL_STRUCTURE db2gQuiesceStartStruct
{
            db2int8 iIsQRequested;
            db2Uint32 iQUsrNameLen;
            char *piQUsrName;
            db2Uint32 iQGrpNameLen;
            char *piQGrpName;
            db2int8 iIsQUsrGrpDef;
} db2gQuiesceStartStruct;
```

db2InstanceStart API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2InstanceStartStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2InstanceStartStruct 데이터 구조 매개변수

iIsRemote

입력. 상수 정수값 참 또는 거짓으로 설정되는 표시기. 이 매개변수는 리모트 시작인 경우 참으로 설정해야 합니다.

piRemoteInstName

입력. 리모트 인스턴스 이름

piCommData

입력. db2DasCommData 구조의 포인터

piStartOpts

입력. db2StartOptionsStruct 구조의 포인터

db2DasCommData 데이터 구조 매개변수

iCommParam

입력. 참 또는 거짓으로 설정되는 표시기. 이 매개변수는 리모트 시작인 경우 참으로 설정해야 합니다.

piNodeOrHostName

입력. 데이터베이스 파티션 또는 호스트 이름

piUserId

입력. 사용자 이름

piUserPw

입력. 사용자 암호

db2StartOptionsStruct 데이터 구조 매개변수

iIsProfile

입력. 프로파일 지정 여부를 표시합니다. 이 필드가 프로파일이 지정되지 않는 것을 나타내는 경우에는 db2profile 파일이 사용됩니다.

piProfile

입력. DB2 환경을 정의하기 위해 각 노드에서 실행되는 프로파일 파일 이름 (MPP 전용). 이 파일은 노드가 시작되기 전에 실행됩니다. 디폴트값은 db2profile입니다.

iIsNodeNum

입력. 노드 번호 지정 여부를 표시합니다. 지정하는 경우 시작 명령은 지정한 노드에만 영향을 줍니다.

iNodeNum

입력. 데이터베이스 파티션 번호

iOption

입력. 조치를 지정합니다. OPTION에 유효한 값(include 디렉토리에 있는 sqlenv 헤더 파일에 정의)은 다음과 같습니다.

SQLE_NONE

일반 db2start 조작을 실행합니다.

SQLE_ADDNODE

ADD NODE 명령을 실행합니다.

SQLE_RESTART

RESTART DATABASE 명령을 실행합니다.

SQLE_RESTART_PARALLEL
병렬 실행을 위해 RESTART DATABASE 명령을 실행합니다.

SQLE_STANDALONE
STANDALONE 모드로 노드를 시작합니다.

iIsHostName
입력. 호스트 이름 지정 여부를 나타냅니다.

piHostName
입력. 시스템 이름

iIsPort
입력. 포트 번호 지정 여부를 나타냅니다.

iPort 입력. 포트 번호

iIsNetName
입력. net 이름 지정 여부를 나타냅니다.

piNetName
입력. 네트워크 이름

iTblspaceType
입력. 추가 중인 노드에 사용되는 시스템 임시 테이블 스페이스 정의 유형을 지정합니다. 가능한 값은 다음과 같습니다.

SQLE_TABLESPACES_NONE
시스템 임시 테이블 스페이스를 작성하지 않습니다.

SQLE_TABLESPACES_LIKE_NODE
시스템 임시 테이블 스페이스 컨테이너는 지정한 노드의 컨테이너와 동일해야 합니다.

SQLE_TABLESPACES_LIKE_CATALOG
시스템 임시 테이블 스페이스 컨테이너는 각 데이터베이스의 카탈로그 노드의 컨테이너와 동일해야 합니다.

iTblspaceNode
입력. 시스템 임시 테이블 스페이스 정의가 확보해야 하는 노드 번호를 지정합니다. 노드 번호는 db2nodes.cfg 파일에 있어야 하며 tblspace_type 필드를 SQLE_TABLESPACES_LIKE_NODE로 설정한 경우에만 사용됩니다.

iIsComputer
입력. 컴퓨터 이름 지정 여부를 나타냅니다. Windows 운영 체제에서만 유효합니다.

piComputer
입력. 컴퓨터 이름. Windows 운영 체제에서만 유효합니다.

piUserName

입력. 로그온 어카운트 사용자 이름 Windows 운영 체제에서만 유효합니다.

piPassword

입력. 로그온 어카운트 사용자 이름에 해당하는 암호

iQuiesceOpts

입력. db2QuiesceStartStruct 구조의 포인터

db2QuiesceStartStruct 데이터 구조 매개변수

iIsQRequested

입력. 참 또는 거짓으로 설정되는 표시기. 이 매개변수는 Quiesce가 요청된 경우 참으로 설정해야 합니다.

piQUsrName

입력. Quiesce 상태의 사용자 이름

piQGrpName

입력. Quiesce 상태의 그룹 이름

iIsQUsrGrpDef

입력. 참 또는 거짓으로 설정되는 표시기. 이 매개변수는 Quiesce 상태의 사용자나 Quiesce 상태의 그룹이 정의된 경우에는 참으로 설정해야 합니다.

db2gInstanceStStruct 데이터 구조 특정 매개변수

iRemoteInstLen

입력. piRemoteInstName 길이를 바이트 단위로 지정합니다.

db2gDasCommData 데이터 구조 특정 매개변수

iNodeOrHostNameLen

입력. piNodeOrHostName 길이를 바이트 단위로 지정합니다.

iUserIdLen

입력. piUserId 길이를 바이트 단위로 지정합니다.

iUserPwLen

입력. piUserPw 길이를 바이트 단위로 지정합니다.

db2gQuiesceStartStruct 데이터 구조 특정 매개변수

iQUsrNameLen

입력. piQusrName 길이를 바이트 단위로 지정합니다.

iQGrpNameLen

입력. piQGrpName 길이를 바이트 단위로 지정합니다.

제 46 장 db2InstanceStop - 인스턴스 중지

로컬 또는 리모트 DB2 인스턴스를 중지합니다.

범위

단일 파티션 데이터베이스 환경에서 범위는 해당하는 단일 데이터베이스 파티션만 해당합니다. 파티션된 데이터베이스 환경에서 노드 구성 파일인 db2nodes.cfg에 정의된 모든 논리적 데이터베이스 파티션 서버의 콜렉션입니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2InstanceStop (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2InstanceStopStruct
{
             db2int8 iIsRemote;
             char *piRemoteInstName;
             db2DasCommData * piCommData;
             db2StopOptionsStruct * piStopOpts;
} db2InstanceStopStruct;

typedef SQL_STRUCTURE db2DasCommData
{
             db2int8 iCommParam;
             char *piNodeOrHostName;
             char *piUserId;
             char *piUserPw;
} db2DasCommData;
```

```
typedef SQL_STRUCTURE db2StopOptionsStruct
{
             db2Uint32 iIsProfile;
             char *piProfile;
             db2Uint32 iIsNodeNum;
             db2NodeType iNodeNum;
             db2Uint32 iStopOption;
             db2Uint32 iCallerac;
} db2StopOptionsStruct;

SQL_API_RC SQL_API_FN
  db2gInstanceStop (
             db2Uint32 versionNumber,
             void * pParmStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInstanceStopStruct
{
             db2int8 iIsRemote;
             db2Uint32 iRemoteInstLen;
             char *piRemoteInstName;
             db2gDasCommData * piCommData;
             db2StopOptionsStruct * piStopOpts;
} db2gInstanceStopStruct;

typedef SQL_STRUCTURE db2gDasCommData
{
             db2int8 iCommParam;
             db2Uint32 iNodeOrHostNameLen;
             char *piNodeOrHostName;
             db2Uint32 iUserIdLen;
             char *piUserId;
             db2Uint32 iUserPwLen;
             char *piUserPw;
} db2gDasCommData;
```

db2InstanceStop API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2InstanceStopStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2InstanceStopStruct 데이터 구조 매개변수

iIsRemote

입력. 참 또는 거짓으로 설정되는 표시기. 이 매개변수는 리모트 시작인 경우 참으로 설정해야 합니다.

piRemoteInstName

입력. 리모트 인스턴스 이름

piCommData

입력. db2DasCommData 구조의 포인터

piStopOpts

입력. db2StopOptionsStruct 구조의 포인터

db2DasCommData 데이터 구조 매개변수

iCommParam

입력. 참 또는 거짓으로 설정되는 표시기. 이 매개변수는 리모트 시작인 경우 참으로 설정해야 합니다.

piNodeOrHostName

입력. 데이터베이스 파티션 또는 호스트 이름

piUserId

입력. 사용자 이름

piUserPw

입력. 사용자 암호

db2StopOptionsStruct 데이터 구조 매개변수

iIsProfile

입력. 프로파일 지정 여부를 표시합니다. 가능한 값은 참 및 거짓입니다. 이 필드가 프로파일이 지정되지 않는 것을 나타내는 경우에는 db2profile 파일이 사용됩니다.

piProfile

입력. 시작된 해당 노드에 대해 DB2 환경을 정의하기 위해 시작 시에 실행된 프로파일 파일 이름(MPP 전용). db2InstanceStart API의 프로파일이 지정된 경우 여기에 동일한 프로파일을 지정해야 합니다.

iIsNodeNum

입력. 노드 번호 지정 여부를 표시합니다. 가능한 값은 참 및 거짓입니다. 지정하는 경우 중지 명령은 지정한 노드에만 영향을 줍니다.

iNodeNum

입력. 데이터베이스 파티션 번호

iStopOption

입력. 옵션. 가능한 값은 다음과 같습니다.

SQLE_NONE

일반 db2stop 조작을 실행합니다.

SQLE_FORCE

FORCE APPLICATION (ALL) 명령을 실행하십시오.

SQLE_DROP

db2nodes.cfg 파일에서 노드를 삭제하십시오.

iCallerac

입력. 이 필드는 OPTION 필드의 SQLE_DROP 값에 대해서만 유효합니다. 가능한 값은 다음과 같습니다.

SQLE_DROP

초기 호출. 이것은 디폴트값입니다.

SQLE_CONTINUE

연속 호출. 프롬프트 이후에 처리가 계속됩니다.

SQLE_TERMINATE

연속 호출. 프롬프트 이후에 처리를 종료합니다.

db2gInstanceStopStruct 데이터 구조 특정 매개변수

iRemoteInstLen

입력. piRemoteInstName 길이를 바이트 단위로 지정합니다.

db2gDasCommData 데이터 구조 특정 매개변수

iNodeOrHostNameLen

입력. piNodeOrHostName 길이를 바이트 단위로 지정합니다.

iUserIdLen

입력. piUserId 길이를 바이트 단위로 지정합니다.

iUserPwLen

입력. piUserPw 길이를 바이트 단위로 지정합니다.

제 47 장 db2InstanceUnquiesce - 인스턴스 Unquiesce

인스턴스의 모든 데이터베이스를 Unquiesce합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2InstanceUnquiesce (
              db2Uint32 versionNumber,
              void * pParmStruct,
              struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2InsUnquiesceStruct
{
              char *piInstanceName;
} db2InsUnquiesceStruct;

SQL_API_RC SQL_API_FN
  db2gInstanceUnquiesce (
              db2Uint32 versionNumber,
              void * pParmStruct,
              struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gInsUnquiesceStruct
{
              db2Uint32 iInstanceNameLen;
              char *piInstanceName;
} db2gInsUnquiesceStruct;
```

db2InstanceUnquiesce API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2InsUnquiesceStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2InsUnquiesceStruct 데이터 구조 매개변수

piInstanceName

입력. 인스턴스 이름

db2gInsUnquiesceStruct 데이터 구조 특정 매개변수

iInstanceNameLen

입력. piInstanceName 길이를 바이트 단위로 지정합니다.

제 48 장 db2LdapCatalogDatabase - LDAP 서버에 데이터베이스 등록

LDAP(Lightweight Directory Access Protocol)에 데이터베이스 항목을 카탈로그합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2LdapCatalogDatabase (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapCatalogDatabaseStruct
{
   char *piAlias;
   char *piDatabaseName;
   char *piComment;
   char *piNodeName;
   char *piGWNodeName;
   char *piParameters;
   char *piARLibrary;
   unsigned short                      iAuthentication;
   char *piDCEPrincipalName;
   char *piBindDN;
   char *piPassword;
} db2LdapCatalogDatabaseStruct;
```

db2LdapCatalogDatabase API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParamStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParamStruct

입력. db2LdapCatalogDatabaseStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LdapCatalogDatabaseStruct 데이터 구조 매개변수

piAlias

입력. 카탈로그 중인 데이터베이스의 대체 이름으로 사용되는 별명을 지정하십시오. 별명을 지정하지 않으면 데이터베이스 관리 프로그램이 데이터베이스 이름을 별명 이름으로 사용합니다.

piDatabaseName

입력. 카탈로그되는 데이터베이스 이름을 지정하십시오. 이 매개변수는 필수입니다.

piComment

입력. DB2 서버를 설명하십시오. 네트워크 디렉토리에 등록된 서버를 설명하는 데 사용되는 주석을 입력할 수 있습니다. 최대 길이는 30자입니다. 캐리지 리턴 또는 줄 바꾸기 문자를 사용할 수 없습니다.

piNodeName

입력. 데이터베이스가 있는 데이터베이스 서버의 노드 이름을 지정하십시오. 데이터베이스가 리모트 데이터베이스 서버에 있는 경우에는 이 매개변수가 필수입니다.

piGWNodename

입력. DB2 Connect 게이트웨이 서버의 노드 이름을 지정하십시오. 데이터베이스 서버 노드 유형이 DCS(호스트 데이터베이스 서버용으로 예약)이고 클라이언트에 DB2 Connect가 설치되지 않은 경우, 클라이언트는 DB2 Connect 게이트웨이 서버에 연결됩니다.

piParameters

입력. 응용프로그램 리퀘스터(AR)에 전달되는 매개변수 문자열을 지정하십시오. 인증 DCE는 지원되지 않습니다.

piARLibrary

입력. 응용프로그램 리퀘스터(AR) 라이브러리 이름을 지정하십시오.

iAuthentication

입력. 인증 유형을 지정하면 성능이 향상됩니다.

piDCEPrincipalName

입력. 목표 서버에 대해 완전한 DCE 핵심부 이름을 지정하십시오.

piBindDN

입력. 사용자의 LDAP 식별 이름(DN)을 지정하십시오. LDAP 사용자 DN에

는 LDAP 디렉토리에서 오브젝트를 작성하고 갱신할 수 있는 권한이 있어야 합니다. 사용자의 LDAP DN을 지정하지 않으면 현재 로그온 사용자의 증명서가 사용됩니다.

piPassword

입력. 어카운트 암호

사용 시 참고사항

다음과 같은 상황인 경우 데이터베이스를 수동으로 등록하거나 카탈로그해야 합니다.

데이터베이스 서버가 LDAP를 지원하지 않습니다. 이 경우 관리자는 LDAP에 각 데이터베이스를 수동으로 등록하여 LDAP를 지원하는 클라이언트가 각 클라이언트 머신에서 데이터베이스를 로컬로 카탈로그하지 않고 데이터베이스에 액세스할 수 있도록 해야 합니다.

응용프로그램이 데이터베이스 연결에 다른 이름을 사용합니다. 이 경우 관리자는 다른 별명 이름을 사용하여 데이터베이스를 카탈로그해야 합니다.

CREATE DATABASE IN LDAP에서 데이터베이스 이름이 이미 LDAP에 있습니다. 여전히 로컬 시스템에 데이터베이스가 작성되어 로컬 응용프로그램에서 액세스할 수 있지만 LDAP의 기존 항목은 새 데이터베이스를 반영하기 위해 수정되지는 않습니다. 이 경우 관리자는 LDAP에서 기존 데이터베이스 항목을 제거하고 LDAP에 수동으로 새 데이터베이스를 등록할 수 있습니다. 다른 별명 이름을 사용하여 LDAP에 새 데이터베이스를 등록할 수도 있습니다.

제 49 장 db2LdapCatalogNode - LDAP 서버에서 노드 이름의 별명 제공

LDAP(Lightweight Directory Access Protocol)에서 노드의 대체 이름을 지정하거나 데이터베이스 서버에 연결을 위한 다른 프로토콜 유형을 지정합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2LdapCatalogNode (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapCatalogNodeStruct
{
   char *piAlias;
   char *piNodeName;
   char *piBindDN;
   char *piPassword;
} db2LdapCatalogNodeStruct;
```

db2LdapCatalogNode API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParamStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParamStruct

입력. db2LdapCatalogNodeStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LdapCatalogNodeStruct 데이터 구조 매개변수

piAlias

입력. 노드의 대체 이름으로 사용되는 새 별명을 지정하십시오.

piNodeName

입력. LDAP에서 DB2 서버를 나타내는 노드 이름을 지정하십시오.

piBindDN

입력. 사용자의 LDAP 식별 이름(DN)을 지정하십시오. LDAP 사용자 DN에는 LDAP 디렉토리에서 오브젝트를 작성하고 갱신할 수 있는 권한이 있어야 합니다. 사용자의 LDAP DN을 지정하지 않으면 현재 로그온 사용자의 증명서가 사용됩니다.

piPassword

입력. 어카운트 암호

제 50 장 db2LdapDeregister - LDAP 서버에서 DB2 서버 및 카탈로그된 데이터베이스 등록 해제

LDAP(Lightweight Directory Access Protocol)에서 DB2 서버의 등록을 해제합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2LdapDeregister (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapDeregisterStruct
{
   char *piNodeName;
   char *piBindDN;
   char *piPassword;
} db2LdapDeregisterStruct;
```

db2LdapDeregister API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParamStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParamStruct

입력. db2LdapDeregisterStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LdapDeregisterStruct 데이터 구조 매개변수

piNodeName

입력. LDAP에서 DB2 서버를 나타내는 단축 이름을 지정하십시오.

piBindDN

입력. 사용자의 LDAP 식별 이름(DN)을 지정하십시오. LDAP 사용자 DN에는 LDAP 디렉토리에서 오브젝트를 삭제하는 데 필요한 권한이 있어야 합니다. 사용자의 LDAP DN을 지정하지 않으면 현재 로그온 사용자의 증명서가 사용됩니다.

piPassword

입력. 어카운트 암호

제 51 장 db2LdapRegister - LDAP 서버에 DB2 서버 등록

LDAP(Lightweight Directory Access Protocol)에 DB2 서버를 등록합니다.

주: NetBIOS는 더 이상 지원되지 않습니다. 해당 API인 APPC, APPN 및 CPI-C가 포함된 SNA가 더 이상 지원되지 않습니다. 이 프로토콜을 사용하는 경우에는 TCP/IP와 같이 지원되는 프로토콜을 사용하여 노드와 데이터베이스를 재카탈로그해야 합니다. 이 프로토콜 참조는 무시해야 합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2LdapRegister (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapRegisterStruct
{
   char *piNodeName;
   char *piComputer;
   char *piInstance;
   unsigned short                      iNodeType;
   unsigned short                      iOsType;
   db2LdapProtocolInfo iProtocol;
   char *piComment;
   char *piBindDN;
   char *piPassword;
} db2LdapRegisterStruct;

typedef SQL_STRUCTURE db2LdapProtocolInfo
{
   char iType;
   char *piHostName;
   char *piServiceName;
   char *piNetbiosName;
   char *piNetworkId;
   char *piPartnerLU;
   char *piTPName;
```

```
   char *piMode;
   unsigned short                       iSecurityType;
   char *piLanAdapterAddress;
   char *piChangePasswordLU;
   char *piIpxAddress;
} db2LdapProtocolInfo;
```

db2LdapRegister API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParamStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParamStruct

입력. db2LdapRegisterStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LdapRegisterStruct 데이터 구조 매개변수

piNodeName

입력. LDAP의 DB2 서버를 나타내는 단축 이름(8자 미만)을 지정하십시오.

piComputer

입력. DB2 서버가 있는 컴퓨터 이름을 지정하십시오. 컴퓨터 이름 값은 서버 머신을 LDAP에 추가할 때 지정한 값과 동일해야 합니다. Windows 운영 체제에서 이 이름은 Windows 컴퓨터 이름입니다. UNIX 기반 시스템에서 이 이름은 TCP/IP 호스트 이름입니다. 로컬 컴퓨터에 DB2 서버를 등록하려면 NULL을 지정하십시오.

piInstance

입력. DB2 서버의 인스턴스 이름을 지정합니다. 리모트 서버를 등록하도록 컴퓨터 이름이 지정된 경우에는 인스턴스 이름을 지정해야 합니다. 현재 인스턴스(DB2SYSTEM 환경 변수에 정의된 대로)를 등록하려면 NULL을 지정하십시오.

iNodeType

입력. 데이터베이스 서버의 노드 유형을 지정하십시오. 가능한 값은 다음과 같습니다.

- SQLF_NT_SERVER
- SQLF_NT_MPP
- SQLF_NT_DCS

iOsType

입력. 서버 머신의 운영 체제 유형을 지정합니다. 운영 체제 유형을 지정하지 않으면 로컬 서버에는 로컬 운영 체제 유형이 사용되고 리모트 서버에는 운영 체제 유형이 사용되지 않습니다.

iProtocol

입력. db2LdapProtocolInfo 구조에 프로토콜 정보를 지정하십시오.

piComment

입력. DB2 서버를 설명하십시오. 네트워크 디렉토리에 등록된 서버를 설명하는 데 사용되는 주석을 입력할 수 있습니다. 최대 길이는 30자입니다. 캐리지 리턴 또는 줄 바꾸기 문자를 사용할 수 없습니다.

piBindDN

입력. 사용자의 LDAP 식별 이름(DN)을 지정하십시오. LDAP 사용자 DN에는 LDAP 디렉토리에서 오브젝트를 작성하고 갱신할 수 있는 권한이 있어야 합니다. 사용자의 LDAP DN을 지정하지 않으면 현재 로그온 사용자의 증명서가 사용됩니다.

piPassword

입력. 어카운트 암호

db2LdapProtocolInfo 데이터 구조 매개변수

iType 입력. 이 서버에서 지원되는 프로토콜 유형을 지정하십시오. 서버가 둘 이상의 프로토콜을 지원하는 경우에는 다중 등록(각각 다른 노드 및 프로토콜 유형)이 필요합니다. 가능한 값은 다음과 같습니다.

SQL_PROTOCOL_TCPIP
TCP/IPv4 또는 TCP/IPv6 지원용

SQL_PROTOCOL_TCPIP4
TCP/IPv4 지원용

SQL_PROTOCOL_TCPIP6
TCP/IPv6 지원용

SQL_PROTOCOL_SOCKS
보안 SOCKS가 포함된 TCP/IP용

SQL_PROTOCOL_SOCKS4
보안 SOCKS가 포함된 TCP/IPv4용

SQL_PROTOCOL_NPIPE
Windows Named Pipe 지원용

piHostName

입력. TCP/IP 호스트 이름 또는 IP 주소를 지정하십시오. IP 주소는 IPv4 또

는 IPv6 주소일 수 있습니다. IP 주소를 지정하는 경우에는 선택한 프로토콜 유형과 일치해야 합니다. 예를 들어, SQL_PROTOCOL_TCPIP4를 선택하는 경우 지정한 IP 주소는 IPv4 주소여야 합니다.

piServiceName

입력. TCP/IP 서비스 이름 또는 포트 번호를 지정하십시오.

piNetbiosName

입력. NetBIOS 워크스테이션 이름을 지정하십시오. NetBIOS 이름은 NetBIOS 지원을 위해 지정되어야 합니다.

piNetworkID

입력. 네트워크 ID를 지정하십시오. 네트워크 ID는 APPC/APPN 지원을 위해 지정되어야 합니다.

piPartnerLU

입력. DB2 서버 머신의 상대 LU 이름을 지정하십시오. 상대 LU는 APPC/APPN 지원을 위해 지정되어야 합니다.

piTPName

입력. 트랜잭션 프로그램 이름을 지정하십시오. 트랜잭션 프로그램 이름은 APPC/APPN 지원을 위해 지정되어야 합니다.

piMode

입력. 모드 이름을 지정하십시오. 모드는 APPC/APPN 지원을 위해 지정되어야 합니다.

iSecurityType

입력. APPC 보안 레벨을 지정하십시오. 가능한 값은 다음과 같습니다.

- SQL_CPIC_SECURITY_NONE(디폴트)
- SQL_CPIC_SECURITY_SAME
- SQL_CPIC_SECURITY_PROGRAM

piLanAdapterAddress

입력. 네트워크 어댑터 주소를 지정하십시오. 이 매개변수는 APPC 지원용으로만 필요합니다. APPN의 경우 이 매개변수는 NULL로 설정할 수 있습니다.

piChangePasswordLU

입력. 호스트 데이터베이스 서버의 암호를 변경할 때 사용할 상대 LU 이름을 지정하십시오.

piIpxAddress

입력. 완전한 IPX 주소를 지정하십시오. IPX 주소는 IPX/SPX 지원을 위해 지정되어야 합니다.

사용 시 참고사항

고유 노드 이름을 지정할 때마다 서버가 지원하는 각 프로토콜에 대해 DB2 서버를 한 번 등록하십시오.

DB2 서버를 로컬로 등록할 때 프로토콜 구성 매개변수를 지정하는 경우 이는 데이터베이스 관리 프로그램 구성 파일에 지정된 값을 겹쳐씁니다.

리모트 DB2 서버는 LDAP에 등록할 수 있습니다. 리모트 서버의 컴퓨터 이름 및 인스턴스 이름은 리모트 서버의 프로토콜 통신과 함께 지정해야 합니다.

호스트 데이터베이스 서버를 등록할 때 iNodeType 매개변수에 대해 SQLF_NT_DCS 값을 지정해야 합니다.

제 52 장 db2LdapUncatalogDatabase - LDAP 서버에서 데이터베이스 등록 해제

LDAP(Lightweight Directory Access Protocol)에서 데이터베이스 항목을 제거합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2LdapUncatalogDatabase (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapUncatalogDatabaseStruct
{
   char *piAlias;
   char *piBindDN;
   char *piPassword;
} db2LdapUncatalogDatabaseStruct;
```

db2LdapUncatalogDatabase API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParamStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParamStruct

입력. db2LdapUncatalogDatabaseStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LdapUncatalogDatabaseStruct 데이터 구조 매개변수

piAlias

입력. 데이터베이스 항목 별명 이름을 지정하십시오. 이 매개변수는 필수입니다.

piBindDN

입력. 사용자의 LDAP 식별 이름(DN)을 지정하십시오. LDAP 사용자 DN에는 LDAP 디렉토리에서 오브젝트를 삭제하는 데 필요한 권한이 있어야 합니다. 사용자의 LDAP DN을 지정하지 않으면 현재 로그온 사용자의 증명서가 사용됩니다.

piPassword

입력. 어카운트 암호

제 53 장 db2LdapUncatalogNode - LDAP 서버에서 노드 이름의 별명 삭제

LDAP(Lightweight Directory Access Protocol)에서 노드 항목을 제거합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2LdapUncatalogNode (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapUncatalogNodeStruct
{
   char *piAlias;
   char *piBindDN;
   char *piPassword;
} db2LdapUncatalogNodeStruct;
```

db2LdapUncatalogNode API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParamStruct

입력. db2LdapUncatalogNodeStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LdapUncatalogNodeStruct 데이터 구조 매개변수

piAlias

입력. LDAP에 카탈로그 해제하려는 노드의 별명을 지정하십시오.

piBindDN

입력. 사용자의 LDAP 식별 이름(DN)을 지정하십시오. LDAP 사용자 DN에는 LDAP 디렉토리에서 오브젝트를 삭제하는 데 필요한 권한이 있어야 합니다. 사용자의 LDAP DN을 지정하지 않으면 현재 로그온 사용자의 증명서가 사용됩니다.

piPassword

입력. 어카운트 암호

제 54 장 db2LdapUpdate - LDAP 서버에서 DB2 서버 속성 갱신

LDAP(Lightweight Directory Access Protocol)에서 DB2 서버의 통신 프로토콜 정보를 갱신합니다.

주: NetBIOS는 더 이상 지원되지 않습니다. 해당 API인 APPC, APPN 및 CPI-C가 포함된 SNA가 더 이상 지원되지 않습니다. 이 프로토콜을 사용하는 경우에는 TCP/IP와 같이 지원되는 프로토콜을 사용하여 노드와 데이터베이스를 재카탈로그해야 합니다. 이 프로토콜 참조는 무시해야 합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2LdapUpdate (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapUpdateStruct
{
   char *piNodeName;
   char *piComment;
   unsigned short                       iNodeType;
   db2LdapProtocolInfo iProtocol;
   char *piBindDN;
   char *piPassword;
} db2LdapUpdateStruct;

typedef SQL_STRUCTURE db2LdapProtocolInfo
{
   char iType;
   char *piHostName;
   char *piServiceName;
   char *piNetbiosName;
   char *piNetworkId;
   char *piPartnerLU;
   char *piTPName;
   char *piMode;
   unsigned short                       iSecurityType;
```

```
   char *piLanAdapterAddress;
   char *piChangePasswordLU;
   char *piIpxAddress;
} db2LdapProtocolInfo;
```

db2LdapUpdate API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParamStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParamStruct

입력. db2LdapUpdateStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LdapUpdateStruct 데이터 구조 매개변수

piNodeName

입력. LDAP에서 DB2 서버를 나타내는 노드 이름을 지정하십시오.

piComment

입력. DB2 서버의 새 설명을 지정하십시오. 최대 길이는 30자입니다. 캐리지 리턴 또는 줄 바꾸기 문자를 사용할 수 없습니다.

iNodeType

입력. 새 노드 유형을 지정하십시오. 가능한 값은 다음과 같습니다.

- SQLF_NT_SERVER
- SQLF_NT_MPP
- SQLF_NT_DCS
- SQL_PARM_UNCHANGE

iProtocol

입력. db2LdapProtocolInfo 구조의 갱신된 프로토콜 정보를 지정하십시오.

piBindDN

입력. 사용자의 LDAP 식별 이름(DN)을 지정하십시오. LDAP 사용자 DN에는 LDAP 디렉토리에서 오브젝트를 작성하고 갱신할 수 있는 권한이 있어야 합니다. 사용자의 LDAP DN을 지정하지 않으면 현재 로그온 사용자의 증명서가 사용됩니다.

piPassword

입력. 어카운트 암호

db2LdapProtocolInfo 데이터 구조 매개변수

iType 입력. 이 서버에서 지원되는 프로토콜 유형을 지정하십시오. 서버가 둘 이상의

프로토콜을 지원하는 경우에는 다중 등록(각각 다른 노드 및 프로토콜 유형)이 필요합니다. 가능한 값은 다음과 같습니다.

SQL_PROTOCOL_TCPIP
TCP/IPv4 또는 TCP/IPv6 지원용

SQL_PROTOCOL_TCPIP4
TCP/IPv4 지원용

SQL_PROTOCOL_TCPIP6
TCP/IPv6 지원용

SQL_PROTOCOL_SOCKS
보안 SOCKS가 포함된 TCP/IP용

SQL_PROTOCOL_SOCKS4
보안 SOCKS가 포함된 TCP/IPv4용

SQL_PROTOCOL_NPIPE
Windows Named Pipe 지원용

piHostName
입력. TCP/IP 호스트 이름 또는 IP 주소를 지정하십시오. IP 주소는 IPv4 또는 IPv6 주소일 수 있습니다. IP 주소를 지정하는 경우에는 선택한 프로토콜 유형과 일치해야 합니다. 예를 들어, SQL_PROTOCOL_TCPIP4를 선택하는 경우 지정한 IP 주소는 IPv4 주소여야 합니다.

piServiceName
입력. TCP/IP 서비스 이름 또는 포트 번호를 지정하십시오.

piNetbiosName
입력. NetBIOS 워크스테이션 이름을 지정하십시오. NetBIOS 이름은 NetBIOS 지원을 위해 지정되어야 합니다.

piNetworkID
입력. 네트워크 ID를 지정하십시오. 네트워크 ID는 APPC/APPN 지원을 위해 지정되어야 합니다.

piPartnerLU
입력. DB2 서버 머신의 상대 LU 이름을 지정하십시오. 상대 LU는 APPC/APPN 지원을 위해 지정되어야 합니다.

piTPName
입력. 트랜잭션 프로그램 이름을 지정하십시오. 트랜잭션 프로그램 이름은 APPC/APPN 지원을 위해 지정되어야 합니다.

piMode
입력. 모드 이름을 지정하십시오. 모드는 APPC/APPN 지원을 위해 지정되어야 합니다.

iSecurityType

입력. APPC 보안 레벨을 지정하십시오. 가능한 값은 다음과 같습니다.

- SQL_CPIC_SECURITY_NONE(디폴트)
- SQL_CPIC_SECURITY_SAME
- SQL_CPIC_SECURITY_PROGRAM

piLanAdapterAddress

입력. 네트워크 어댑터 주소를 지정하십시오. 이 매개변수는 APPC 지원용으로만 필요합니다. APPN의 경우 이 매개변수는 NULL로 설정할 수 있습니다.

piChangePasswordLU

입력. 호스트 데이터베이스 서버의 암호를 변경할 때 사용할 상대 LU 이름을 지정하십시오.

piIpxAddress

입력. 완전한 IPX 주소를 지정하십시오. IPX 주소는 IPX/SPX 지원을 위해 지정되어야 합니다.

제 55 장 db2LdapUpdateAlternateServerForDB - LDAP 서버에서 데이터베이스의 대체 서버 갱신

LDAP(Lightweight Directory Access Protocol)에서 데이터베이스의 대체 서버를 갱신합니다.

권한 부여

LDAP 서버에 대한 읽기/쓰기 액세스

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2LdapUpdateAlternateServerForDB (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LdapUpdateAltServerStruct
{
   char *piDbAlias;
   char *piNode;
   char *piGWNode;
   char *piBindDN;
   char *piPassword;
} db2LdapUpdateAltServerStruct;
```

db2LdapUpdateAlternateServerForDB API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2LdapUpdateAltServerStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LdapUpdateAltServerStruct 데이터 구조 매개변수

piDbAlias

입력. 갱신되는 데이터베이스 별명이 포함된 문자열

piNode

입력. 대체 노드 이름이 포함된 문자열. 이 노드 이름은 LDAP에 있어야 합니다.

piGWNode

입력. 대체 게이트웨이 노드 이름이 포함된 문자열. 이 노드 이름은 LDAP에 있어야 합니다. IBM Data Server Runtime Client가 이를 사용하여 게이트웨이를 통해 호스트에 연결됩니다.

piBindDN

입력. 사용자의 LDAP 식별 이름(DN)을 지정합니다. 사용자의 LDAP DN에는 LDAP 디렉토리에서 오브젝트를 작성하고 갱신할 권한이 있어야 합니다. 사용자의 LDAP DN을 지정하지 않으면 현재 사용자의 증명서가 사용됩니다.

piPassword

입력. 어카운트 암호

제 56 장 db2Load - 테이블에 데이터 로드

DB2 테이블에 데이터를 로드합니다. 서버에 있는 데이터는 파일, 커서, 테이프 또는 Named Pipe 양식이 될 수 있습니다. 리모트로 연결된 클라이언트에 있는 데이터는 완전한 파일 커서 또는 Named Pipe 양식이 될 수 있습니다. 임포트 유틸리티보다 더 빠른 방법이기는 하지만 로드 유틸리티는 계층 구조 레벨의 데이터 로드와 별칭으로 로드를 지원하지 않습니다.

권한 부여

다음 중 하나가 필요합니다.

- *데이터 액세스*
- 데이터베이스에서 로드 권한 및:
  - 로드 유틸리티가 INSERT 모드, TERMINATE 모드(이전의 로드 삽입 조작을 종료) 또는 RESTART 모드(이전의 로드 삽입 조작을 재시작)로 호출되는 경우 테이블에 대한 INSERT 특권
  - 로드 유틸리티를 REPLACE 모드, TERMINATE 모드(이전의 로드 바꾸기 조작을 종료) 또는 RESTART 모드(이전의 로드 바꾸기 조작을 재시작)로 호출하는 경우 테이블에 대한 INSERT 및 DELETE 특권
  - 이 테이블이 로드 조작의 일부로 사용되는 경우 예외 테이블에 대한 INSERT 특권.

FORCE 옵션을 지정한 경우에는 SYSADM 권한이 필요합니다.

주: 일반적으로 모든 로드 프로세스와 모든 DB2 서버 프로세스는 인스턴스 소유자가 소유합니다. 이 프로세스 전체에서는 인스턴스 소유자의 ID를 사용하여 필요한 파일에 액세스합니다. 따라서 인스턴스 소유자는 명령을 실행한 사용자에 상관 없이 입력 파일에 대해 읽기 액세스가 있어야 합니다.

필수 연결

데이터베이스. 내재된 연결이 사용 가능한 경우 디폴트 데이터베이스에 연결됩니다. Linux, UNIX 또는 Windows 클라이언트에서 Linux, UNIX 또는 Windows 데이터베이스 서버에 대한 유틸리티 액세스는 DB2 Connect 게이트웨이 또는 루프백 환경을 통해서가 아니라 엔진을 통해 직접 연결되어야 합니다.

인스턴스. 명시적 첨부는 필요하지 않습니다. 데이터베이스에 대한 연결이 설정되면 로컬 인스턴스에 대한 내재적 접속이 시도됩니다.

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Load (
    db2Uint32 versionNumber,
    void * pParmStruct,
    struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LoadStruct
{
   struct sqlu_media_list *piSourceList;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piLocalMsgFileName;
   char *piTempFilesPath;
   struct sqlu_media_list *piVendorSortWorkPaths;
   struct sqlu_media_list *piCopyTargetList;
   db2int32 *piNullIndicators;
   struct db2LoadIn *piLoadInfoIn;
   struct db2LoadOut *poLoadInfoOut;
   struct db2PartLoadIn *piPartLoadInfoIn;
   struct db2PartLoadOut *poPartLoadInfoOut;
   db2int16 iCallerAction;
   struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2LoadStruct;

typedef SQL_STRUCTURE db2LoadUserExit
{
   db2Char iSourceUserExitCmd;
   struct db2Char *piInputStream;
   struct db2Char *piInputFileName;
   struct db2Char *piOutputFileName;
   db2Uint16 *piEnableParallelism;
} db2LoadUserExit;

typedef SQL_STRUCTURE db2LoadIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   char *piUseTablespace;
   db2Uint32 iSavecount;
   db2Uint32 iDataBufferSize;
   db2Uint32 iSortBufferSize;
   db2Uint32 iWarningcount;
   db2Uint16 iHoldQuiesce;
   db2Uint16 iCpuParallelism;
   db2Uint16 iDiskParallelism;
   db2Uint16 iNonrecoverable;
   db2Uint16 iIndexingMode;
   db2Uint16 iAccessLevel;
   db2Uint16 iLockWithForce;
   db2Uint16 iCheckPending;
   char iRestartphase;
   char iStatsOpt;
   db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
   db2Uint16 iSetIntegrityPending;
   struct db2LoadUserExit *piSourceUserExit;
} db2LoadIn;
```

```
typedef SQL_STRUCTURE db2LoadOut
{
   db2Uint64 oRowsRead;
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsLoaded;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsDeleted;
   db2Uint64 oRowsCommitted;
} db2LoadOut;

typedef SQL_STRUCTURE db2PartLoadIn
{
   char *piHostname;
   char *piFileTransferCmd;
   char *piPartFileLocation;
   struct db2LoadNodeList *piOutputNodes;
   struct db2LoadNodeList *piPartitioningNodes;
   db2Uint16 *piMode;
   db2Uint16 *piMaxNumPartAgents;
   db2Uint16 *piIsolatePartErrs;
   db2Uint16 *piStatusInterval;
   struct db2LoadPortRange *piPortRange;
   db2Uint16 *piCheckTruncation;
   char *piMapFileInput;
   char *piMapFileOutput;
   db2Uint16 *piTrace;
   db2Uint16 *piNewline;
   char *piDistfile;
   db2Uint16 *piOmitHeader;
   SQL_PDB_NODE_TYPE *piRunStatDBPartNum;
} db2PartLoadIn;

typedef SQL_STRUCTURE db2LoadNodeList
{
   SQL_PDB_NODE_TYPE *piNodeList;
   db2Uint16 iNumNodes;
} db2LoadNodeList;

typedef SQL_STRUCTURE db2LoadPortRange
{
   db2Uint16 iPortMin;
   db2Uint16 iPortMax;
} db2LoadPortRange;

typedef SQL_STRUCTURE db2PartLoadOut
{
   db2Uint64 oRowsRdPartAgents;
   db2Uint64 oRowsRejPartAgents;
   db2Uint64 oRowsPartitioned;
   struct db2LoadAgentInfo *poAgentInfoList;
   db2Uint32 iMaxAgentInfoEntries;
   db2Uint32 oNumAgentInfoEntries;
} db2PartLoadOut;

typedef SQL_STRUCTURE db2LoadAgentInfo
{
   db2int32 oSqlcode;
   db2Uint32 oTableState;
   SQL_PDB_NODE_TYPE oNodeNum;
   db2Uint16 oAgentType;
} db2LoadAgentInfo;

SQL_API_RC SQL_API_FN
  db2gLoad (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gLoadStruct
```

```
{
   struct sqlu_media_list *piSourceList;
   struct sqlu_media_list *piLobPathList;
   struct sqldcol *piDataDescriptor;
   struct sqlchar *piActionString;
   char *piFileType;
   struct sqlchar *piFileTypeMod;
   char *piLocalMsgFileName;
   char *piTempFilesPath;
   struct sqlu_media_list *piVendorSortWorkPaths;
   struct sqlu_media_list *piCopyTargetList;
   db2int32 *piNullIndicators;
   struct db2gLoadIn *piLoadInfoIn;
   struct db2LoadOut *poLoadInfoOut;
   struct db2gPartLoadIn *piPartLoadInfoIn;
   struct db2PartLoadOut *poPartLoadInfoOut;
   db2int16 iCallerAction;
   db2Uint16 iFileTypeLen;
   db2Uint16 iLocalMsgFileLen;
   db2Uint16 iTempFilesPathLen;
   struct sqlu_media_list *piXmlPathList;
   struct sqllob *piLongActionString;
} db2gLoadStruct;

typedef SQL_STRUCTURE db2gLoadIn
{
   db2Uint64 iRowcount;
   db2Uint64 iRestartcount;
   char *piUseTablespace;
   db2Uint32 iSavecount;
   db2Uint32 iDataBufferSize;
   db2Uint32 iSortBufferSize;
   db2Uint32 iWarningcount;
   db2Uint16 iHoldQuiesce;
   db2Uint16 iCpuParallelism;
   db2Uint16 iDiskParallelism;
   db2Uint16 iNonrecoverable;
   db2Uint16 iIndexingMode;
   db2Uint16 iAccessLevel;
   db2Uint16 iLockWithForce;
   db2Uint16 iCheckPending;
   char iRestartphase;
   char iStatsOpt;
   db2Uint16 iUseTablespaceLen;
   db2Uint16 iSetIntegrityPending;
   db2Uint16 *piXmlParse;
   db2DMUXmlValidate *piXmlValidate;
   struct db2LoadUserExit *piSourceUserExit;
} db2gLoadIn;

typedef SQL_STRUCTURE db2gPartLoadIn
{
   char *piHostname;
   char *piFileTransferCmd;
   char *piPartFileLocation;
   struct db2LoadNodeList *piOutputNodes;
   struct db2LoadNodeList *piPartitioningNodes;
   db2Uint16 *piMode;
   db2Uint16 *piMaxNumPartAgents;
   db2Uint16 *piIsolatePartErrs;
   db2Uint16 *piStatusInterval;
   struct db2LoadPortRange *piPortRange;
   db2Uint16 *piCheckTruncation;
   char *piMapFileInput;
   char *piMapFileOutput;
   db2Uint16 *piTrace;
   db2Uint16 *piNewline;
   char *piDistfile;
   db2Uint16 *piOmitHeader;
```

```
   void *piReserved1;
   db2Uint16 iHostnameLen;
   db2Uint16 iFileTransferLen;
   db2Uint16 iPartFileLocLen;
   db2Uint16 iMapFileInputLen;
   db2Uint16 iMapFileOutputLen;
   db2Uint16 iDistfileLen;
} db2gPartLoadIn;


/* Definitions for iUsing value of db2DMUXmlValidate structure               */
#define DB2DMU_XMLVAL_XDS              1         /* Use XDS                   */
#define DB2DMU_XMLVAL_SCHEMA           2         /* Use a specified schema    */
#define DB2DMU_XMLVAL_SCHEMALOC_HINTS  3         /* Use schemaLocation hints  */
#define DB2DMU_XMLVAL_ORIGSCHEMA       4         /* Use schema that document was
                                                    originally validated against
                                                    (load from cursor only) */
```

db2Load API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2LoadStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2LoadStruct 데이터 구조 매개변수

piSourceList

입력. 소스 파일, 디바이스, 벤더, 파이프 또는 SQL문 목록을 제공하는 데 사용된 sqlu_media_list 구조의 포인터.

이 구조에서 제공되는 정보는 media_type 필드 값에 따라 달라집니다. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_SQL_STMT

media_type 필드가 이 값으로 설정된 경우에 호출자는 대상 필드의 pStatement 필드를 사용하여 SQL 쿼리를 제공합니다. pStatement 필드는 sqlu_statement_entry 유형입니다. 로드 유틸리티가 로드당 한 개의 SQL 쿼리만 승인하기 때문에 세션 필드의 값은 1로 설정되어야 합니다.

SQLU_SERVER_LOCATION

media_type 필드가 이 값으로 설정된 경우에 호출자는 sqlu_location_entry 구조를 통해 정보를 제공합니다. 세션 필드는 제공된 sqlu_location_entry 구조 수를 나타냅니다. 이는 파일, 디바이스 및 Named Pipes에 대해 사용됩니다.

SQLU_CLIENT_LOCATION

media_type 필드가 이 값으로 설정된 경우에 호출자는 sqlu_location_entry 구조를 통해 정보를 제공합니다. 세션 필드는 제공된 sqlu_location_entry 구조 수를 나타냅니다. 이는 완전한 파일 및 Named Pipes에 대해 사용됩니다. 이 media_type은 리모트로 연결된 클라이언트를 통해 API가 호출 중인 경우에만 유효합니다.

SQLU_TSM_MEDIA

media_type 필드를 이 값으로 설정하면 파일 이름이 로드되는 데이터의 고유 ID인 경우에 sqlu_vendor 구조가 사용됩니다. 세션 값에 상관 없이 sqlu_vendor 항목은 한 개만 있어야 합니다. 세션 필드는 시작할 TSM 세션 번호를 나타냅니다. 로드 유틸리티는 다른 시퀀스 번호로 세션을 시작하지만 하나의 sqlu_vendor 항목의 동일한 데이터를 사용합니다.

SQLU_OTHER_MEDIA

media_type 필드가 이 값으로 설정되면 sqlu_vendor 구조가 사용되며 이 경우 shr_lib는 공유 라이브러리 이름이고 filename은 로드되는 데이터의 고유 ID입니다. 세션 값에 상관 없이 sqlu_vendor 항목은 한 개만 있어야 합니다. 세션 필드는 시작할 기타 벤더 세션 수입니다. 로드 유틸리티는 다른 시퀀스 번호로 세션을 시작하지만 하나의 sqlu_vendor 항목의 동일한 데이터를 사용합니다.

SQLU_REMOTEFETCH

media_type 필드가 이 값으로 설정된 경우에 호출자는 sqlu_remotefetch_entry 구조를 통해 정보를 제공합니다. 세션 필드의 값은 1로 설정되어야 합니다.

piLobPathList

입력. sqlu_media_list 구조의 포인터. IXF, ASC 및 DEL 파일 유형의 경우 로드하려는 각 LOB 파일의 위치를 식별하는 완전한 경로 또는 디바이스 목록. 파일 이름은 IXF, ASC 또는 DEL 파일에 있으며 제공된 경로에 추가됩니다.

이 구조에서 제공되는 정보는 media_type 필드 값에 따라 달라집니다. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_LOCAL_MEDIA

이 값으로 설정하면 호출자는 sqlu_media_entry 구조를 통해 정보를 제공합니다. 세션 필드는 제공된 sqlu_media_entry 구조 수를 나타냅니다.

SQLU_TSM_MEDIA

이 값으로 설정되면 sqlu_vendor 구조가 사용되며 이 경우 filename은 로드되는 데이터의 고유 ID입니다. 세션 값에 상관 없이 sqlu_vendor 항목은 한 개만 있어야 합니다. 세션 필드는 시작할 TSM 세션 수를

나타냅니다. 로드 유틸리티는 다른 시퀀스 번호로 세션을 시작하지만 하나의 sqlu_vendor 항목의 동일한 데이터를 사용합니다.

SQLU_OTHER_MEDIA

이 값으로 설정되면 sqlu_vendor 구조가 사용되며 이 경우 shr_lib는 공유 라이브러리 이름이고 filename은 로드되는 데이터의 고유 ID입니다. 세션 값에 상관 없이 sqlu_vendor 항목은 한 개만 있어야 합니다. 세션 필드는 시작할 기타 벤더 세션 수입니다. 로드 유틸리티는 다른 시퀀스 번호로 세션을 시작하지만 하나의 sqlu_vendor 항목의 동일한 데이터를 사용합니다.

piDataDescriptor

입력. 외부 파일에서 로드하기 위해 선택한 컬럼에 대한 정보를 포함하는 sqldcol 구조의 포인터.

piFileType 매개변수를 SQL_ASC로 설정한 경우 이 구조의 dcolmeth 필드를 SQL_METH_L로 설정해야 합니다. 사용자는 로드하려는 각 컬럼의 시작 및 종료 위치를 지정합니다.

파일 유형이 SQL_DEL인 경우 dcolmeth는 SQL_METH_P 또는 SQL_METH_D일 수 있습니다. SQL_METH_P인 경우 사용자는 소스 컬럼 위치를 제공해야 합니다. SQL_METH_D인 경우 파일의 첫 번째 컬럼은 테이블의 첫 번째 컬럼에 로드되는 방식입니다.

파일 유형이 SQL_IXF인 경우 dcolmeth는 SQL_METH_P, SQL_METH_D 또는 SQL_METH_N 중 하나일 수 있습니다. DEL 파일의 규칙이 여기에 적용되며 SQL_METH_N이 파일 컬럼 이름이 sqldcol 구조에서 제공되도록 표시된 경우는 제외입니다.

piActionString

더 이상 사용되지 않으며 piLongActionString으로 교체됩니다.

piLongActionString

입력. 테이블에 영향을 주는 조치를 지정하는 문자 배열이 뒤에 오는 4바이트 길이의 필드를 포함하는 *sqllob* 구조의 포인터.

문자 배열은 다음과 같은 양식입니다.

```
"INSERT|REPLACE KEEPDICTIONARY|REPLACE RESETDICTIONARY|RESTART|TERMINATE
INTO tbname [(column_list)]
[FOR EXCEPTION e_tbname]"
```

INSERT

기존 테이블 데이터를 변경하지 않고 로드된 데이터를 테이블에 추가합니다.

REPLACE

테이블에서 기존 데이터를 모두 삭제하고 로드된 데이터를 삽입합니다. 테이블 정의 및 인덱스 정의는 변경되지 않습니다.

RESTART

이전에 인터럽트된 로드 조작을 재시작합니다. 로드 조작은 로드, 빌드 또는 삭제 단계의 마지막 일관성 지점에서 자동으로 계속됩니다.

TERMINATE

일관성 지점이 패스된 경우라도 이전에 인터럽트된 로드 조작을 종료하고 조작이 시작된 특정 시점으로 롤백합니다. 조작에 관계된 테이블 스페이스의 상태가 정상으로 리턴되고 모든 테이블 오브젝트는 일관성을 갖게 됩니다(인덱스 오브젝트가 유효하지 않은 것으로 표시되는 경우, 인덱스 재빌드가 다음 액세스에서 자동으로 발생). 테이블이 있는 테이블 스페이스가 로드 보류 상태가 아닌 경우 이 옵션은 테이블 스페이스 상태에 영향을 주지 않습니다.

로드 종료 옵션은 테이블 스페이스에서 백업 보류 상태를 제거하지 않습니다.

tbname

데이터가 로드되는 테이블 이름. 테이블은 시스템 테이블, 선언된 임시 테이블 또는 작성된 임시 테이블일 수 없습니다. 별명 또는 완전하거나 규정에 맞지 않는 테이블 이름을 지정할 수 있습니다. 규정된 테이블 이름의 양식은 schema.tablename입니다. 규정에 맞지 않는 테이블 이름이 지정되면 테이블은 CURRENT SCHEMA로 규정됩니다.

(column_list)

데이터가 삽입되는 테이블 컬럼 이름 목록. 컬럼 이름은 쉼표로 구분해야 합니다. 이름에 공백이나 소문자가 포함된 경우에는 인용 부호로 묶어야 합니다.

FOR EXCEPTION e_tbname

오류 행이 복사될 예외 테이블을 지정합니다. 예외 테이블을 사용하여 고유 인덱스 규칙, 범위 제한조건 및 보안 규정을 위반하는 행 사본을 저장합니다.

NORANGEEXC

행이 범위 위반으로 거부되는 경우에는 예외 테이블에 삽입되지 않는다는 것을 표시합니다.

NOUNIQUEEXC

행이 고유 제한조건 위반으로 거부되는 경우에는 예외 테이블에 삽입되지 않는다는 것을 표시합니다.

piFileType

입력. 입력 데이터 소스 형식을 나타내는 문자열. 지원되는 외부 형식(sqlutil에 정의)은 다음과 같습니다.

SQL_ASC

컬럼 식별자가 없는 ASCII

SQL_DEL

컬럼 식별자가 있는 ASCII, dBase, BASIC 및 IBM Personal Decision Series 프로그램과 교환에 사용되는 다수의 데이터베이스 관리 프로그램 및 파일 관리자

SQL_IXF

통합 교환 형식의 PC 버전으로 테이블에서 데이터를 익스포트할 때 이후에 동일한 테이블이나 다른 데이터베이스 관리 프로그램 테이블에 로드할 수 있기 때문에 자주 사용되는 방법입니다.

SQL_CURSOR

SQL 쿼리. piSourceList 매개변수로 전달되는 sqlu_media_list 구조는 SQLU_SQL_STMT 또는 SQLU_REMOTEFETCH 유형으로 SQL 쿼리나 테이블 이름입니다.

piFileTypeMod

입력. sqlchar 구조의 포인터로 뒤에 하나 이상의 처리 옵션을 지정하는 문자 배열이 옵니다. 이 포인터가 NULL이거나 지시된 구조에 영(0) 문자가 있는 경우 이 조치는 디폴트 스펙 선택으로 해석됩니다.

지원되는 모든 파일 유형에 모든 옵션을 사용할 수 있는 것은 아닙니다. 관련 링크인 ″로드 유틸리티의 파일 유형 수정자″를 참조하십시오.

piLocalMsgFileName

입력. 출력 메시지가 기록되는 로컬 파일 이름이 포함된 문자열

piTempFilesPath

입력. 임시 파일을 위해 서버에서 사용되는 경로 이름이 포함된 문자열. 임시 파일은 메시지, 일관성 지점을 저장하고 단계 정보를 삭제하기 위해 작성됩니다.

piVendorSortWorkPaths

입력. 벤더 정렬 작업 디렉토리를 지정하는 sqlu_media_list 구조의 포인터

piCopyTargetList

입력. 사본 이미지가 작성되는 경우 이를 위한 목표 경로, 디바이스 또는 공유 라이브러리 목록을 제공하는 데 사용되는 sqlu_media_list 구조의 포인터

이 구조에서 제공되는 값은 media_type 필드 값에 따라 다릅니다. 이 매개변수의 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_LOCAL_MEDIA

사본이 로컬 미디어에 작성되는 경우 media_type을 이 값으로 설정하

고 sqlu_media_entry 구조에 목표에 대한 정보를 제공하십시오. 세션 필드는 제공된 sqlu_media_entry 구조 수를 지정합니다.

SQLU_TSM_MEDIA
사본이 TSM에 작성되는 경우 이 값을 사용하십시오. 추가 정보는 필요하지 않습니다.

SQLU_OTHER_MEDIA
벤더 제품을 사용하려는 경우 이 값을 사용하여 sqlu_vendor 구조를 통해 추가 정보를 제공하십시오. 이 구조의 shr_lib 필드를 벤더 제품의 공유 라이브러리로 설정하십시오. 세션 값에 상관 없이 한 개의 sqlu_vendor 항목만 제공하십시오. 세션 필드는 제공된 sqlu_media_entry 구조 수를 지정합니다. 로드 유틸리티는 다른 시퀀스 번호로 세션을 시작하지만 하나의 sqlu_vendor 항목에서 제공된 동일한 데이터를 사용합니다.

piNullIndicators
입력. ASC 파일에만 해당. 컬럼 데이터가 널(NULL) 입력 가능한지 여부를 표시하는 정수 배열. 이 배열 요소와 데이터 파일에서 로드 중인 컬럼은 1대1 대응 방식입니다. 즉, 요소 수는 piDataDescriptor 매개변수의 dcolnum 필드와 동일해야 합니다. 배열의 각 요소에는 널(NULL) 표시기 필드로 사용되는 데이터 파일의 위치를 식별하는 번호 또는 테이블 컬럼이 널(NULL) 입력 가능한 것을 표시하는 영(0)이 포함됩니다. 요소가 영(0)이 아닌 경우 데이터 파일의 식별된 위치에는 Y 또는 N이 포함됩니다. Y는 테이블 컬럼 데이터가 NULL임을 표시하고 N은 테이블 컬럼 데이터가 NULL이 아닌 것을 표시합니다.

piLoadInfoIn
입력. db2LoadIn 구조의 포인터

poLoadInfoOut
출력. db2LoadOut 구조의 포인터

piPartLoadInfoIn
입력. db2PartLoadIn 구조의 포인터

poPartLoadInfoOut
출력. db2PartLoadOut 구조의 포인터

iCallerAction
입력. 호출자가 요청한 조치. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_INITIAL
초기 호출. 이 값(또는
SQLU_NOINTERRUPT)은 API의 첫 번째 호출에 사용되어야 합니다.

SQLU_NOINTERRUPT

초기 호출. 처리를 일시중단하지 않습니다. 이 값(또는 SQLU_INITIAL)은 API의 첫 번째 호출에 사용되어야 합니다.

초기 호출이나 이후에 임의의 후속 호출이 리턴되고 요청된 로드 조작을 완료하기 전에 일부 조치를 수행하는 응용프로그램을 호출해야 하는 경우 호출자 조치는 다음 중 하나로 설정해야 합니다.

SQLU_CONTINUE

처리를 계속합니다. 이 값은 초기 호출에서 사용자 입력을 요청(예: 테이프 끝에 응답 조건)하는 유틸리티를 리턴한 후에 API의 후속 호출에만 사용할 수 있습니다. 유틸리티로 요청된 사용자 조치가 완료되고 유틸리티는 초기 요청 처리를 계속하도록 지정합니다.

SQLU_TERMINATE

처리를 종료합니다. 로드 유틸리티가 너무 빨리 종료되어 로드 중인 테이블 스페이스는 LOAD_PENDING 상태가 됩니다. 이 옵션은 데이터의 추가 처리가 완료되지 않아야 하는 경우에 지정해야 합니다.

SQLU_ABORT

처리를 종료합니다. 로드 유틸리티가 너무 빨리 종료되어 로드 중인 테이블 스페이스는 LOAD_PENDING 상태가 됩니다. 이 옵션은 데이터의 추가 처리가 완료되지 않아야 하는 경우에 지정해야 합니다.

SQLU_RESTART

처리를 재시작합니다.

SQLU_DEVICE_TERMINATE

디바이스 하나를 종료합니다. 유틸리티가 디바이스에서 데이터 읽기를 중단하지만 데이터의 추가 처리는 완료해야 하는 경우에 이 옵션을 지정해야 합니다.

piXmlPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 xml 파일이 있는 클라이언트 경로를 나열하는 sqlu_media_list의 포인터

db2LoadUserExit 데이터 구조 매개변수

iSourceUserExitCmd

입력. 데이터를 유틸리티에 공급하는 데 사용되는 실행 파일의 완전한 이름. 보안상의 이유로 실행 파일은 서버의 `sqllib/bin` 디렉토리에 있어야 합니다. 이 매개변수는 piSourceUserExit 구조가 NULL이 아닌 경우 필수입니다.

piInputStream, piInputFileName, piOutputFileName 및 piEnableParallelism 필드는 선택적입니다.

piInputStream

입력. STDIN을 통해 userexit 응용프로그램에 직접 전달되는 일반 바이트스트림. 이 바이트스트림에 포함되는 데이터와 해당 형식을 완벽하게 제어해야 합니다. 로드 유틸리티는 서버에서 이 바이트스트림을 단순하게 처리하고 프로세스의 STDIN을 공급하여 userexit 응용프로그램에 전달합니다(코드 페이지를 변환하거나 바이트스트림을 수정하지는 않음). userexit 응용프로그램은 STDIN에서 인수를 읽고 데이터를 의도한 대로 사용합니다.

이 기능의 중요한 한 가지 속성은 민감한 정보(사용자 ID/암호)를 숨기는 기능입니다.

piInputFileName

입력. 프로세스의 STDIN을 공급하여 컨텐츠가 userexit 응용프로그램에 전달되는 완전한 클라이언트 측 파일의 이름이 포함됩니다.

piOutputFileName

입력. 서버 측 파일의 완전한 이름. userexit 응용프로그램을 실행하는 프로세스의 STDOUT 및 STDERR 스트림이 이 파일로 스트림됩니다. piEnableParallelism이 참인 경우 다중 파일이 작성되고(userexit 인스턴스당 1개) 각 파일 이름에는 3자리의 숫자 노드 번호 값이 추가됩니다(예: `<filename>.000`).

piEnableParallelism

입력. 유틸리티가 userexit 응용프로그램 호출을 병렬 처리하도록 표시하는 플래그.

db2LoadIn 데이터 구조 매개변수

iRowcount

입력. 로드할 실제 레코드 수. 사용자가 파일의 첫 번째 rowcnt 행만 로드하도록 허용합니다.

iRestartcount

입력. 나중에 사용하기 위해 예약됨

piUseTablespace

입력. 인덱스가 재빌드되는 경우, 인덱스의 쉐도우 사본이 테이블 스페이스 iUseTablespaceName에 빌드되고 로드 종료 시에 원래 테이블 스페이스에 복사됩니다. 이 옵션에는 시스템 임시 테이블 스페이스만 사용할 수 있습니다. 테이블 스페이스를 지정하지 않을 경우 음영 인덱스는 인덱스 오브젝트와 동일한 테이블 스페이스에 작성됩니다.

음영 사본이 인덱스 오브젝트와 동일한 테이블 스페이스에 작성된 경우, 이전 인덱스 오브젝트에 대한 음영 인덱스 오브젝트의 복사는 순간적입니다. 음영 사본이 인덱스 오브젝트와 다른 테이블 스페이스에 있을 경우 실제 복사가 수행

됩니다. 상당한 I/O 및 시간이 포함될 수 있습니다. 복사는 테이블이 로드 종료 시에 오프라인 상태에서 수행됩니다.

이 필드는 iAccessLevel이 SQLU_ALLOW_NO_ACCESS인 경우 무시됩니다.

사용자가 INDEXING MODE REBUILD 또는 INDEXING MODE AUTOSELECT를 지정하지 않을 경우 이 옵션은 무시됩니다. IDEXING MODE AUTOSELECT가 선택되고 로드가 인덱스를 점차적으로 갱신하도록 선택할 경우에도 이 옵션이 무시됩니다.

iSavecount

일관성 지점을 작성하기 전에 로드되는 레코드 수. 이 값은 쪽 수로 변환되며 Extent *크기*의 간격으로 반올림됩니다. 메시지는 각각의 일관성 지점에서 발행되므로 db2LoadQuery - 로드 쿼리를 사용하여 로드 조작을 모니터하는 경우 이 옵션을 선택해야 합니다. savecount 값이 충분히 높지 않으면 각 일관성 지점에서 수행된 활동의 동기화가 성능에 영향을 줍니다.

디폴트값은 0으로, 필요하지 않으면 일관성 지점을 설정하지 않음을 의미합니다.

iDataBufferSize

유틸리티에서 데이터 전송에 필요한 버퍼 스페이스로 사용할 4KB 페이지의 수 (병렬 처리 수준과 무관함). 지정된 값이 알고리즘의 최소 값보다 작으면, 필요한 최소값이 사용되고 경고는 리턴되지 않습니다.

이 메모리는 유틸리티 힙에서 바로 할당되며 *크기*는 util_heap_sz 데이터베이스 구성 매개변수로 수정할 수 있습니다.

값이 지정되지 않으면 런타임시 유틸리티에 의해 적절한 디폴트값이 계산됩니다. 디폴트값은 테이블의 일부 등록 정보 뿐 아니라 로더의 인스턴스화 시간에 유틸리티 힙에서 사용 가능한 여유 공간의 백분율을 기초로 합니다.

iSortBufferSize

입력. 이 옵션은 로드 조작 중 SORTHEAP 데이터베이스 구성 매개변수를 겹쳐쓰는 값을 지정합니다. 이 옵션은 인덱스와 함께 테이블을 로드하는 경우와 iIndexingMode 매개변수가 SQLU_INX_DEFERRED로 지정되지 않은 경우에만 의미가 있습니다. 지정된 값은 SORTHEAP의 값을 초과할 수 없습니다. 이 매개변수는 SORTHEAP의 값을 변경하지 않은 채로 LOAD에서 사용되는 정렬 메모리를 조절하는 데 유용하며 일반 쿼리 처리에도 영향을 줍니다.

iWarningcount

입력. warningcnt 경고 후에 로드 조작을 중지합니다. 경고가 예상되지는 않지만 올바른 파일 및 테이블이 사용되고 있는지에 대한 검증을 원할 경우 이 매개변수를 설정하십시오. 로드 파일이나 목표 테이블이 잘못 지정된 경우, 로드

유틸리티가 로드를 시도하는 각 행에 대해 경고를 생성하므로 로드에 실패하게 됩니다. warningcnt가 0이거나 이 옵션을 지정하지 않으면 로드 조작이 발생된 경고 수에 상관 없이 계속됩니다.

경고 임계값이 초과되어 로드 조작이 중지된 경우 다른 로드 조작은 RESTART 모드로 시작할 수 있습니다. 로드 조작은 마지막 일관성 지점에서 자동으로 계속됩니다. 또는 다른 로드 조작을 REPLACE 모드로 초기화하여 입력 파일의 처음부터 시작하게 할 수 있습니다.

iHoldQuiesce

입력. 유틸리티가 로드 후에 테이블을 Quiesce 독점 상태로 유지하는 경우 값이 참으로 설정되고 이 외의 경우에는 값이 거짓으로 설정되는 플래그.

iCpuParallelism

입력. 테이블 오브젝트를 빌드할 때 레코드를 구문 분석, 변환 및 포맷팅하기 위해 로드 유틸리티가 작성하는 프로세스 또는 스레드의 수. 이 매개변수는 파티션 내 병렬 처리를 이용하도록 설계되었습니다. 소스 데이터에 있는 기록 순서가 보존되므로 사전에 정렬된 데이터를 로드할 때 특히 유용합니다. 이 매개변수의 값이 0인 경우 로드 유틸리티는 런타임에서 인텔리전트한 디폴트값을 사용합니다. 참고: 이 매개변수는 LOB 또는 LONG VARCHAR 필드 중 하나가 있는 테이블과 함께 사용되면 시스템 CPU 수나 사용자가 지정한 값과 관계없이 값은 1이 됩니다.

iDiskParallelism

입력. 로드 유틸리티가 데이터를 테이블 스페이스 컨테이너에 기록하기 위해 작성하는 프로세스나 스레드 수. 값이 지정되지 않으면 유틸리티가 테이블 스페이스 컨테이너의 수 및 테이블의 등록 정보를 기반으로 하여 적절한 디폴트값을 선택합니다.

iNonrecoverable

입력. 로드 트랜잭션이 복구 불가능으로 표시되어 후속 롤 포워드 조치로 복구할 수 없는 경우에는 SQLU_NON_RECOVERABLE_LOAD로 설정하십시오. 롤 포워드 유틸리티가 트랜잭션을 건너뛰고 데이터가 로드 중인 테이블을 ″유효하지 않음″으로 표시합니다. 또한 이 유틸리티는 해당 테이블에 대한 후속 트랜잭션을 무시합니다. 롤 포워드가 완료되면 이러한 테이블은 삭제할 수만 있습니다. 이 옵션을 사용하면 로드 조작 후에 테이블 스페이스가 백업 보류 상태가 되지 않으며 로드 조작 중에 로드된 데이터의 사본을 작성할 필요가 없습니다. 로드 트랜잭션이 복구 가능으로 표시된 경우 SQLU_RECOVERABLE_LOAD로 설정하십시오.

iIndexingMode

입력. 인덱스 모드를 지정합니다. 유효한 값(include 디렉토리에 있는 sqlutil 헤더 파일에 정의)은 다음과 같습니다.

SQLU_INX_AUTOSELECT
LOAD는 REBUILD 및 INCREMENTAL 인덱스 모드에서 선택합니다.

SQLU_INX_REBUILD
테이블 인덱스를 재빌드합니다.

SQLU_INX_INCREMENTAL
기존 인덱스를 확장합니다.

SQLU_INX_DEFERRED
테이블 인덱스를 갱신하지 않습니다.

iAccessLevel
입력. 액세스 등급을 지정합니다. 가능한 값은 다음과 같습니다.

SQLU_ALLOW_NO_ACCESS
로드가 테이블을 배타적으로 잠그도록 지정합니다.

SQLU_ALLOW_READ_ACCESS
테이블의 원래 데이터(델타가 아닌 부분)가 로드 진행 중에도 판독기에 표시되도록 지정합니다. 이 옵션은 로드 삽입과 같은 로드 추가에만 유효하며 로드 교체에 대해서는 무시됩니다.

iLockWithForce
입력. 부울 플래그. 참으로 설정되면 로드는 테이블 잠금을 즉시 확보하도록 다른 응용프로그램을 강제 실행합니다. 이 옵션을 사용하려면 FORCE APPLICATIONS 명령(SYSADM 또는 SYSCTRL)과 동일한 권한이 필요합니다.

SQLU_ALLOW_NO_ACCESS 로드는 로드 조작 시작 시에 충돌하는 응용프로그램을 강제 실행할 수도 있습니다. 로드 시작 시에 유틸리티는 테이블을 쿼리하거나 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다.

SQLU_ALLOW_READ_ACCESS 로드는 로드 조작 시작 또는 종료 시에 충돌하는 응용프로그램을 강제 실행할 수도 있습니다. 로드 시작 시에 로드 유틸리티는 테이블을 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다. 로드 종료 시에 로드 유틸리티는 테이블을 쿼리하거나 수정하려고 시도 중인 응용프로그램을 강제 실행할 수 있습니다.

iCheckPending
이 매개변수는 버전 9.1부터 더 이상 사용되지 않습니다. iSetIntegrityPending 매개변수를 대신 사용하십시오.

iRestartphase
입력. 예약됨. 유효한 값은 단일 공백 문자 ' '입니다.

iStatsOpt

입력. 수집할 통계의 세분화도. 가능한 값은 다음과 같습니다.

SQLU_STATS_NONE

수집할 통계가 없습니다.

SQLU_STATS_USE_PROFILE

현재 테이블에 대해 정의된 프로파일을 사용하여 통계가 수집됩니다. 이 프로파일은 RUNSTATS 명령을 사용하여 작성해야 합니다. 현재 테이블에 대한 프로파일이 없는 경우 경고가 리턴되고 통계가 수집되지 않습니다.

iSetIntegrityPending

입력. 테이블을 무결성 설정 보류 상태로 지정합니다.

SQLU_SI_PENDING_CASCADE_IMMEDIATE 값이 지정된 경우 무결성 설정 보류 상태가 즉시 모든 종속 및 하위 테이블에 중첩됩니다. SQLU_SI_PENDING_CASCADE_DEFERRED 값이 지정된 경우 종속 테이블의 무결성 설정 보류 상태 중첩이 목표 테이블의 무결성 위반이 점검될 때까지 지연됩니다. SQLU_SI_PENDING_CASCADE_DEFERRED는 옵션을 지정하지 않는 경우 디폴트값입니다.

piSourceUserExit

입력. db2LoadUserExit 구조의 포인터

piXmlParse

입력. XML 문서에 대해 수행해야 하는 구문 분석 유형. include 디렉토리의 db2ApiDf 헤더 파일에 있는 유효한 값은 다음과 같습니다.

DB2DMU_XMLPARSE_PRESERVE_WS

공백은 유지해야 합니다.

DB2DMU_XMLPARSE_STRIP_WS

공백은 스트립되어야 합니다.

piXmlValidate

입력. db2DMUXmlValidate 구조의 포인터. XML 스키마 유효성 확인이 XML 문서에 대해 수행되어야 하는 것을 나타냅니다.

```
/* XML Validate structure                                              */
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                       iUsing;      /* What to use to perform   */
                                                /* validation               */
   struct db2DMUXmlValidateXds    *piXdsArgs;   /* Arguments for            */
                                                /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Arguments for           */
                                                /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

db2LoadOut 데이터 구조 매개변수

oRowsRead

출력. 로드 조작에서 읽은 레코드 수

oRowsSkipped

출력. 로드 조작 시작 전에 건너뛴 레코드 수

oRowsLoaded

출력. 목표 테이블로 로드된 행 수

oRowsRejected

출력. 로드할 수 없던 레코드 수

oRowsDeleted

출력. 삭제된 중복 행 수

oRowsCommitted

출력. 처리된 총 레코드 수 즉, 데이터베이스에 성공적으로 로드되어 커미트된 레코드 수 더하기 건너뛰고 거부된 레코드 수

db2PartLoadIn 데이터 구조 매개변수

piHostname

입력. iFileTransferCmd 매개변수의 호스트 이름. NULL인 경우 호스트 이름의 디폴트는 ″nohost″입니다. 이 매개변수는 더 이상 사용되지 않습니다.

piFileTransferCmd

입력. 파일 전송 명령 매개변수. 필요하지 않는 경우 NULL로 설정해야 합니다. 이 매개변수는 더 이상 사용되지 않습니다. piSourceUserExit 매개변수를 대신 사용하십시오.

piPartFileLocation

입력. PARTITION_ONLY, LOAD_ONLY, LOAD_ONLY_VERIFY_PART 모드에서 이 매개변수를 사용하여 파티션된 파일의 위치를 지정할 수 있습니다. 이 위치는 piOutputNodes 옵션으로 지정한 각 데이터베이스 파티션에 있어야 합니다.

SQL_CURSOR 파일 유형의 경우 이 매개변수는 NULL일 수 없으며 위치는 경로가 아니라 완전한 파일 이름입니다. PARTITION_ONLY 모드에서 각 출력 데이터베이스 파티션에서 작성된 파티션된 파일의 완전한 기본 파일 이름이거나 LOAD_ONLY 모드에서 각 데이터베이스 파티션에서 읽은 파일의 위치입니다. PARTITION_ONLY 모드의 경우 목표 테이블에 LOB 컬럼이 있는 경우 다중 파일이 지정한 기본 이름으로 작성될 수도 있습니다. SQL_CURSOR 이외의 파일 유형의 경우 이 매개변수 값이 NULL인 경우 현재 디렉토리가 디폴트입니다.

piOutputNodes

입력. 로드 출력 데이터베이스 파티션 목록. NULL은 목표 테이블의 모든 노드가 정의되는 것을 나타냅니다.

piPartitioningNodes

입력. 파티셔닝 노드의 목록. NULL은 디폴트를 표시합니다.

piMode

입력. 파티션된 데이터베이스의 로드 모드를 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

- DB2LOAD_PARTITION_AND_LOAD

데이터가 분산(병렬 가능)되고 해당 데이터베이스 파티션에서 동시에 로드됩니다.

- DB2LOAD_PARTITION_ONLY

데이터가 분산(병렬 가능)되고 출력은 각 로딩 데이터베이스 파티션에 지정한 위치에 있는 파일에 기록됩니다. SQL_CURSOR 이외의 파일 유형의 경우 각 데이터베이스 파티션의 출력 파일 이름은 filename.xxx 이며 여기서 filename은 piSourceList로 지정한 첫 번째 입력 파일 이름이며 xxx는 데이터베이스 파티션 번호입니다. SQL_CURSOR 파일 유형의 경우 각 데이터베이스 파티션에 있는 출력 파일은 piPartFileLocation 매개변수로 판별합니다. 각 데이터베이스 파티션에서 데이터베이스 파티션 위치를 지정하는 방법에 대한 정보는 piPartFileLocation 매개변수를 참조하십시오.

주: CLI LOAD에는 이 모드를 사용할 수 없습니다.

DB2LOAD_LOAD_ONLY

데이터가 이미 분산된 것으로 간주됩니다. 분산 프로세스는 건너뛰고 데이터가 해당 데이터베이스 파티션에서 동시에 로드됩니다. SQL_CURSOR 이외의 파일 유형의 경우 각 데이터베이스 파티션의 입력 파일 이름은 filename.xxx 양식이어야 하며 여기서 filename은 piSourceList로 지정한 첫 번째 파일 이름이고 xxx는 13자리의 데이터베이스 파티션 번호입니다. SQL_CURSOR 파일 유형의 경우 각 데이터베이스 파티션에 있는 입력 파일은 piPartFileLocation 매개변수로 판별합니다. 각 데이터베이스 파티션에서 데이터베이스 파티션 위치를 지정하는 방법에 대한 정보는 piPartFileLocation 매개변수를 참조하십시오.

주: 이 모드는 리모트 클라이언트에 있는 데이터 파일 로드 시 또는 CLI LOAD에는 사용할 수 없습니다.

DB2LOAD_LOAD_ONLY_VERIFY_PART

데이터가 이미 분산된 것으로 간주되지만 데이터 파일에는 데이터베이

스 파티션 헤더가 포함되지 않습니다. 분산 프로세스는 건너뛰고 데이터가 해당 데이터베이스 파티션에서 동시에 로드됩니다. 로드 조작 중에 각 행은 올바른 데이터베이스 파티션에 있는지 확인하기 위해 점검됩니다. 데이터베이스 파티션 위반을 포함하는 행은 덤프 파일 파일 유형 수정자가 지정된 경우 덤프 파일에 배치됩니다. 그 이외의 경우에는 행이 버려집니다. 데이터베이스 파티션 위반이 특정 로드 데이터베이스 파티션에 있는 경우 해당 데이터베이스 파티션에 대한 로드 메시지에 한 개의 경고가 기록됩니다. 각 데이터베이스 파티션의 입력 파일 이름은 filename.xxx 양식이어야 하며 여기서 filename은 piSourceList로 지정한 첫 번째 파일 이름이고 xxx는 13자리의 데이터베이스 파티션 번호입니다.

주: 이 모드는 리모트 클라이언트에 있는 데이터 파일 로드 시 또는 CLI LOAD에는 사용할 수 없습니다.

DB2LOAD_ANALYZE

모든 데이터베이스 파티션의 분산에서도 최적의 분산 맵이 생성됩니다.

piMaxNumPartAgents

입력. 초대 파티셔닝 에이전트 수. NULL 값은 25인 디폴트입니다.

piIsolatePartErrs

입력. 로드 조작이 각 데이터베이스 파티션에서 발생한 오류에 대응하는 방법을 나타냅니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOAD_SETUP_ERRS_ONLY

이 모드에서 데이터베이스 파티션 액세스 문제 또는 테이블 스페이스 또는 데이터베이스 파티션의 테이블 액세스 문제와 같이 설정 중에 데이터베이스 파티션에서 발생하는 오류로 인해 로드 조작이 실패한 데이터베이스 파티션에서 중지되지만 나머지 데이터베이스 파티션에서는 계속됩니다. 데이터가 로드되는 중에 데이터베이스 파티션에서 발생한 오류로 인해 전체 조작이 실패하고 각 데이터베이스 파티션의 마지막 일관성 시점으로 롤백됩니다.

DB2LOAD_LOAD_ERRS_ONLY

이 모드에서는 설정 중에 데이터베이스 파티션에서 발생한 오류로 인해 전체 로드 조작이 실패합니다. 데이터 로드 중에 오류가 발생하면 오류가 발생한 데이터베이스 파티션이 최종 일관성 지점으로 롤백됩니다. 실패가 발생하거나 모든 데이터가 로드될 때까지 나머지 데이터베이스 파티션에서 로드 조작이 계속됩니다. 모든 데이터가 로드된 데이터베이스 파티션에서 로드 조작 후에 데이터가 표시되지 않습니다. 기타 데이터베이스 파티션의 오류로 인해 트랜잭션이 중단됩니다. 모든 데이터베이스 파티션의 데이터는 로드 시작 조작이 수행될 때까지 표시되

지 않습니다. 이로 인해 새로 로드된 데이터는 로드 조작이 완료되는 데이터베이스 파티션에 표시되고 오류가 발생한 데이터베이스 파티션에서 로드 조작이 다시 시작됩니다.

주: 이 모드는 iAccessLevel이 SQLU_ALLOW_READ_ACCESS로 설정되고 사본 대상도 지정된 경우에는 사용할 수 없습니다.

DB2LOAD_SETUP_AND_LOAD_ERRS

이 모드에서는 설정이나 데이터 로드 중에 발생한 데이터베이스 파티션 오류로 인해 영향 받는 데이터베이스 파티션에서만 처리가 중지됩니다. DB2LOAD_LOAD_ERRS_ONLY 모드를 사용하면 데이터 로드 중에 데이터베이스 파티션 오류가 발생하지 않은 경우 모든 데이터베이스 파티션의 데이터는 로드 시작 조작이 수행될 때까지 표시되지 않습니다.

주: 이 모드는 iAccessLevel이 SQLU_ALLOW_READ_ACCESS로 설정되고 사본 대상도 지정된 경우에는 사용할 수 없습니다.

DB2LOAD_NO_ISOLATION

로드 조작 중의 오류로 인해 트랜잭션이 중단됩니다. 이 매개변수가 NULL인 경우 iAccessLevel이 SQLU_ALLOW_READ_ACCESS로 설정되고 사본 대상도 지정된 경우에는 디폴트가 DB2LOAD_NO_ISOLATION인 경우 DB2LOAD_LOAD_ERRS_ONLY를 디폴트로 사용합니다.

piStatusInterval

입력. 진행 상황 메시지를 생성하기 전에 로드할 데이터 수(메가바이트(MB))를 지정합니다. 유효한 값은 1 - 4000 사의의 정수입니다. NULL을 지정하면 디폴트값인 100이 사용됩니다.

piPortRange

입력. 내부 통신을 위한 TCP 포트. NULL인 경우 사용되는 포트 범위는 6000 - 6063입니다.

piCheckTruncation

입력. 로드가 입/출력(I/O) 시에 레코드 절단을 점검합니다. 유효한 값은 참 및 거짓입니다. NULL인 경우 디폴트는 거짓입니다.

piMapFileInput

입력. 분산 맵 입력 파일 이름. 모드가 ANALYZE가 아닌 경우 이 매개변수는 NULL로 설정해야 합니다. 모드가 ANALYZE인 경우 이 매개변수를 지정해야 합니다.

piMapFileOutput

입력. 분산 맵 출력 파일 이름. piMapFileInput의 규칙이 여기에도 적용됩니다.

piTrace

입력. 모든 데이터 변환 프로세스 및 해시 값 출력의 덤프를 검토해야 하는 경우 추적할 레코드 수를 지정합니다. NULL인 경우 레코드 수의 디폴트는 0입니다.

piNewline

입력. RECLEN 파일 유형 수정자도 지정된 경우 ASC 데이터 레코드 끝에서 로드가 개행 문자를 강제로 점검하도록 합니다. 가능한 값은 참 및 거짓입니다. NULL인 경우 디폴트는 거짓입니다.

piDistfile

입력. 데이터베이스 파티션 분산 파일 이름. NULL을 지정하면 디폴트 값은 ″DISTFILE″입니다.

piOmitHeader

입력. 분산 맵 헤더가 DB2LOAD_PARTITION_ONLY 모드 사용 시에 데이터베이스 파티션 파일에 포함되지 않는 것을 나타냅니다. 가능한 값은 참 및 거짓입니다. NULL인 경우 디폴트는 거짓입니다.

piRunStatDBPartNum

통계를 수집하는 데이터베이스 파티션을 지정합니다. 디폴트값은 출력 데이터베이스 파티션 목록의 첫 번째 데이터베이스 파티션입니다.

db2LoadNodeList 데이터 구조 매개변수

piNodeList

입력. 노드 번호 배열.

iNumNodes

입력. piNodeList 배열의 노드 수. 0이 디폴트이며 목표 테이블이 정의된 모든 노드입니다.

db2LoadPortRange 데이터 구조 매개변수

iPortMin

입력. 낮은 포트 번호

iPortMax

입력. 높은 포트 번호

db2PartLoadOut 데이터 구조 매개변수

oRowsRdPartAgents

출력. 모든 파티셔닝 에이전트가 읽는 총 행 수

oRowsRejPartAgents

출력. 모든 파티셔닝 에이전트가 거부하는 총 행 수

oRowsPartitioned

출력. 모든 파티셔닝 에이전트로 파티션된 총 행 수

poAgentInfoList

출력. 파티션된 데이터베이스의 로드 조작 중에 다음 로드 처리 엔티티가 사용될 수 있습니다. 로드 에이전트, 파티셔닝 에이전트, 프리파티셔닝 에이전트, 파일 전송 명령 에이전트 및 파일에 로드 에이전트(이들에 대해서는 데이터 이동 안내서에서 설명됨). poAgentInfoList 출력 매개변수는 로드 조작에 사용되는 각 로드 에이전트에 대한 호출자 정보를 리턴하는 데 사용됩니다. 목록의 각 항목에는 다음 정보가 포함됩니다.

oAgentType

항목이 설명하는 로드 에이전트 종류를 나타내는 태그

oNodeNum

에이전트가 실행된 데이터베이스 파티션 번호

oSqlcode

에이전트 처리 결과로 작성된 최종 sqlcode

oTableState

에이전트가 실행된 데이터베이스 파티션에서 테이블의 최종 상태(로드 에이전트에만 관련).

API 호출 전에 이 목록에 대해 API 호출자가 메모리를 할당해야 합니다. 호출자는 iMaxAgentInfoEntries 매개변수에 메모리를 할당한 항목 수도 표시해야 합니다. 호출자가 poAgentInfoList를 NULL로 설정하거나 iMaxAgentInfoEntries를 0으로 설정한 경우 로드 에이전트에 대한 정보가 리턴되지 않습니다.

iMaxAgentInfoEntries

입력. 사용자가 poAgentInfoList에 대해 할당한 에이전트 정보 항목의 최대 수. 일반적으로 이 매개변수는 로드 조작에 사용된 데이터베이스 파티션 수의 3배로 설정하면 충분합니다.

oNumAgentInfoEntries

출력. 로드 조작으로 작성된 에이전트 정보 항목의 실제 수. 이 항목 수는 iMaxAgentInfoEntries가 oNumAgentInfoEntries 이상인 경우 poAgentInfoList 매개변수로 사용자에게 리턴됩니다. iMaxAgentInfoEntries가 oNumAgentInfoEntries보다 작은 경우 poAgentInfoList로 리턴된 항목 수는 iMaxAgentInfoEntries와 같습니다.

db2LoadAgentInfo 데이터 구조 매개변수

oSqlcode

출력. 에이전트 처리 결과로 작성된 최종 sqlcode

oTableState

출력. 이 출력 매개변수는 로드 조작 후에 테이블의 가능한 모든 상태를 보고하는 데 사용되지는 않습니다. 오히려 호출자에게 로드 처리 중에 발생하는 일반적인 정보를 제공하기 위해 가능한 테이블 상태의 일부 서브세트만을 보고합니다. 이 값은 로드 에이전트에만 연관됩니다. 가능한 값은 다음과 같습니다.

DB2LOADQUERY_NORMAL

데이터베이스 파티션에서 로드가 완료되었으며 테이블이 LOAD IN PROGRESS(또는 LOAD PENDING) 상태를 벗어났음을 나타냅니다. 이 경우 테이블은 이후 제한조건 처리를 위해 SET INTEGRITY PENDING 상태일 수는 있지만 이는 정상적이기 때문에 보고되지는 않습니다.

DB2LOADQUERY_UNCHANGED

오류로 인해 로드 작업이 처리를 중단했지만 db2Load 호출 이전의 상태에 관계 없이 데이터베이스 파티션의 테이블 상태가 아직 변경되지 않았음을 나타냅니다. 이런 데이터베이스 파티션에서 로드 재시작을 수행하거나 조작을 종료할 필요는 없습니다.

DB2LOADQUERY_LOADPENDING

처리 중에 로드 작업이 중단되었지만 데이터베이스 파티션의 테이블이 LOAD PENDING 상태로 해당 데이터베이스 파티션의 로드 작업은 종료 또는 재시작해야 함을 나타냅니다.

oNodeNum

출력. 에이전트가 실행된 데이터베이스 파티션 번호

oAgentType

출력. 에이전트 유형. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

- DB2LOAD_LOAD_AGENT
- DB2LOAD_PARTITIONING_AGENT
- DB2LOAD_PRE_PARTITIONING_AGENT
- DB2LOAD_FILE_TRANSFER_AGENT
- DB2LOAD_LOAD_TO_FILE_AGENT

db2gLoadStruct 데이터 구조 특정 매개변수

iFileTypeLen

입력. iFileType 매개변수 길이를 바이트 단위로 지정합니다.

iLocalMsgFileLen

입력. iLocalMsgFileName 매개변수 길이를 바이트 단위로 지정합니다.

iTempFilesPathLen

입력. iTempFilesPath 매개변수 길이를 바이트 단위로 지정합니다.

piXmlPathList

입력. 해당 media_type 필드가 SQLU_LOCAL_MEDIA로 설정되고 해당 sqlu_media_entry 구조는 xml 파일이 있는 클라이언트 경로를 나열하는 sqlu_media_list의 포인터

db2gLoadIn 데이터 구조 특정 매개변수

iUseTablespaceLen

입력. piUseTablespace 매개변수의 길이(바이트)

piXmlParse

입력. XML 문서에 대해 수행해야 하는 구문 분석 유형. include 디렉토리의 db2ApiDf 헤더 파일에 있는 유효한 값은 다음과 같습니다.

DB2DMU_XMLPARSE_PRESERVE_WS

공백은 유지해야 합니다.

DB2DMU_XMLPARSE_STRIP_WS

공백은 스트립되어야 합니다.

piXmlValidate

입력. db2DMUXmlValidate 구조의 포인터. XML 스키마 유효성 확인이 XML 문서에 대해 수행되어야 하는 것을 나타냅니다.

```
/* XML Validate structure                                              */
typedef SQL_STRUCTURE db2DMUXmlValidate
{
   db2Uint16                       iUsing;       /* What to use to perform   */
                                                 /* validation               */
   struct db2DMUXmlValidateXds    *piXdsArgs;    /* Arguments for            */
                                                 /* XMLVALIDATE USING XDS    */
   struct db2DMUXmlValidateSchema *piSchemaArgs; /* Arguments for            */
                                                 /* XMLVALIDATE USING SCHEMA */
} db2DMUXmlValidate;
```

db2gPartLoadIn 데이터 구조 특정 매개변수

piReserved1

나중에 사용하기 위해 예약됨

iHostnameLen

입력. piHostname 매개변수의 길이(바이트)

iFileTransferLen

입력. piFileTransferCmd 매개변수의 길이(바이트)

iPartFileLocLen

입력. piPartFileLocation 매개변수의 길이(바이트)

iMapFileInputLen

입력. piMapFileInput 매개변수의 길이(바이트)

iMapFileOutputLen

입력. piMapFileOutput 매개변수의 길이(바이트)

iDistfileLen

입력. piDistfile 매개변수의 길이(바이트)

사용 시 참고사항

데이터는 입력 파일에 나타나는 시퀀스로 로드됩니다. 특정 시퀀스를 원할 경우 로드를 시도하기 전에 데이터를 정렬해야 합니다.

로드 유틸리티는 기존 정의에 따라 인덱스를 빌드합니다. 고유 키에 대한 중복을 처리하기 위해 예외 테이블을 사용합니다. 이 유틸리티는 참조 무결성을 강제하거나 제한조건 점검을 수행하거나 로드되는 테이블에 종속된 요약 테이블을 갱신하지 않습니다. 참조 또는 점검 제한조건을 포함하는 테이블은 무결성 설정 보류 상태에 놓입니다. 로드되고 있는 테이블에 종속되고 REFRESH IMMEDIATE를 사용하여 정의된 요약 테이블도 무결성 설정 보류 상태에 놓입니다. 이들 테이블을 무결성 설정 보류 상태에서 해제하려면 SET INTEGRITY 문을 발행하십시오. 복제된 요약 테이블에서는 로드 조작을 수행할 수 없습니다.

클러스터된 인덱스의 경우 로드하기 전에 클러스터링 인덱스에 대해 데이터를 정렬해야 합니다. 다차원적으로 클러스터된(MDC) 테이블에 로드하는 경우에는 데이터를 정렬하지 않아도 됩니다.

제 57 장 db2LoadQuery - 로드 조작 상태 가져오기

처리 중에 로드 조작 상태를 점검합니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2LoadQuery (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2LoadQueryStruct
{
   db2Uint32 iStringType;
   char *piString;
   db2Uint32 iShowLoadMessages;
   struct db2LoadQueryOutputStruct *poOutputStruct;
   char *piLocalMessageFile;
} db2LoadQueryStruct;

typedef SQL_STRUCTURE db2LoadQueryOutputStruct
{
   db2Uint32 oRowsRead;
   db2Uint32 oRowsSkipped;
   db2Uint32 oRowsCommitted;
   db2Uint32 oRowsLoaded;
   db2Uint32 oRowsRejected;
   db2Uint32 oRowsDeleted;
   db2Uint32 oCurrentIndex;
   db2Uint32 oNumTotalIndexes;
   db2Uint32 oCurrentMPPNode;
   db2Uint32 oLoadRestarted;
   db2Uint32 oWhichPhase;
   db2Uint32 oWarningCount;
   db2Uint32 oTableState;
} db2LoadQueryOutputStruct;

typedef SQL_STRUCTURE db2LoadQueryOutputStruct64
{
   db2Uint64 oRowsRead;
```

```
   db2Uint64 oRowsSkipped;
   db2Uint64 oRowsCommitted;
   db2Uint64 oRowsLoaded;
   db2Uint64 oRowsRejected;
   db2Uint64 oRowsDeleted;
   db2Uint32 oCurrentIndex;
   db2Uint32 oNumTotalIndexes;
   db2Uint32 oCurrentMPPNode;
   db2Uint32 oLoadRestarted;
   db2Uint32 oWhichPhase;
   db2Uint32 oWarningCount;
   db2Uint32 oTableState;
} db2LoadQueryOutputStruct64;

typedef SQL_STRUCTURE db2LoadQueryStruct64
{
   db2Uint32 iStringType;
   char *piString;
   db2Uint32 iShowLoadMessages;
   struct db2LoadQueryOutputStruct64 *poOutputStruct;
   char *piLocalMessageFile;
} db2LoadQueryStruct64;

SQL_API_RC SQL_API_FN
  db2gLoadQuery (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gLoadQueryStruct
{
   db2Uint32 iStringType;
   db2Uint32 iStringLen;
   char *piString;
   db2Uint32 iShowLoadMessages;
   struct db2LoadQueryOutputStruct *poOutputStruct;
   db2Uint32 iLocalMessageFileLen;
   char *piLocalMessageFile;
} db2gLoadQueryStruct;

typedef SQL_STRUCTURE db2gLoadQueryStru64
{
   db2Uint32 iStringType;
   db2Uint32 iStringLen;
   char *piString;
   db2Uint32 iShowLoadMessages;
   struct db2LoadQueryOutputStruct64 *poOutputStruct;
   db2Uint32 iLocalMessageFileLen;
   char *piLocalMessageFile;
} db2gLoadQueryStru64;
```

db2LoadQuery API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2LoadQueryStruct 구조의 포인터. 버전이 버전 9 이상인 경우 db2LoadQueryStruct64 구조의 포인터입니다. 그 외의 경우에는 db2LoadQueryStruct 구조의 포인터입니다.

pSqlca

출력. sqlca 구조의 포인터

db2LoadQueryStruct 데이터 구조 매개변수

iStringType

입력. piString 유형을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOADQUERY_TABLENAME

db2LoadQuery API에서 사용하는 테이블 이름을 지정합니다.

piString

입력. iStringType 값에 따라 임시 파일 경로 이름 또는 테이블 이름을 지정합니다.

iShowLoadMessages

입력. 로드 유틸리티에서 리턴되는 메시지 레벨을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOADQUERY_SHOW_ALL_MSGS

모든 로드 메시지를 리턴합니다.

DB2LOADQUERY_SHOW_NO_MSGS

로드 메시지를 리턴하지 않습니다.

DB2LOADQUERY_SHOW_NEW_MSGS

이 API를 마지막으로 호출한 이후에 생성된 메시지만 리턴합니다.

poOutputStruct

출력. 로드 요약 정보를 포함하는 db2LoadQueryOutputStruct 구조의 포인터. 요약이 필요없는 경우에는 NULL로 설정하십시오.

piLocalMessageFile

입력. 출력 메시지에 사용되는 로컬 파일 이름을 지정합니다.

db2LoadQueryOutputStruct 데이터 구조 매개변수

oRowsRead

출력. 로드 유틸리티에서 현재까지 읽은 레코드 수

oRowsSkipped

출력. 로드 조작이 시작되기 전에 건너뛴 레코드 수

oRowsCommitted

출력. 현재까지 목표 테이블에 커미트된 행 수

oRowsLoaded

출력. 현재까지 목표 테이블에 로드된 행 수

oRowsRejected

출력. 현재까지 목표 테이블에 거부된 행 수

oRowsDeleted

출력. 현재까지 목표 테이블에서 삭제된 행 수(삭제 단계에서)

oCurrentIndex

출력. 현재 빌드 중인 인덱스(빌드 단계에서)

oNumTotalIndexes

출력. 빌드되는 총 인덱스 수(빌드 단계에서)

oCurrentMPPNode

출력. 쿼리 중인 데이터베이스 파티션 서버를 식별합니다(파티션된 데이터베이스 환경 모드 전용)

oLoadRestarted

출력. 쿼리 중인 로드 조작이 로드 재시작 조작인 경우 값이 참인 플래그

oWhichPhase

출력. 쿼리 중인 로드 조작의 현재 단계를 표시합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOADQUERY_LOAD_PHASE

로드 단계.

DB2LOADQUERY_BUILD_PHASE

빌드 단계.

DB2LOADQUERY_DELETE_PHASE

삭제 단계.

DB2LOADQUERY_INDEXCOPY_PHASE

인덱스 복사 단계

oWarningCount

출력. 현재까지 리턴된 총 경고 수

oTableState

출력. 테이블 상태. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOADQUERY_NORMAL

테이블 상태가 테이블에 영향을 주지 않습니다.

DB2LOADQUERY_SI_PENDING

테이블에 제한조건이 있으며 제한조건이 아직 검증되지 않았습니다. SET INTEGRITY 명령을 사용하여 테이블을 B2LOADQUERY_SI_PENDING 상태에서 해제하십시오. 로드 유틸리티는 제한조건이 포함된 테이블 로드를 시작할 때 테이블을 DB2LOADQUERY_SI_PENDING 상태로 설정합니다.

DB2LOADQUERY_LOAD_IN_PROGRESS

이 테이블에서 진행 중인 로드 활동이 있습니다.

DB2LOADQUERY_LOAD_PENDING

이 테이블에서 로드가 활성화되어 있지만 로드가 커미트하기 전에 중단되었습니다. 로드 종료, 로드 재시작 또는 로드 교체를 실행하여 테이블을 DB2LOADQUERY_LOAD_PENDING 상태에서 해제하십시오.

DB2LOADQUERY_REORG_PENDING

reorg 권장 변경이 이 테이블에서 수행되었습니다. 일반 reorg는 테이블에 액세스할 수 있기 전에 수행해야 합니다.

DB2LOADQUERY_READ_ACCESS

읽기 액세스 쿼리에 테이블 데이터를 사용할 수 있습니다. DB2LOADQUERY_READ_ACCESS 옵션을 사용한 로드는 테이블을 읽기 액세스 전용 상태로 설정합니다.

DB2LOADQUERY_NOTAVAILABLE

테이블이 사용 불가능합니다. 테이블을 삭제하거나 백업에서 리스토어만 할 수 있습니다. 복구 불가능한 로드를 통한 롤 포워드의 경우 테이블을 사용 불가능 상태로 설정합니다.

DB2LOADQUERY_NO_LOAD_RESTART

테이블은 로드 재시작이 불가능한 부분 로드 상태입니다. 테이블은 로드 보류 상태가 되기도 합니다. 로드 종료 또는 로드 교체를 실행하여 테이블을 로드 재시작 불가능 상태에서 해제하십시오. 테이블은 롤 포워드 조작 중에 DB2LOADQUERY_NO_LOAD_RESTART 상태로 설정할 수도 있습니다. 로드 조작 종료 이전의 특정 시점으로 롤 포워드하거나 중단된 로드 조작을 통해 롤 포워드하지만 로드 종료 조작이나 로드 재시작 조작 종료 시점으로 롤 포워드하지 않는 경우에 발생할 수 있습니다.

DB2LOADQUERY_TYPE1_INDEXES

테이블은 현재 유형 1 인덱스를 사용합니다. 유형 1 인덱스는 더 이상 지원되지 않으며 다음에 테이블에 액세스할 때 유형 2 인덱스로 변환됩니다. 결과적으로, 이 값은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다.

db2LoadQueryOutputStruct64 데이터 구조 매개변수

oRowsRead

출력. 로드 유틸리티에서 현재까지 읽은 레코드 수

oRowsSkipped

출력. 로드 조작이 시작되기 전에 건너뛴 레코드 수

oRowsCommitted

출력. 현재까지 목표 테이블에 커미트된 행 수

oRowsLoaded

출력. 현재까지 목표 테이블에 로드된 행 수

oRowsRejected

출력. 현재까지 목표 테이블에 거부된 행 수

oRowsDeleted

출력. 현재까지 목표 테이블에서 삭제된 행 수(삭제 단계에서)

oCurrentIndex

출력. 현재 빌드 중인 인덱스(빌드 단계에서)

oNumTotalIndexes

출력. 빌드되는 총 인덱스 수(빌드 단계에서)

oCurrentMPPNode

출력. 쿼리 중인 데이터베이스 파티션 서버를 식별합니다(파티션된 데이터베이스 환경 모드 전용)

oLoadRestarted

출력. 쿼리 중인 로드 조작이 로드 재시작 조작인 경우 값이 참인 플래그

oWhichPhase

출력. 쿼리 중인 로드 조작의 현재 단계를 표시합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOADQUERY_LOAD_PHASE

로드 단계.

DB2LOADQUERY_BUILD_PHASE

빌드 단계.

DB2LOADQUERY_DELETE_PHASE

삭제 단계.

DB2LOADQUERY_INDEXCOPY_PHASE

인덱스 복사 단계

oWarningCount

출력. 현재까지 리턴된 총 경고 수

oTableState

출력. 테이블 상태. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOADQUERY_NORMAL

테이블 상태가 테이블에 영향을 주지 않습니다.

DB2LOADQUERY_SI_PENDING

테이블에 제한조건이 있으며 제한조건이 아직 검증되지 않았습니다. SET INTEGRITY 명령을 사용하여 테이블을 B2LOADQUERY_SI_PENDING 상태에서 해제하십시오. 로드 유틸리티는 제한조건이 포함된 테이블 로드를 시작할 때 테이블을 DB2LOADQUERY_SI_PENDING 상태로 설정합니다.

DB2LOADQUERY_LOAD_IN_PROGRESS

이 테이블에서 진행 중인 로드 활동이 있습니다.

DB2LOADQUERY_LOAD_PENDING

이 테이블에서 로드가 활성화되어 있지만 로드가 커미트하기 전에 중단되었습니다. 로드 종료, 로드 재시작 또는 로드 교체를 실행하여 테이블을 DB2LOADQUERY_LOAD_PENDING 상태에서 해제하십시오.

DB2LOADQUERY_REORG_PENDING

reorg 권장 변경이 이 테이블에서 수행되었습니다. 일반 reorg는 테이블에 액세스할 수 있기 전에 수행해야 합니다.

DB2LOADQUERY_READ_ACCESS

읽기 액세스 쿼리에 테이블 데이터를 사용할 수 있습니다. DB2LOADQUERY_READ_ACCESS 옵션을 사용한 로드는 테이블을 읽기 액세스 전용 상태로 설정합니다.

DB2LOADQUERY_NOTAVAILABLE

테이블이 사용 불가능합니다. 테이블을 삭제하거나 백업에서 리스토어만 할 수 있습니다. 복구 불가능한 로드를 통한 롤 포워드의 경우 테이블을 사용 불가능 상태로 설정합니다.

DB2LOADQUERY_NO_LOAD_RESTART

테이블은 로드 재시작이 불가능한 부분 로드 상태입니다. 테이블은 로드 보류 상태가 되기도 합니다. 로드 종료 또는 로드 교체를 실행하여 테이블을 로드 재시작 불가능 상태에서 해제하십시오. 테이블은 롤 포워드 조작 중에 DB2LOADQUERY_NO_LOAD_RESTART 상태로 설정할 수도 있습니다. 로드 조작 종료 이전의 특정 시점으로 롤 포워

드하거나 중단된 로드 조작을 통해 롤 포워드하지만 로드 종료 조작이나 로드 재시작 조작 종료 시점으로 롤 포워드하지 않는 경우에 발생할 수 있습니다.

DB2LOADQUERY_TYPE1_INDEXES
테이블은 현재 유형 1 인덱스를 사용합니다. 유형 1 인덱스는 더 이상 지원되지 않으며 다음에 테이블에 액세스할 때 유형 2 인덱스로 변환됩니다. 결과적으로, 이 값은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다.

db2LoadQueryStruct64 데이터 구조 매개변수

iStringType
입력. piString 유형을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOADQUERY_TABLENAME
db2LoadQuery API에서 사용하는 테이블 이름을 지정합니다.

piString
입력. iStringType 값에 따라 임시 파일 경로 이름 또는 테이블 이름을 지정합니다.

iShowLoadMessages
입력. 로드 유틸리티에서 리턴되는 메시지 레벨을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2LOADQUERY_SHOW_ALL_MSGS
모든 로드 메시지를 리턴합니다.

DB2LOADQUERY_SHOW_NO_MSGS
로드 메시지를 리턴하지 않습니다.

DB2LOADQUERY_SHOW_NEW_MSGS
이 API를 마지막으로 호출한 이후에 생성된 메시지만 리턴합니다.

poOutputStruct
출력. 로드 요약 정보를 포함하는 db2LoadQueryOutputStruct 구조의 포인터. 요약이 필요없는 경우에는 NULL로 설정하십시오.

piLocalMessageFile
입력. 출력 메시지에 사용되는 로컬 파일 이름을 지정합니다.

db2gLoadQueryStruct 데이터 구조 특정 매개변수

iStringLen
입력. piString 매개변수 길이를 바이트 단위로 지정합니다.

iLocalMessageFileLen

입력. piLocalMessageFile 매개변수 길이를 바이트 단위로 지정합니다.

db2gLoadQueryStru64 데이터 구조 특정 매개변수

iStringLen

입력. piString 매개변수 길이를 바이트 단위로 지정합니다.

iLocalMessageFileLen

입력. piLocalMessageFile 매개변수 길이를 바이트 단위로 지정합니다.

사용 시 참고사항

이 API는 piString으로 지정한 테이블에서 로드 조작 상태를 읽고 piLocalMsgFileName으로 지정한 파일에 상태를 기록합니다.

제 58 장 db2MonitorSwitches - 모니터 스위치 설정 가져오기 또는 갱신

데이터베이스 관리 프로그램이 수집하는 모니터 데이터 그룹의 스위치를 선택적으로 해제하거나 설정할 수 있습니다. 호출을 발행하는 응용프로그램에 대해 이 스위치의 현재 상태를 리턴합니다.

범위

이 API는 인스턴스의 데이터베이스 파티션 서버에 대한 정보 또는 인스턴스의 모든 데이터베이스 파티션에 대한 정보를 리턴할 수 있습니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

리모트 인스턴스(또는 다른 로컬 인스턴스)의 설정을 표시하려면 우선 해당 인스턴스에 접속해야 합니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2MonitorSwitches (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2MonitorSwitchesData
{
   struct sqlm_recording_group *piGroupStates;
   void *poBuffer;
   db2Uint32 iBufferSize;
   db2Uint32 iReturnData;
   db2Uint32 iVersion;
```

```
   db2int32 iNodeNumber;
   db2Uint32 *poOutputFormat;
} db2MonitorSwitchesData;

SQL_API_RC SQL_API_FN
  db2gMonitorSwitches (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gMonitorSwitchesData
{
   struct sqlm_recording_group *piGroupStates;
   void *poBuffer;
   db2Uint32 iBufferSize;
   db2Uint32 iReturnData;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
   db2Uint32 *poOutputFormat;
} db2gMonitorSwitchesData;
```

db2MonitorSwitches API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다. 위에서 설명한 대로 구조를 사용하려면 db2Version810을 지정하십시오. 이 구조의 다른 버전을 사용하려면 include 디렉토리의 db2ApiDf.h 헤더 파일에서 지원되는 버전의 전체 목록을 점검하십시오. 지정한 버전 번호에 해당하는 db2MonitorSwitchesStruct 구조의 버전을 사용하고 있는지 확인하십시오.

pParmStruct

입력. db2MonitorSwitchesStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2MonitorSwitchesData 데이터 구조 매개변수

piGroupStates

입력. 스위치 목록을 포함하는 sqlm-recording-group 구조의 포인터(sqlmon.h에 정의)

poBuffer

스위치 상태 데이터가 기록되는 버퍼의 포인터

iBufferSize

입력. 출력 버퍼 크기를 지정합니다.

iReturnData

입력. 현재 스위치 상태가 poBuffer로 포인트된 데이터 버퍼에 기록되는지 여부를 지정하는 플래그.

iVersion

입력. 수집할 데이터베이스 모니터 데이터의 버전 ID. 데이터베이스 모니터는 요청한 버전에서 사용할 수 있는 데이터만 리턴합니다. 이 매개변수를 다음 기호 상수 중 하나로 설정하십시오.

- SQLM_DBMON_VERSION1
- SQLM_DBMON_VERSION2
- SQLM_DBMON_VERSION5
- SQLM_DBMON_VERSION5_2
- SQLM_DBMON_VERSION6
- SQLM_DBMON_VERSION7
- SQLM_DBMON_VERSION8
- SQLM_DBMON_VERSION9
- SQLM_DBMON_VERSION9_5

주: SQLM_DBMON_VERSION1을 버전으로 지정한 경우 API는 리모트로 실행할 수 없습니다.

주: 상수 SQLM_DBMON_VERSION5_2 및 이전은 더 이상 사용되지 않으며 DB2의 추후 릴리스에서 제거될 수 있습니다.

iNodeNumber

입력. 요청을 보내는 데이터베이스 파티션 서버. 요청은 이 값을 사용하여 현재 데이터베이스 파티션 서버, 모든 데이터베이스 파티션 서버 또는 사용자가 지정한 데이터베이스 파티션 서버에 대해 처리됩니다. 가능한 값은 다음과 같습니다.

- SQLM_CURRENT_NODE
- SQLM_ALL_NODES
- 노드 값

주: 독립형 인스턴스의 경우 SQLM_CURRENT_NODE를 사용해야 합니다.

poOutputFormat

서버에서 리턴되는 스트림 형식. 다음 중 하나입니다.

SQLM_STREAM_STATIC_FORMAT

스위치 상태가 버전 7 이전의 스위치 구조로 정적으로 리턴되는 것을 나타냅니다.

SQLM_STREAM_DYNAMIC_FORMAT

스위치가 db2GetSnapshot에 대해 리턴되는 형식과 유사한 자체 기술적 형식으로 리턴되는 것을 나타냅니다.

사용 시 참고사항

데이터베이스 관리 프로그램 레벨에서 스위치 상태를 확보하려면 OBJ_TYPE에 SQLMA_DB2를 지정하여 db2GetSnapshot을 호출하십시오(데이터베이스 관리 프로그램에 대한 스냅샷 가져오기)

iVersion이 SQLM_DBMON_VERSION8보다 작으면 시간소인 스위치를 사용할 수 없습니다.

제 59 장 db2Prune - 사용 중인 로그 경로에서 실행기록 파일 항목 또는 로그 파일 삭제

사용 중인 로그 경로에 있는 실행기록 파일 또는 로그 파일에서 항목을 삭제합니다.

권한 부여

다음 중 하나가 필요합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*
- *dbadm*

필수 연결

데이터베이스. 디폴트 데이터베이스 이외의 다른 데이터베이스에 대해 실행기록 파일에서 항목을 삭제하려면 이 API를 호출하기 전에 데이터베이스에 연결해야 합니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Prune (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2PruneStruct
{
  char *piString;
  db2HistoryEID iEID;
  db2Uint32 iAction;
  db2Uint32 iOptions;
} db2PruneStruct;

SQL_API_RC SQL_API_FN
  db2gPrune (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gPruneStruct
{
  db2Uint32 iStringLen;
  char *piString;
```

```
  db2HistoryEID iEID;
  db2Uint32 iAction;
  db2Uint32 iOptions;
} db2gPruneStruct;
```

db2Prune API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2PruneStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2PruneStruct 데이터 구조 매개변수

piString

입력. 시간소인 또는 로그 시퀀스 번호(LSN)를 지정하는 문자열의 포인터. 시간소인 또는 시간소인 일부(최소 yyyy 또는 연도)를 사용하여 삭제할 레코드를 선택합니다. 시간소인과 동등하거나 이전의 모든 항목이 삭제됩니다. 유효한 시간소인를 제공해야 합니다. NULL 매개변수 값은 유효하지 않습니다.

이 매개변수를 사용하여 비활성된 로그를 프룬되도록 LSN을 전달할 수 있습니다.

iEID 입력. 실행기록 파일에서 한 개의 항목을 프룬하는 데 사용할 수 있는 고유 ID를 지정합니다.

iAction

입력. 수행할 조치 유형을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2PRUNE_ACTION_HISTORY

실행기록 파일 항목을 제거합니다.

DB2PRUNE_ACTION_LOG

사용 중인 로그 경로에서 로그 파일을 제거합니다.

iOptions

입력. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2PRUNE_OPTION_FORCE

강제로 최종 백업을 제거합니다.

DB2PRUNE_OPTION_DELETE

실행기록 파일에서 프룬된 로그 파일을 삭제합니다.

auto_del_rec_obj 데이터베이스 구성 매개변수를 ON으로 설정한 경우에 DB2PRUNE_OPTION_DELETE를 지정하여 db2Prune을 호출하면 관련된 백업 이미지와 로드 사본 이미지도 삭제됩니다.

DB2PRUNE_OPTION_LSNSTRING

호출자 조치인 DB2PRUNE_ACTION_LOG를 지정할 때 사용되는 LSN으로 piString 값을 지정하십시오.

db2gPruneStruct 데이터 구조 특정 매개변수

iStringLen

입력. piString 길이를 바이트 단위로 지정합니다.

사용 시 참고사항

do_not_delete 상태인 해당 항목은 삭제되거나 프룬되지 않습니다. UPDATE HISTORY 명령, UPDATE_HISTORY를 지정한 ADMIN_CMD 또는 db2HistoryUpdate API를 사용하여 복구 실행기록 파일 항목 상태를 do_not_delete로 설정할 수 있습니다. do_not_delete 상태를 사용하여 키 복구 실행기록 파일 항목이 프룬되거나 삭제되지 않도록 할 수 있습니다.

미디어에서 최신의 전체 데이터베이스 백업이 삭제된 경우(실행기록 파일에서 프룬된 것 이외에도) 사용자 테이블 스페이스 및 사용자 테이블 스페이스를 포함한 모든 테이블 스페이스가 백업되었는지 확인해야 합니다. 모두 백업되지 않은 경우에는 데이터베이스를 복구할 수 없거나 데이터베이스의 사용자 데이터 일부가 손실될 수 있습니다.

db2Prune을 사용하여 백업 데이터베이스 실행기록 파일 항목을 프룬할 수는 있지만 DB2PRUNE_OPTION_DELETE 매개변수를 사용하여 관련된 실제 복구 오브젝트를 삭제할 수는 없습니다. db2acsutil 명령을 사용해서만 스냅샷 백업 오브젝트를 삭제할 수 있습니다.

REXX API 구문

```
PRUNE RECOVERY HISTORY BEFORE :timestamp [WITH FORCE OPTION]
```

REXX API 매개변수

시간소인

시간소인이 포함된 호스트 변수. 시간소인이 제공된 시간소인과 동등하거나 이전의 모든 항목이 실행기록 파일에서 삭제됩니다.

WITH FORCE OPTION

지정한 경우에는 가장 최신 리스토어 세트의 일부 항목이 파일에서 삭제되어도 지정된 시간소인에 따라 실행기록 파일이 프룬됩니다. 지정하지 않으면 시간소인이 입력으로 지정한 시간소인과 동등하거나 이전인 경우에도 가장 최신의 리스토어 세트가 보존됩니다.

제 60 장 db2QuerySatelliteProgress - Satellite 동기화 세션 상태 가져오기

Satellite 동기화 세션 상태를 점검합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2QuerySatelliteProgress (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2QuerySatelliteProgressStruct
{
   db2int32 oStep;
   db2int32 oSubstep;
   db2int32 oNumSubsteps;
   db2int32 oScriptStep;
   db2int32 oNumScriptSteps;
   char *poDescription;
   char *poError;
   char *poProgressLog;
} db2QuerySatelliteProgressStruct;
```

db2QuerySatelliteProgress API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2QuerySatelliteProgressStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2QuerySatelliteProgressStruct 데이터 구조 매개변수

oStep 출력. 동기화 세션의 현재 단계(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)

oSubstep

출력. 매개변수로 표시된 동기화 단계인 oStep은 하위 단계로 분리할 수 있으며 이는 현재 하위 상태가 됩니다.

oNumSubsteps

출력. 동기화 세션의 현재 단계에 대한 하위 단계(oSubstep)가 있는 경우에는 동기화 단계를 구성하는 총 하위 단계 수가 됩니다.

oScriptStep

출력. 현재 하위 단계가 스크립트 실행인 경우 이 매개변수는 스크립트 실행 진행 상태를(가능한 경우) 보고합니다.

oNumScriptSteps

출력. 스크립트 단계가 보고되면 이 매개변수에는 스크립트 실행을 구성하는 총 단계 수가 포함됩니다.

poDescription

출력. Satellite 동기화 세션 상태에 대한 설명

poError

출력. 동기화 세션이 오류 상태인 경우 이 매개변수가 오류 설명을 전달합니다.

poProgressLog

출력. 이 매개변수가 Satellite 동기화 세션의 전체 로그를 리턴합니다.

제 61 장 db2ReadLog - 로그 레코드 읽기

DB2 데이터베이스 로그에서 로그 레코드를 읽고 현재 로그 상태 정보를 위해 로그 관리 프로그램을 쿼리합니다. 이 API는 복구 가능한 데이터베이스에서만 사용할 수 있습니다. 데이터베이스는 데이터베이스 구성 매개변수 logarchmeth1 및/또는 logarchmeth2가 OFF로 설정되지 않은 경우에 복구 가능합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- dbadm

필수 연결

데이터베이스

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2ReadLog (
       db2Uint32 versionNumber,
       void * pDB2ReadLogStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogStruct
{
   db2Uint32 iCallerAction;
   db2LSN *piStartLSN;
   db2LSN *piEndLSN;
   char *poLogBuffer;
   db2Uint32 iLogBufferSize;
   db2Uint32 iFilterOption;
   db2ReadLogInfoStruct *poReadLogInfo;
} db2ReadLogStruct;

typedef SQL_STRUCTURE db2ReadLogInfoStruct
{
   db2LSN initialLSN;
   db2LSN firstReadLSN;
   db2LSN nextStartLSN;
   db2LSN firstReusedLSN;
   db2Uint32 logRecsWritten;
   db2Uint32 logBytesWritten;
   db2Uint32 timeOfLSNReuse;
   db2TimeOfLog currentTimeValue;
```

```
} db2ReadLogInfoStruct;

typedef SQL_STRUCTURE db2TimeOfLog
{
   db2Uint32 seconds;
   db2Uint32 accuracy;
} db2TimeOfLog;

typedef SQL_STRUCTURE db2ReadLogFilterData
{
  db2LSN    recordLSN;
  db2Uint32 realLogRecLen;
  db2int32  sqlcode;
}
```

db2ReadLog API 매개변수

versionNumber

입력. 두 번째 매개변수(pDB2ReadLogStruct)로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2ReadLogStruct

입력. db2ReadLogStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ReadLogStruct 데이터 구조 매개변수

iCallerAction

입력. 수행할 조치를 지정합니다.

DB2READLOG_READ

시작 로그 시퀀스에서 종료 로그 시퀀스 번호까지 데이터베이스 로그를 읽고 이 범위 내의 로그 레코드를 리턴합니다.

DB2READLOG_READ_SINGLE

시작 로그 시퀀스 번호로 식별된 단일 로그 레코드(전파 가능 또는 불가능)를 읽습니다.

DB2READLOG_QUERY

데이터베이스 로그를 쿼리합니다. 쿼리 결과는 db2ReadLogInfoStruct 구조를 통해 다시 전송됩니다.

piStartLsn

입력. 시작 로그 시퀀스 번호는 로그 읽기에 대한 시작 상대 바이트 주소를 지정합니다. 이 값은 실제 로그 레코드의 시작이어야 합니다.

piEndLsn

입력. 종료 로그 시퀀스 번호는 로그 읽기에 대한 종료 상대 바이트 주소를 지정합니다. 이 값은 startLsn 매개변수보다 커야 하며 실제 로그 레코드의 끝일 필요는 없습니다.

poLogBuffer

출력. 지정된 범위 내에서 전파 가능한 모든 로그 레코드가 읽혀지는 버퍼는 순차적으로 저장됩니다. 이 버퍼는 단일 로그 레코드를 보유할 만큼 충분히 커야 합니다. 가이드라인으로, 이 버퍼는 최소 56바이트여야 합니다. 최대 *크기*는 요청된 범위의 *크기*에 따라 다릅니다.

- iFilterOption이 ON인 경우, db2ReadLogFilterData 구조는 각 로그 레코드 앞에 붙습니다.
- iFilterOption이 OFF인 경우, 버퍼의 각 로그 레코드 앞에는 다음 로그 레코드의 LSN을 표시하는 8바이트 로그 시퀀스 번호(LSN)가 앞에 붙습니다.

iLogBufferSize

입력. 로그 버퍼의 *크기*(바이트)를 지정합니다.

iFilterOption

입력. 로그 레코드를 읽을 때 사용되는 로그 레코드 필터링 레벨을 지정합니다. 가능한 값은 다음과 같습니다.

DB2READLOG_FILTER_OFF

지정한 LSN 범위에서 모든 로그 레코드를 읽으십시오.

DB2READLOG_FILTER_ON

전파 가능으로 표시된 지정한 LSN 범위의 로그 레코드만 읽습니다. 이는 비동기 로그 읽기 API의 일반적인 동작입니다. 이 값이 사용되는 경우 리턴되는 로그 레코드는 ″DB2 로그 레코드″ 주제에 설명되어 있습니다. 다른 모든 로그 레코드는 IBM의 내부 전용이므로 문서화되지 않습니다.

poReadLogInfo

출력. 호출 및 데이터베이스 로그에 대한 정보를 자세히 설명하는 구조.

db2ReadLogInfoStruct 데이터 구조 매개변수

initialLSN

활성화된 이후로 데이터베이스에서 사용된, 또는 사용될 첫 번째 LSN

firstReadLSN

poLogBuffer 매개변수에 있는 첫 번째 LSN

nextStartLSN

호출자가 읽어야 하는 다음 로그 레코드의 시작. 일부 로그 레코드는 필터되어 poLogBuffer 매개변수에서 리턴되지 않을 수도 있으므로, poLogBuffer 매개

변수의 마지막 로그 레코드 끝 대신 다음 읽기의 시작으로 이 LSN을 사용하면 이미 필터된 로그 레코드는 다시 스캔하지 않습니다.

firstReusedLSN
데이터베이스 리스토어 또는 롤 포워드 조작으로 재사용할 첫 번째 LSN

logRecsWritten
poLogBuffer 매개변수에 기록된 로그 레코드 수

logBytesWritten
poLogBuffer 매개변수에 기록된 데이터의 총 바이트 수

timeOfLSNReuse
firstReusedLSN으로 표시된 LSN이 재사용된 시간. 시간은 1970년 1월 1일 이후의 초 수입니다.

currentTimeValue
데이터베이스에 따른 현재 시간

db2TimeOfLog 데이터 구조 매개변수

seconds
1970년 1월 1일 이후의 초 수

accuracy
호출자가 동일한 초 내에 발생한 시간소인을 비교할 때 이벤트 순서를 구별할 수 있도록 하는 높은 정밀도의 카운터.

db2ReadLogFilterData 데이터 구조 매개변수

출력. db2ReadLogFilterData 구조는 다음과 같이 로그 레코드에 대한 메타데이터를 보유합니다.

recordLSN
다음 로그 레코드의 LSN

realLogRecLen
DB2 로그에서 물리 로그 레코드 길이

sqlcode
이 필드는 로그 레코드에서 압축된 행 이미지의 압축을 해제하려고 하는 동안 오류가 발생하는 경우 0이 아닌 값이 됩니다. 오류가 발생한 경우, 오류와 연관된 SQL 코드를 나타내는 정수가 포함됩니다. 영구 오류의 경우, SQL0204N이 리턴될 가능성이 가장 높습니다. API 요청을 다시 제출해도 동일한 오류가 리턴될 수 있습니다. 임시적 오류인 경우, 리턴되는 SQL 코드는 수정하기 위한 사용자 조치가 필요하거나 필요하지 않을 수 있는 오류 원인에 해당됩니다.

사용 시 참고사항

요청된 조치가 로그를 읽는 것이면, 로그 시퀀스 번호 범위와 로그 레코드를 보유할 버퍼를 제공해야 합니다. 이 API는 요청된 LSN 범위로 바운드된 로그를 순차적으로 읽고, DATA CAPTURE CHANGES 절로 정의된 테이블과 연관되는 로그 레코드와 현재 활성 로그 정보가 있는 db2ReadLogInfoStruct 구조를 리턴합니다. 요청된 조치가 데이터베이스 로그의 쿼리인 경우(DB2READLOG_QUERY 값을 지정하여 표시됨), API는 현재 활성 로그 정보가 있는 db2ReadLogInfoStruct 구조를 리턴합니다.

비동기 로그 판독기를 사용하려면, 먼저 유효한 시작 LSN을 찾기 위해 데이터베이스 로그를 쿼리하십시오. 쿼리 호출 다음에, 읽기 로그 정보 구조(db2ReadLogInfoStruct)에는 읽기 호출에 사용될 유효한 시작 LSN(initialLSN 구성원에서)이 포함됩니다. 읽기에서 종료 LSN으로 사용되는 값은 다음 중 하나가 될 수 있습니다.

- initialLSN보다 큰 값
- 비동기 로그 판독기에 의해 현재 로그의 끝으로 해석되는 FFFF FFFF FFFF FFFF

시작 및 종료 LSN 범위 내에서 읽혀진 전파 가능한 로그 레코드가 로그 버퍼에서 리턴됩니다. iFilterOption 옵션을 DB2READLOG_FILTER_ON으로 설정한 경우, LSN은 버퍼에서 db2ReadLogFilterData 데이터 구조로 교체됩니다. db2ReadLog에 의해 리턴되는 다양한 DB2 로그 레코드의 설명은 ″DB2 로그 레코드″ 주제에서 찾을 수 있습니다.

초기 읽기 이후에 다음 순차 로그 레코드를 읽으려면, db2ReadLogStruct에서 리턴된 nextStartLSN 필드를 사용하십시오. 이 새로운 시작 LSN 및 유효한 종료 LSN으로 호출을 다시 제출하십시오. 다음 레코드 블록을 읽습니다.
SQLU_RLOG_READ_TO_CURRENT의 sqlca 코드는 로그 판독기가 현재 활성 로그의 끝까지 읽었음을 의미합니다.

이 API는 DB2 로그에서 데이터를 읽습니다. 레이블 기반 액세스 제어(LBAC)는 이와 같은 로그에서 시행되지 않습니다. 따라서 이 API를 호출하는 응용프로그램은 호출자가 API를 호출하기 위한 충분한 권한을 가지고 있고 로그 레코드 형식을 이해할 수 있는 경우 테이블 데이터에 대한 액세스를 얻을 수 있습니다.

db2ReadLog API는 현재 데이터베이스 연결에서 작동합니다. 여러 개의 데이터베이스 연결이 동일한 프로세스에서 작성되는 경우, 동시 액세스 API를 사용하여 여러 컨텍스트를 관리하십시오.

응용프로그램에서 db2ReadLog API를 호출하면 연결이 끊어지기 전에 커미트 또는 롤백이 수행되지 않는 경우에 데이터베이스에서 응용프로그램 연결이 끊어질 때 오류가 발생할 수 있습니다.

- CLI 응용프로그램에서 db2ReadLog API가 호출되는 경우 CLI0116E 오류가 생성될 수 있습니다.

- C로 작성된 Embedded SQL 응용프로그램에서 db2ReadLog API가 호출되는 경우 SQL0428N 오류가 생성될 수 있습니다.

일시적인 해결책 1: Embedded SQL 응용프로그램이 아닌 SQL 응용프로그램의 경우, db2ReadLog API를 호출하기 전에 자동 커미트 모드를 설정하십시오.

일시적인 해결책 2: db2ReadLog API를 호출한 후, 데이터베이스에서 연결이 끊어지기 전에 COMMIT 또는 ROLLBACK 문을 발행하십시오.

제 62 장 db2ReadLogNoConn - 데이터베이스에 연결하지 않고 데이터베이스 로그 읽기

DB2 데이터베이스 로그에서 로그 레코드를 추출하고 현재 로그 상태 정보를 위해 로그 관리 프로그램을 쿼리합니다. 이 API를 사용하기 전에, db2ReadLogNoConnInit API를 호출하여 이 API에 입력 매개변수로 전달되는 메모리를 할당하십시오. 이 API를 호출한 후, db2ReadLogNoConnTerm API를 호출하여 메모리 할당을 해제하십시오.

권한 부여

없음

필수 연결

없음

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2ReadLogNoConn (
       db2Uint32 versionNumber,
       void * pDB2ReadLogNoConnStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnStruct
{
   db2Uint32 iCallerAction;
   db2LSN *piStartLSN;
   db2LSN *piEndLSN;
   char *poLogBuffer;
   db2Uint32 iLogBufferSize;
   char *piReadLogMemPtr;
   db2ReadLogNoConnInfoStruct *poReadLogInfo;
} db2ReadLogNoConnStruct;

typedef SQL_STRUCTURE db2ReadLogNoConnInfoStruct
{
   db2LSN firstAvailableLSN;
   db2LSN firstReadLSN;
   db2LSN nextStartLSN;
   db2Uint32 logRecsWritten;
   db2Uint32 logBytesWritten;
   db2Uint32 lastLogFullyRead;
   db2TimeOfLog currentTimeValue;
} db2ReadLogNoConnInfoStruct;
```

db2ReadLogNoConn API 매개변수

versionNumber

입력. 두 번째 매개변수(pDB2ReadLogNoConnStruct)로 전달되는 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2ReadLogNoConnStruct

입력. db2ReadLogNoConnStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ReadLogNoConnStruct 데이터 구조 매개변수

iCallerAction

입력. 수행할 조치를 지정합니다. 가능한 값은 다음과 같습니다.

DB2READLOG_READ

시작 로그 시퀀스에서 종료 로그 시퀀스 번호까지 데이터베이스 로그를 읽고 이 범위 내의 로그 레코드를 리턴합니다.

DB2READLOG_READ_SINGLE

시작 로그 시퀀스 번호로 식별된 단일 로그 레코드(전파 가능 또는 불가능)를 읽습니다.

DB2READLOG_QUERY

데이터베이스 로그를 쿼리합니다. 쿼리 결과는 db2ReadLogNoConnInfoStruct 구조를 통해 다시 전송됩니다.

piStartLSN

입력. 시작 로그 시퀀스 번호는 로그 읽기에 대한 시작 상대 바이트 주소를 지정합니다. 이 값은 실제 로그 레코드의 시작이어야 합니다.

piEndLSN

입력. 종료 로그 시퀀스 번호는 로그 읽기에 대한 종료 상대 바이트 주소를 지정합니다. 이 값은 piStartLsn보다 커야 하며 실제 로그 레코드의 끝일 필요는 없습니다.

poLogBuffer

출력. 지정된 범위 내에서 전파 가능한 모든 로그 레코드를 읽는 버퍼는 순차적으로 저장됩니다. 이 버퍼는 단일 로그 레코드를 보유할 만큼 충분히 커야 합니다. 가이드라인으로, 이 버퍼는 최소 48바이트여야 합니다. 최대 크기는 요청된 범위의 크기에 따라 다릅니다.

버퍼의 각 로그 레코드에는 다음 로그 레코드의 LSN을 표시하는 8바이트 로그 시퀀스 번호(LSN)가 앞에 붙습니다.

iLogBufferSize

입력. 로그 버퍼의 *크기*(바이트)를 지정합니다.

piReadLogMemPtr

입력. *초기화* 호출에서 할당된 iReadLogMemoryLimit *크기*의 메모리 블록. 이 메모리에는 각 호출 시마다 API에 필요한 지속적 데이터가 포함됩니다. 이 메모리 블록은 호출자가 다시 할당하거나 수정하면 안됩니다.

poReadLogInfo

출력. db2ReadLogNoConnInfoStruct 구조의 포인터

db2ReadLogNoConnInfoStruct 데이터 구조 매개변수

firstAvailableLSN

사용 가능한 로그에서 첫 번째 사용 가능한 LSN

firstReadLSN

해당 호출에서 읽은 첫 번째 LSN

nextStartLSN

읽을 수 있는 다음 LSN

logRecsWritten

로그 버퍼 필드(poLogBuffer)에 기록된 로그 레코드 수

logBytesWritten

로그 버퍼 필드(poLogBuffer)에 기록된 바이트 수

lastLogFullyRead

완료하기 위해 읽은 마지막 로그 파일을 표시하는 번호

currentTimeValue

나중에 사용하기 위해 예약됨

사용 시 참고사항

db2ReadLogNoConn API에는 db2ReadLogNoConnInit API를 사용하여 할당해야 하는 메모리 블록이 필요합니다. 메모리 블록은 모든 연속 db2ReadLogNoConn API 호출에 입력 매개변수로 전달해야 하며 변경할 수 없습니다.

로그의 순차 읽기를 요청하는 경우, API에는 로그 시퀀스 번호(LSN) 범위와 할당된 메모리가 필요합니다. API는 초기화할 때 지정한 필터 옵션과 LSN 범위를 기초로 로그 레코드 시퀀스를 리턴합니다. 쿼리를 요청할 때, 읽기 로그 정보 구조에는 읽기 호출에 사용될 유효한 시작 LSN이 포함됩니다. 읽기에서 종료 LSN으로 사용되는 값은 다음 중 하나가 될 수 있습니다.

- 호출자 지정 startLSN보다 큰 값

- 비동기 로그 판독기에 의해 사용 가능한 로그의 끝으로 해석되는 FFFF FFFF FFFF FFFF

시작 및 종료 LSN 범위 내에서 읽혀진 전파 가능한 로그 레코드가 로그 버퍼에서 리턴됩니다. 로그 레코드는 자체의 LSN을 포함하지 않고, 이 LSN은 실제 로그 레코드 이전에 버퍼에 포함됩니다. db2ReadLogNoConn에 의해 리턴되는 다양한 DB2 로그 레코드의 설명은 DB2 로그 레코드 섹션에서 찾을 수 있습니다.

초기 읽기 이후에 다음 순차 로그 레코드를 읽으려면 db2ReadLogNoConnInfoStruct에서 리턴된 nextStartLSN 값을 사용하십시오. 이 새로운 시작 LSN 및 유효한 종료 LSN으로 호출을 다시 제출하십시오. 다음 레코드 블록을 읽습니다.
SQLU_RLOG_READ_TO_CURRENT의 sqlca 코드는 로그 판독기가 사용 가능한 로그 파일의 끝까지 읽었음을 의미합니다.

API가 더 이상 사용되지 않는 경우 db2ReadLogNoConnTerm을 사용하여 메모리를 종료하십시오.

이 API는 DB2 로그에서 데이터를 읽습니다. 레이블 기반 액세스 제어(LBAC)는 이와 같은 로그에서 시행되지 않습니다. 따라서 이 API를 호출하는 응용프로그램은 호출자가 API를 호출하기 위한 충분한 권한을 가지고 있고 로그 레코드 형식을 이해할 수 있는 경우 테이블 데이터에 대한 액세스를 잠재적으로 얻을 수 있습니다.

주: 이 API는 데이터베이스에 연결해야 하는 로그 레코드의 압축된 행 이미지의 포맷팅을 지원하지 않습니다. 이를 수행하려면 대신 db2ReadLog API를 사용하십시오. 압축된 행 이미지의 포맷팅은 지원되지 않으므로, 이 API는 압축된 행 이미지를 그대로 리턴합니다.

제 63 장 db2ReadLogNoConnInit - 데이터베이스에 연결하지 않고 데이터베이스 로그 읽기 초기화

DB2 데이터베이스 로그에서 로그 레코드를 추출하고 현재 로그 상태 정보를 위해 로그 관리 프로그램을 쿼리하기 위해 db2ReadLogNoConn에서 사용되는 메모리를 할당합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2ReadLogNoConnInit (
       db2Uint32 versionNumber,
       void * pDB2ReadLogNoConnInitStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnInitStruct
{
   db2Uint32 iFilterOption;
   char *piLogFilePath;
   char *piOverflowLogPath;
   db2Uint32 iRetrieveLogs;
   char *piDatabaseName;
   char *piNodeName;
   db2Uint32 iReadLogMemoryLimit;
   char                                **poReadLogMemPtr;
} db2ReadLogNoConnInitStruct;
```

db2ReadLogNoConnInit API 매개변수

versionNumber

입력. 두 번째 매개변수 pDB2ReadLogNoConnInitStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2ReadLogNoConnInitStruct

입력. db2ReadLogNoConnInitStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ReadLogNoConnInitStruct 데이터 구조 매개변수

iFilterOption

입력. 로그 레코드를 읽을 때 사용되는 로그 레코드 필터링 레벨을 지정합니다. 가능한 값은 다음과 같습니다.

DB2READLOG_FILTER_OFF

지정한 LSN 범위에서 모든 로그 레코드를 읽으십시오.

DB2READLOG_FILTER_ON

전파 가능으로 표시된 지정한 LSN 범위의 로그 레코드만 읽습니다. 이는 비동기 로그 읽기 API의 일반적인 동작입니다.

piLogFilePath

입력. 읽으려는 로그 파일이 있는 경로

piOverflowLogPath

입력. 읽으려는 로그 파일이 위치할 수도 있는 대체 경로

iRetrieveLogs

입력. 로그 파일 경로 또는 오버플로우 로그 경로에 없는 로그 파일을 검색하기 위해 userexit를 호출해야 하는지를 지정하는 옵션. 가능한 값은 다음과 같습니다.

DB2READLOG_RETRIEVE_OFF

누락된 로그 파일을 검색하기 위해 userexit를 호출하면 안됩니다.

DB2READLOG_RETRIEVE_LOGPATH

지정한 로그 파일 경로에서 누락된 로그 파일을 검색하기 위해 userexit를 호출해야 합니다.

DB2READLOG_RETRIEVE_OVERFLOW

지정한 오버플로우 로그 경로에서 누락된 로그 파일을 검색하기 위해 userexit를 호출해야 합니다.

piDatabaseName

입력. 읽고 있는 복구 로그를 소유하는 데이터베이스 이름위의 검색 옵션을 지정한 경우에는 필수입니다.

piNodeName

입력. 읽고 있는 복구 로그를 소유하는 노드 이름. 위의 검색 옵션을 지정한 경우에는 필수입니다.

iReadLogMemoryLimit

입력. API가 내부적으로 할당할 수 있는 최대 바이트 수

poReadLogMemPtr

출력. iReadLogMemoryLimit *크기*의 API 할당 메모리 블록. 이 메모리에는 각 호출 시마다 API에 필요한 지속적 데이터가 포함됩니다. 이 메모리 블록은 호출자가 다시 할당하거나 수정하면 안됩니다.

사용 시 참고사항

db2ReadLogNoConnInit로 *초기화*된 메모리는 변경하면 안됩니다.

db2ReadLogNoConn이 더 이상 사용되지 않는 경우 db2ReadLogNoConnTerm을 호출하여 db2ReadLogNoConnInit로 *초기화*된 메모리를 할당 해제하십시오.

제 64 장 db2ReadLogNoConnTerm - 데이터베이스에 연결하지 않고 데이터베이스 로그 읽기 종료

원래 db2ReadLogNoConnInit API로 초기화된 db2ReadLogNoConn API에서 사용된 메모리를 할당 해제합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2ReadLogNoConnTerm (
       db2Uint32 versionNumber,
       void * pDB2ReadLogNoConnTermStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReadLogNoConnTermStruct
{
   char                                **poReadLogMemPtr;
} db2ReadLogNoConnTermStruct;
```

db2ReadLogNoConnTerm API 매개변수

versionNumber

입력. 두 번째 매개변수 pDB2ReadLogNoConnTermStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2ReadLogNoConnTermStruct

입력. db2ReadLogNoConnTermStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ReadLogNoConnTermStruct 데이터 구조 매개변수

poReadLogMemPtr

출력. 초기화 호출에서 할당된 메모리 블록의 포인터. 이 포인터는 사용 가능화되어 NULL로 설정됩니다.

제 65 장 db2Recover - 데이터베이스 리스토어 및 롤 포워드

특정 시점 또는 로그 끝으로 데이터베이스를 리스토어하거나 롤 포워드합니다.

범위

파티션된 데이터베이스 환경에서는 이 API를 카탈로그 파티션에서만 호출할 수 있습니다. 데이터베이스 파티션 서버를 지정하지 않은 경우 db2nodes.cfg 파일에 나열된 모든 데이터베이스 파티션 서버에만 영향을 줍니다. 특정 시점이 지정된 경우 API는 모든 데이터베이스 파티션에 영향을 줍니다.

권한 부여

다음 중 하나의 기존 데이터베이스를 복구하려면 다음을 수행하십시오.

- sysadm
- sysctrl
- sysmaint

다음 중 하나의 새 데이터베이스를 복구하려면 다음을 수행하십시오.

- sysadm
- sysctrl

필수 연결

기존 데이터베이스를 복구하려면 데이터베이스에 연결해야 합니다. 이 API는 자동으로 지정한 데이터베이스에 연결하고 복구 조작이 완료되면 연결을 해제합니다. 새 데이터베이스로 복구하려는 인스턴스 및 데이터베이스. 데이터베이스를 작성하려면 인스턴스에 접속해야 합니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Recover (
       db2Uint32 versionNumber,
       void * pDB2RecovStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RecoverStruct
{
  char *piSourceDBAlias;
  char *piUsername;
```

```
  char *piPassword;
  db2Uint32 iRecoverCallerAction;
  db2Uint32 iOptions;
  sqlint32 *poNumReplies;
  struct sqlurf_info *poNodeInfo;
  char *piStopTime;
  char *piOverflowLogPath;
  db2Uint32 iNumChngLgOvrflw;
  struct sqlurf_newlogpath *piChngLogOvrflw;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  SQL_PDB_NODE_TYPE *piNodeList;
  db2int32 iNumNodeInfo;
  char *piHistoryFile;
  db2Uint32 iNumChngHistoryFile;
  struct sqlu_histFile *piChngHistoryFile;
  char *piComprLibrary;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
} db2RecoverStruct;

SQL_STRUCTURE sqlu_histFile
{
   SQL_PDB_NODE_TYPE nodeNum;
   unsigned short  filenameLen;
   char filename[SQL_FILENAME_SZ+1];
};

SQL_API_RC SQL_API_FN
  db2gRecover (
       db2Uint32 versionNumber,
       void * pDB2gRecoverStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRecoverStruct
{
  char *piSourceDBAlias;
  db2Uint32 iSourceDBAliasLen;
  char *piUserName;
  db2Uint32 iUserNameLen;
  char *piPassword;
  db2Uint32 iPasswordLen;
  db2Uint32 iRecoverCallerAction;
  db2Uint32 iOptions;
  sqlint32 *poNumReplies;
  struct sqlurf_info *poNodeInfo;
  char *piStopTime;
  db2Uint32 iStopTimeLen;
  char *piOverflowLogPath;
  db2Uint32 iOverflowLogPathLen;
  db2Uint32 iNumChngLgOvrflw;
  struct sqlurf_newlogpath *piChngLogOvrflw;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  SQL_PDB_NODE_TYPE *piNodeList;
  db2int32 iNumNodeInfo;
  char *piHistoryFile;
  db2Uint32 iHistoryFileLen;
```

```
  db2Uint32 iNumChngHistoryFile;
  struct sqlu_histFile *piChngHistoryFile;
  char *piComprLibrary;
  db2Uint32 iComprLibraryLen;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
} db2gRecoverStruct;
```

db2Recover API 매개변수

versionNumber

입력. 두 번째 매개변수 pDB2RecoverStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2RecoverStruct

입력. db2RecoverStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2RecoverStruct 데이터 구조 매개변수

piSourceDBAlias

입력. 복구하려는 데이터베이스의 데이터베이스 별명이 포함된 문자열

piUserName

입력. 연결 시도 시에 사용되는 사용자 이름이 포함된 문자열. NULL일 수 있습니다.

piPassword

입력. 사용자 이름에 사용되는 암호가 포함된 문자열. NULL일 수 있습니다.

iRecoverCallerAction

입력. 가능한 값은 다음과 같습니다.

DB2RECOVER

복구 조작을 시작합니다. 복구가 자동으로 실행되며 일반적으로 사용자가 개입해야 하는 시나리오가 호출자에게 우선 리턴되지 않고 시도되거나 오류를 생성하도록 지정합니다. 예를 들어, 이 호출자 조치를 사용하면, 복구에 필요한 모든 미디어가 마운트된 것을 알고 있는 경우 유틸리티 프롬프트가 필요하지 않습니다.

DB2RECOVER_RESTART

사용자가 이전 복구를 무시하고 처음부터 시작할 수 있도록 합니다.

DB2RECOVER_CONTINUE

경고 메시지를 생성한 디바이스를 사용하여 계속합니다(예를 들어, 새로운 테이프가 마운트된 경우).

DB2RECOVER_LOADREC_TERM

로드 복구에서 사용 중인 모든 디바이스를 종료합니다.

DB2RECOVER_DEVICE_TERM

경고 메시지를 생성한 디바이스를 사용하여 중지합니다(예를 들어, 더 이상의 테이프가 없는 경우).

DB2RECOVER_PARM_CHK_ONLY

복구 조작을 수행하지 않고 매개변수의 유효성을 확인하는 데 사용됩니다. 이 호출이 리턴되기 전에 이 호출로 작성된 데이터베이스 연결이 종료되며 연속된 호출은 필요하지 않습니다.

DB2RECOVER_DEVICE_TERMINATE

복구 조작에서 사용되는 디바이스 목록에서 특정 디바이스를 제거합니다. 특정 디바이스가 해당 입력에서 모두 사용된 경우 복구는 호출자에 경고를 리턴합니다. 이 호출자 조치로 복구 유틸리티를 다시 호출하여 경고를 생성한 디바이스를 사용 중인 디바이스 목록에서 제거하십시오.

iOptions

입력. 가능한 값은 다음과 같습니다.

- DB2RECOVER_EMPTY_FLAG

플래그가 지정되지 않습니다.

- DB2RECOVER_LOCAL_TIME

piStopTime으로 중지 시간에 대해 지정된 값은 GMT가 아닌 로컬 시간임을 나타냅니다. 이는 디폴트 설정입니다.

- DB2RECOVER_GMT_TIME

이 플래그는 piStopTime으로 중지 시간에 대해 지정된 값이 GMT(Greenwich Mean Time)임을 나타냅니다.

poNumReplies

출력. 수신된 응답 수

poNodeInfo

출력. 데이터베이스 파티션 응답 정보

piStopTime

입력. ISO 형식의 시간소인을 포함하는 문자열. 이 시간소인이 초과되면 데이터베이스 복구가 중지됩니다. 최대한 롤 포워드하려면 SQLUM_INFINITY_TIMESTAMP를 지정하십시오. DB2ROLLFORWARD_QUERY, DB2ROLLFORWARD_PARM_CHECK 및 모든 로그 복구(DB2ROLLFORWARD_LOADREC_) 호출자 조치에 대해 NULL일 수 있습니다.

piOverflowLogPath

입력. 이 매개변수를 사용하여 사용하려는 대체 로그 경로를 지정합니다. 사용 중인 로그 파일 뿐만 아니라 이 유틸리티에서 사용하기 전에 아카이브된 로그 파일을 logpath 구성 매개변수로 지정한 위치로 사용자가 이동해야 합니다. 로그 경로에 충분한 스페이스가 없는 경우에는 문제가 발생할 수도 있습니다. 이런 이유로 인해 오버플로우 로그 경로가 제공됩니다. 롤 포워드 복구 중에는 필요한 로그 파일이 로그 경로에서 먼저 검색되고 오버플로우 로그 경로에서 두 번째로 검색됩니다. 테이블 스페이스 롤 포워드 복구에 필요한 로그 파일은 로그 경로 또는 오버플로우 로그 경로로 가져올 수 있습니다. 호출자가 오버플로우 로그 경로를 지정하지 않으면 디폴트값은 로그 경로입니다.

파티션된 데이터베이스 환경에서 오버플로우 로그 경로는 유효한 완전한 경로여야 합니다. 디폴트 경로는 각 데이터베이스 파티션의 디폴트 오버플로우 로그 경로입니다. 단일 파티션 데이터베이스 환경에서 오버플로우 로그 경로는 서버가 로컬인 경우 상대 경로입니다.

iNumChngLgOvrflw

입력. 파티션된 데이터베이스 환경에만 적용됩니다. 변경된 오버플로우 로그 경로 수. 이 새 로그 경로는 지정된 데이터베이스 파티션 서버에 대해서만 디폴트 오버플로우 로그 경로를 겹쳐씁니다.

piChngLogOvrflw

입력. 파티션된 데이터베이스 환경에만 적용됩니다. 변경된 오버플로우 로그 경로의 완전한 이름이 포함된 구조의 포인터. 이 새 로그 경로는 지정된 데이터베이스 파티션 서버에 대해서만 디폴트 오버플로우 로그 경로를 겹쳐씁니다.

iAllNodeFlag

입력. 파티션된 데이터베이스 환경에만 적용됩니다. 롤 포워드 조작이 db2nodes.cfg에 정의된 모든 데이터베이스 파티션 서버에 적용되는지를 나타냅니다. 가능한 값은 다음과 같습니다.

DB2_NODE_LIST

piNodeList로 전달된 목록의 데이터베이스 파티션 서버에 적용됩니다.

DB2_ALL_NODES

모든 데이터베이스 파티션 서버에 적용됩니다. piNodeList는 NULL이어야 합니다. 이것은 디폴트값입니다.

DB2_ALL_EXCEPT

piNodeList로 전달된 목록에 있는 데이터베이스 파티션 서버를 제외한 모든 데이터베이스 파티션 서버에 적용됩니다.

DB2_CAT_NODE_ONLY

카탈로그 파티션에만 적용됩니다. piNodeList는 NULL이어야 합니다.

iNumNodes

입력. piNodeList 배열의 데이터베이스 파티션 서버 수를 지정합니다.

piNodeList

입력. 롤 포워드 복구를 수행해야 하는 데이터베이스 파티션 서버 수 배열의 포인터

iNumNodeInfo

입력. 출력 매개변수 poNodeInfo *크기*를 정의하며 이는 롤 포워드되는 각 데이터베이스 파티션의 상태 정보를 모두 포함할 수 있는 *크기*여야 합니다. 단일 파티션의 데이터베이스 환경의 경우 이 매개변수는 1로 설정해야 합니다. 이 매개변수 값은 이 API가 호출되는 데이터베이스 파티션 서버의 수와 동일해야 합니다.

piHistoryFile

실행기록 파일

iNumChngHistoryFile

목록의 실행기록 파일 수

piChngHistoryFile

실행기록 파일 목록

piComprLibrary

입력. 이미지가 압축된 경우 백업 이미지 압축 해제 수행에 사용되는 외부 라이브러리의 이름을 표시합니다. 이름은 서버의 파일을 나타내는 완전한 경로여야 합니다. 값이 NULL 포인터 또는 빈 문자열의 포인터인 경우 DB2는 이미지에 저장된 라이브러리를 사용하려고 합니다. 백업이 압축되지 않은 경우 이 매개변수 값은 무시됩니다. 지정한 라이브러리를 찾을 수 없는 경우 리스토어가 실패합니다.

piComprOptions

입력. 압축 해제 라이브러리의 *초기화* 루틴으로 전달될 2진 데이터 블록을 설명합니다. DB2는 이 문자열을 클라이언트에서 서버로 직접 전달하여 바이트 리버설이나 코드 페이지 변환에 관련된 모든 문제는 압축 라이브러리가 처리하도록 합니다. 데이터 블록의 첫 번째 문자가 '@'인 경우 데이터의 나머지는 서버에 있는 파일 이름으로 DB2가 해석합니다. 그런 다음 DB2가 piComprOptions 및 iComprOptionsSize의 컨텐츠를 이 파일의 컨텐츠와 *크기*로 교체하고 대신 이 새 값을 초기화 루틴에 전달합니다.

iComprOptionsSize

입력. piComprOptions로 전달된 데이터 블록 *크기*를 나타냅니다. piComprOptions가 NULL 포인터인 경우에는 iComprOptionsSize가 영(0)이 됩니다.

sqlu_histFile 데이터 구조 매개변수

nodeNum

입력. 이 항목이 사용되어야 하는 데이터베이스 파티션을 지정합니다.

filenameLen

입력. 파일 이름 길이(바이트)

filename

입력. 이 데이터베이스 파티션의 실행기록 파일 경로. 경로는 슬래시로 종료되어야 합니다.

db2gRecoverStruct 데이터 구조 특정 매개변수

iSourceDBAliasLen

piSourceDBAlias 매개변수 길이를 바이트 단위로 지정합니다.

iUserNameLen

piUsername 매개변수 길이를 바이트 단위로 지정합니다.

iPasswordLen

piPassword 매개변수 길이를 바이트 단위로 지정합니다.

iStopTimeLen

piStopTime 매개변수 길이를 바이트 단위로 지정합니다.

iOverflowLogPathLen

piOverflowLogPath 매개변수 길이를 바이트 단위로 지정합니다.

iHistoryFileLen

piHistoryFile 매개변수 길이를 바이트 단위로 지정합니다.

iComprLibraryLen

입력. piComprLibrary 매개변수에 지정된 라이브러리 이름 길이를 바이트 단위로 지정합니다. 라이브러리 이름을 지정하지 않은 경우 영(0)으로 설정하십시오.

제 66 장 db2Reorg - 인덱스 또는 테이블 재구성

분할된 데이터를 제거하기 위해 정보를 압축하고 행 또는 인덱스 데이터를 재구성하여 테이블에 정의된 테이블 및 모든 인덱스를 재구성합니다.

권한 부여

다음 중 하나가 필요합니다.

- SYSADM
- SYSCTRL
- SYSMAINT
- DBADM
- SQLADM
- 테이블에 대한 CONTROL 특권

필수 연결

데이터베이스

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Reorg (
             db2Uint32 versionNumber,
             void * pReorgStruct,
             struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ReorgStruct
{
  db2Uint32 reorgType;
  db2Uint32 reorgFlags;
  db2int32 nodeListFlag;
  db2Uint32 numNodes;
  SQL_PDB_NODE_TYPE *pNodeList;
  union db2ReorgObject        reorgObject;
} db2ReorgStruct;

union db2ReorgObject
{
  struct db2ReorgTable             tableStruct;
  struct db2ReorgIndexesAll        indexesAllStruct;
};

typedef SQL_STRUCTURE db2ReorgTable
```

```
{
  char *pTableName;
  char *pOrderByIndex;
  char *pSysTempSpace;
  char *pLongTempSpace;
  char *pPartitionName;
} db2ReorgTable;

typedef SQL_STRUCTURE db2ReorgIndexesAll
{
  char *pTableName;
  char *pIndexName;
  char *pPartitionName;
} db2ReorgIndexesAll;

SQL_API_RC SQL_API_FN
  db2gReorg (
              db2Uint32 versionNumber,
              void * pReorgStruct,
              struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gReorgStruct
{
  db2Uint32 reorgType;
  db2Uint32 reorgFlags;
  db2int32 nodeListFlag;
  db2Uint32 numNodes;
  SQL_PDB_NODE_TYPE *pNodeList;
  union db2gReorgObject       reorgObject;
} db2gReorgStruct;

typedef SQL_STRUCTURE db2gReorgNodes
{
  SQL_PDB_NODE_TYPE nodeNum[SQL_PDB_MAX_NUM_NODE];
} db2gReorgNodes;

union db2gReorgObject
{
  struct db2gReorgTable            tableStruct;
  struct db2gReorgIndexesAll       indexesAllStruct;
};

typedef SQL_STRUCTURE db2gReorgTable
{
  db2Uint32 tableNameLen;
  char *pTableName;
  db2Uint32 orderByIndexLen;
  char *pOrderByIndex;
  db2Uint32 sysTempSpaceLen;
  char *pSysTempSpace;
  db2Uint32 longTempSpaceLen;
  char *pLongTempSpace;
  db2Uint32 partitionNameLen;
  char *pPartitionName;
} db2gReorgTable;

typedef SQL_STRUCTURE db2gReorgIndexesAll
{
```

```
  db2Uint32 tableNameLen;
  char *pTableName;
  db2Uint32 indexNameLen;
  char *pIndexName;
  db2Uint32 indexNameLen;
  char *pPartitionName;
} db2gReorgIndexesAll;
```

db2Reorg API 매개변수

versionNumber

입력. 두 번째 매개변수인 **pReorgStruct**로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pReorgStruct

입력. db2ReorgStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ReorgStruct 데이터 구조 매개변수

reorgType

입력. 재구성 유형을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의된)은 다음과 같습니다.

DB2REORG_OBJ_TABLE_OFFLINE

테이블을 오프라인에서 재구성합니다.

DB2REORG_OBJ_TABLE_INPLACE

테이블을 제 위치에서 재구성합니다.

DB2REORG_OBJ_INDEXESALL

모든 인덱스를 재구성합니다.

DB2REORG_OBJ_INDEX

하나의 인덱스를 재구성합니다.

DB2REORG_RECLAIM_EXTENTS

테이블 스페이스의 비어 있는 Extent를 재개하기 위해 다차원적으로 클러스터된(MDC) 테이블을 재구성합니다.

reorgFlags

입력. 재구성 옵션. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의된)은 다음과 같습니다.

DB2REORG_OPTION_NONE

디폴트 조치

DB2REORG_LONGLOB

긴 필드 및 로그를 재구성하며

DB2REORG_OBJ_TABLE_OFFLINE이 **reorgType**으로 지정된 경우에 사용됩니다. DB2REORG_RESETDICTIONARY 또는 DB2REORG_KEEPDICTIONARY 옵션도 지정한 경우, 옵션은 테이블 오브젝트 외에 테이블의 XML 스토리지 오브젝트에 적용됩니다.

DB2REORG_INDEXSCAN
인덱스 스캔의 활용을 다시 클러스터하며 DB2REORG_OBJ_TABLE_OFFLINE이 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_START_ONLINE
온라인 재구성을 시작하며 DB2REORG_OBJ_TABLE_INPLACE가 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_PAUSE_ONLINE
기존 온라인 재구성을 일시정지하며 DB2REORG_OBJ_TABLE_INPLACE가 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_STOP_ONLINE
기존 온라인 재구성을 중지하며 DB2REORG_OBJ_TABLE_INPLACE가 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_RESUME_ONLINE
일시정지된 온라인 재구성을 다시 시작하며 DB2REORG_OBJ_TABLE_INPLACE가 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_NOTRUNCATE_ONLINE
테이블 절단을 수행하지 않으며 DB2REORG_OBJ_TABLE_INPLACE가 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_ALLOW_NONE
테이블에 대한 읽기 쓰기 액세스가 없습니다. 이 매개변수는 DB2REORG_OBJ_TABLE_INPLACE가 **reorgType**으로 지정된 경우에 지원되지 않습니다.

DB2REORG_ALLOW_WRITE
테이블에 대한 읽기 쓰기 액세스를 허용합니다. 이 매개변수는 DB2REORG_OBJ_TABLE_OFFLINE이 **reorgType**으로 지정된 경우에 지원되지 않습니다.

DB2REORG_ALLOW_READ
테이블에 대한 읽기 액세스만 허용합니다.

DB2REORG_CLEANUP_NONE

정리가 필요하지 않으며 DB2REORG_OBJ_INDEXESALL 또는 DB2REORG_OBJ_INDEX가 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_CLEANUP_ALL

커미트된 의사 삭제된 키와 커미트된 의사 비어 있는 페이지를 정리하며 DB2REORG_OBJ_INDEXESALL 또는 DB2REORG_OBJ_INDEX가 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_CLEANUP_PAGES

커미트된 의사 비어 있는 페이지만 정리하며 의사 비어 있지 않은 페이지에서 의사 삭제된 키는 정리하지 않습니다.
DB2REORG_OBJ_INDEXESALL 또는 DB2REORG_OBJ_INDEX가 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_CONVERT_NONE

변환이 필요하지 않으며 DB2REORG_OBJ_INDEXESALL 또는 DB2REORG_OBJ_INDEX가 **reorgType**으로 지정된 경우에 사용됩니다.

DB2REORG_CONVERT

유형 2 인덱스로 변환하며
DB2REORG_OBJ_TABLE_INDEXESALL이 **reorgType**으로 지정된 경우에 사용됩니다. 유형 1 인덱스는 더 이상 지원되지 않으며 다음에 테이블에 액세스할 때 유형 2 인덱스로 변환됩니다. 결과적으로, 이 값은 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다.

DB2REORG_RESET_DICTIONARY

DB2REORG_LONGLOB 옵션도 지정한 경우,
DB2REORG_RESETDICTIONARY는 테이블의 XML 스토리지 오브젝트에도 적용됩니다. 테이블에 대한 COMPRESS 속성이 YES이면 새 압축 사전이 빌드됩니다. 재구성 중 처리된 모든 행은 이 새 사전을 사용하여 압축되기 쉽습니다. 이 사전은 오브젝트에서 이전 사전을 교체합니다. 테이블에 대한 COMPRESS 속성이 NO이고 테이블 오브젝트나 XML 스토리지 오브젝트에 기존 압축 사전이 있으면 REORG 처리는 사전을 제거하고 새로 재구성된 테이블의 모든 행은 압축되지 않은 형식이 됩니다. 이 매개변수는
DB2REORG_OBJ_TABLE_OFFLINE **reorgType**의 경우에만 지원됩니다.

DB2REORG_KEEP_DICTIONARY

DB2REORG_LONGLOB 키워드도 지정한 경우,

DB2REORG_KEEPDICTIONARY는 테이블 오브젝트와 테이블의 XML 스토리지 오브젝트에 적용됩니다. DB2REORG_LONGLOB를 지정하지 않은 경우 다음은 테이블 오브젝트에만 적용됩니다.

테이블의 COMPRESS 속성이 YES이고 사전이 존재하면 그 사전이 보존됩니다. 테이블의 COMPRESS 속성이 YES이고 사전이 존재하지 않으면 사전이 빌드됩니다. 이와 같은 경우 옵션 디폴트가 DB2REORG_RESET_DICTIONARY이기 때문입니다. 재구성으로 처리되는 모든 행이 압축됩니다. 테이블의 COMPRESS 속성이 NO이면 사전은 보유되고(사전이 있는 경우) 새로 재구성된 테이블의 모든 행은 압축되지 않은 형식이 됩니다. 이 매개변수는 DB2REORG_OBJ_TABLE_OFFLINE **reorgType**의 경우에만 지원됩니다.

nodeListFlag

입력. 재구성할 노드를 지정합니다. 유효한 값(`include` 디렉토리에 있는 `db2ApiDf` 헤더 파일에 정의된)은 다음과 같습니다.

DB2REORG_NODE_LIST

노드 목록 배열의 모든 노드에 제출합니다.

DB2REORG_ALL_NODES

데이터베이스 파티션 그룹의 모든 노드에 제출합니다.

DB2REORG_ALL_EXCEPT

노드 목록 매개변수에 지정된 노드를 제외한 모든 노드에 제출합니다.

numNodes

입력. 노드 목록 배열의 노드 수.

pNodeList

노드 번호 배열에 대한 포인터

reorgObject

입력. 재구성할 오브젝트의 유형을 지정합니다.

db2ReorgObject 통합 매개변수

tableStruct

테이블 재구성 옵션을 지정합니다.

indexesAllStruct

인덱스 재구성 옵션을 지정합니다.

db2ReorgTable 데이터 구조 매개변수

pTableName

입력. 재구성할 테이블의 이름을 지정합니다.

pOrderByIndex

입력. 테이블 순서를 지정할 인덱스를 지정합니다.

pSysTempSpace

입력. 임시 오브젝트가 작성되는 시스템 임시 테이블 스페이스를 지정합니다. REORG 명령은 컬럼이 테이블에 추가되고(즉, ALTER TABLE ADD COLUMN을 통해) 컬럼이 추가되기 전에 행이 삽입된 경우에 행을 확장할 수 있습니다. 파티션되지 않은 테이블의 경우, 이 매개변수는 새 테이블 오브젝트를 작성하기 위한 충분한 영역이 있는 테이블 스페이스를 지정해야 합니다. 파티션된 테이블에서는 한 번에 하나의 데이터 파티션이 재구성됩니다. 이 경우, 테이블의 가장 큰 데이터 파티션을 보유하기 위해 충분한 여유 공간이 있어야 합니다.

이 매개변수가 파티션되지 않은 테이블에 지정되지 않으면 테이블이 있는 테이블 스페이스가 사용됩니다. 파티션된 테이블에 이 매개변수를 지정하지 않으면, 각 데이터 파티션이 위치된 테이블 스페이스가 해당 데이터 파티션의 임시 스토리지에 사용됩니다. 각 데이터 파티션의 테이블 스페이스에는 데이터베이스 파티션의 사본을 보유하기에 충분한 여유 공간이 있어야 합니다.

pLongTempSpace

입력. 테이블 재구성 중에 긴 오브젝트(LONG VARCHAR 및 LOB 컬럼)를 작성하기 위한 임시 테이블 스페이스를 지정합니다. **pSysTempSpace** 매개변수를 지정하지 않은 경우, 이 매개변수는 무시됩니다. 이 매개변수를 지정하지 않았지만 **pSysTempSpace** 매개변수가 지정된 경우, DB2는 **pSysTempSpace** 매개변수에 지정된 테이블 스페이스에서 긴 데이터 오브젝트를 작성합니다(페이지 *크기*가 다르지 않은 경우).

페이지 *크기*가 다른데 **pSysTempSpace**는 지정하고 이 매개변수는 지정하지 않은 경우, DB2는 페이지 *크기*가 일치하는 기존 테이블 스페이스를 찾아서 긴 오브젝트를 작성하려고 합니다.

pPartitionName

입력. 재구성할 데이터 파티션의 이름을 지정합니다. MDC RECLAIM 옵션에만 지원됩니다(SQL2222N).

db2ReorgIndexesAll 데이터 구조 매개변수

pTableName

입력. 인덱스 재구성 대상 테이블의 이름을 지정합니다. DB2REORG_OBJ_INDEX를 **reorgType**으로 지정한 경우, **pTableName** 매개변수는 필요하지 않으며 NULL이 될 수 있습니다. 그러나 **pTableName** 매개변수를 지정한 경우 인덱스가 정의된 테이블이어야 합니다.

pIndexName

입력. 재구성할 인덱스의 이름을 지정합니다. 이 매개변수는 **reorgType** 매개

변수가 DB2REORG_OBJ_INDEX 값으로 설정된 경우에만 사용됩니다. 그렇지 않으면 **pIndexName** 매개변수를 NULL로 설정하십시오.

pPartitionName

입력. 해당 인덱스를 재구성할 데이터 파티션의 이름을 지정합니다. MDC RECLAIM 옵션에만 지원됩니다(SQL2222N).

db2gReorgTable 데이터 구조 특정 매개변수

tableNameLen

입력. **pTableName**의 길이(바이트)를 지정합니다.

orderByIndexLen

입력. **pOrderByIndex**의 길이(바이트)를 지정합니다.

sysTempSpaceLen

입력. **pSysTempSpace**의 길이(바이트)를 지정합니다.

longTempSpaceLen

입력. **pLongTempSpace**에 저장된 이름의 길이를 지정합니다.

partitionNameLen

입력. **pPartitionName**의 길이(바이트)를 지정합니다. MDC RECLAIM 옵션에만 지원됩니다(SQL2222N).

pPartitionName

입력. 재구성할 데이터 파티션의 이름을 지정합니다. MDC RECLAIM 옵션에만 지원됩니다(SQL2222N).

db2gReorgIndexesAll 데이터 구조 특정 매개변수

tableNameLen

입력. **pTableName**의 길이(바이트)를 지정합니다.

indexNameLen

입력. **pIndexName** 매개변수의 길이(바이트)를 지정합니다.

partitionNameLen

입력. **pPartitionName**의 길이(바이트)를 지정합니다. MDC RECLAIM 옵션에만 지원됩니다(SQL2222N).

pPartitionName

입력. 인덱스에 대한 데이터 파티션의 이름을 지정합니다. MDC RECLAIM 옵션에만 지원됩니다(SQL2222N).

사용 시 참고사항

- 테이블 데이터를 여러 번 수정하여 분할되고 클러스터되지 않으면 테이블 액세스, 인덱스 스캔 성능과 인덱스 페이지 프리페칭의 효과에 나쁜 영향을 줄 수 있습니다. 테

이블 또는 해당 인덱스가 재구성 후보인지 여부를 판별하려면 REORGCHK를 사용하십시오. Reorg 처리 중에 모든 작업이 커미트되고 모든 열린 커서가 닫힙니다. 테이블 또는 해당 인덱스를 재구성한 후, db2Runstats를 사용하여 통계를 갱신하고 sqlarbnd를 사용하여 테이블을 사용하는 패키지를 리바인드하십시오.

- 테이블 데이터가 몇 개의 노드에 분산되어 있는데 영향을 받는 노드 중에서 재구성에 실패하는 경우, 실패하는 노드에서만 재구성이 롤백됩니다. 테이블 재구성이 실패하면 임시 파일은 삭제하면 안됩니다. 데이터베이스 관리 프로그램은 이 파일을 사용하여 데이터베이스를 복구합니다.
- 테이블 재구성의 경우, 인덱스 이름을 지정하면 데이터베이스 관리 프로그램이 인덱스 순서에 따라 데이터를 재구성합니다. 성능을 최대화하려면 SQL 쿼리에서 자주 사용되는 인덱스를 지정하십시오. 인덱스 이름이 지정되지 않고 클러스터링 인덱스가 존재할 경우 데이터는 클러스터링 인덱스에 따라 정렬됩니다.
- 테이블의 PCTFREE 값은 페이지 당 지정된 여유 공간의 크기를 결정합니다. 이 값이 설정되어 있지 않으면 유틸리티는 각 페이지에서 가능한 많은 공간을 채웁니다.
- 테이블 재구성 후 테이블 스페이스 롤 포워드 복구를 완료하려면 데이터 및 LONG 테이블 스페이스가 롤 포워드 사용 가능 상태여야 합니다.
- 테이블에 COMPACT 옵션으로 정의되지 않은 LOB 컬럼이 있는 경우, 테이블 재구성 후 LOB DATA 스토리지 오브젝트가 훨씬 더 커질 수 있습니다. 이는 행이 재구성된 순서와 사용된 테이블 스페이스의 유형(SMS/DMS)의 결과가 될 수 있습니다.
- 다음 표는 테이블 및 Reorg 유형을 기초로 선택되는 디폴트 테이블 액세스를 나타냅니다.

표 8. 테이블 및 Reorg 유형을 기초로 선택되는 디폴트 테이블 액세스

디폴트 테이블 액세스에 영향을 줄 수 있는 **Reorg** 및 적용 가능 플래그의 유형		각 테이블 유형에 대해 선택되는 액세스 모드	
reorgType	**reorgFlags**(적용 가능한 경우)	파티션되지 않은 테이블	파티션된 테이블
DB2REORG_OBJ_TABLE_OFFLINE		DB2REORG_ALLOW_READ	DB2REORG_ALLOW_NONE
DB2REORG_OBJ_TABLE_OFFLINE		DB2REORG_ALLOW_READ	DB2REORG_ALLOW_READ (*)
DB2REORG_OBJ_TABLE_INPLACE		DB2REORG_ALLOW_WRITE	N/A
DB2REORG_OBJ_INDEXESALL		DB2REORG_ALLOW_READ	DB2REORG_ALLOW_NONE
DB2REORG_OBJ_INDEXESALL		N/A	DB2REORG_ALLOW_READ
DB2REORG_OBJ_INDEXESALL	DB2REORG_CLEANUP_ ALL, DB2REORG_CLEANUP_PAGES	DB2REORG_ALLOW_READ	DB2REORG_ALLOW_READ
DB2REORG_OBJ_INDEX		N/A	DB2REORG_ALLOW_READ

표 8. 테이블 및 Reorg 유형을 기초로 선택되는 디폴트 테이블 액세스 (계속)

디폴트 테이블 액세스에 영향을 줄 수 있는 **Reorg** 및 적용 가능 플래그의 유형		각 테이블 유형에 대해 선택되는 액세스 모드	
reorgType	**reorgFlags**(적용 가능한 경우)	파티션되지 않은 테이블	파티션된 테이블
DB2REORG_OBJ_INDEX	DB2REORG_CLEANUP_ ALL, DB2REORG_CLEANUP_ PAGES	N/A	DB2REORG_ALLOW_ READ

(*) 파티션되지 않은 인덱스가 없는 경우 DB2REORG_ALLOW_READ. 그렇지 않으면 DB2REORG_ALLOW_NONEN/A: 현재 지원되지 않으므로 적용할 수 없습니다.

주: 일부 액세스 모드는 특정 유형의 테이블 또는 인덱스에 대해 지원되지 않을 수도 있습니다. 이와 같은 경우와 가능한 경우에, 최소의 제한 액세스 모드가 사용됩니다. (대부분의 제한 액세스 모드는 DB2REORG_ALLOW_NONE이며 그 다음에는 DB2REORG_ALLOW_READ이고, 최소의 제한은 DB2REORG_ALLOW_WRITE입니다.) 기존 테이블 또는 인덱스 유형에 대한 지원이 변경되거나 새 테이블 또는 인덱스 유형이 제공되므로, 디폴트도 한층 제한되는 액세스 모드에서 덜 제한되는 모드로 변경될 수 있습니다. 디폴트로 선택된 최소의 제한 모드는 **reorgType**이 DB2REORG_OBJ_TABLE_INPLACE가 아닐 때 DB2REORG_ALLOW_READ를 넘지 못합니다. DB2REORG_ALLOW_NONE, DB2REORG_ALLOW_READ 또는 DB2REORG_ALLOW_WRITE 플래그 중 어느 것도 지정되지 않은 경우에 디폴트 액세스 모드가 선택됩니다.

- 인덱스를 재구성할 때, 다른 트랜잭션이 테이블에 대해 읽기 전용 또는 읽기-쓰기 액세스할 수 있도록 허용하려면 액세스 옵션을 사용하십시오.
- 인덱스 재빌드가 필요하여 읽기 허용 또는 쓰기 허용 액세스가 있는 인덱스 재구성이 실패하는 경우, 재구성은 액세스를 허용하지 않도록 전환된 후에 계속됩니다. 메시지는 액세스 모드에서의 변경에 대한 진단 로그 및 관리 통지 로그 둘 다에 기록됩니다. 파티션된 테이블에 대해 DB2REORG_OBJ_INDEX가 지정된 경우, 재빌드해야 하는 인덱스는 오프라인에서 재빌드되고, 지정된 인덱스는 재구성됩니다(재빌드되지 않은 것으로 가정하고). 이와 같은 재구성에는 지정된 액세스 모드가 사용됩니다(즉, 액세스 모드는 처리 중에 변경되지 않음). 메시지는 오프라인에서 재빌드되는 인덱스에 대한 진단 로그 및 관리 통지 로그에 기록됩니다.
- Non-inplace 테이블 재구성의 경우, DB2REORG_RESET_DICTIONARY 또는 DB2REORG_KEEP_DICTIONARY를 지정하지 않으면 디폴트는 DB2REORG_KEEP_DICTIONARY입니다.
- 액세스가 없는 인덱스 재구성이 실패하면, 일부 또는 모든 인덱스는 사용할 수 없고 다음 테이블 액세스에서 재빌드됩니다.
- 이 API는 다음과 함께 사용할 수 없습니다.
  - 인덱스 확장을 기초로 하는 뷰 또는 인덱스
  - 선언된 임시 테이블

– 작성된 임시 테이블

제 67 장 db2ResetAlertCfg - Health 표시기의 경보 구성 재설정

특정 오브젝트의 Health 표시기 설정값을 해당 오브젝트 유형의 현재 디폴트값으로 재설정하거나 오브젝트 유형의 현재 디폴트 Health 표시기 설정값을 설치 디폴트값으로 재설정합니다.

중요사항: Health Monitor가 버전 9.7에서 사용되지 않으므로
이 명령 또는 API는 사용되지 않으며 이후 릴리스에서 제거될 수 있습니다. 자세한 정보는 *버전 9.7의 새로운 내용* 책에 있는 『Health Monitor는 사용되지 않음』 주제를 참조하십시오.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysmaint
- sysctrl

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2ResetAlertCfg (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ResetAlertCfgData
{
   db2Uint32 iObjType;
   char *piObjName;
   char *piDbName;
   db2Uint32 iIndicatorID;
} db2ResetAlertCfgData;
```

db2ResetAlertCfg API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2ResetAlertCfgData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ResetAlertCfgData 데이터 구조 매개변수

iObjType

입력. 구성이 재설정되어야 하는 오브젝트 유형을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

- DB2ALERTCFG_OBJTYPE_DBM
- DB2ALERTCFG_OBJTYPE_DATABASES
- DB2ALERTCFG_OBJTYPE_TABLESPACES
- DB2ALERTCFG_OBJTYPE_TS_CONTAINERS
- DB2ALERTCFG_OBJTYPE_DATABASE
- DB2ALERTCFG_OBJTYPE_TABLESPACE
- DB2ALERTCFG_OBJTYPE_TS_CONTAINER

piObjName

입력. 오브젝트 유형 iObjType이 DB2ALERTCFG_OBJTYPE_TS_CONTAINER 또는 DB2ALERTCFG_OBJTYPE_TABLESPACE로 설정된 경우 테이블 스페이스 또는 테이블 스페이스 컨테이너의 이름. 테이블 스페이스 컨테이너 이름은 <tablespace-numericalID>.<tablespace-container-name>으로 정의됩니다.

piDbname

입력. 오브젝트 유형 iObjType이 DB2ALERTCFG_OBJTYPE_TS_CONTAINER, DB2ALERTCFG_OBJTYPE_TABLESPACE 및 DB2ALERTCFG_OBJTYPE_DATABASE로 설정된 경우 구성이 재설정되어야 하는 데이터베이스의 별명 이름.

iIndicatorID

입력. 구성 재설정이 적용되는 Health 표시기

사용 시 참고사항

오브젝트 유형의 현재 디폴트는 iObjType이 DB2ALERTCFG_OBJTYPE_DBM, DB2ALERTCFG_OBJTYPE_DATABASES, DB2ALERTCFG_OBJTYPE_TABLESPACES, DB2ALERTCFG_OBJTYPE_TS_CONTAINERS이거나 piObjName 및 piDbName 모두 NULL인 경우 재설정됩니다. iObjType이

DB2ALERTCFG_OBJTYPE_DATABASE,
DB2ALERTCFG_OBJTYPE_TABLESPACE,
DB2ALERTCFG_OBJTYPE_TS_CONTAINER이고 piDbName 및 piObjName(데이터베이스에는 필요하지 않음)이 지정된 경우 해당 특정 오브젝트의 현재 설정값이 재설정됩니다.

제 68 장 db2ResetMonitor - 데이터베이스 시스템 모니터 데이터 재설정

지정한 데이터베이스 또는 사용 중인 모든 데이터베이스의 데이터베이스 시스템 모니터 데이터를 호출 중인 응용프로그램에 대해 재설정합니다.

범위

이 API는 인스턴스의 지정된 데이터베이스 파티션 또는 인스턴스의 모든 데이터베이스 파티션에 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

리모트 인스턴스(또는 다른 로컬 인스턴스)의 모니터 스위치를 재설정하려면 우선 해당 인스턴스에 접속해야 합니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2ResetMonitor (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2ResetMonitorData
{
   db2Uint32 iResetAll;
   char *piDbAlias;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
} db2ResetMonitorData;

SQL_API_RC SQL_API_FN
```

```
  db2gResetMonitor (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gResetMonitorData
{
   db2Uint32 iResetAll;
   char *piDbAlias;
   db2Uint32 iDbAliasLength;
   db2Uint32 iVersion;
   db2int32 iNodeNumber;
} db2gResetMonitorData;
```

db2ResetMonitor API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2ResetMonitorData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2ResetMonitorData 데이터 구조 매개변수

iResetAll

입력. 재설정 플래그.

piDbAlias

입력. 데이터베이스 별명의 포인터

iVersion

입력. 수집할 데이터베이스 모니터 데이터의 버전 ID. 데이터베이스 모니터는 요청한 버전에서 사용할 수 있는 데이터만 리턴합니다. 이 매개변수를 다음 기호 상수 중 하나로 설정하십시오.

- SQLM_DBMON_VERSION1
- SQLM_DBMON_VERSION2
- SQLM_DBMON_VERSION5
- SQLM_DBMON_VERSION5_2
- SQLM_DBMON_VERSION6
- SQLM_DBMON_VERSION7
- SQLM_DBMON_VERSION8
- SQLM_DBMON_VERSION9
- SQLM_DBMON_VERSION9_5

주: SQLM_DBMON_VERSION1을 버전으로 지정한 경우 API는 리모트로 실행할 수 없습니다.

주: 상수 SQLM_DBMON_VERSION5_2 및 이전은 더 이상 사용되지 않으며 DB2의 추후 릴리스에서 제거될 수 있습니다.

iNodeNumber

입력. 요청을 보내는 데이터베이스 파티션 서버. 요청은 이 값을 사용하여 현재 데이터베이스 파티션 서버, 모든 데이터베이스 파티션 서버 또는 사용자가 지정한 데이터베이스 파티션 서버에 대해 처리됩니다. 가능한 값은 다음과 같습니다.

- SQLM_CURRENT_NODE
- SQLM_ALL_NODES
- 노드 값

주: 독립형 인스턴스의 경우 SQLM_CURRENT_NODE 값을 사용해야 합니다.

db2gResetMonitorData 데이터 구조 특정 매개변수

iDbAliasLength

입력. piDbAlias 매개변수 길이를 바이트 단위로 지정합니다.

사용 시 참고사항

각 프로세스(접속)에는 모니터 데이터의 자체 개인용 뷰가 있습니다. 임의 사용자가 모니터 스위치를 재설정하거나 꺼도 다른 사용자에게는 영향이 없습니다. 응용프로그램이 임의 데이터베이스 모니터 함수를 처음 호출하는 경우 데이터베이스 관리 프로그램 구성 파일에서 디폴트 스위치 설정을 상속합니다. 이 설정은 db2MonitorSwitches - 모니터 스위치 가져오기/갱신으로 겹쳐쓸 수 있습니다.

모든 사용 중인 데이터베이스가 재설정되면 일부 데이터베이스 관리 프로그램 정보가 리턴된 데이터의 일관성을 유지하기 위해 재설정됩니다.

이 API는 선택적으로 특정 데이터 항목을 재설정하거나 특정 모니터 그룹을 재설정하는 데 사용할 수 없습니다. 그러나 db2MonitorSwitches - 모니터 스위치 가져오기/갱신을 사용하여 특정 그룹은 해당 스위치를 끈 다음 켜서 재설정할 수 있습니다.

제 69 장 db2Restore - 데이터베이스 또는 테이블 스페이스 리스토어

db2Backup API를 사용하여 백업한 손상된 데이터베이스를 재작성합니다. 리스토어된 데이터베이스 백업 사본이 작성된 상태와 동일한 상태가 됩니다.

이 유틸리티를 사용하여 다음을 수행할 수도 있습니다.

- 새 데이터베이스로 리스토어할 수 있을 뿐만 아니라 백업 이미지의 데이터베이스 이름과 다른 이름으로 데이터베이스를 리스토어합니다(백업 이미지 데이터베이스 이름이 동일해야 하는 스냅샷 리스토어는 예외).
- 이전 2 릴리스에서 작성된 DB2 데이터베이스를 리스토어합니다.
- 테이블 스페이스 레벨 백업에서 리스토어하거나 테이블 스페이스를 데이터베이스 백업 이미지에서 리스토어합니다.

범위

이 API는 호출된 데이터베이스 파티션에만 영향을 줍니다.

권한 부여

기존 데이터베이스를 리스토어하려면 다음 중 하나가 있어야 합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*

새 데이터베이스로 리스토어하려면 다음 중 하나가 있어야 합니다.

- *sysadm*
- *sysctrl*

필수 연결

데이터베이스 - 기존 데이터베이스로 리스토어하려는 경우. 이 API는 지정한 데이터베이스에 자동으로 연결하고 리스토어 조작이 완료되면 연결을 해제합니다.

인스턴스 및 *데이터베이스* - 새 데이터베이스로 리스토어하려는 경우. 데이터베이스를 작성하려면 인스턴스에 접속해야 합니다.

스냅샷 리스토어의 경우 *인스턴스* 및 *데이터베이스*에 연결해야 합니다.

현재 인스턴스와 다른 인스턴스에서 새 데이터베이스를 리스토어하려면(**DB2INSTANCE** 환경 변수의 값으로 정의) 우선 새 데이터베이스가 상주할 인스턴스에 접속해야 합니다.

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Restore (
       db2Uint32 versionNumber,
       void * pDB2RestoreStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RestoreStruct
{
  char *piSourceDBAlias;
  char *piTargetDBAlias;
  char oApplicationId[SQLU_APPLID_LEN+1];
  char *piTimestamp;
  char *piTargetDBPath;
  char *piReportFile;
  struct db2TablespaceStruct *piTablespaceList;
  struct db2MediaListStruct *piMediaList;
  char *piUsername;
  char *piPassword;
  char *piNewLogPath;
  void *piVendorOptions;
  db2Uint32 iVendorOptionsSize;
  db2Uint32 iParallelism;
  db2Uint32 iBufferSize;
  db2Uint32 iNumBuffers;
  db2Uint32 iCallerAction;
  db2Uint32 iOptions;
  char *piComprLibrary;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
  char *piLogTarget;
  struct db2StoragePathsStruct *piStoragePaths;
  char *piRedirectScript;
} db2RestoreStruct;

typedef SQL_STRUCTURE db2TablespaceStruct
{
  char                         **tablespaces;
  db2Uint32 numTablespaces;
} db2TablespaceStruct;

typedef SQL_STRUCTURE db2MediaListStruct
{
  char                         **locations;
  db2Uint32 numLocations;
  char locationType;
} db2MediaListStruct;

typedef SQL_STRUCTURE db2StoragePathsStruct
{
```

```
  char                         **storagePaths;
  db2Uint32 numStoragePaths;
} db2StoragePathsStruct;

SQL_API_RC SQL_API_FN
  db2gRestore (
       db2Uint32 versionNumber,
       void * pDB2gRestoreStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRestoreStruct
{
  char *piSourceDBAlias;
  db2Uint32 iSourceDBAliasLen;
  char *piTargetDBAlias;
  db2Uint32 iTargetDBAliasLen;
  char *poApplicationId;
  db2Uint32 iApplicationIdLen;
  char *piTimestamp;
  db2Uint32 iTimestampLen;
  char *piTargetDBPath;
  db2Uint32 iTargetDBPathLen;
  char *piReportFile;
  db2Uint32 iReportFileLen;
  struct db2gTablespaceStruct *piTablespaceList;
  struct db2gMediaListStruct *piMediaList;
  char *piUsername;
  db2Uint32 iUsernameLen;
  char *piPassword;
  db2Uint32 iPasswordLen;
  char *piNewLogPath;
  db2Uint32 iNewLogPathLen;
  void *piVendorOptions;
  db2Uint32 iVendorOptionsSize;
  db2Uint32 iParallelism;
  db2Uint32 iBufferSize;
  db2Uint32 iNumBuffers;
  db2Uint32 iCallerAction;
  db2Uint32 iOptions;
  char *piComprLibrary;
  db2Uint32 iComprLibraryLen;
  void *piComprOptions;
  db2Uint32 iComprOptionsSize;
  char *piLogTarget;
  db2Uint32 iLogTargetLen;
  struct db2gStoragePathsStruct *piStoragePaths;
  char *piRedirectScript;
  db2Uint32 iRedirectScriptLen;
} db2gRestoreStruct;

typedef SQL_STRUCTURE db2gTablespaceStruct
{
  struct db2Char *tablespaces;
  db2Uint32 numTablespaces;
} db2gTablespaceStruct;

typedef SQL_STRUCTURE db2gMediaListStruct
{
```

```
  struct db2Char *locations;
  db2Uint32 numLocations;
  char locationType;
} db2gMediaListStruct;

typedef SQL_STRUCTURE db2gStoragePathsStruct
{
  struct db2Char *storagePaths;
  db2Uint32 numStoragePaths;
} db2gStoragePathsStruct;

typedef SQL_STRUCTURE db2Char
{
   char *pioData;
   db2Uint32 iLength;
   db2Uint32 oLength;
} db2Char;
```

db2Restore API 매개변수

versionNumber

입력. 두 번째 매개변수인 **pDB2RestoreStruct**로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2RestoreStruct

입력. db2RestoreStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2RestoreStruct 데이터 구조 매개변수

piSourceDBAlias

입력. 소스 데이터베이스 백업 이미지의 데이터베이스 별명을 포함하는 문자열.

piTargetDBAlias

입력. 목표 데이터베이스 별명을 포함하는 문자열. 이 매개변수가 널(NULL)인 경우에는 **piSourceDBAlias** 매개변수 값이 사용됩니다.

oApplicationId

출력. API는 에이전트 서비스 응용프로그램을 식별하는 문자열을 리턴합니다. 데이터베이스 모니터를 사용하는 백업 조작 진행 상태에 대한 정보를 가져오는 데 사용할 수 있습니다.

piTimestamp

입력. 백업 이미지의 시간소인을 나타내는 문자열. 지정한 소스에 백업 이미지가 한 개만 있는 경우에 이 필드는 선택적입니다.

piTargetDBPath

입력. 서버에서 목표 데이터베이스 디렉토리의 상대 이름 또는 완전한 이름을

포함하는 문자열. 리스토어된 백업을 위해 새 데이터베이스가 작성되는 경우에 사용되며 이 외의 경우에는 사용되지 않습니다.

piReportFile

입력. 파일 이름을 지정하는 경우에는 완전한 이름으로 지정해야 합니다.

주: 이 매개변수는 사용되지 않지만 정의는 되어 있습니다.

piTablespaceList

입력. 리스토어되는 테이블 스페이스 목록. 데이터베이스 또는 테이블 스페이스 백업 이미지에서 테이블 스페이스 서브세트를 리스토어하는 경우에 사용됩니다. 재빌드하는 경우, 데이터베이스를 재빌드하는 데 사용된 테이블 스페이스의 포함 목록 또는 제외 목록이 될 수 있습니다. DB2TablespaceStruct 구조를 참조하십시오. 다음과 같은 제한사항이 적용됩니다.

- 데이터베이스는 복구 가능해야 합니다(재빌드가 아닌 경우만 해당). 즉, 로그 유지 또는 userexit가 사용 가능해야 합니다.
- 리스토어 중인 데이터베이스는 백업 이미지를 작성할 때 사용한 데이터베이스와 동일해야 합니다. 즉, 테이블 스페이스는 테이블 스페이스 리스토어 함수를 통해 데이터베이스에 추가할 수 없습니다.
- 롤 포워드 유틸리티를 사용하여 파티션된 데이터베이스 환경에 리스토어된 테이블 스페이스가 동일한 테이블 스페이스를 포함하는 다른 데이터베이스 파티션과 동기화됩니다. 테이블 스페이스 리스토어 조작이 요청되고 **piTablespaceList**가 NULL인 경우 리스토어 유틸리티는 백업 이미지의 모든 테이블 스페이스를 리스토어하려고 합니다.
- 백업된 이후에 이름이 변경된 테이블 스페이스를 리스토어하는 경우 새 테이블 스페이스 이름을 리스토어 명령에 사용해야 합니다. 이전 테이블 스페이스 이름을 사용하면 찾을 수 없게 됩니다.
- 재빌드인 경우 목록은 다음과 같은 5개의 재빌드 유형 중 3개로 지정되어야 합니다. DB2RESTORE_ALL_TBSP_IN_DB_EXC, DB2RESTORE_ALL_TBSP_IN_IMG_EXC 및 DB2RESTORE_ALL_TBSP_IN_LIST.

piMediaList

입력. 백업 이미지의 소스 미디어.

자세한 정보는 아래 db2MediaListStruct 구조를 참조하십시오.

piUsername

입력. 연결 시도 시에 사용되는 사용자 이름이 포함된 문자열. NULL일 수 있습니다.

piPassword

입력. 사용자 이름에 사용되는 암호가 포함된 문자열. NULL일 수 있습니다.

piNewLogPath

입력. 리스토어 완료 후에 로깅에 사용되는 경로를 나타내는 문자열. 이 필드가 널(NULL)인 경우 디폴트 로그 경로가 사용됩니다.

piVendorOptions

입력. 응용프로그램의 정보를 벤더 기능에 전달하는 데 사용됩니다. 이 데이터 구조는 플랫해야 합니다. 즉, 간접 레벨이 지원되지 않습니다. 이 데이터에 대해서는 바이트 리버설이 수행되지 않고 코드 페이지가 점검되지 않습니다.

iVendorOptionsSize

입력. **piVendorOptions** 매개변수의 길이(바이트 단위)로 65535바이트를 초과할 수 없습니다.

iParallelism

입력. 병렬 처리 수준(버퍼 조작 수). 최소는 1이며 최대는 1024입니다.

iBufferSize

입력. 4KB 할당 단위(페이지)로 버퍼 *크기*를 백업하십시오. 최소 단위는 8입니다. 리스토어에 입력한 *크기*는 백업 이미지 작성에 사용된 버퍼 *크기*와 같거나 정수 배수여야 합니다.

iNumBuffers

입력. 사용된 리스토어 버퍼 수를 지정합니다.

iCallerAction

입력. 수행할 조치를 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

- DB2RESTORE_RESTORE - 리스토어 조작을 시작합니다.
- DB2RESTORE_NOINTERRUPT - 리스토어를 시작합니다. 리스토어가 자동으로 실행되며 일반적으로 사용자가 개입해야 하는 시나리오가 호출자에게 우선 리턴되지 않고 시도되거나 오류를 생성하도록 지정합니다. 예를 들어, 이 호출자 조치를 사용하면 리스토어에 필요한 모든 미디어가 마운트된 것을 알고 있는 경우 유틸리티 프롬프트가 필요하지 않습니다.
- DB2RESTORE_CONTINUE - 유틸리티에서 요청된 일부 조치(예: 새 테이프 마운트)를 수행한 후에 리스토어가 계속됩니다.
- DB2RESTORE_TERMINATE - 유틸리티에서 요청된 일부 조치 수행이 실패한 후에는 리스토어를 종료합니다.
- DB2RESTORE_DEVICE_TERMINATE - 리스토어에서 사용되는 디바이스 목록에서 특정 디바이스를 제거합니다. 특정 디바이스가 해당 입력에서 모두 사용된 경우 리스토어는 호출자에 경고를 리턴합니다. 이 호출자 조치로 리스토어를 다시 호출하여 경고를 생성한 디바이스를 사용 중인 디바이스 목록에서 제거하십시오.

- DB2RESTORE_PARM_CHK - 리스토어를 수행하지 않고 매개변수의 유효성을 확인하는 데 사용됩니다. 이 옵션은 호출이 리턴된 후에도 데이터베이스 연결을 종료하지 않습니다. 이 호출이 성공적으로 리턴되면 리스토어를 계속하기 위해 **iCallerAction** 매개변수를 DB2RESTORE_CONTINUE 값으로 설정하여 이 API에 대한 다른 호출을 실행합니다.
- DB2RESTORE_PARM_CHK_ONLY - 리스토어를 수행하지 않고 매개변수의 유효성을 확인하는 데 사용됩니다. 이 호출이 리턴되기 전에 이 호출로 작성된 데이터베이스 연결이 종료되며 연속된 호출은 필요하지 않습니다.
- DB2RESTORE_TERMINATE_INCRE - 완료 이전에 증분 리스토어 조작을 종료합니다.
- DB2RESTORE_RESTORE_STORDEF - 초기 호출. 테이블 스페이스 컨테이너를 재정의해야 합니다.
- DB2RESTORE_STORDEF_NOINTERRUPT - 초기 호출. 리스토어가 인터럽트되지 않고 실행됩니다. 테이블 스페이스 컨테이너를 재정의해야 합니다.

iOptions

입력. 리스토어 등록 정보의 비트맵. **iOptions** 값을 작성하기 위해 비트 방향의 OR 연산자를 사용하여 옵션이 조합됩니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

- DB2RESTORE_OFFLINE - 오프라인 리스토어 조작을 수행합니다.
- DB2RESTORE_ONLINE - 온라인 리스토어 조작을 수행합니다.
- DB2RESTORE_DB - 데이터베이스에서 모든 테이블 스페이스를 리스토어합니다. 오프라인으로 실행해야 합니다.
- DB2RESTORE_TABLESPACE - 백업 이미지에서 **piTablespaceList** 매개변수에 나열된 테이블 스페이스만 리스토어합니다. 온라인 또는 오프라인으로 수행할 수 있습니다.
- DB2RESTORE_HISTORY - 실행기록 파일만 리스토어합니다.
- DB2RESTORE_COMPR_LIB - 압축 라이브러리가 리스토어되는 것을 나타냅니다. 이 옵션은 다른 유형의 리스토어 프로세스와 같이는 사용할 수 없습니다. 오브젝트가 백업 이미지에 있는 경우 데이터베이스 디렉토리로 리스토어됩니다. 오브젝트가 백업 이미지에 없는 경우 리스토어 조작이 실패합니다.
- DB2RESTORE_LOGS - 백업 이미지에 포함된 로그 파일 세트만 리스토어되는 것을 나타냅니다. 백업 이미지에 로그 파일이 없는 경우 리스토어 조작이 실패합니다. 이 옵션을 지정하는 경우 **piLogTarget** 매개변수도 지정해야 합니다.

- DB2RESTORE_INCREMENTAL - 수동의 누적된 리스토어 조작을 수행합니다.
- DB2RESTORE_AUTOMATIC - 자동으로 누적된(증분식) 리스토어 조작을 수행합니다. DB2RESTORE_INCREMENTAL도 같이 지정해야 합니다.
- DB2RESTORE_ROLLFWD - 데이터베이스가 성공적으로 리스토어된 후에 롤 포워드 보류 상태로 설정됩니다.
- DB2RESTORE_NOROLLFWD - 데이터베이스가 성공적으로 리스토어된 후에 롤 포워드 보류 상태로 설정하지 않습니다. 온라인으로 수행된 백업이나 테이블 스페이스 레벨 리스토어에 대해서는 지정할 수 없습니다. 리스토어가 성공한 후에 데이터베이스가 롤 포워드 보류 상태인 경우 db2Rollforward API는 데이터베이스를 사용하기 전에 호출해야 합니다.
- DB2RESTORE_GENERATE_SCRIPT - 경로 재지정 리스토어 수행에 사용할 수 있는 스크립트를 작성합니다. **piRedirectScript**에는 유효한 파일 이름이 포함되어야 합니다. **iCallerAction**은 DB2RESTORE_RESTORE_STORDEF 또는 DB2RESTORE_STORDEF_NOINTERRUPT여야 합니다.

다음 값은 재빌드 조작에만 사용해야 합니다.

- DB2RESTORE_ALL_TBSP_IN_DB - 이미지 리스토어 시에 데이터베이스에 인식된 모든 테이블 스페이스와 같이 데이터베이스를 리스토어합니다. 이 재빌드는 데이터베이스가 이미 있는 경우 겹쳐씁니다.
- DB2RESTORE_ALL_TBSP_IN_DB_EXC - **piTablespaceList** 매개변수로 지시된 목록에 지정된 이미지를 제외하고 이미지 리스토어 시에 데이터베이스에 인식된 모든 테이블 스페이스와 같이 데이터베이스를 리스토어합니다. 이 재빌드는 데이터베이스가 이미 있는 경우 겹쳐씁니다.
- DB2RESTORE_ALL_TBSP_IN_IMG - 리스토어 중인 이미지의 테이블 스페이스만 데이터베이스와 같이 리스토어합니다. 이 재빌드는 데이터베이스가 이미 있는 경우 겹쳐씁니다.
- DB2RESTORE_ALL_TBSP_IN_IMG_EXC - **piTablespaceList** 매개변수로 지시된 목록에 있는 테이블 스페이스는 제외하고 리스토어 중인 이미지에 있는 테이블 스페이스와 같이 데이터베이스를 리스토어합니다. 이 재빌드는 데이터베이스가 이미 있는 경우 겹쳐씁니다.
- DB2RESTORE_ALL_TBSP_IN_LIST - **piTablespaceList** 매개변수로 지시된 목록에 지정된 테이블 스페이스만 포함하여 데이터베이스를 리스토어합니다. 이 재빌드는 데이터베이스가 이미 있는 경우 겹쳐씁니다.

참고: 백업 이미지가 복구 가능한 데이터베이스인 WITHOUT ROLLING FORWARD(DB2RESTORE_NOROLLFWD)는 위의 재빌드 조치에 전혀 지정할 수 없습니다.

piComprLibrary

입력. 이미지가 압축된 경우 백업 이미지 압축 해제에 사용된 외부 라이브러리 이름을 나타냅니다. 이름은 서버의 파일을 나타내는 완전한 경로여야 합니다. 값이 NULL 포인터 또는 빈 문자열의 포인터인 경우 DB2 데이터베이스 시스템은 이미지에 저장된 라이브러리를 사용하려고 합니다. 백업이 압축되지 않은 경우 이 매개변수 값은 무시됩니다. 지정한 라이브러리를 찾을 수 없는 경우 리스토어 조작이 실패합니다.

piComprOptions

입력. 이 API 매개변수는 압축 해제 라이브러리의 *초기화* 루틴으로 전달되는 2진 데이터 블록을 설명합니다. DB2 데이터베이스 시스템은 이 문자열을 클라이언트에서 서버로 직접 전달하여 바이트 리버설이나 코드 페이지 변환에 관련된 모든 문제는 압축 라이브러리가 처리하도록 합니다. 데이터 블록의 첫 번째 문자가 '@'일 경우, 데이터의 나머지 부분은 서버에 있는 파일 이름으로 해석합니다. 그런 다음 DB2데이터베이스 시스템이 **piComprOptions** 및 **iComprOptionsSize**의 컨텐츠를 이 파일의 컨텐츠와 *크기*로 교체하고 이 새 값을 *초기화* 루틴에 전달합니다.

iComprOptionsSize

입력. **piComprOptions**으로 전달된 데이터 블록의 *크기*를 나타내는 4바이트의 부호없는 정수. **piComprOptions** 값이 NULL 포인터인 경우에만 **iComprOptionsSize**가 0이어야 합니다.

piLogTarget

입력. 백업 이미지에서 로그 파일을 추출하는 대상 디렉토리로 사용되어야 하는 데이터베이스 서버의 디렉토리 절대 경로를 지정합니다. 이 매개변수를 지정하는 경우 백업 이미지에 포함된 모든 로그 파일이 대상 디렉토리에 추출됩니다. 이 매개변수를 지정하지 않으면 백업 이미지에 포함된 로그 파일은 추출되지 않습니다. 백업 이미지에서 로그 파일만 추출하려면 DB2RESTORE_LOGS 값을 **iOptions** 매개변수에 전달해야 합니다.

스냅샷 리스토어의 경우 다음 중 하나를 지정해야 합니다.

- DB2RESTORE_LOGTARGET_INCLUDE ″INCLUDE″

  스냅샷 이미지에서 로그 디렉토리 볼륨을 리스토어합니다. 이 옵션을 지정하고 백업 이미지에 로그 디렉토리가 포함된 경우 해당 디렉토리가 리스토어됩니다. 디스크의 기존 로그 디렉토리 및 로그 파일은 백업 이미지의 로그 디렉토리와 충돌하지 않으면 그대로 유지됩니다. 디스크의 기존 로그 디렉토리가 백업 이미지의 로그 디렉토리와 충돌하는 경우에는 오류가 리턴됩니다.

- DB2RESTORE_LOGTARGET_EXCLUDE ″EXCLUDE″

  로그 디렉토리 볼륨을 리스토어하지 않습니다. 이 옵션을 지정하면 로그 디렉토리가 백업 이미지에서 리스토어되지 않습니다. 디스크의 기존 로그 디렉

토리 및 로그 파일은 백업 이미지의 로그 디렉토리와 충돌하지 않으면 그대로 유지됩니다. 이로 인해 데이터베이스에 속한 경로가 리스토어되고 로그 디렉토리가 내재적으로 리스토어되어 로그 디렉토리가 겹쳐써지면 오류가 리턴됩니다.

- DB2RESTORE_LOGTARGET_INCFORCE ″INCLUDE FORCE″

  스냅샷 이미지를 리스토어할 때 기존 로그 디렉토리가 겹쳐쓰여지고 교체됩니다. 이 옵션을 지정하고 백업 이미지에 로그 디렉토리가 포함된 경우 해당 디렉토리가 리스토어됩니다. 디스크의 기존 로그 디렉토리 및 로그 파일은 백업 이미지의 로그 디렉토리와 충돌하지 않으면 그대로 유지됩니다. 디스크의 기존 로그 디렉토리가 백업 이미지의 로그 디렉토리와 충돌하는 경우에는 백업 이미지의 디렉토리로 겹쳐써집니다.

- DB2RESTORE_LOGTARGET_EXCFORCE ″EXCLUDE FORCE″

  스냅샷 이미지를 리스토어할 때 기존 로그 디렉토리가 겹쳐쓰여지고 교체됩니다. 이 옵션을 지정하면 로그 디렉토리가 백업 이미지에서 리스토어되지 않습니다. 디스크의 기존 로그 디렉토리 및 로그 파일은 백업 이미지의 로그 디렉토리와 충돌하지 않으면 그대로 유지됩니다. 이로 인해 데이터베이스에 속한 경로가 리스토어되고 로그 디렉토리가 내재적으로 리스토어되어 로그 디렉토리가 겹쳐써지면 리스토어가 계속되고 충돌하는 로그 디렉토리가 겹쳐써집니다.

여기서 DB2RESTORE_LOGTARGET_EXCLUDE가 디폴트입니다.

piStoragePaths

입력. 자동 스토리지에 사용되는 스토리지 경로 목록을 설명하는 필드가 포함된 구조. 데이터베이스에 대해 자동 스토리지가 사용 가능하지 않은 경우에는 NULL로 설정하십시오.

piRedirectScript

입력. 클라이언트 측에 작성되는 경로 재지정 리스토어 스크립트의 파일 이름. 파일 이름은 상대 또는 절대로 지정할 수 있습니다. **iOptions** 필드에는 DB2RESTORE_GENERATE_SCRIPT 비트 세트가 포함되어야 합니다.

db2TablespaceStruct 데이터 구조 특정 매개변수

tablespaces

입력. 백업되는 테이블 스페이스 목록의 포인터. C의 경우 목록은 널(null)로 끝나는 문자열입니다. 일반적으로 db2Char 구조의 목록입니다.

numTablespaces

입력. **tablespaces** 매개변수의 엔트리 수입니다.

db2MediaListStruct 데이터 구조 매개변수

locations

입력. 미디어 위치 목록의 포인터. C의 경우 목록은 널(null)로 끝나는 문자열입니다. 일반적으로 db2Char 구조의 목록입니다.

numLocations

입력. **locations** 매개변수의 엔트리 수

locationType

입력. 미디어 유형을 나타내는 문자. 유효한 값(include 디렉토리에 있는 `sqlutil` 헤더 파일에 정의)은 다음과 같습니다.

SQLU_LOCAL_MEDIA: 'L'

로컬 디바이스(테이프, 디스크, 디스켓 또는 Named Pipes).

SQLU_XBSA_MEDIA: 'X'

XBSA 인터페이스.

SQLU_TSM_MEDIA: 'A'

Tivoli Storage Manager.

SQLU_OTHER_MEDIA: 'O'

벤더 라이브러리.

SQLU_SNAPSHOT_MEDIA: 'F'

스냅샷 백업에서 데이터가 리스토어되도록 지정합니다.

다음 중 어느 것과도 함께 SQLU_SNAPSHOT_MEDIA를 사용할 수 없습니다.

- 호출자 조치: DB2RESTORE_RESTORE_STORDEF, DB2RESTORE_STORDEF_NOINTERRUPT, DB2RESTORE_TERMINATE_INCRE
- DB2RESTORE_REPLACE_HISTORY
- DB2RESTORE_TABLESPACE
- DB2RESTORE_COMPR_LIB
- DB2RESTORE_INCREMENTAL
- DB2RESTORE_HISTORY
- DB2RESTORE_LOGS
- **piStoragePaths** - 사용하려면 NULL 또는 비어 있어야 함
- **piTargetDBPath**
- **piTargetDBAlias**
- **piNewLogPath**
- **iNumBuffers**

- **iBufferSize**
- **piRedirectScript**
- **iRedirectScriptLen**
- **iParallelism**
- **piComprLibrary**, **iComprLibraryLen**, **piComprOptions** 또는 **iComprOptionsSize**
- 이 구조의 **numLocations** 필드는 스냅샷 리스토어의 경우 1이어야 합니다.

또한 SNAPSHOT 매개변수를 테이블 스페이스 목록에 연관된 모든 리스토어 조작에 사용할 수 없습니다.

스냅샷 백업 이미지에서 데이터를 리스토어할 때 디폴트 동작은 시간소인이 제공되지 않는 경우 모든 컨테이너, 로컬 볼륨 디렉토리, 데이터베이스 경로(**DBPATH**), 1차 로그 및 가장 최신 스냅샷 백업의 미러 로그 경로를 포함하여 데이터베이스를 구성하는 모든 경로의 FULL DATABASE OFFLINE 리스토어입니다(**INCLUDE LOGS**는 **EXCLUDE LOGS**가 명시적으로 언급된 경우를 제외하고는 모든 스냅샷 백업의 디폴트임). 시간소인이 제공되는 경우에는 스냅샷 백업 이미지가 리스토어됩니다.

IBM Data Server로의 통합은 다음 스토리지 하드웨어에 대한 DB2 ACS API 드라이버입니다.

- IBM TotalStorage SAN Volume Controller
- IBM Enterprise Storage Server Model 800
- IBM System Storage DS6000
- IBM System Storage DS8000
- IBM System Storage N Series
- NetApp V-series
- NetApp FAS

db2StoragePathsStruct 데이터 구조 매개변수

storagePaths

입력. 자동 스토리지 테이블 스페이스에 사용되는 서버 스토리지 경로의 완전한 이름이 포함된 문자열의 배열. 다중 파티션 데이터베이스에서 동일한 스토리지 경로가 모든 데이터베이스 파티션에 사용됩니다. 다중 파티션 데이터베이스가 새 스토리지 경로로 리스토어되는 경우 카탈로그 파티션은 다른 데이터베이스 파티션이 리스토어되기 전에 리스토어해야 합니다.

numStoragePaths

입력. db2StoragePathsStruct 구조의 **storagePaths** 매개변수에 대한 스토리지 경로 수.

db2gRestoreStruct 데이터 구조 특정 매개변수

iSourceDBAliasLen

입력. **piSourceDBAlias** 매개변수의 길이를 바이트 단위로 지정합니다.

iTargetDBAliasLen

입력. **piTargetDBAlias** 매개변수의 길이를 바이트 단위로 지정합니다.

iApplicationIdLen

입력. **poApplicationId** 매개변수의 길이를 바이트 단위로 지정합니다. SQLU_APPLID_LEN + 1과 동일해야 합니다. 상수 SQLU_APPLID_LEN은 include 디렉토리에 있는 `sqlutil` 헤더 파일에 정의됩니다.

iTimestampLen

입력. **piTimestamp** 매개변수의 길이를 바이트 단위로 지정합니다.

iTargetDBPathLen

입력. **piTargetDBPath** 매개변수의 길이를 바이트 단위로 지정합니다.

iReportFileLen

입력. **piReportFile** 매개변수의 길이를 바이트 단위로 지정합니다.

iUsernameLen

입력. **piUsername** 매개변수의 길이를 바이트 단위로 지정합니다. 사용자 이름을 제공하지 않는 경우에는 영(0)으로 설정하십시오.

iPasswordLen

입력. **piPassword** 매개변수의 길이를 바이트 단위로 지정합니다. 암호를 제공하지 않은 경우에는 영(0)으로 설정하십시오.

iNewLogPathLen

입력. **piNewLogPath** 매개변수의 길이를 바이트 단위로 지정합니다.

iLogTargetLen

입력. **piLogTarget** 매개변수의 길이를 바이트 단위로 지정합니다.

iRedirectScriptLen

입력. **piRedirectScript**에 지정된 라이브러리 이름 길이(바이트 단위)를 나타내는 4바이트의 부호없는 정수. 스크립트 이름을 지정하지 않은 경우 영(0)으로 설정하십시오.

db2Char 데이터 구조 매개변수

pioData

문자 데이터 버퍼의 포인터. NULL인 경우 데이터가 리턴되지 않습니다.

iLength

입력. **pioData** 버퍼 크기

oLength

출력. **pioData** 버퍼에 있는 데이터의 유효한 문자 수

사용 시 참고사항

- 오프라인 리스토어의 경우 이 유틸리티는 독점 모드로 데이터베이스에 연결됩니다. 호출 응용프로그램을 포함한 임의의 응용프로그램이 리스토어 중인 데이터베이스에 이미 연결되어 있는 경우에는 유틸리티가 실패합니다. 또한 리스토어 수행에 사용 중인 리스토어 유틸리티 및 호출 응용프로그램을 포함한 임의의 응용프로그램이 이 동일한 워크스테이션의 데이터베이스에 이미 연결되어 있는 경우에도 요청이 실패합니다. 성공적으로 연결되면 리스토어가 완료될 때까지 API가 다른 응용프로그램을 잠급니다.
- 현재 데이터베이스 구성 파일은 사용 불가능해지기 전에는 백업 사본으로 교체되지 않습니다. 이 경우 파일이 교체되면 경고 메시지가 리턴됩니다.
- 데이터베이스나 테이블 스페이스는 db2Backup API를 사용하여 백업해야 합니다.
- 호출자 조치 값이 DB2RESTORE_NOINTERRUPT인 경우 응용프로그램을 프롬프트하지 않고 리스토어가 계속됩니다. 호출자 조치 값이 DB2RESTORE_RESTORE이고 유틸리티가 기존 데이터베이스를 리스토어하는 경우 유틸리티는 일부 사용자의 상호 작용을 요청하는 메시지와 같이 응용프로그램에 제어를 리턴합니다. 사용자 상호 작용을 처리한 후에 응용프로그램은 호출자 조치 값이 처리를 계속하거나 (DB2RESTORE_CONTINUE) 또는 후속 호출에서 종료하도록 (DB2RESTORE_TERMINATE) 표시하여 RESTORE DATABASE를 다시 호출합니다. 유틸리티는 처리를 종료하고 sqlca에 SQLCODE를 리턴합니다.
- 완료 시에 디바이스를 종료하려면 호출자 조치 값을 DB2RESTORE_DEVICE_TERMINATE로 설정하십시오. 예를 들어, 사용자가 2개의 테이프 디바이스를 사용하여 3개의 테이프 볼륨에서 리스토어 중이고 이 테이프 중 하나가 리스토어된 경우 응용프로그램은 테이프 끝을 나타내는 SQLCODE가 포함된 API에서 제어를 확보합니다. 응용프로그램은 사용자에게 다른 테이프를 마운트하도록 프롬프트할 수 있으며 사용자가 "더 이상 필요하지 않음"을 표시한 경우 미디어 디바이스 마지막을 알리도록 호출자 조치 값 DB2RESTORE_DEVICE_TERMINATE를 사용하여 API로 리턴하십시오. 디바이스 드라이버는 종료되지만 리스토어에 사용되는 나머지 디바이스는 리스토어 설정의 모든 세그먼트가 리스토어될 때까지 계속 입력이 처리됩니다(리스토어 설정의 세그먼트 수는 백업 프로세스 중에 최종 미디어 디바이스에 배치됨). 이 호출자 조치는 테이프 이외의 디바이스에도 사용할 수 있습니다(벤더 지원 디바이스).
- 응용프로그램으로 리턴하기 전에 매개변수 점검을 수행하려면 호출자 조치 값을 DB2RESTORE_PARM_CHK로 설정하십시오.

- 경로 재지정 리스토어 수행 시에 호출자 조치 값을 DB2RESTORE_RESTORE_STORDEF로 설정하고 이는 sqlbstsc API와 같이 사용됩니다.
- 데이터베이스 리스토어의 중요한 단계에서 시스템 오류가 발생하면 성공적으로 리스토어가 수행될 때까지 데이터베이스에 제대로 연결할 수 없습니다. 이 조건은 연결 시도 시에 발견되어 오류 메시지가 리턴됩니다. 백업된 데이터베이스를 롤 포워드 복구되도록 구성할 수 없지만 리스토어 후에 이 매개변수 중 사용 가능한 매개변수가 포함된 사용 가능한 현재 구성 파일이 있는 경우 데이터베이스에 연결하기 전에 사용자는 데이터베이스의 새 백업을 작성하거나 로그 유지 및 userexit 매개변수를 사용 불가능하도록 설정해야 합니다.
- 리스토어된 데이터베이스는 삭제할 수 없지만(현재 없는 데이터베이스로 복구하는 경우는 제외) 리스토어가 실패하면 사용할 수 없게 됩니다.
- 리스토어 유형으로 백업의 실행기록 파일이 리스토어되도록 지정한 경우 데이터베이스의 기존 실행기록 파일에 대해 리스토어되며 현재 리스토어 중인 백업 후에 실행기록 파일의 모든 변경사항은 효율적으로 지워집니다. 이를 원하지 않는 경우 수행된 모든 갱신사항을 삭제하지 않고도 해당 컨텐츠를 확인할 수 있도록 실행기록 파일을 새 파일로 리스토어하거나 데이터베이스를 테스트하십시오.
- 백업 조작 시에 데이터베이스를 롤 포워드 복구에 사용할 수 있는 경우 db2Restore를 올바르게 실행한 후에 db2Rollforward를 실행하여 손상되기 전 상태로 데이터베이스를 되돌릴 수 있습니다. 데이터베이스가 복구 가능한 경우 리스토어가 완료되면 디폴트로 롤 포워드 보류 상태가 됩니다.
- 데이터베이스 백업 이미지가 오프라인이고 호출자가 리스토어 후에 데이터베이스를 롤 포워드하지 않으려는 경우 리스토어에 DB2RESTORE_NOROLLFWD 옵션을 사용할 수 있습니다. 그러면 리스토어 후에 데이터베이스를 즉시 사용할 수 있게 됩니다. 백업 이미지가 온라인인 경우 호출자는 리스토어 완료 시에 해당 로그 레코드를 통해 롤 포워드해야 합니다.
- 로그 파일이 포함된 백업 이미지에서 로그 파일을 리스토어하려면 완전한 유효한 경로가 DB2 서버에 있는 것으로 가정하여 **LOGTARGET** 옵션을 지정해야 합니다. 해당 조건이 충족되면 리스토어 유틸리티가 이미지의 로그 파일을 목표 경로에 기록합니다. 로그를 포함하지 않는 백업 이미지 리스토어 중에 **LOGTARGET**을 지정한 경우 리스토어 조작은 테이블 스페이스 데이터 리스토어 시도 전에 오류를 리턴합니다. 유효하지 않거나 읽기 전용의 **LOGTARGET** 경로를 지정한 경우에도 리스토어 조작이 실패합니다.
- RESTORE DATABASE 명령 실행 시에 **LOGTARGET** 경로에 로그 파일이 없는 경우 경고 프롬프트가 사용자에게 리턴됩니다. **WITHOUT PROMPTING**을 지정한 경우에는 이 경고가 리턴되지 않습니다.
- **LOGTARGET**이 지정된 리스토어 조작 중에 로그 파일을 추출할 수 없으면 리스토어 조작이 실패하고 오류가 리턴됩니다. 백업 이미지에 추출 중인 로그 파일이

LOGTARGET 경로에 있는 기존 파일과 이름이 같은 경우 리스토어 조작이 실패하고 오류가 리턴됩니다. 리스토어 유틸리티는 **LOGTARGET** 디렉토리의 기존 로그 파일을 겹쳐쓰지 않습니다.

- 백업 이미지에서는 저장된 로그 세트만 리스토어할 수 있습니다. 로그 파일만 리스토어되도록 표시하려면 **LOGTARGET** 경로와 함께 LOGS 옵션을 지정하십시오. **LOGTARGET** 경로 없이 **LOGS** 옵션만 지정하면 오류가 발생합니다. 이 모드에서 로그 파일 리스토어 중에 문제가 발생하면 리스토어 조작이 즉시 종료되고 오류가 리턴됩니다.
- 자동 증분 리스토어 조작 중에는 리스토어 조작의 목표 이미지에 포함된 로그만 백업 이미지에서 검색됩니다. 증분 리스토어 프로세스 중에 참조되는 중간 이미지에 포함된 로그는 해당 중간 백업 이미지에서 추출되지 않습니다. 수동 증분 리스토어 조작 중에 **LOGTARGET** 경로는 최종 리스토어 명령에만 지정해야 합니다.
- 백업이 압축된 경우 DB2 데이터베이스 시스템은 이 상태를 발견하고 이를 리스토어하기 전에 데이터를 자동으로 압축 해제합니다. 라이브러리가 db2Restore API에 지정된 경우 데이터 압축 해제에 사용됩니다. 라이브러리가 db2Restore API에 지정되지 않은 경우에는 백업 이미지에 저장된 라이브러리가 사용됩니다. 백업 이미지에 저장된 라이브러리가 없는 경우에는 데이터를 압축 해제할 수 없으며 리스토어 조작이 실패합니다.
- 백업 이미지에서 압축 라이브러리를 리스토어 중인 경우 (DB2RESTORE_COMPR_LIB 리스토어 유형을 지정하여 명시적으로 또는 압축된 백업의 일반 리스토어를 수행하여 내재적으로) 리스토어 조작은 백업이 수행된 동일한 플랫폼 및 운영 체제에서 수행해야 합니다. DB2 데이터베이스 시스템이 일반적으로 관련된 두 시스템에서 여러 플랫폼에 걸친 리스토어 조작을 지원하는 경우에도 플랫폼이 다른 경우 리스토어 조작이 실패합니다.
- 자동 스토리지가 사용 가능한 데이터베이스를 리스토어하는 경우 데이터베이스에 연관된 스토리지 경로를 재정의하거나 이전과 동일하게 유지할 수 있습니다. 현재와 동일하게 스토리지 경로를 유지하려면 리스토어 조작 중에 스토리지 경로를 제공하지 마십시오. 그 이외의 경우에는 데이터베이스에 연관시키려는 새 스토리지 경로 설정을 지정하십시오. 자동 스토리지 테이블 스페이스는 리스토어 조작 중에 자동으로 새 스토리지 경로로 경로 재지정됩니다.

스냅샷 리스토어

일반(비스냅샷) 리스토어와 유사하게 스냅샷 백업 이미지 리스토어 시의 디폴트 동작은 로그 디렉토리를 리스토어하지 않습니다. - DB2RESTORE_LOGTARGET_EXCLUDE.

DB2 관리 프로그램이 리스토어하려는 기타 경로에서 로그 디렉토리의 그룹 ID가 공유되는 것을 발견하면 오류가 리턴됩니다. 이 경우

DB2RESTORE_LOGTARGET_INCLUDE 또는
DB2RESTORE_LOGTARGET_INCFORCE는 로그 디렉토리가 리스토어의 파트여야 하기 때문에 지정해야 합니다.

DB2 관리 프로그램은 백업 이미지의 경로 리스토어가 수행되기 전에 기존 로그 디렉토리를 모두 저장하려고 합니다(기본, 미러 및 오버플로우).

로그 디렉토리를 리스토어하려는 경우 DB2가 디스크의 기존 로그 디렉토리가 백업 이미지의 로그 디렉토리와 충돌하는 것을 발견하면 DB2 관리 프로그램이 오류를 보고합니다. 이 경우 DB2RESTORE_LOGTARGET_INCFORCE를 지정하면 이 오류가 표시되지 않고 이미지의 로그 디렉토리가 리스토어되며 기존의 모든 사항은 삭제됩니다.

DB2RESTORE_LOGTARGET_EXCLUDE 옵션이 지정되고 로그 디렉토리 경로가 데이터베이스 디렉토리(예: /NODE*xxxx*/SQL*xxxxx*/SQLOGDIR/)에 있는 특수한 상황도 있습니다. 이 경우 리스토어는 로그 디렉토리를 데이터베이스 경로로 겹쳐쓰고 포함된 모든 컨텐츠가 리스토어됩니다. DB2 관리 프로그램이 이 시나리오를 발견하고 로그 파일이 이 로그 디렉토리에 있는 경우 오류가 보고됩니다.
DB2RESTORE_LOGTARGET_EXCLUDE를 지정하고 이 오류가 표시되지 않으면 백업 이미지의 해당 로그 디렉토리는 디스크의 충돌하는 로그 디렉토리를 겹쳐씁니다.

제 70 장 db2Rollforward - 데이터베이스 롤 포워드

데이터베이스 로그 파일에 기록된 트랜잭션을 적용하여 데이터베이스를 복구합니다. 데이터베이스나 테이블 스페이스 백업이 리스토어된 후에 호출되거나 미디어 오류로 인해 데이터베이스에서 테이블 스페이스를 오프라인으로 설정한 경우에 호출됩니다. 데이터베이스는 롤 포워드 복구를 사용하여 복구하기 전에 복구 가능해야 합니다(즉, **logarchmeth1** 데이터베이스 구성 매개변수 또는 **logarchmeth2** 데이터베이스 구성 매개변수가 OFF 이외의 값으로 설정되어야 함)

범위

파티션된 데이터베이스 환경에서 이 API는 카탈로그 파티션에서 호출해야 합니다. 롤 포워드되는 파티션은 TO절에 지정한 사항에 따라 다릅니다.

- 특정 시점 롤 포워드 호출은 db2nodes.cfg 파일에 나열된 모든 데이터베이스 파티션 서버에 영향을 줍니다.
- END OF LOGS 롤 포워드 호출은 ON DATABASE PARTITION절에 지정된 데이터베이스 파티션 서버에만 영향을 줍니다. 데이터베이스 파티션 서버를 지정하지 않으면 롤 포워드 호출이 db2nodes.cfg 파일에 나열된 모든 데이터베이스 파티션 서버에 영향을 줍니다.
- 백업 종료를 지정하는 데이터베이스 또는 테이블 스페이스 호출은 db2nodes.cfg 파일에 나열된 모든 데이터베이스 파티션 서버에 영향을 줍니다.

특정 데이터베이스 파티션 서버의 모든 트랜잭션이 현재 데이터베이스에 적용되어서 해당 트랜잭션 중에서 롤 포워드할 트랜잭션이 없는 경우에는 해당 데이터베이스 파티션 서버가 무시됩니다.

파티션된 테이블을 특정 시점으로 롤 포워드하는 경우 해당 테이블을 포함하는 테이블 스페이스도 동일한 특정 시점으로 롤 포워드해야 합니다. 그러나 테이블 스페이스를 롤 포워드할 때 해당 테이블 스페이스의 모든 테이블을 롤 포워드할 필요는 없습니다.

권한 부여

다음 중 하나가 필요합니다.

- *sysadm*
- *sysctrl*
- *sysmaint*

필수 연결

없음. 이 API는 데이터베이스 연결을 작성합니다.

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Rollforward (
       db2Uint32 versionNumber,
       void * pDB2RollforwardStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RollforwardStruct
{
  struct db2RfwdInputStruct *piRfwdInput;
  struct db2RfwdOutputStruct *poRfwdOutput;
} db2RollforwardStruct;

typedef SQL_STRUCTURE db2RfwdInputStruct
{
  sqluint32 iVersion;
  char *piDbAlias;
  db2Uint32 iCallerAction;
  char *piStopTime;
  char *piUserName;
  char *piPassword;
  char *piOverflowLogPath;
  db2Uint32 iNumChngLgOvrflw;
  struct sqlurf_newlogpath *piChngLogOvrflw;
  db2Uint32 iConnectMode;
  struct sqlu_tablespace_bkrst_list *piTablespaceList;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  SQL_PDB_NODE_TYPE *piNodeList;
  db2int32 iNumNodeInfo;
  char *piDroppedTblID;
  char *piExportDir;
  db2Uint32 iRollforwardFlags;
} db2RfwdInputStruct;

typedef SQL_STRUCTURE db2RfwdOutputStruct
{
  char *poApplicationId;
  sqlint32 *poNumReplies;
  struct sqlurf_info *poNodeInfo;
  db2Uint32 oRollforwardFlags;
} db2RfwdOutputStruct;

SQL_STRUCTURE sqlurf_newlogpath
{
   SQL_PDB_NODE_TYPE nodenum;
   unsigned short  pathlen;
   char logpath[SQL_LOGPATH_SZ+SQL_LOGFILE_NAME_SZ+1];
};

typedef SQL_STRUCTURE sqlu_tablespace_bkrst_list
{
   sqlint32 num_entry;
   struct sqlu_tablespace_entry *tablespace;
```

```
} sqlu_tablespace_bkrst_list;

typedef SQL_STRUCTURE sqlu_tablespace_entry
{
   sqluint32 reserve_len;
   char tablespace_entry[SQLU_MAX_TBS_NAME_LEN+1];
   char filler[1];
} sqlu_tablespace_entry;

SQL_STRUCTURE sqlurf_info
{
   SQL_PDB_NODE_TYPE nodenum;
   sqlint32 state;
   unsigned char   nextarclog[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   firstarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastarcdel[SQLUM_ARCHIVE_FILE_LEN+1];
   unsigned char   lastcommit[SQLUM_TIMESTAMP_LEN+1];
};

SQL_API_RC SQL_API_FN
  db2gRollforward (
       db2Uint32 versionNumber,
       void * pDB2gRollforwardStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRollforwardStruct
{
  struct db2gRfwdInputStruct *piRfwdInput;
  struct db2RfwdOutputStruct *poRfwdOutput;
} db2gRollforwardStruct;

typedef SQL_STRUCTURE db2gRfwdInputStruct
{
  db2Uint32 iDbAliasLen;
  db2Uint32 iStopTimeLen;
  db2Uint32 iUserNameLen;
  db2Uint32 iPasswordLen;
  db2Uint32 iOvrflwLogPathLen;
  db2Uint32 iDroppedTblIDLen;
  db2Uint32 iExportDirLen;
  sqluint32 iVersion;
  char *piDbAlias;
  db2Uint32 iCallerAction;
  char *piStopTime;
  char *piUserName;
  char *piPassword;
  char *piOverflowLogPath;
  db2Uint32 iNumChngLgOvrflw;
  struct sqlurf_newlogpath *piChngLogOvrflw;
  db2Uint32 iConnectMode;
  struct sqlu_tablespace_bkrst_list *piTablespaceList;
  db2int32 iAllNodeFlag;
  db2int32 iNumNodes;
  SQL_PDB_NODE_TYPE *piNodeList;
  db2int32 iNumNodeInfo;
```

```
  char *piDroppedTblID;
  char *piExportDir;
  db2Uint32 iRollforwardFlags;
} db2gRfwdInputStruct;
```

db2Rollforward API 매개변수

versionNumber

입력. 두 번째 매개변수로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pDB2RollforwardStruct

입력. db2RollforwardStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2RollforwardStruct 데이터 구조 매개변수

piRfwdInput

입력. db2RfwdInputStruct 구조의 포인터

poRfwdOutput

출력. db2RfwdOutputStruct 구조의 포인터

db2RfwdInputStruct 데이터 구조 매개변수

iVersion

입력. 롤 포워드된 매개변수의 버전 ID. SQLUM_RFWD_VERSION으로 정의됩니다.

piDbAlias

입력. 데이터베이스 별명이 포함된 문자열. 시스템 데이터베이스 디렉토리에서 카탈로그되는 별명입니다.

iCallerAction

입력. 수행할 조치를 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2ROLLFORWARD_ROLLFWD

piStopTime 매개변수로 지정된 특정 시점으로 롤 포워드됩니다. 데이터베이스 롤 포워드를 위해 데이터베이스는 롤 포워드 보류 상태로 유지됩니다. 특정 시점으로 테이블 스페이스를 롤 포워드하기 위해 테이블 스페이스는 롤 포워드 진행 중 상태로 유지됩니다.

DB2ROLLFORWARD_STOP

사용 가능한 로그 파일을 사용하여 롤 포워드한 후에 다시 롤백하여 롤 포워드 복구를 종료합니다. 언커미트된 트랜잭션은 취소된 다음 데이터베이스나 테이블 스페이스의 롤 포워드 보류 상태가 해제됩니다. 이 값

의 동의어는
DB2ROLLFORWARD_RFWD_COMPLETE입니다.

DB2ROLLFORWARD_RFWD_STOP
piStopTime으로 지정된 특정 시점으로 롤 포워드하고 롤 포워드 복구를 종료합니다. 데이터베이스나 테이블 스페이스의 롤 포워드 보류 상태가 해제됩니다. 이 값의 동의어는
DB2ROLLFORWARD_RFWD_COMPLETE입니다.

DB2ROLLFORWARD_QUERY
nextarclog, firstarcdel, lastarcdel, lastcommit의 쿼리 값. 데이터베이스 상태와 노드 번호가 리턴됩니다.

DB2ROLLFORWARD_PARM_CHECK
롤 포워드를 수행하지 않고 매개변수의 유효성을 확인합니다.

DB2ROLLFORWARD_CANCEL
현재 실행 중인 롤 포워드 조작을 취소합니다. 데이터베이스나 테이블 스페이스는 복구 보류 상태가 됩니다.

주: 롤 포워드가 실제로 실행 중인 경우에는 이 옵션을 사용할 수 없습니다. 롤 포워드가 일시정지되거나(즉, STOP 대기) 또는 롤 포워드 중에 시스템 오류가 발생한 경우에 사용할 수 있습니다. 사용 시 주의해야 합니다.
데이터베이스를 롤 포워드하려면 테이프 디바이스를 사용하는 로드 복구가 필요합니다. 디바이스에서 사용자의 개입이 필요한 경우에는 롤 포워드 API가 경고 메시지를 리턴합니다. 다음 세 호출자 조치 중 하나를 사용하여 API를 다시 호출할 수 있습니다.

DB2ROLLFORWARD_LOADREC_CONT
경고 메시지를 생성한 디바이스를 사용하여 계속합니다(예를 들어, 새로운 테이프가 마운트된 경우)

DB2ROLLFORWARD_DEVICE_TERM
경고 메시지를 생성한 디바이스를 사용하여 중지합니다(예를 들어, 더 이상의 테이프가 없는 경우)

DB2ROLLFORWARD_LOAD_REC_TERM
로드 복구에서 사용 중인 모든 디바이스를 종료합니다.

piStopTime
입력. ISO 형식의 시간소인을 포함하는 문자열. 이 시간소인이 초과되면 데이터베이스 복구가 중지됩니다. 최대한 롤 포워드하려면
SQLUM_INFINITY_TIMESTAMP를 지정하십시오.

DB2ROLLFORWARD_QUERY, DB2ROLLFORWARD_PARM_CHECK 및 모든 로그 복구(DB2ROLLFORWARD_LOADREC_xxx) 호출자 조치에 대해 NULL일 수 있습니다.

piUserName

입력. 응용프로그램의 사용자 이름이 포함된 문자열. NULL일 수 있습니다.

piPassword

입력. 입력한 사용자 이름(있는 경우)의 암호가 포함된 문자열. NULL일 수 있습니다.

piOverflowLogPath

입력. 이 매개변수를 사용하여 사용하려는 대체 로그 경로를 지정합니다. 사용중인 로그 파일 뿐만 아니라 이 유틸리티에서 사용하기 전에 사용자는 이 아카이브된 로그 파일도 logpath로 이동해야 합니다. 데이터베이스가 logpath에 공간이 충분하지 않은 경우에 문제가 발생할 수 있습니다. 이런 이유로 인해 오버플로우 로그 경로가 제공됩니다. 롤 포워드 복구 중에는 필요한 로그 파일이 logpath에서 먼저 검색되고 오버플로우 로그 경로에서 두 번째로 검색됩니다. 테이블 스페이스 롤 포워드 복구에 필요한 로그 파일은 logpath 또는 오버플로우 로그 경로로 가져올 수 있습니다. 호출자가 오버플로우 로그 경로를 지정하지 않으면 디폴트값은 logpath입니다. 파티션된 데이터베이스 환경에서 오버플로우 로그 경로는 유효한 완전한 경로여야 합니다. 디폴트 경로는 각 노드의 디폴트 오버플로우 로그 경로입니다. 단일 파티션 데이터베이스 환경에서 오버플로우 로그 경로는 서버가 로컬인 경우 상대 경로입니다.

iNumChngLgOvrflw

입력. 파티션된 데이터베이스 환경에만 적용됩니다. 변경된 오버플로우 로그 경로 수. 이 새 로그 경로는 지정된 데이터베이스 파티션 서버에 대해서만 디폴트 오버플로우 로그 경로를 겹쳐씁니다.

piChngLogOvrflw

입력. 파티션된 데이터베이스 환경에만 적용됩니다. 변경된 오버플로우 로그 경로의 완전한 이름이 포함된 구조의 포인터. 이 새 로그 경로는 지정된 데이터베이스 파티션 서버에 대해서만 디폴트 오버플로우 로그 경로를 겹쳐씁니다.

iConnectMode

입력. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2ROLLFORWARD_OFFLINE

오프라인 롤 포워드. 이 값은 데이터베이스 롤 포워드 복구에 대해 지정해야 합니다.

DB2ROLLFORWARD_ONLINE

온라인 롤 포워드

piTablespaceList

입력. 로그 끝 또는 특정 시점으로 롤 포워드되는 테이블 스페이스의 이름이 포함된 구조의 포인터. 지정하지 않으면 롤 포워드해야 하는 테이블 스페이스가 선택됩니다.

파티션된 테이블의 경우 파티션된 테이블의 임의 부분이 포함된 테이블 스페이스의 특정 시점(PIT)의 롤 포워드도 테이블이 동일한 특정 시점에 상주해야 하는 다른 모든 테이블 스페이스를 롤 포워드해야 합니다. 파티션된 테이블 일부를 포함하는 단일 테이블 스페이스에 대해 로그 끝으로 롤 포워드하는 것도 허용됩니다.

파티션된 테이블에 접속, 접속 해제 또는 삭제된 데이터 파티션이 있는 경우에는 PIT 롤 포워드는 이 데이터 파티션에 대한 모든 테이블 스페이스도 포함해야 합니다. 파티션된 테이블에 접속, 접속 해제 또는 삭제된 데이터 파티션이 있는지 판별하려면 SYSDATAPARTITIONS 카탈로그 테이블의 상태 필드를 쿼리하십시오.

파티션된 테이블은 여러 테이블 스페이스에 상주할 수 있기 때문에 일반적으로 여러 테이블 스페이스를 롤 포워드해야 합니다. 삭제된 테이블 복구를 통해 복구된 데이터는 piExportDir 매개변수에 지정된 익스포트 디렉토리에 기록됩니다. 하나의 명령을 실행하여 모든 테이블 스페이스를 롤 포워드하거나 관련된 테이블 스페이스의 서브세트에 대해 롤 포워드 조작을 반복할 수도 있습니다. db2Rollforward API가 테이블의 모든 데이터를 복구하는 데 필요한 전체 테이블 스페이스 세트를 지정하지 않은 경우 경고가 통지 로그에 기록됩니다. 관리 통지 로그에 있는 명령으로 복구되지 않은 모든 파티션의 전체 세부사항이 포함된 경고가 리턴됩니다.

테이블 스페이스 서브세트를 롤 포워드할 수 있으면 단일 익스포트 디렉토리에 맞는 데이터보다 더 많은 데이터를 복구해야 하는 경우에 이를 더 쉽게 처리할 수 있습니다.

iAllNodeFlag

입력. 파티션된 데이터베이스 환경에만 적용됩니다. 롤 포워드 조작이 db2nodes.cfg에 정의된 모든 데이터베이스 파티션 서버에 적용되는지를 나타냅니다. 가능한 값은 다음과 같습니다.

DB2_NODE_LIST

piNodeList로 전달된 목록의 데이터베이스 파티션 서버에 적용됩니다.

DB2_ALL_NODES

모든 데이터베이스 파티션 서버에 적용됩니다. 디폴트값입니다. 이 값을 사용하는 경우에는 piNodeList 매개변수를 NULL로 설정해야 합니다.

DB2_ALL_EXCEPT

piNodeList로 전달된 목록에 있는 데이터베이스 파티션 서버를 제외한 모든 데이터베이스 파티션 서버에 적용됩니다.

DB2_CAT_NODE_ONLY

카탈로그 파티션에만 적용됩니다. 이 값을 사용하는 경우에는 piNodeList 매개변수를 NULL로 설정해야 합니다.

iNumNodes

입력. piNodeList 배열의 데이터베이스 파티션 서버 수를 지정합니다.

piNodeList

입력. 롤 포워드 복구를 수행해야 하는 데이터베이스 파티션 서버 수 배열의 포인터

iNumNodeInfo

입력. 출력 매개변수 poNodeInfo *크기*를 정의하며 이는 롤 포워드되는 각 데이터베이스 파티션의 상태 정보를 모두 포함할 수 있는 *크기*여야 합니다. 단일 파티션의 데이터베이스 환경의 경우 이 매개변수는 1로 설정해야 합니다. 이 매개변수 값은 이 API가 호출되는 데이터베이스 파티션 서버의 수와 동일해야 합니다.

piDroppedTblID

입력. 복구 시도 중인 삭제된 테이블의 ID가 포함된 문자열. 파티션된 테이블의 경우 drop-table-id는 테이블을 전체로 식별하기 때문에 테이블의 모든 데이터 파티션은 단일 롤 포워드 명령으로 복구할 수 있습니다.

piExportDir

입력. 삭제된 테이블이 익스포트되는 디렉토리 이름

iRollforwardFlags

입력. 롤 포워드 플래그를 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2ROLLFORWARD_EMPTY_FLAG

*플래그*가 지정되지 않습니다.

DB2ROLLFORWARD_LOCAL_TIME

GMT 시간이 아닌 사용자의 로컬 시점인 특정 시점으로 롤 포워드할 수 있습니다. 그러면, 사용자의 로컬 시스템의 특정 시점으로 롤 포워드 하기가 쉽고 로컬을 GMT 시간으로 변환함으로서 초래되는 사용자의 잠재적인 오류도 방지할 수 있습니다.

DB2ROLLFORWARD_NO_RETRIEVE

아카이브된 로그 검색을 사용하지 않도록 설정하여 대기 머신에서 롤 포워드되는 로그 파일을 제어할 수 있습니다. 롤 포워드되는 로그 파일을 제어하여 대기 머신이 프로덕션 머신보다 X 시간 늦도록 설정하

여 사용자가 두 시스템 모두를 변경하지 않도록 할 수 있습니다. 이 옵션은 대기 시스템이 아카이브에 액세스할 수 없는 경우에 유용합니다. 예를 들어 TSM이 아카이브인 경우 원래 머신만 파일을 검색할 수 있습니다. 또한, 프로덕션 시스템이 파일을 아카이브하고 대기 시스템이 동일할 파일을 검색할 때 대기 시스템이 완료되지 않은 로그 파일을 검색하게 되는 가능성도 제거합니다.

DB2ROLLFORWARD_END_OF_BACKUP
데이터베이스는 *최소 복구 시간*으로 롤 포워드하도록 지정합니다.

db2RfwdOutputStruct 데이터 구조 매개변수

poApplicationId
출력. 응용프로그램 ID

poNumReplies
출력. 수신된 응답 수

poNodeInfo
출력. 데이터베이스 파티션 응답 정보

oRollforwardFlags
출력. 롤 포워드 출력 플래그. 가능한 값은 다음과 같습니다.

DB2ROLLFORWARD_OUT_LOCAL_TIME
마지막으로 커미트한 트랜잭션 시간소인이 UTC가 아닌 로컬 시간에 표시되도록 표시합니다. 로컬 시간은 클라이언트가 아니라 서버의 로컬 시간입니다. 파티션된 데이터베이스 환경에서 로컬 시간은 카탈로그 파티션의 로컬 시간입니다.

sqlurf_newlogpath 데이터 구조 매개변수

nodenum
입력. 이 구조에서 설명하는 데이터베이스 파티션 번호

pathlen
입력. logpath 필드의 총 길이

logpath
입력. 롤 포워드 조작을 위해 특정 노드에 사용되는 완전한 경로

sqlu_tablespace_bkrst_list 데이터 구조 매개변수

num_entry
입력. 테이블 스페이스 매개변수로 지시된 목록에 포함된 구조 수

tablespace
입력. sqlu_tablespace_entry 구조 목록의 포인터

sqlu_tablespace_entry 데이터 구조 매개변수

reserve_len

입력. tablespace_entry 매개변수의 길이를 바이트 단위로 지정합니다.

tablespace_entry

입력. 롤 포워드되는 테이블 스페이스 이름

filler 메모리에서 데이터 구조를 적절하게 맞추는 데 사용되는 필러

sqlurf_info 데이터 구조 매개변수

nodenum

출력. 이 구조가 포함하고 있는 데이터베이스 파티션 수.

state 출력. 데이터베이스 파티션에서 롤 포워드에 포함된 데이터베이스 또는 테이블 스페이스의 현재 상태.

nextarclog

출력. 롤 포워드가 완료되면 이 필드는 비어 있습니다. 롤 포워드가 완료되지 않으면 이는 롤 포워드에 대해 처리되는 다음 로그 파일의 이름입니다.

firstarcdel

출력. 롤 포워드로 재생되는 첫 번째 로그 파일

lastarcdel

출력. 롤 포워드로 재생되는 마지막 로그 파일

lastcommit

출력. 마지막으로 커미트된 트랜잭션 시간.

db2gRfwdInputStruct 데이터 구조 특정 매개변수

iDbAliasLen

입력. 데이터베이스 별명 길이를 바이트 단위로 지정합니다.

iStopTimeLen

입력. 중지 시간 매개변수 길이를 바이트 단위로 지정합니다. 중지 시간을 제공하지 않는 경우에는 영(0)으로 설정하십시오.

iUserNameLen

입력. 사용자 이름의 길이를 바이트 단위로 지정합니다. 사용자 이름을 제공하지 않는 경우에는 영(0)으로 설정하십시오.

iPasswordLen

입력. 암호 길이를 바이트 단위로 지정합니다. 암호를 제공하지 않은 경우에는 영(0)으로 설정하십시오.

iOverflowLogPathLen

입력. 오버플로우 로그 경로 길이를 바이트 단위로 지정합니다. 오버플로우 로그 경로를 제공하지 않는 경우에는 영(0)으로 설정하십시오.

iDroppedTblIDLen

입력. 삭제된 테이블 ID(piDroppedTblID 매개변수) 길이를 바이트 단위로 지정합니다. 삭제된 테이블 ID를 제공하지 않은 경우에는 영(0)으로 설정하십시오.

iExportDirLen

입력. 삭제된 테이블 익스포트 디렉토리(piExportDir 매개변수) 길이를 바이트 단위로 지정합니다. 삭제된 테이블 익스포트 디렉토리를 제공하지 않은 경우에는 영(0)으로 설정하십시오.

사용 시 참고사항

데이터베이스 관리 프로그램은 아카이브 및 사용 중인 로그 파일에 저장된 정보를 사용하여 마지막으로 백업된 이후 데이터베이스에서 수행된 트랜잭션을 재구성합니다.

이 API가 호출될 때 수행되는 조치는 호출 이전에 데이터베이스의 rollforward_pending 플래그에 따라 다릅니다. db2CfgGet - 구성 매개변수 가져오기를 사용하여 쿼리할 수 있습니다. rollforward_pending 플래그는 데이터베이스가 롤 포워드 보류 상태인 경우 DATABASE로 설정됩니다. 하나 이상의 테이블 스페이스가 SQLB_ROLLFORWARD_PENDING 또는 SQLB_ROLLFORWARD_IN_PROGRESS 상태인 경우 TABLESPACE로 설정됩니다. 데이터베이스나 테이블 스페이스를 롤 포워드할 필요가 없는 경우 rollforward_pending 플래그가 NO로 설정됩니다.

이 API가 호출될 때 데이터베이스가 롤 포워드 보류 상태인 경우 데이터베이스가 롤 포워드됩니다. 비정상 상태로 하나 이상의 테이블 스페이스가 오프라인이 된 경우를 제외하고는 데이터베이스 롤 포워드가 제대로 완료되면 테이블 스페이스는 일반 상태로 리턴됩니다. rollforward_pending 플래그가 TABLESPACE로 설정되면 롤 포워드 보류 상태인 테이블 스페이스 또는 이름으로 요청된 테이블 스페이스만 롤 포워드됩니다.

주: 테이블 스페이스 롤 포워드가 비정상 종료되면 롤 포워드 중이던 테이블 스페이스의 상태는 SQLB_ROLLFORWARD_IN_PROGRESS가 됩니다. ROLLFORWARD DATABASE의 다음 호출에서 SQLB_ROLLFORWARD_IN_PROGRESS 상태의 테이블 스페이스만 처리됩니다. 선택한 테이블 스페이스 이름 세트에 SQLB_ROLLFORWARD_IN_PROGRESS 상태인 모든 테이블 스페이스가 포함되지 않은 경우 필요하지 않은 테이블 스페이스는 SQLB_RESTORE_PENDING 상태가 됩니다.

데이터베이스가 롤 포워드 보류 상태가 아니고 특정 시점이 지정되지 않은 경우 rollforward-in-progress 상태인 모든 테이블 스페이스가 로그 끝으로 롤 포워드됩니다.

rollforward-in-progress 상태인 테이블 스페이스가 없는 경우 롤 포워드 보류 상태인 모든 테이블 스페이스가 로그 끝으로 롤 포워드됩니다.

이 API는 백업 이미지에 일치하는 로그 파일을 시작으로 로그 파일을 읽습니다. 이 로그 파일 이름은 로그 파일을 롤 포워드하기 전에 호출자 조치인 DB2ROLLFORWARD_QUERY가 포함된 이 API를 호출하여 판별할 수 있습니다.

로그 파일에 포함된 트랜잭션이 데이터베이스에 재적용됩니다. 로그는 정보가 사용 가능해진 시점 또는 중지 시간 매개변수로 지정한 시간까지 포워드 처리됩니다.

다음 이벤트 중 하나라도 발생하면 복구가 중지됩니다.

- 로그 파일이 더 이상 없습니다.
- 로그 파일의 시간소인이 중지 시간 매개변수로 지정한 완료 시간소인을 초과했습니다.
- 로그 파일을 읽는 중에 오류가 발생했습니다.

일부 트랜잭션이 복구되지 않을 수도 있습니다. 마지막 커미트에서 리턴된 값은 데이터베이스에 적용된 마지막 커미트된 트랜잭션의 시간소인입니다.

응용프로그램이나 사용자의 실수로 데이터베이스를 복구해야 하는 경우 오류 발생 시점 이전에 복구가 중지되는 것을 나타내는 piStopTime의 시간소인 값을 제공할 수도 있습니다. 이는 전체 데이터베이스 롤 포워드 복구 및 특정 시점의 테이블 스페이스 롤 포워드에만 적용됩니다. 복구 시도 중에 이전 실패로 인해 판별된 사항을 바탕으로 로그 읽기 오류가 발생하기 전에 복구가 중지되도록 합니다.

rollforward_recovery 플래그가 DATABASE로 설정된 경우 롤 포워드 복구가 종료되기 전에는 데이터베이스를 사용할 수 없습니다. 데이터베이스를 롤 포워드 보류 상태에서 해제하는 호출자 조치인 DB2ROLLFORWARD_STOP 또는 DB2ROLLFORWARD_RFWRD_STOP가 포함된 API를 호출하여 종료할 수 있습니다. rollforward_recovery 플래그가 TABLESPACE인 경우 데이터베이스는 사용 가능합니다. 그러나 SQLB_ROLLFORWARD_PENDING 및 SQLB_ROLLFORWARD_IN_PROGRESS 상태의 테이블 스페이스는 테이블 스페이스 롤 포워드 복구를 수행하도록 API를 호출하지 않으면 사용할 수 없습니다. 특정 시점으로 테이블 스페이스를 롤 포워드하는 경우 테이블 스페이스는 롤 포워드가 제대로 완료된 이후에 백업 보류 상태가 됩니다.

RollforwardFlags 옵션을 DB2ROLLFORWARD_LOCAL_TIME으로 설정한 경우 사용자에게 리턴되는 모든 메시지도 로컬 시간입니다. 파티션된 데이터베이스 환경인 경우 서버 및 카탈로그 파티션에서 모든 시간이 변환됩니다. 시간소인 문자열은 서버에서 GMT로 변환되기 때문에 시간은 클라이언트가 아닌 서버의 시간대입니다. 클라이언트와 서버의 시간대가 다른 경우에는 서버의 로컬 시간이 사용되어야 합니다. 이는 클라이언트의 로컬인 제어 센터의 로컬 시간 옵션과는 다릅니다. 시간소인 문자열이 서머타

임제로 인해 클럭의 시간 변경에 가까워지면 중지 시간이 클럭 변경의 이전 또는 이후인지 파악하여 제대로 지정해야 합니다.

제 71 장 db2Runstats - 테이블 및 인덱스의 통계 갱신

테이블 및/또는 연관된 인덱스 또는 통계적 뷰의 특성에 대한 통계를 갱신하십시오. 이 특성에는 다른 많은 특성 중에 레코드 수, 페이지 수 및 평균 레코드 길이가 포함됩니다. 옵티마이저는 데이터에 대한 액세스 경로를 판별할 때 이들 통계를 사용합니다.

테이블에 대해 사용되는 경우, 테이블을 재구성한 후 또는 새 인덱스를 작성한 후 테이블에 갱신사항이 많이 있을 때 이 유틸리티를 호출해야 합니다.

통계는 API가 실행하는 데이터베이스 파티션에 있는 테이블의 분할 영역을 기초로 합니다. 전역 테이블 통계는 데이터베이스 파티션에서 얻은 값에 테이블이 완전히 저장된 데이터베이스 파티션 수를 곱하여 파생됩니다. 전역 통계는 카탈로그 테이블에 저장됩니다. API가 호출되는 데이터베이스 파티션은 테이블의 분할 영역을 포함할 필요가 없습니다.

- 테이블의 분할 영역을 포함하는 데이터베이스 파티션에서 API가 호출되면 유틸리티는 이 데이터베이스 파티션에서 실행됩니다.
- 테이블의 분할 영역을 포함하지 않는 데이터베이스 파티션에서 API가 호출되면 요청은 데이터베이스 파티션 그룹에서 테이블의 분할 영역을 포함하는 첫 번째 데이터베이스 파티션으로 보냅니다. 유틸리티는 이 데이터베이스 파티션에서 실행됩니다. 통계 뷰에 대한 통계를 수집할 때, 통계는 모든 데이터베이스 파티션에 대해 수집됩니다.

통계 뷰에서 사용되는 경우, 기본이 되는 테이블에 대한 변경사항이 뷰에서 리턴된 행에 실질적으로 영향을 줄 때 이 유틸리티를 호출해야 합니다. 이 뷰는 ″ALTER VIEW ... ENABLE QUERY OPTIMIZATION″을 사용하여 쿼리 최적화에 사용될 수 있도록 설정되어야 합니다.

범위

이 API는 db2nodes.cfg 파일의 모든 데이터베이스 파티션 서버에서 호출할 수 있습니다. 카탈로그 데이터베이스 파티션에 있는 카탈로그를 갱신하는 데 사용될 수 있습니다.

권한 부여

테이블에서 사용되는 경우 다음 중 하나이어야 합니다.

- sysadm
- sysctrl
- sysmaint
- dbadm

- sqladm
- 테이블에 대한 CONTROL 특권
- LOAD

통계 뷰에서 사용되는 경우 다음 중 하나이어야 합니다.

- sysadm
- sysctrl
- sysmaint
- dbadm
- sqladm
- 뷰에 대한 CONTROL 특권

또한 사용자는 뷰의 행에 액세스하기 위한 적절한 권한 또는 특권을 가지고 있어야 합니다. 특히, 뷰 정의에 참조된 각 테이블, 뷰 또는 별명에 대해, 사용자는 다음 권한 또는 특권 중 하나를 가지고 있어야 합니다.

- dataaccess
- CONTROL 특권
- SELECT 특권

필수 연결

데이터베이스

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2Runstats (
                db2Uint32 versionNumber,
                void * data,
                struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2RunstatsData
{
  double iSamplingOption;
  unsigned char *piTablename;
  struct db2ColumnData        **piColumnList;
  struct db2ColumnDistData    **piColumnDistributionList;
  struct db2ColumnGrpData     **piColumnGroupList;
  unsigned char               **piIndexList;
  db2Uint32 iRunstatsFlags;
  db2int16 iNumColumns;
  db2int16 iNumColdist;
  db2int16 iNumColGroups;
  db2int16 iNumIndexes;
```

```
  db2int16 iParallelismOption;
  db2int16 iTableDefaultFreqValues;
  db2int16 iTableDefaultQuantiles;
  db2Uint32 iSamplingRepeatable;
  db2Uint32 iUtilImpactPriority;
} db2RunstatsData;

typedef SQL_STRUCTURE db2ColumnData
{
  unsigned char *piColumnName;
  db2int16 iColumnFlags;
} db2ColumnData;

typedef SQL_STRUCTURE db2ColumnDistData
{
  unsigned char *piColumnName;
  db2int16 iNumFreqValues;
  db2int16 iNumQuantiles;
} db2ColumnDistData;

typedef SQL_STRUCTURE db2ColumnGrpData
{
  unsigned char               **piGroupColumnNames;
  db2int16 iGroupSize;
  db2int16 iNumFreqValues;
  db2int16 iNumQuantiles;
} db2ColumnGrpData;

SQL_API_RC SQL_API_FN
  db2gRunstats (
                db2Uint32 versionNumber,
                void * data,
                struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gRunstatsData
{
  double iSamplingOption;
  unsigned char *piTablename;
  struct db2gColumnData       **piColumnList;
  struct db2gColumnDistData   **piColumnDistributionList;
  struct db2gColumnGrpData    **piColumnGroupList;
  unsigned char               **piIndexList;
  db2Uint16 *piIndexNamesLen;
  db2Uint32 iRunstatsFlags;
  db2Uint16 iTablenameLen;
  db2int16 iNumColumns;
  db2int16 iNumColdist;
  db2int16 iNumColGroups;
  db2int16 iNumIndexes;
  db2int16 iParallelismOption;
  db2int16 iTableDefaultFreqValues;
  db2int16 iTableDefaultQuantiles;
  db2Uint32 iSamplingRepeatable;
  db2Uint32 iUtilImpactPriority;
} db2gRunstatsData;

typedef SQL_STRUCTURE db2gColumnData
{
```

```
  unsigned char *piColumnName;
  db2Uint16 iColumnNameLen;
  db2int16 iColumnFlags;
} db2gColumnData;

typedef SQL_STRUCTURE db2gColumnDistData
{
  unsigned char *piColumnName;
  db2Uint16 iColumnNameLen;
  db2int16 iNumFreqValues;
  db2int16 iNumQuantiles;
} db2gColumnDistData;

typedef SQL_STRUCTURE db2gColumnGrpData
{
  unsigned char               **piGroupColumnNames;
  db2Uint16 *piGroupColumnNamesLen;
  db2int16 iGroupSize;
  db2int16 iNumFreqValues;
  db2int16 iNumQuantiles;
} db2gColumnGrpData;
```

db2Runstats API 매개변수

versionNumber

입력. 두 번째 매개변수 데이터로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

데이터 입력. db2RunstatsData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2RunstatsData 데이터 구조 매개변수

iSamplingOption

입력. 통계가 테이블 또는 뷰 데이터 샘플에서 수집됨을 표시합니다. iSamplingOption은 샘플 *크기*를 백분율 P로 표시합니다. 이 값은 100 이하의 양수여야 하지만, 1 및 0 사이일 수도 있습니다. 예를 들어, 0.01 값은 백분율의 1/100을 나타냅니다(평균적으로 10,000의 1 행이 샘플링됨). 값 0 또는 100은 DB2RUNSTATS_SAMPLING_SYSTEM이 지정되었는지 여부에 관계없이 DB2에 의해 샘플링이 지정되지 않은 것처럼 처리됩니다. 100보다 *크거나* 0보다 작은 값은 DB2에 의해 오류(SQL1197N)로 처리됩니다. 샘플링의 가능한 두 유형은 BERNOULLI 및 SYSTEM입니다. 샘플링 유형 스펙은 iRunstatsFlags에 표시된 DB2RUNSTATS_SAMPLING_SYSTEM 설정으로 제어됩니다.

piTablename

입력. 통계가 수집될 테이블 또는 통계 뷰의 완전한 이름에 대한 포인터. 이름은 별명이 될 수 있습니다. 행 유형의 경우 piTablename은 계층 구조의 루트 테이블 이름이어야 합니다.

piColumnList

입력. db2ColumnData 요소의 배열. 이 배열의 각 요소는 두 개의 하위 요소로 구성됩니다.

- 통계를 수집할 컬럼의 이름을 표시하는 문자열.
- 컬럼의 통계 옵션을 표시하는 플래그 필드

iNumColumns가 0이면 piColumnList를 제공해도 무시됩니다.

piColumnDistributionList

입력. db2ColumnDistData 요소의 배열. 이 요소는 특정 컬럼에 대한 분산 통계를 수집하려고 할 때 제공됩니다. 이 배열의 각 요소는 세 개의 하위 요소로 구성됩니다.

- 분산 통계를 수집할 컬럼의 이름을 표시하는 문자열.
- 수집할 자주 사용되는 값 수.
- 수집할 Quantile 수.

piColumnLis에는 표시되지 않고 piColumnDistributionList에는 표시되는 컬럼은 기본 컬럼 통계가 수집됩니다. 효과는 첫 번째 위치에서 piColumnList에 이 컬럼을 포함시키는 것과 같습니다. iNumColdist가 0이면 piColumnDistributionList는 무시됩니다.

piColumnGroupList

입력. db2ColumnGrpData 요소의 배열. 이 요소는 컬럼 그룹에 대해 컬럼 통계를 수집할 때 제공됩니다. 즉, 행마다 그룹의 각 컬럼에 있는 값을 함께 연결하여 단일 값으로 처리합니다. 각각의 db2ColumnGrpData는 세 개까지의 정수 필드와 문자열 배열로 구성됩니다. 첫 번째 정수 필드는 piGroupColumns 문자열 배열의 문자열 수를 나타냅니다. 이 배열의 각 문자열에는 하나의 컬럼 이름이 포함됩니다. 예를 들어, 컬럼 조합 통계가 컬럼 그룹 (c1,c2) 및 (c3,c4,c5)에서 수집되는 경우 piGroupColumns에는 두 개의 db2ColumnGrpData 요소가 있습니다.

첫 번째 db2ColumnGrpData 요소는 piGroupSize = 2이고 문자열 배열에는 두 개의 요소인 c1 및 c2가 포함됩니다.

두 번째 db2ColumnGrpData 요소는 piGroupSize = 3이고 문자열 배열에는 세 개의 요소인 c3, c4 및 c5가 포함됩니다.

두 번째 및 세 번째 정수 필드는 컬럼 그룹에 대해 분산 통계를 수집할 때 각각 자주 사용되는 값 수와 Quantile 수를 나타냅니다. 이는 현재 지원되지 않습니다.

piColumnLis에 표시되지 않지만 piColumnGroupList에는 표시되는 컬럼은 기본 컬럼 통계가 수집됩니다. 효과는 첫 번째 위치에서 piColumnList에 이 컬럼을 포함시키는 것과 같습니다. iNumColGroups가 0이면 piColumnGroupList는 무시됩니다.

piIndexList

입력. 문자열 배열. 각 문자열에는 하나의 완전한 인덱스 이름이 포함됩니다. NumIndexes가 0이면 piIndexList는 무시됩니다.

iRunstatsFlags

입력. 통계 옵션을 지정하기 위해 사용되는 비트 마스크 필드. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2RUNSTATS_ALL_COLUMNS

테이블 또는 통계 뷰의 모든 컬럼에 대해 통계를 수집합니다. 이 옵션은 컬럼, 컬럼 분산, 컬럼 그룹 또는 인덱스 구조 목록과 조합으로 지정할 수 있습니다. 이는 테이블이나 뷰의 모든 컬럼에 대해 통계를 수집하지만 특정 컬럼에 대해 통계 옵션을 제공하려고 하는 경우에 유용합니다.

DB2RUNSTATS_KEY_COLUMNS

테이블에 정의된 모든 인덱스로 구성되는 컬럼에 대한 통계만 수집합니다. 이 옵션은 통계 뷰에 사용할 수 없습니다. 테이블에서는 컬럼, 컬럼 분산, 컬럼 그룹 또는 인덱스 구조 목록과 조합으로 지정할 수 있습니다. 이는 테이블의 모든 키 컬럼에 대해 통계를 수집하지만 일부 키가 아닌 컬럼에 대한 통계도 수집하거나 특정 키 컬럼에 대해 통계 옵션을 제공하려고 하는 경우에 유용합니다. 정의에 따라, XML 유형 컬럼은 키 컬럼이 아니므로 iRunstatsFlags 매개변수가 DB2RUNSTATS_KEY_COLUMNS 값으로 설정될 때 통계 콜렉션에 포함되지 않습니다.

DB2RUNSTATS_DISTRIBUTION

분산 통계를 수집합니다. 이 옵션은 단지 DB2RUNSTATS_ALL_COLUMNS 및 DB2RUNSTATS_KEY_COLUMNS와 함께 사용할 수 있습니다. DB2RUNSTATS_ALL_COLUMNS와 함께 사용하는 경우, 테이블 또는 통계 뷰의 모든 컬럼에 대해 분산 통계가 수집됩니다. DB2RUNSTATS_KEY_COLUMNS와 함께 사용하는 경우, 테이블에 정의된 모든 인덱스로 구성되는 모든 컬럼에 대해 분산 통계가 수집됩니다. DB2RUNSTATS_ALL_COLUMNS 및

DB2RUNSTATS_KEY_COLUMNS와 함께 사용하는 경우, 테이블의 모든 컬럼에 대해 기본 통계가 수집되고 테이블에 정의된 모든 인덱스로 구성되는 컬럼에 대해서만 분산 통계가 수집됩니다.

DB2RUNSTATS_ALL_INDEXES

테이블에 정의된 모든 인덱스에 대해 통계를 수집합니다. 이 옵션은 통계 뷰에 사용할 수 없습니다.

DB2RUNSTATS_EXT_INDEX

자세한 인덱스 통계를 수집합니다. 옵션은 DB2RUNSTATS_ALL_INDEXES나 명시적 인덱스 이름 목록(piIndexList 및 iNumIndexes > 0)과 함께 지정해야 합니다. 이 옵션은 통계 뷰에 사용할 수 없습니다.

DB2RUNSTATS_EXT_INDEX_SAMPLED

샘플링 메소드를 사용하여 자세한 인덱스 통계를 수집합니다. 옵션은 DB2RUNSTATS_ALL_INDEXES나 명시적 인덱스 이름 목록(piIndexList 및 iNumIndexes > 0)과 함께 지정해야 합니다. DB2RUNSTATS_EXT_INDEX는 동시에 지정된 경우 무시됩니다. 이 옵션은 통계 뷰에 사용할 수 없습니다.

DB2RUNSTATS_ALLOW_READ

통계가 수집되는 동안 다른 사용자가 읽기 전용 액세스를 갖도록 허용합니다. 디폴트는 읽기 및 쓰기 액세스를 허용하는 것입니다.

DB2RUNSTATS_SAMPLING_SYSTEM

iSamplingOption 매개변수를 통해 사용자가 지정한 데이터 페이지의 백분율에 대한 통계를 수집합니다. SYSTEM 샘플링은 각 페이지를 개별적으로 검토하여 확률이 P/100인 페이지는 포함시키고(여기서 P는 iSamplingOption의 값임) 확률이 1 - P/100인 페이지는 제외시킵니다. 따라서 iSamplingOption의 값이 10(10퍼센트 샘플을 나타냄)인 경우 확률이 0.1인 각 행은 포함되고 확률이 0.9인 행은 제외됩니다.

통계 뷰에서, SYSTEM 샘플링은 정의가 단일 기본 테이블을 통한 선택인 뷰로 제한됩니다. 뷰가 다중 테이블을 포함하면, 다음의 경우 SYSTEM 샘플링도 가능합니다.

- 테이블 간에 정의된 참조 무결성 제한조건에 포함된 모든 기본 키와 외부 키에 대해 등호 술어를 사용하여 테이블이 조인됩니다.
- 어떤 검색 조건도 관계의 모든 상위 테이블 행을 필터하지 않습니다.
- 상위 테이블이 아닌 단일 하위 테이블은 모든 테이블 간에 식별될 수 있습니다.

통계 뷰가 이들 조건을 충족시키지 않는 경우, BERNOULLI 샘플링이 대신 사용되며 경고가 리턴됩니다(SQL2317W).

DB2RUNSTATS_SAMPLING_SYSTEM을 지정하지 않은 경우, DB2는 샘플링 메소드로 BERNOULLI 샘플링이 사용될 것으로 가정합니다. BERNOULLI 샘플링은 각 행을 개별적으로 고려하여, 확률이 P/100인 행은 포함시키고(여기서 P는 iSamplingOption의 값임) 확률이 1 - P/100인 행은 제외시킵니다.

DB2RUNSTATS_SAMPLING_REPEAT 플래그가 지정된 경우를 제외하고는 SYSTEM 및 BERNOULLI 샘플링 모두에서, 통계 수집이 실행될 때마다 보통 다른 테이블 또는 통계 뷰 샘플이 생성됩니다.

DB2RUNSTATS_SAMPLING_REPEAT

시드(seed)가 iSamplingRepeatable 매개변수를 통해 전달되었음을 지정합니다. iSamplingRepeatable 값은 데이터 샘플을 생성하기 위한 시드(seed)로 사용됩니다. 샘플링 비율을 표시하려면 iSamplingOption 매개변수도 지정해야 합니다.

DB2RUNSTATS_USE_PROFILE

테이블 또는 뷰의 카탈로그에 이미 등록된 통계 프로파일을 사용하여 테이블 또는 통계 뷰에 대한 통계를 수집합니다. iRunstatsFlags 비트 마스크에 설정된 이 플래그에 의해 USE PROFILE 옵션이 지정되면, db2RunstatsData에 있는 다른 모든 옵션이 무시됩니다.

DB2RUNSTATS_SET_PROFILE

지정된 통계 옵션을 기록하는 카탈로그에서 프로파일을 생성하여 저장하고 동일한 옵션을 사용하여 통계를 수집합니다.

DB2RUNSTATS_SET_PROFILE_ONLY

테이블 또는 뷰에 대해 실제로 통계를 수집하지 않고 지정된 통계 옵션을 기록하는 카탈로그에서 프로파일을 생성하여 저장합니다.

DB2RUNSTATS_UNSET_PROFILE

통계 프로파일 설정 취소는 SYSCAT.STATISTICS_PROFILE을 NULL로 설정하여 시스템 카탈로그에서 통계 프로파일을 제거합니다. 통계 프로파일이 존재하지 않으면 설정 취소하려고 할 때 오류가 발생합니다(SQLCODE -2315).

DB2RUNSTATS_UPDATE_PROFILE

카탈로그에서 기존 통계 프로파일을 수정하고 갱신된 프로파일에서 옵션을 사용하여 통계를 수집합니다.

DB2RUNSTATS_UPDA_PROFILE_ONLY

테이블 또는 뷰에 대해 실제로 통계를 수집하지 않고 카탈로그에서 기존 통계 프로파일을 수정합니다.

DB2RUNSTATS_EXCLUDING_XML

XML 유형 컬럼에 대해 통계를 수집하지 않습니다. 통계는 XML이 아닌 유형을 가지고 있는 지정된 모든 컬럼에 대해 계속 수집됩니다. 이 옵션은 XML 컬럼을 지정하는 다른 모든 메소드보다 우선합니다.

iNumColumns

입력. piColumnList 목록에 지정된 항목 수

iNumColdist

입력. piColumnDistributionList 목록에 지정된 항목 수

iNumColGroups

입력. piColumnGroupList 목록에 지정된 항목 수

iNumIndexes

입력. piIndexList 목록에 지정된 항목 수

iParallelismOption

입력. 나중에 사용하기 위해 예약됩니다. 유효한 값은 0입니다.

iTableDefaultFreqValues

입력. 테이블 또는 뷰에 대해 수집할 디폴트 자주 사용되는 값 수를 지정합니다. 가능한 값은 다음과 같습니다.

n 컬럼 레벨에서 달리 지정하지 않으면 n개의 자주 사용되는 값이 수집됩니다.

0 컬럼 레벨에서 달리 지정하지 않으면 자주 사용되는 값이 수집되지 않습니다.

-1 수집할 자주 사용되는 값 수에 대해 디폴트 데이터베이스 구성 매개변수 NUM_FREQVALUES를 사용합니다.

iTableDefaultQuantiles

입력. 테이블 또는 뷰에 대해 수집할 디폴트 Quantile 수를 지정합니다. 가능한 값은 다음과 같습니다.

n 컬럼 레벨에서 달리 지정하지 않으면 n개의 Quantile이 수집됩니다.

0 컬럼 레벨에서 달리 지정하지 않으면 Quantile이 수집되지 않습니다.

-1 수집할 Quantile 수에 대해 디폴트 데이터베이스 구성 매개변수 NUM_QUANTILE를 사용합니다.

iSamplingRepeatable

입력. 테이블 또는 뷰 샘플링에 사용될 시드(seed)를 나타내는 음수가 아닌 정수. 음수 시드를 패스하면 오류가 발생합니다(SQL1197N).

이 시드(seed)를 사용하려면 DB2RUNSTATS_SAMPLING_REPEAT 플래그를 설정해야 합니다. 이 옵션은 후속 통계 콜렉션에서 동일한 데이터 샘플을 생

성하기 위해 iSamplingOption 매개변수와 함께 사용됩니다. 반복 가능 요청이 마지막으로 실행된 후 테이블 또는 뷰에 대한 활동으로 테이블 또는 뷰 데이터가 변경된 경우 샘플 세트는 반복 가능한 요청 사이에서 계속 다를 수 있습니다. 또한 일치된 결과를 얻기 위해서는 샘플이 확보된 메소드(BERNOULLI 또는 SYSTEM)도 같아야 합니다.

iUtilImpactPriority

입력. runstats 호출에 대한 우선순위. 유효한 값은 범위 0 - 100에 속해야 하며 70은 조절하지 않은 상태를 나타내고 100은 최상의 가능한 우선순위를 나타냅니다. 이 옵션은 통계 뷰에 사용할 수 없습니다.

db2ColumnData 데이터 구조 매개변수

piColumnName

입력. 컬럼 이름을 나타내는 문자열에 대한 포인터

iColumnFlags

입력. 컬럼에 대한 통계 옵션을 지정하기 위해 사용되는 비트 마스크 필드. 가능한 값은 다음과 같습니다.

DB2RUNSTATS_COLUMN_LIKE_STATS
컬럼에 대해 LIKE 통계를 수집합니다.

db2ColumnDistData 데이터 구조 매개변수

piColumnName

입력. 컬럼 이름을 나타내는 문자열에 대한 포인터

iNumFreqValues

입력. 컬럼에 대해 수집할 자주 사용되는 값 수. 가능한 값은 다음과 같습니다.

n 컬럼에 대해 n개의 자주 사용되는 값을 수집합니다.

-1 테이블 디폴트 자주 사용되는 값 수(예: iTableDefaultFreqValues(설정된 경우)), 또는 데이터베이스 구성 매개변수 NUM_FREQVALUES를 사용합니다.

iNumQuantiles

입력. 컬럼에 대해 수집할 Quantile 수. 가능한 값은 다음과 같습니다.

n 컬럼에 대해 n개의 Quantile을 수집합니다.

-1 테이블 디폴트 Quantile 수(iTableDefaultQuantiles(설정된 경우)), 또는 데이터베이스 구성 매개변수 NUM_QUANTILES를 사용합니다.

db2ColumnGrpData 데이터 구조 매개변수

piGroupColumnNames

입력. 문자열 배열. 각 문자열은 통계를 수집할 컬럼 그룹의 일부인 컬럼 이름을 나타냅니다.

iGroupSize

입력. 컬럼 그룹의 컬럼 수. 가능한 값은 다음과 같습니다.

n 컬럼 그룹은 n개의 컬럼으로 구성됩니다.

iNumFreqValues

입력. 나중에 사용하기 위해 예약됩니다.

iNumQuantiles

입력. 나중에 사용하기 위해 예약됨

db2gRunstatsData 데이터 구조 특정 매개변수

piIndexNamesLen

입력. 인덱스 목록에 있는 인덱스 이름 각각의 길이(바이트)를 나타내는 값 배열. NumIndexes가 0이면 piIndexNamesLen은 무시됩니다.

iTablenameLen

입력. 테이블 또는 뷰 이름의 길이(바이트 단위)를 나타내는 값.

db2gColumnData 데이터 구조 특정 매개변수

iColumnNameLen

입력. 컬럼 이름의 길이(바이트 단위)를 나타내는 값.

db2gColumnDistData 데이터 구조 특정 매개변수

iColumnNameLen

입력. 컬럼 이름의 길이(바이트 단위)를 나타내는 값.

db2gColumnGrpData 데이터 구조 특정 매개변수

piGroupColumnNamesLen

입력. 컬럼 이름 목록에 있는 컬럼 이름 각각의 길이(바이트)를 나타내는 값 배열.

사용 시 참고사항

다음의 경우에 통계를 갱신하려면 db2Runstats를 사용하십시오.

- 여러 번 수정된 테이블의 경우(예를 들어, 여러 번 갱신되었거나 상당한 데이터 양이 삽입 또는 삭제된 경우)
- 재구성된 테이블의 경우
- 새 인덱스가 작성된 경우

- 기본이 되는 테이블이 실질적으로 수정되어 뷰에서 리턴되는 행이 변경되는 뷰의 경우

통계가 갱신된 후 sqlabndx - 바인드를 사용하여 패키지를 리바인드하여 테이블에 대한 새 액세스 경로를 작성할 수 있습니다.

인덱스 통계가 요청되었으며 인덱스가 있는 테이블에서 통계가 실행된 적이 없는 경우, 테이블 및 인덱스에 대한 통계가 계산됩니다.

db2Runstats API가 인덱스에 대해서만 통계를 수집하는 경우 이전에 수집된 분산 통계는 보유됩니다. 그렇지 않으면, API는 이전에 수집된 분산 통계를 삭제합니다. db2Runstats API가 XML 컬럼에 대해서만 통계를 수집하는 경우 이전에 수집된 기본 컬럼 통계 및 분산 통계는 보유됩니다. 일부 XML 컬럼에 대한 통계가 이전에 수집된 경우, 최신 db2Runstats API 호출로 해당 XML 컬럼에 대한 통계가 수집되지 않으면 XML 컬럼에 대해 이전에 수집된 통계는 삭제되며 최신 db2Runstats API 호출로 해당 XML 컬럼에 대한 통계가 수집되면 이전에 수집된 통계를 교체합니다. 그렇지 않으면, API는 이전에 수집된 분산 통계를 삭제합니다.

iRunstatsFlags 매개변수가 값 DB2RUNSTATS_EXCLUDING_XML로 설정되면, 통계는 XML 컬럼에 대해 수집되지 않습니다. 이 값은 XML 컬럼을 지정하는 다른 모든 메소드보다 우선합니다.

이 API를 호출한 후, 응용프로그램은 COMMIT를 발행하여 잠금을 해제해야 합니다.

새 액세스 플랜이 생성될 수 있도록 하려면 API 호출 후에 목표 테이블을 참조하는 패키지를 리바인드해야 합니다.통계 뷰를 이용할 수 있는 쿼리가 있는 패키지도 이와 같은 뷰에 대한 통계를 갱신한 후에는 리바인드해야 합니다.

통계 뷰에 대해 통계가 수집될 때 SQL 쿼리는 내부적으로 실행됩니다. EXPLAIN 기능을 사용하여 이 쿼리에 대해 선택된 액세스 플랜을 조사하여 통계 콜렉션에 성능 문제점이 있는지 조사할 수 있습니다. EXPLAIN 테이블에서 쿼리 액세스 플랜을 저장하려면, CURRENT EXPLAIN MODE 특수 레지스터를 YES로 설정하십시오.

테이블에서만 이 API를 실행하면 테이블 레벨 통계가 기존 인덱스 레벨 통계와 일치하지 않은 상황이 발생할 수 있습니다. 예를 들어, 특정 테이블에 대해 인덱스 레벨 통계가 수집되고 나중에 테이블에서 많은 수의 행이 삭제되는 경우, 테이블에 대해서만 이 API를 발행하면 테이블 카디널리티(cardinality)가 FIRSTKEYCARD (FIRSTKEYCARD는 SYSCAT.INDEXES 및 SYSSTAT.INDEXES 카탈로그 뷰의 카탈로그 통계 필드임)보다 적게 됩니다. 이는 불일치 상태입니다. 마찬가지로, 인덱스에 대해서만 이 API를 발행하면 기존의 테이블 레벨 통계가 불일치 상태로 남아 있을 수 있습니다. 예를 들어, 특정 테이블에서 테이블 레벨 통계를 수집하고 나중에 이 테이블에서 많은 수의 행을 삭제한 경우, 인덱스에 대해서만 db2Runstats API를 발행하면 COLCARD(COLCARD는 SYSCAT.COLUMNS 및 SYSSTAT.COLUMNS 카

탈로그 뷰의 카탈로그 통계 필드임)가 있는 일부 컬럼이 테이블 카디널리티보다 크게 됩니다. 이와 같은 불일치가 발견되면 경고가 리턴됩니다.

구조화된 유형이 있는 컬럼에 대한 통계는 수집되지 않습니다. 통계가 지정되면, 해당 데이터 유형이 있는 컬럼은 무시됩니다.

LOB 또는 LONG 데이터 유형이 있는 컬럼에 대한 AVGCOLLEN 및 NUMNULLS 만 수집됩니다.

컬럼이 데이터베이스 메모리 또는 임시 테이블에 저장될 때 AVGCOLLEN은 바이트 단위로 평균 스페이스를 표시합니다. LOB 데이터가 데이터 페이지에서 인라인될 때를 제외하고 이 값은 LOB 또는 LONG 데이터 유형의 데이터 디스크립터의 길이를 표시합니다.

주: 디스크에서 컬럼을 저장하기 위해 필요한 평균 스페이스는 이 통계로 표시되는 값과 다를 수 있습니다.

제 72 장 db2SelectDB2Copy - 응용프로그램에서 사용되는 DB2 사본 선택

응용프로그램이 지정된 특정 DB2 사본 또는 위치를 사용하는 데 필요한 환경을 설정합니다. 환경에서 사용하려는 DB2 사본을 이미 설정한 경우 이 API를 호출할 필요가 없습니다. 그러나 다른 DB2 사본을 사용해야 하는 경우에는 이 API를 호출해야 합니다. 프로세스에서 DB2 dll 파일을 로드하기 전에 이 API를 호출하십시오. 이 호출은 프로세스별로 하나만 수행할 수 있습니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiInstall.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2SelectDB2Copy (
   db2Uint32 versionNumber,
   void *pDB2SelectDB2CopyStruct);

typedef enum DB2CopyParmType
{
   DB2CopyInvalid=0,
   DB2CopyName,
   DB2CopyPath
} db2CopyParmType;

typedef struct DB2SelectDB2CopyStruct
{
   DB2CopyParmType  Type;
   char *pszDB2Copy;
} db2SelectDB2CopyStruct
```

db2SelectDB2Copy API 매개변수

versionNumber

입력. 두 번째 매개변수인 pDB2SelectInstallationStruct로 전달된 변수의 버전 번호 및 릴리스 레벨을 지정합니다.

pDB2SelectDB2CopyStruct

입력. DB2SelectDB2CopyStruct 구조의 포인터

DB2SelectDB2CopyStruct 데이터 구조 매개변수

유형 입력. 이는 DB2CopyName 또는 DB2CopyPath일 수 있습니다.

psziDB2Copy

입력. Type이 DB2CopyName으로 지정된 경우 psziDB2Copy는 DB2 사본의 이름입니다. Type이 db2CopyPath로 지정된 경우 psziDB2Copy는 DB2 설치 경로입니다. NULL도 가능합니다.

사용 시 참고사항

API를 사용하려면 응용프로그램이 정적으로 db2ApiInstall.lib에 강제로 링크하는 db2ApiInstall.h를 포함해야 합니다.

또한 이 API는 DB2 라이브러리를 로드하기 전에 호출해야 하며 응용프로그램에서 한 번만 호출할 수 있습니다. DB2 라이브러리를 링크할 때 /delayload 옵션을 사용하여 DB2 라이브러리를 로드하지 않거나 LoadLibraryEx를 사용하고 LOAD_WITH_ALTERED_SEA를 지정하여 이 라이브러리를 동적으로 로드할 수도 있습니다.

제 73 장 db2SetSyncSession - Satellite 동기화 세션 설정

Satellite의 동기화 세션을 설정합니다. 동기화 세션은 Satellite에서 실행 중인 사용자 응용프로그램의 버전과 연관됩니다. 응용프로그램의 각 버전은 특정 데이터베이스 구성으로 지원되며 특정 데이터 세트를 조작하고 각각은 중심 사이트에서 동기화할 수 있습니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2SetSyncSession (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2SetSyncSessionStruct
{
   char *piSyncSessionID;
} db2SetSyncSessionStruct;
```

db2SetSyncSession API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2SetSyncSessionStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2SetSyncSessionStruct 데이터 구조 매개변수

piSyncSessionID

입력. Satellite가 사용하는 동기화 세션 ID를 지정합니다. 지정된 값은 Satellite 제어 서버에 정의된 대로 Satellite 그룹에 대해 해당 응용프로그램 버전과 일치해야 합니다.

제 74 장 db2SetWriteForDB - 데이터베이스에 대한 입출력 쓰기 일시중단 또는 다시 시작

입출력 쓰기가 일시중단되는 데이터베이스를 설정하거나 디스크에 입출력 쓰기를 다시 시작합니다. 입출력 쓰기는 미러 분할을 수행하기 전에 데이터베이스에 대해 일시중단되어야 합니다. 잠재적으로 발생할 수 있는 문제를 방지하려면 쓰기 일시중단과 다시 시작을 수행하는 데 동일한 연결을 사용하십시오.

범위

이 명령은 실행되는 데이터베이스 파티션에만 영향을 미칩니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

데이터베이스

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2SetWriteForDB (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef struct db2SetWriteDbStruct
{
   db2int32 iOption;
   char *piTablespaceNames;
} db2SetWriteDbStruct;
```

db2SetWriteForDB API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2SetWriteDbStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2SetWriteDbStruct 데이터 구조 매개변수

iOption

입력. 조치를 지정합니다. 가능한 값은 다음과 같습니다.

- **DB2_DB_SUSPEND_WRITE**

  디스크에 대한 입출력 쓰기를 일시중단합니다.

- **DB2_DB_RESUME_WRITE**

  디스크에 입출력 쓰기를 다시 시작합니다.

piTablespaceNames

입력. 나중에 사용하기 위해 예약됩니다.

제 75 장 db2SpmListIndTrans - SPM 인다우트(Indoubt) 트랜잭션 나열

동기점 관리 프로그램에서 인다우트(Indoubt) 상태인 트랜잭션의 목록을 제공합니다.

범위

이 API는 발행된 데이터베이스 파티션에만 영향을 줍니다.

권한 부여

없음

필수 연결

동기점 관리 프로그램에 연결

API 내장 파일

```
sqlxa.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
db2SpmListIndTrans (
  db2Uint32 versionNumber,
  void * pParmStruct,
  struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2SpmListIndTransStruct
{
db2SpmRecoverStruct * piIndoubtData;
db2Uint32            iIndoubtDataLen;
db2Uint32            oNumIndoubtsReturned;
db2Uint32            oNumIndoubtsTotal;
db2Uint32            oReqBufferLen;
} db2XaListIndTransStruct;

typedef SQL_STRUCTURE db2SpmRecoverStruct
{
   SQLXA_XID    xid;
   char         luwid[SQLCSPQY_LUWID_SZ+1];
   char         corrtok[SQLCSPQY_APPLID_SZ+1];
   char         partner[SQLCSPQY_LUNAME_SZ+1];
   char         dbname[SQLCSPQY_DBNAME_SZ+1];
   char         dbalias[SQLCSPQY_DBNAME_SZ+1];
   char         role;
   char         uow_status;
   char         partner_status;
} db2SpmRecoverStruct;
```

db2SpmListIndTrans API 매개변수

versionNumber

입력. 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2SpmListIndTransStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2SpmListIndTransStruct 데이터 구조 매개변수

piIndoubtData

입력. 인다우트(Indoubt) 데이터가 리턴되는 응용프로그램 입력 버퍼의 포인터. 인다우트(Indoubt) 데이터는 db2SpmRecoverStruct 형식입니다. 응용프로그램은 이 매개변수에서 제공되는 주소를 시작으로 db2SpmRecoverStruct 구조 크기를 사용하여 인다우트(Indoubt) 트랜잭션의 목록을 트래버스합니다. 값이 NULL인 경우 필요한 버퍼 크기가 계산되어 oReqBufferLen에 리턴됩니다. oNumIndoubtsTotal 에서는 총 인다우트(Indoubt) 트랜잭션 수가 포함됩니다. 응용프로그램은 필요한 버퍼 크기를 할당하고 API를 다시 발행합니다.

oNumIndoubtsReturned

출력. pIndoubtData에서 지정된 버퍼에 리턴되는 인다우트(Indoubt) 트랜잭션 레코드 수

oNumIndoubtsTotal

출력. API 호출 시에 사용 가능한 총 인다우트(Indoubt) 트랜잭션 레코드 수 piIndoubtData 버퍼가 너무 작아서 모든 레코드를 포함하지 못하는 경우 oNumIndoubtsTotal이 oNumIndoubtsReturned에 대한 총계보다 크게 됩니다. 응용프로그램은 모든 레코드를 확보하기 위해 API를 재발행합니다.

이 수는 자동 또는 경험적 인다우트(Indoubt) 트랜잭션 재동기화 결과 또는 인다우트(Indoubt) 상태가 되는 다른 트랜잭션 결과로 API 호출 간에 변경될 수 있습니다.

oReqBufferLen

출력. API 호출 시에 모든 인다우트(Indoubt) 트랜잭션 레코드를 보유하는 데 필요한 버퍼 길이. 응용프로그램은 pIndoubtData를 NULL로 설정하고 API를 호출하여 필요한 버퍼 크기를 판별하는 데 이 값을 사용할 수 있습니다. 그러면 이 값을 사용하여 필요한 버퍼를 할당하고 API는 pIndoubtData를 할당된 버퍼 주소로 설정하여 발행할 수 있습니다.

필요한 버퍼 크기는 자동 또는 경험적 인다우트(Indoubt) 트랜잭션 재동기화 결과 또는 인다우트(Indoubt) 상태가 되는 다른 트랜잭션 결과로 API 호출 간에 변경될 수 있습니다. 이를 위해 응용프로그램은 더 큰 버퍼를 어카운트에 할당할 수도 있습니다.

db2SpmRecoverStruct 데이터 구조 매개변수

xid 출력. 트랜잭션 관리 프로그램이 전역 트랜잭션을 고유하게 식별하기 위해 지정하는 XA ID를 지정합니다.

luwid 출력. 동기점 관리 프로그램이 상대 시스템에서 XA ID(XID)를 식별하기 위해 지정하는 논리적 작업 단위(LUWID)를 지정합니다.

corrtok

출력. 이 트랜잭션을 위해 동기점 관리 프로그램이 지정하는 응용프로그램 ID를 지정합니다.

partner

출력. 상대 시스템 이름을 지정합니다.

dbname

출력. 상대 시스템의 데이터베이스

dbalias

출력. 인다우트(Indoubt) 트랜잭션이 있는 데이터베이스 별명을 지정합니다.

role 출력. 동기점 관리 프로그램의 역할

SQLCSPQY_AR

동기점 관리 프로그램은 응용프로그램 리퀘스터(AR)입니다.

SQLCSPQY_AS

동기점 관리 프로그램은 응용프로그램 서버(AS)입니다.

uow_status

출력. 동기점 관리 프로그램에서 이 인다우트(Indoubt) 트랜잭션의 상태를 표시합니다. 가능한 값은 다음과 같습니다.

SQLCSPQY_STATUS_COM

트랜잭션은 동기점 관리 프로그램에서 커미트 상태입니다. 트랜잭션은 다음 재동기화 간격에서 상대 시스템의 재동기화를 대기 중입니다.

SQLCSPQY_STATUS_RBK

트랜잭션은 동기점 관리 프로그램에서 롤백 상태입니다. 상대 시스템이 재동기화를 시작하고 인다우트(Indoubt)를 해결하도록 대기합니다.

SQLCSPQY_STATUS_IDB

트랜잭션은 동기점 관리 프로그램에서 준비 상태입니다. 연결된 매개변

수를 사용하여 트랜잭션이 일반 커미트 처리의 두 번째 단계를 대기 중인지 또는 오류가 발생하여 트랜잭션 관리 프로그램을 재동기화해야 하는지 판별할 수 있습니다.

SQLCSPQY_STATUS_HCM
트랜잭션이 경험적으로 커미트되었습니다.

SQLCSPQY_STATUS_HRB
트랜잭션이 경험적으로 롤백되었습니다.

사용 시 참고사항

일반 응용프로그램은 현재 연결을 동기점 관리 프로그램으로 설정한 후에 다음을 수행합니다.*

1. piIndoubtData를 NULL로 설정하여 db2SpmListIndTrans API를 호출합니다. 그러면 oReqBufferLen 및 oNumIndoubtsTotal에 값이 리턴됩니다.
2. oReqBufferLen에 리턴된 값을 사용하여 버퍼를 할당하십시오. oReqBufferLen 확보를 위한 이 API의 초기 호출로 인해 추가 인다우트(Indoubt) 트랜잭션이 있는 경우 이 버퍼의 크기가 충분하지 않을 수도 있습니다. 응용프로그램은 oReqBufferLen 보다 큰 버퍼를 제공할 수 있습니다.
3. 모든 인다우트(Indoubt) 트랜잭션 레코드가 확보되었는지 판별하십시오. 이는 oNumIndoubtsReturned를 oNumIndoubtsTotal에 비교하여 수행할 수 있습니다. oNumIndoubtsTotal이 oNumIndoubtsReturned보다 큰 경우 응용프로그램은 위의 단계를 반복할 수 있습니다.

* 동기점 관리 프로그램에 연결하려면 DB2 Connect 서버에서 사용 중인 동기점 관리 프로그램 이름을 판별하십시오. 이는 데이터베이스 구성 매개변수 spm_name를 DB2 Connect 서버에서 쿼리하여 판별할 수 있습니다. spm_name을 연결 API의 데이터베이스 별명으로 지정하여 연결을 발행하십시오.

제 76 장 db2SyncSatellite - Satellite 동기화 시작

Satellite를 동기화합니다. Satellite의 동기화를 위해서는 Satellite를 해당 그룹의 다른 Satellite와 일관성 있는 상태로 설정해야 합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2SyncSatellite (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

db2SyncSatellite API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. NULL로 설정하십시오.

pSqlca

출력. sqlca 구조의 포인터

제 77 장 db2SyncSatelliteStop - Satellite 동기화 일시정지

현재 활성화되어 있는 Satellite의 동기화 세션을 중지합니다. 이 Satellite의 동기화가 사용되지 않는 경우 db2SyncSatellite를 호출하여 다시 시작할 수 있는 방식으로 세션이 중지됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2SyncSatelliteStop (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

db2SyncSatelliteStop API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. NULL로 설정하십시오.

pSqlca

출력. sqlca 구조의 포인터

제 78 장 db2SyncSatelliteTest - Satellite 동기화 가능 여부 테스트

Satellite의 동기화 기능을 테스트합니다. 즉, Satellite가 해당 그룹의 다른 Satellite와 일관성이 있는 상태로 설정할 수 있는지 여부를 테스트합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2SyncSatelliteTest (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);
```

db2SyncSatelliteTest API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. NULL로 설정하십시오.

pSqlca

출력. sqlca 구조의 포인터

제 79 장 db2UpdateAlertCfg - Health 표시기의 경보 구성 설정 갱신

Health 표시기의 경보 구성 설정값을 갱신합니다.

중요사항: Health Monitor가 버전 9.7에서 사용되지 않으므로
이 명령 또는 API는 사용되지 않으며 이후 릴리스에서 제거될 수 있습니다. 자세한 정보는 *버전 9.7의 새로운 내용* 책에 있는 『Health Monitor는 사용되지 않음』 주제를 참조하십시오.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2UpdateAlertCfg (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateAlertCfgData
{
   db2Uint32 iObjType;
   char *piObjName;
   char *piDbName;
   db2Uint32 iIndicatorID;
   db2Uint32 iNumIndAttribUpdates;
   struct db2AlertAttrib *piIndAttribUpdates;
   db2Uint32 iNumActionUpdates;
   struct db2AlertActionUpdate *piActionUpdates;
   db2Uint32 iNumActionDeletes;
   struct db2AlertActionDelete *piActionDeletes;
   db2Uint32 iNumNewActions;
   struct db2AlertActionNew *piNewActions;
} db2UpdateAlertCfgData;
```

```
typedef SQL_STRUCTURE db2AlertAttrib
{
   db2Uint32 iAttribID;
   char *piAttribValue;
} db2AlertAttrib;

typedef SQL_STRUCTURE db2AlertActionUpdate
{
   db2Uint32 iActionType;
   char *piActionName;
   db2Uint32 iCondition;
   db2Uint32 iNumParmUpdates;
   struct db2AlertAttrib *piParmUpdates;
} db2AlertActionUpdate;

typedef SQL_STRUCTURE db2AlertActionDelete
{
   db2Uint32 iActionType;
   char *piName;
   db2Uint32 iCondition;
} db2AlertActionDelete;

typedef SQL_STRUCTURE db2AlertActionNew
{
   db2Uint32 iActionType;
   struct db2AlertScriptAction *piScriptAttribs;
   struct db2AlertTaskAction *piTaskAttribs;
} db2AlertActionNew;

typedef SQL_STRUCTURE db2AlertScriptAction
{
   db2Uint32 scriptType;
   db2Uint32 condition;
   char *pPathName;
   char *pWorkingDir;
   char *pCmdLineParms;
   char stmtTermChar;
   char *pUserID;
   char *pPassword;
   char *pHostName;
} db2AlertScriptAction;

typedef SQL_STRUCTURE db2AlertTaskAction
{
   char *pTaskName;
   db2Uint32 condition;
   char *pUserID;
   char *pPassword;
   char *pHostName;
} db2AlertTaskAction;
```

db2UpdateAlertCfg API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2UpdateAlertCfgData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2UpdateAlertCfgData 데이터 구조 매개변수

iObjType

입력. 구성이 요청되는 오브젝트 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2ALERTCFG_OBJTYPE_DBM
- DB2ALERTCFG_OBJTYPE_DATABASES
- DB2ALERTCFG_OBJTYPE_TABLESPACES
- DB2ALERTCFG_OBJTYPE_TS_CONTAINERS
- DB2ALERTCFG_OBJTYPE_DATABASE
- DB2ALERTCFG_OBJTYPE_TABLESPACE
- DB2ALERTCFG_OBJTYPE_TS_CONTAINER

piObjName

입력. 오브젝트 유형 iObjType이 DB2ALERTCFG_OBJTYPE_TABLESPACE 또는 DB2ALERTCFG_OBJTYPE_TS_CONTAINER로 설정된 경우 테이블 스페이스 또는 테이블 스페이스 컨테이너의 이름이며 그 이외의 경우에는 NULL로 설정됩니다.

piDbName

입력. 오브젝트 유형 iObjType이 DB2ALERTCFG_OBJTYPE_TS_CONTAINER, DB2ALERTCFG_OBJTYPE_TABLESPACE, 및 DB2ALERTCFG_OBJTYPE_DATABASE인 경우 데이터베이스 별명 이름이며 그 이외의 경우에는 NULL로 설정됩니다.

iIndicatorID

입력. 구성 갱신이 적용되는 Health 표시기

iNumIndAttribUpdates

입력. iIndicatorID Health 표시기에 대해 갱신되는 경보 속성 수

piIndAttribUpdates

입력. db2AlertAttrib 구조 배열의 포인터

iNumActionUpdates

입력. iIndicatorID Health 표시기에 대해 갱신되는 경보 조치 수

piActionUpdates

입력. db2AlertActionUpdate 구조 배열의 포인터

iNumActionDeletes

입력. iIndicatorID Health 표시기에 대해 삭제되는 경보 조치 수

piActionDeletes

입력. db2AlertActionDelete 구조 배열의 포인터

iNumNewActions

입력. iIndicatorID Health 표시기에 대해 추가되는 새 경보 조치 수

piNewActions

입력. db2AlertActionNew 구조 배열의 포인터

db2AlertAttrib 데이터 구조 매개변수

iAttribID

입력. 갱신되는 경보 속성을 지정합니다. 유효한 값에는 다음이 포함됩니다.

- DB2ALERTCFG_ALARM
- DB2ALERTCFG_WARNING
- DB2ALERTCFG_SENSITIVITY
- DB2ALERTCFG_ACTIONS_ENABLED
- DB2ALERTCFG_THRESHOLD_CHECK

piAttribValue

입력. 경보 속성의 새 값. 가능한 값은 다음과 같습니다.

- DB2ALERTCFG_ALARM
- DB2ALERTCFG_WARNING
- DB2ALERTCFG_SENSITIVITY
- DB2ALERTCFG_ACTIONS_ENABLED
- DB2ALERTCFG_THRESHOLD_CHECK

db2AlertActionUpdate 데이터 구조 매개변수

iActionType

입력. 경보 조치를 지정합니다. 가능한 값은 다음과 같습니다.

- DB2ALERTCFG_ACTIONTYPE_SCRIPT
- DB2ALERTCFG_ACTIONTYPE_TASK

piActionName

입력. 경보 조치 이름. 스크립트 조치 이름은 스크립트의 절대 경로 이름입니다. 태스크 조치 이름은 <task-numberical-ID>.<task-numberical-suffix> 양식의 문자열입니다.

iCondition

조치 실행 조건. 임계값 기본 Health 표시기의 유효한 값은 다음과 같습니다.

- DB2ALERTCFG_CONDITION_ALL
- DB2ALERTCFG_CONDITION_WARNING
- DB2ALERTCFG_CONDITION_ALARM

상태 기반 Health 표시기의 경우 sqlmon에 정의된 숫자 값을 사용하십시오.

iNumParmUpdates

입력. piParmUpdates 배열에서 갱신되는 조치 속성 수

piParmUpdates

입력. db2AlertAttrib 구조의 포인터

db2AlertActionDelete 데이터 구조 매개변수

iActionType

입력. 경보 조치를 지정합니다. 가능한 값은 다음과 같습니다.

- DB2ALERTCFG_ACTIONTYPE_SCRIPT
- DB2ALERTCFG_ACTIONTYPE_TASK

piName

입력. 경보 조치 또는 스크립트 조치 이름. 스크립트 조치 이름은 스크립트의 절대 경로 이름이지만 태스크 조치의 이름은 <task-numerical-ID>.<task-numerical-suffix> 양식의 문자열입니다.

iCondition

조치 실행 조건. 임계값 기본 Health 표시기의 유효한 값은 다음과 같습니다.

- DB2ALERTCFG_CONDITION_ALL
- DB2ALERTCFG_CONDITION_WARNING
- DB2ALERTCFG_CONDITION_ALARM

상태 기반 Health 표시기의 경우 sqlmon에 정의된 숫자 값을 사용하십시오.

db2AlertActionNew 데이터 구조 매개변수

iActionType

입력. 경보 조치를 지정합니다. 가능한 값은 다음과 같습니다.

- DB2ALERTCFG_ACTIONTYPE_SCRIPT
- DB2ALERTCFG_ACTIONTYPE_TASK

piScriptAttribs

입력. db2AlertScriptAction 구조의 포인터

piTaskAttribs

입력. db2AlertTaskAction 구조의 포인터

db2AlertScriptAction 데이터 구조 매개변수

scriptType

스크립트 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2ALERTCFG_SCRIPTTYPE_DB2CMD
- DB2ALERTCFG_SCRIPTTYPE_OS

condition

조치 실행 조건. 임계값 기본 Health 표시기의 유효한 값은 다음과 같습니다.

- DB2ALERTCFG_CONDITION_ALL
- DB2ALERTCFG_CONDITION_WARNING
- DB2ALERTCFG_CONDITION_ALARM

상태 기반 Health 표시기의 경우 sqlmon에 정의된 숫자 값을 사용하십시오.

pPathname

스크립트의 절대 경로 이름

pWorkingDir

스크립트가 실행되는 디렉토리의 절대 경로 이름

pCmdLineParms

스크립트가 호출될 때 스크립트에 전달되는 명령 매개변수.
DB2ALERTCFG_SCRIPTTYPE_OS에만 선택적

stmtTermChar

명령문을 종료하기 위해 스크립트에서 사용되는 문자.
DB2ALERTCFG_SCRIPTTYPE_DB2CMD에만 선택적

pUserID

스크립트가 실행되는 사용자 어카운트

pPassword

사용자 어카운트 pUserId의 암호

pHostName

스크립트를 실행하는 호스트 이름. 태스크와 스크립트 모두에 적용됩니다.

Script 스크립트가 상주하고 실행되는 호스트 이름

Task 스케줄러가 있는 호스트 이름

db2AlertTaskAction 데이터 구조 매개변수

pTaskname

태스크 이름

condition

조치를 실행하는 조건

pUserID

스크립트가 실행되는 사용자 어카운트

pPassword

사용자 어카운트 pUserId의 암호

pHostName

스크립트를 실행하는 호스트 이름. 태스크와 스크립트 모두에 적용됩니다.

Script 스크립트가 상주하고 실행되는 호스트 이름

Task 스케줄러가 있는 호스트 이름

제 80 장 db2UpdateAlternateServerForDB - 시스템 데이터베이스 디렉토리에서 데이터베이스 별명에 대한 대체 서버 갱신

시스템 데이터베이스 디렉토리에서 데이터베이스 별명의 대체 서버를 갱신합니다.

범위

이 API는 시스템 데이터베이스 디렉토리에 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2UpdateAlternateServerForDB (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateAltServerStruct
{
   char *piDbAlias;
   char *piHostName;
   char *piPort;
} db2UpdateAltServerStruct;

SQL_API_RC SQL_API_FN
  db2gUpdateAlternateServerForDB (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gUpdateAltServerStruct
{
   db2Uint32 iDbAlias_len;
   char *piDbAlias;
   db2Uint32 iHostName_len;
```

```
   char *piHostName;
   db2Uint32 iPort_len;
   char *piPort;
} db2gUpdateAltServerStruct;
```

db2UpdateAlternateServerForDB API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2UpdateAltServerStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2UpdateAltServerStruct 데이터 구조 매개변수

piDbAlias

입력. 데이터베이스 별명이 포함된 문자열

piHostName

입력. 데이터베이스의 대체 서버가 있는 노드의 호스트 이름 또는 IP 주소가 포함된 문자열. 호스트 이름은 TCP/IP 네트워크에서 인식되는 노드의 이름입니다. 호스트 이름의 최대 길이는 255자입니다. IP 주소는 IPv4 또는 IPv6 주소일 수 있습니다.

piPort 입력. 대체 서버 데이터베이스 관리 프로그램 인스턴스의 포트 번호. 포트 번호의 최대 길이는 14자입니다.

db2gUpdateAltServerStruct 데이터 구조 특정 매개변수

iDbAlias_len

입력. piDbAlias의 길이(바이트)

iHostName_len

입력. piHostName의 길이(바이트)

iPort_len

입력. piPort의 길이(바이트)

사용 시 참고사항

API는 시스템 데이터베이스 디렉토리에만 적용됩니다.

API는 서버에서만 사용해야 합니다. 클라이언트에서 발행되면 무시되고 경고 SQL1889W가 발행됩니다.

LDAP(Lightweight Directory Access Protocol) 지원이 현재 머신에서 사용 가능한 경우에는 데이터베이스의 대체 서버가 자동으로 LDAP 디렉토리에서 갱신됩니다.

제 81 장 db2UpdateContact - 문의처 속성 갱신

문의처 속성을 갱신합니다. 문의처는 통지 메시지를 수신할 수 있는 사용자입니다. 문의처는 시스템에 로컬로 또는 전역 목록으로 정의할 수 있습니다. DB2 관리 서버(DAS) 구성 매개변수 contact_host의 설정으로 목록이 로컬 또는 전역인지 판별합니다.

권한 부여

없음

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2UpdateContact (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateContactData
{
   char *piUserid;
   char *piPassword;
   char *piContactName;
   db2Uint32 iNumAttribsUpdated;
   struct db2ContactAttrib *piAttribs;
} db2UpdateContactData;

typedef SQL_STRUCTURE db2ContactAttrib
{
   db2Uint32 iAttribID;
   char *piAttribValue;
} db2ContactAttrib;
```

db2UpdateContact API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2UpdateContactData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2UpdateContactData 데이터 구조 매개변수

piContactName

입력. 갱신한 문의처 이름을 지정합니다.

iNumAttribsUpdated

입력. 갱신한 속성 수

piAttribs

입력. db2ContactAttrib 구조의 포인터

db2ContactAttrib 데이터 구조 매개변수

iAttribID

입력. 문의처 속성을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2CONTACT_ADDRESS
- DB2CONTACT_TYPE
- DB2CONTACT_MAXPAGELEN
- DB2CONTACT_DESCRIPTION

piAttribValue

입력. 문의처 속성의 새 값

사용 시 참고사항

이 API는 UNIX 및 Linux에서는 지원되지 않습니다. 그러나 SQL 인터페이스를 통해 동일한 기능에 액세스할 수 있습니다.

제 82 장 db2UpdateContactGroup - 문의처 그룹의 속성 갱신

문의처 그룹 속성을 갱신합니다. 문의처 그룹에는 통지 메시지를 수신할 수 있는 사용자 목록이 포함됩니다. 문의처 그룹은 시스템에 로컬로 또는 전역 목록으로 정의할 수 있습니다. DB2 관리 서버(DAS) 구성 매개변수 contact_host의 설정으로 목록이 로컬 또는 전역인지 판별합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
db2ApiDf.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2UpdateContactGroup (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateContactGroupData
{
   char *piUserid;
   char *piPassword;
   char *piGroupName;
   db2Uint32 iNumNewContacts;
   struct db2ContactTypeData *piNewContacts;
   db2Uint32 iNumDroppedContacts;
   struct db2ContactTypeData *piDroppedContacts;
   char *piNewDescription;
} db2UpdateContactGroupData;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

db2UpdateContactGroup API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2ResetMonitorData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2UpdateContactGroupData 데이터 구조 매개변수

piUserid

입력. 사용자 이름

piPassword

입력. piUserid의 암호

piGroupName

입력. 갱신한 문의처 그룹 이름

iNumNewContacts

입력. 그룹에 추가되는 새 문의처 수

piNewContacts

입력. db2ContactTypeData 구조의 포인터

iNumDroppedContacts

입력. 삭제되는 그룹의 문의처 수

piDroppedContacts

입력. db2ContactTypeData 구조의 포인터

piNewDescription

입력. 그룹에 대한 새 설명. 이전 설명을 변경하지 않으려면 이 매개변수를 NULL로 설정하십시오.

db2ContactTypeData 데이터 구조 매개변수

contactType

문의처 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName

contactType을 DB2CONTACT_SINGLE로 설정한 경우 문의처 그룹 이름 또는 담당자

사용 시 참고사항

이 API는 UNIX 및 Linux에서는 지원되지 않습니다. 그러나 SQL 인터페이스를 통해 동일한 기능에 액세스할 수 있습니다.

제 83 장 db2UpdateHealthNotificationList - Health 경보를 수신할 수 있는 문의처 목록 갱신

인스턴스에 의해 발행된 Health 경보에 대한 통지의 문의처 목록을 갱신합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

인스턴스. 인스턴스 접속이 없는 경우에는 디폴트 인스턴스 접속이 작성됩니다.

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2UpdateHealthNotificationList (
   db2Uint32 versionNumber,
   void * pParmStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UpdateHealthNotificationListData
{
   db2Uint32 iNumUpdates;
   struct db2HealthNotificationListUpdate *piUpdates;
} db2UpdateHealthNotificationListData;

typedef SQL_STRUCTURE db2HealthNotificationListUpdate
{
   db2Uint32 iUpdateType;
   struct db2ContactTypeData *piContact;
} db2HealthNotificationListUpdate;

typedef SQL_STRUCTURE db2ContactTypeData
{
   db2Uint32 contactType;
   char *pName;
} db2ContactTypeData;
```

db2UpdateHealthNotificationList API 매개변수

versionNumber

입력. 두 번째 매개변수 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2UpdateHealthNotificationListData 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2UpdateHealthNotificationListData 데이터 구조 매개변수

iNumUpdates

입력. 갱신사항 수.

piUpdates

입력. db2HealthNotificationListUpdate 구조의 포인터

db2HealthNotificationListUpdate 데이터 구조 매개변수

iUpdateType

입력. 갱신 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2HEALTHNOTIFICATIONLIST_ADD
- DB2HEALTHNOTIFICATIONLIST_DROP

piContact

입력. db2ContactTypeData 구조의 포인터

db2ContactTypeData 데이터 구조 매개변수

contactType

문의처 유형을 지정합니다. 가능한 값은 다음과 같습니다.

- DB2CONTACT_SINGLE
- DB2CONTACT_GROUP

pName

contactType을 DB2CONTACT_SINGLE로 설정한 경우 문의처 그룹 이름 또는 담당자

제 84 장 db2UtilityControl - 실행 중인 유틸리티의 우선순위 레벨 설정

실행 중인 유틸리티의 우선순위 레벨을 제어합니다. 조절 및 조절되지 않은 유틸리티 호출에 사용할 수 있습니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

인스턴스

API 내장 파일

db2ApiDf.h

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2UtilityControl (
   db2Uint32 version,
   void * pUtilityControlStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2UtilityControlStruct
{
   db2Uint32 iID;
   db2Uint32 iAttribute;
   void *pioValue;
} db2UtilityControlStruct;

SQL_API_RC SQL_API_FN
  db2gUtilityControl (
   db2Uint32 version,
   void * pgUtilityControlStruct,
   struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2gUtilityControlStruct
{
   db2Uint32 iID;
   db2Uint32 iAttribute;
   void *pioValue;
} db2gUtilityControlStruct;
```

db2UtilityControl API 매개변수

version

입력. 두 번째 매개변수인 pUtilityControlStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pUtilityControlStruct

입력. db2UtilityControlStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2UtilityControlStruct 데이터 구조 매개변수

iId 입력. 수정할 유틸리티 ID를 지정합니다.

iAttribute

입력. 수정할 속성을 지정합니다. 유효한 값(include 디렉토리에 있는 db2ApiDf 헤더 파일에 정의)은 다음과 같습니다.

DB2UTILCTRL_PRIORITY_ATTRIB

유틸리티의 조절 우선순위를 수정하십시오.

pioValue

입력. iAttribute 매개변수와 연관된 새 속성 값을 지정합니다.

주: iAttribute 매개변수가 DB2UTILCTRL_PRIORITY_ATTRIB로 설정된 경우 pioValue 매개변수는 우선순위를 포함하는 db2Uint32를 지시해야 합니다.

사용 시 참고사항

지정한 iId의 기존 유틸리티가 없는 경우에는 SQL1153N이 리턴됩니다. 이는 함수가 유효하지 않은 인수로 호출되었거나 유틸리티가 완료된 것을 나타낼 수도 있습니다.

유틸리티가 조절을 지원하지 않으면 SQL1154N이 리턴됩니다.

제 85 장 sqlabndx - 패키지 작성을 위해 응용프로그램 바인드

프리컴파일러에서 생성된 바인드 파일에 저장된 SQL 문을 준비하고 데이터베이스에 저장되는 패키지를 작성하는 바인드 유틸리티를 호출합니다.

범위

이 API는 db2nodes.cfg의 모든 데이터베이스 파티션 서버에서 호출할 수 있습니다. 이 API는 카탈로그 파티션에서 데이터베이스 카탈로그를 갱신합니다. 모든 데이터베이스 파티션 서버에서 갱신된 사항을 확인할 수 있습니다.

권한 부여

다음 권한 중 하나여야 합니다.

- *dbadm* 권한
- SQLERROR CHECK 또는 EXPLAIN ONLY가 지정된 경우 EXPLAIN 또는 SQLADM 권한으로도 충분합니다.
- 패키지가 없는 경우에는 BINDADD 권한 및:
  - 패키지의 스키마 이름이 없는 경우에는 데이터베이스의 IMPLICIT_SCHEMA 권한.
  - 패키지의 스키마 이름이 없는 경우에는 스키마의 CREATEIN 특권.
- 패키지가 있는 경우에는 다음 특권 중 하나여야 합니다.
  - 스키마에서 ALTERIN 특권
  - 패키지에서 BIND 특권

응용프로그램에서 모든 정적 SQL 문을 컴파일할 때 필요한 모든 특권도 필요합니다. 그룹에 부여된 특권은 정적 명령문의 인증 점검에는 사용되지 않습니다.

필수 연결

데이터베이스

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlabndx (
        _SQLOLDCHAR * pBindFileName,
        _SQLOLDCHAR * pMsgFileName,
        struct sqlopt * pBindOptions,
```

```
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgbndx (
        unsigned short MsgFileNameLen,
        unsigned short BindFileNameLen,
        struct sqlca * pSqlca,
        struct sqlopt * pBindOptions,
        _SQLOLDCHAR * pMsgFileName,
        _SQLOLDCHAR * pBindFileName);
```

sqlabndx API parameters

pBindFileName

입력. 바인드 파일 이름을 포함하는 문자열 또는 바인드 파일 이름 목록을 포함하는 파일 이름. 바인드 파일 이름의 확장자에는 .bnd가 포함되어야 합니다. 이 파일 경로를 지정할 수 있습니다.

바인드 목록 파일 이름 앞에는 (@) 기호를 추가하십시오. 예를 들어, 완전한 바인드 목록 파일 이름은 다음과 같습니다.

```
/u/user1/bnd/@all.lst
```

바인드 목록 파일에는 하나 이상의 바인드 목록 파일이 포함되어야 하며 확장자는 .lst여야 합니다.

첫 번째 바인드 파일을 제외한 모든 바인드 파일 이름 앞에는 더하기 기호(+)를 추가하십시오. 바인드 파일 이름은 하나 이상의 라인으로 될 수도 있습니다. 예를 들어 바인드 목록 파일 all.lst에는 다음이 포함됩니다.

```
mybind1.bnd+mybind2.bnd+
mybind3.bnd+
mybind4.bnd
```

목록 파일에서 바인드 파일 이름의 경로 스펙을 사용할 수 있습니다. 경로를 지정하지 않으면 데이터베이스 관리 프로그램이 바인드 목록 파일의 정보를 사용합니다.

pMsgFileName

입력. 오류, 경고 및 정보 메시지 목적지를 포함하는 문자열. 운영 체제 파일 또는 표준 디바이스의 경로 및 이름일 수 있습니다. 파일이 이미 있는 경우에는 겹쳐씁니다. 없는 경우에는 파일이 작성됩니다.

pBindOptions

입력. 바인드 옵션을 API에 전달하는 데 사용되는 구조. 이 구조에 대한 자세한 정보는 SQLOPT를 참조하십시오.

pSqlca

출력. sqlca 구조의 포인터

sqlgbndx API 특정 매개변수

pMsgFileName

입력. 오류, 경고 및 정보 메시지 목적지를 포함하는 문자열. 운영 체제 파일 또는 표준 디바이스의 경로 및 이름일 수 있습니다. 파일이 이미 있는 경우에는 겹쳐씁니다. 없는 경우에는 파일이 작성됩니다.

BindFileNameLen

입력. pBindFileName 매개변수 길이(바이트)

사용 시 참고사항

응용프로그램 소스 파일에 대한 프리컴파일 프로세스 중 또는 이후에 별도의 단계를 수행하여 바인드를 완료할 수 있습니다. 별도의 프로세스를 통해 바인드를 수행할 때는 BIND를 사용하십시오.

패키지 작성 시에 사용한 이름은 바인드 파일에 저장되고 생성된 소스 파일을 기반으로 합니다(기존 경로나 확장자는 버림). 예를 들어 프리컴파일된 소스 파일인 myapp.sqc는 디폴트 바인드 파일로 myapp.bnd를 생성하고 디폴트 패키지 이름은 MYAPP입니다. (그러나 바인드 파일 이름 및 패키지 이름은 sqlaprep의 SQL_BIND_OPT 및 SQL_PKG_OPT 옵션을 사용하여 프리컴파일 시에 겹쳐쓸 수 있습니다.)

BIND는 사용자가 시작한 트랜잭션에서 실행합니다. 바인드를 수행한 후에 BIND는 COMMIT(바인드가 성공한 경우) 또는 ROLLBACK(바인드가 실패한 경우) 조작을 발행하여 현재 트랜잭션을 종료하고 다른 트랜잭션을 시작합니다.

치명적 오류가 발생하거나 오류가 100개 이상 발생하면 바인드가 정지합니다. 바인딩 중에 치명적 오류가 발생한 경우 BIND는 바인드를 중지하고 모든 파일을 닫은 후에 패키지를 버립니다.

바인드 응용프로그램에는 이 매뉴얼 범위에 포함되지 않은 전제조건 요구사항 및 제한사항이 있습니다. 예를 들어, 응용프로그램은 버전 8 클라이언트에서 버전 8 서버로 바인드할 수 없으며 버전 7 서버에 대해 실행할 수 없습니다.

바인드 옵션 유형 및 값은 sql.h에 정의됩니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 86 장 sqlaintp - 오류 메시지 가져오기

sqlca 구조의 sqlcode 필드로 지정한 오류 조건과 연관된 메시지를 검색합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlaintp (
        char * pBuffer,
        short BufferSize,
        short LineWidth,
        const char * pMsgFileName,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgintp (
        short BufferSize,
        short LineWidth,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pBuffer);
```

sqlaintp API 매개변수

pBuffer

출력. 메시지 텍스트가 배치되는 문자열 버퍼의 포인터. 버퍼에 맞도록 메시지가 절단되는 경우에는 널(NULL) 문자열 종단자 문자를 절단 시에 사용할 수 있습니다.

BufferSize

입력. 검색된 메시지 텍스트를 보유하는 문자열 버퍼 크기(바이트)

LineWidth

입력. 각 메시지 텍스트 라인의 최대 라인 너비. 라인은 단어를 경계로 분리됩니다. 값이 0인 경우에는 메시지 텍스트가 라인 중단 없이 리턴됩니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

호출별로 한 개의 메시지가 리턴됩니다.

새 라인(줄 바꾸기, LF 또는 캐리지 리턴/줄 바꾸기, CR/LF) 시퀀스가 각 메시지 끝에 추가됩니다.

양수의 라인 너비를 지정하는 경우 새 라인 시퀀스가 단어 사이에 삽입되어 라인이 라인 너비를 초과하지 않습니다.

단어가 라인 너비보다 큰 경우에는 라인이 최대한의 문자로 채워진 다음 새 라인이 삽입되고 나머지 문자가 다음 라인에 표시됩니다.

멀티스레드 응용프로그램에서 sqlaintp는 유효한 컨텍스트에 접속되어야 합니다. 그렇지 않으면 SQLCODE - 1445 메시지 텍스트를 가져올 수 없습니다.

리턴 코드

코드	메시지
+i	형식화된 메시지의 바이트 수를 나타내는 양수. 이 값이 호출자의 버퍼 크기 입력보다 큰 경우에 메시지가 잘립니다.
-1	함수의 메시지 포맷팅 서비스에 사용할 수 있는 메모리가 부족합니다. 요청된 메시지가 리턴되지 않습니다.
-2	오류 없음. sqlca에 오류 코드가 없습니다(SQLCODE = 0).
-3	메시지 파일에 액세스할 수 없거나 해당 파일이 올바르지 않습니다.
-4	라인 너비가 영(0)보다 작습니다.
-5	유효하지 않은 sqlca, 잘못된 버퍼 주소 또는 버퍼 길이.

리턴 코드가 -1 또는 -3인 경우 메시지 버퍼에는 문제점에 대한 추가 정보가 포함됩니다.

REXX API 구문

```
GET MESSAGE INTO :msg [LINEWIDTH width]
```

REXX API 매개변수

msg 메시지 텍스트가 배치되는 REXX 변수

width 텍스트 메시지의 각 라인에 대한 최대 라인 너비. 라인은 단어를 경계로 분리됩니다. 너비를 지정하지 않거나 0으로 설정하면 메시지 텍스트가 라인 중단 없이 리턴됩니다.

제 87 장 sqlaprep - 응용프로그램 프리컴파일

Embedded SQL 문이 포함된 응용프로그램 소스 파일을 처리합니다. SQL문에 대한 호스트 언어 호출이 포함된 수정된 소스 파일이 작성되고 디폴트로 데이터베이스에 패키지가 작성됩니다.

범위

이 API는 db2nodes.cfg의 모든 데이터베이스 파티션 서버에서 호출할 수 있습니다. 이 API는 카탈로그 파티션에서 데이터베이스 카탈로그를 갱신합니다. 모든 데이터베이스 파티션 서버에서 갱신된 사항을 확인할 수 있습니다.

권한 부여

다음 권한 중 하나여야 합니다.

- *dbadm* 권한
- SQLERROR CHECK 또는 EXPLAIN ONLY가 지정된 경우 EXPLAIN 또는 SQLADM 권한으로도 충분합니다.
- 패키지가 없는 경우에는 BINDADD 권한 및:
  - 패키지의 스키마 이름이 없는 경우에는 데이터베이스의 IMPLICIT_SCHEMA 권한.
  - 패키지의 스키마 이름이 없는 경우에는 스키마의 CREATEIN 특권.
- 패키지가 있는 경우에는 다음 특권 중 하나여야 합니다.
  - 스키마에서 ALTERIN 특권
  - 패키지에서 BIND 특권

응용프로그램에서 모든 정적 SQL 문을 컴파일할 때 필요한 모든 특권도 필요합니다. 그룹에 부여된 특권은 정적 명령문의 인증 점검에는 사용되지 않습니다.

필수 연결

데이터베이스

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlaprep (
        _SQLOLDCHAR * pProgramName,
        _SQLOLDCHAR * pMsgFileName,
```

```
        struct sqlopt * pPrepOptions,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgprep (
        unsigned short MsgFileNameLen,
        unsigned short ProgramNameLen,
        struct sqlca * pSqlca,
        struct sqlopt * pPrepOptions,
        _SQLOLDCHAR * pMsgFileName,
        _SQLOLDCHAR * pProgramName);
```

sqlaprep API 매개변수

pProgramName

입력. 프리컴파일되는 응용프로그램 이름이 포함된 문자열. 다음과 같은 확장자를 사용하십시오.

- .sqb: Cobol 응용프로그램
- .sqc: C 응용프로그램
- .sqC: UNIX C++ 응용프로그램
- .sqf: FORTRAN 응용프로그램
- .sqx: C++ 응용프로그램

TARGET 옵션을 사용하면 입력 파일 이름 확장자는 이 사전 정의된 목록에서 가져올 필요가 없습니다.

UNIX 기반 시스템에서 Embedded SQL이 포함된 C++ 응용프로그램의 선호 확장자는 sqC입니다. 그러나 대소문자를 구분하지 않는 시스템용으로 개발된 sqx 규칙은 UNIX 기반 시스템에서 허용됩니다.

pMsgFileName

입력. 오류, 경고 및 정보 메시지 목적지를 포함하는 문자열. 운영 체제 파일 또는 표준 디바이스의 경로 및 이름일 수 있습니다. 파일이 이미 있는 경우에는 겹쳐씁니다. 없는 경우에는 파일이 작성됩니다.

pPrepOptions

입력. 프리컴파일 옵션을 API에 전달하는 데 사용되는 구조. 이 구조에 대한 자세한 정보는 SQLOPT를 참조하십시오.

pSqlca

출력. sqlca 구조의 포인터

sqlgprep API 특정 매개변수

MsgFileNameLen

입력. pMsgFileName 매개변수 길이(바이트)

ProgramNameLen

입력. pProgramName 매개변수 길이(바이트)

사용 시 참고사항

SQL문과 동등한 호스트 언어가 포함된 수정된 소스 파일이 작성됩니다. 디폴트로 패키지는 연결된 데이터베이스에 작성됩니다. 패키지 이름은 최대 8자로 프로그램 파일 이름과 동일합니다(확장자 제외 및 대문자로 변환)

데이터베이스 연결 후에 시작된 트랜잭션에서 sqlaprep가 실행됩니다. 그러면, PRECOMPILE PROGRAM이 COMMIT 또는 ROLLBACK 조작을 실행하여 현재 트랜잭션을 종료하고 다른 트랜잭션을 시작합니다.

치명적 오류가 발생하거나 오류가 100개 이상 발생하면 프리컴파일이 정지합니다. 치명적 오류가 발생하면 PRECOMPILE PROGRAM이 프리컴파일을 중지하고 모든 파일을 닫은 후에 패키지를 버립니다.

프리컴파일 옵션 유형 및 값은 sql.h에 정의됩니다.

PRECOMPILE 명령이나 sqlaprep API를 사용하는 경우 패키지 이름은 PACKAGE USING 옵션을 지정할 수 있습니다. 이 옵션을 사용하는 경우 패키지 이름에 최대 128 바이트를 지정할 수 있습니다. 이 옵션을 사용하지 않으면 프리컴파일러가 패키지 이름을 생성합니다. 응용프로그램 소스 파일 이름(확장자 제외 및 대문자로 변환)은 최대 8자까지 가능합니다. 생성된 이름은 이전 버전의 DB2와 호환되도록 최대 8자가 사용됩니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 88 장 sqlarbnd - 패키지 리바인드

바인드 파일을 사용하지 않고도 데이터베이스에 저장된 패키지를 재작성할 수 있습니다.

권한 부여

다음 중 하나가 필요합니다.

- dbadm 권한
- 스키마에서 ALTERIN 특권
- 패키지의 BIND 특권

패키지의 가장 최신 바인더의 ID인 SYSCAT.PACKAGES 시스템 카탈로그 테이블의 BOUNDBY 컬럼에 로그인된 권한 부여 ID가 리바인드의 바인더 권한 부여 ID 및 패키지의 테이블 참조를 위한 디폴트 스키마용으로 사용됩니다. 이 디폴트 규정자는 리바인드 요청을 실행 중인 사용자의 권한 부여 ID와는 다를 수 있습니다. REBIND는 패키지 작성 시에 지정한 동일한 바인드 옵션을 사용합니다.

필수 연결

데이터베이스

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlarbnd (
        char * pPackageName,
        struct sqlca * pSqlca,
        struct sqlopt * pRebindOptions);

SQL_API_RC SQL_API_FN
  sqlgrbnd (
        unsigned short PackageNameLen,
        char * pPackageName,
        struct sqlca * pSqlca,
        struct sqlopt * pRebindOptions);
```

sqlarbnd API 매개변수

pPackageName

입력. 리바인드되는 패키지에 지정된 규정되었거나 규정되지 않은 이름이 포함된 문자열. 규정되지 않은 패키지 이름은 현재 권한 부여 ID로 내재적으로 규정합니다. 이 이름은 패키지 버전을 포함하지 않습니다. 비어 있는 문자열이 아

닌 버전을 포함하는 패키지를 지정하는 경우에는 SQL_VERSION_OPT 리바인드 옵션을 사용하여 버전 ID를 지정해야 합니다.

pSqlca

출력. sqlca 구조의 포인터

pRebindOptions

입력. 리바인드 옵션을 API에 전달하는 데 사용되는 SQLOPT 옵션의 포인터. 이 구조에 대한 자세한 정보는 SQLOPT를 참조하십시오.

sqlgrbnd API 특정 매개변수

PackageNameLen

입력. pPackageName 매개변수 길이(바이트)

사용 시 참고사항

REBIND는 리바인드가 성공한 후에 자동으로 트랜잭션을 커미트하지는 않습니다. 사용자가 내재적으로 트랜잭션을 커미트해야 합니다. 그러면, 사용자가 특정 통계를 갱신하고 변경사항을 확인하기 위해 패키지를 리바인드 할 때 ″what if″ 분석을 사용할 수 있습니다. 작업 단위(UOW)에서 다중 리바인드를 허가합니다.

이 API는

- 패키지 재작성을 위한 빠른 방법을 제공합니다. 사용자는 원래 바인드 파일 .fs를 사용하지 않고도 시스템에서 변경사항을 사용할 수 있습니다. 예를 들어, 특정 SQL문이 새로 작성된 인덱스를 사용할 수 있으면 REBIND를 패키지 재작성에 사용할 수 있습니다. db2Runstats가 실행된 후에 새 통계를 사용하여 REBIND를 사용하여 패키지를 재작성할 수 있습니다.
- 작동 불능 패키지 재작성 방법을 제공합니다. 작동 불능 패키지는 바인드 유틸리티 또는 리바인드 유틸리티를 호출하여 명시적으로 리바인드해야 합니다. 패키지가 종속된 함수 인스턴스가 삭제된 경우에는 패키지가 작동 불능으로 표시됩니다 (SYSCAT.PACKAGES 시스템 카탈로그의 VALID 컬럼이 X로 설정됨). 작동 불능 패키지에는 리바인드 제한 옵션이 지원되지 않습니다.
- 유효하지 않은 패키지 리바인드를 사용자가 제어할 수 있도록 합니다. 유효하지 않은 패키지는 실행될 때 데이터베이스 관리 프로그램이 자동으로(또는 내재적으로) 리바인드합니다. 그러면, 유효하지 않은 패키지에 대한 첫 번째 SQL 요청 실행이 상당히 지연될 수 있습니다. 초기 지연을 방지하고 내재적인 리바인드가 실패하게 되는 예기치 않은 SQL 오류 메시지가 표시되지 않도록 하려면 시스템이 자동으로 유효하지 않은 패키지를 리바인드하는 것보다 명시적으로 유효하지 않은 패키지를 리바인드하는 것이 좋을 수도 있습니다. 예를 들어 다음 데이터베이스 업그레이드에서 데이터베이스에 저장된 모든 패키지는 UPGRADE DATABASE 명령으로 무효화됩니다. 많은 패키지가 관련되어 있는 경우 한 번에 유효하지 않은 패키지 전체를 명시

적으로 리바인드하는 것이 좋을 수도 있습니다. BIND, REBIND 또는 db2rbind 도구를 사용하여 이 명시적 리바인드를 수행할 수 있습니다.

명시적으로 패키지를 리바인드하기 위해 BIND 또는 REBIND를 사용하는지는 환경에 따라 다릅니다. REBIND를 사용하는 것이 BIND 사용 시보다 성능이 훨씬 좋기 때문에 반드시 BIND를 사용하도록 지정된 경우가 아닐 때는 REBIND를 사용하는 것이 좋습니다. 그러나 다음과 같은 경우에는 BIND를 사용해야 합니다.

- 프로그램이 수정된 경우(예를 들어 SQL문이 추가 또는 삭제되었거나 패키지가 프로그램 실행 파일에 일치하지 않는 경우)
- 리바인드 중에 임의의 바인드 옵션을 수정하려는 경우. REBIND는 바인드 옵션을 모두 지원하지 않습니다. 예를 들어 바인드 프로세스 중에 부여되는 패키지의 특권을 사용하려면 BIND에 SQL_GRANT_OPT 옵션이 있기 때문에 반드시 BIND를 사용해야 합니다.
- 현재 패키지가 데이터베이스에 없는 경우.
- 모든 바인드 오류를 발견해야 하는 경우. REBIND는 발견한 첫 번째 오류만 리턴하고 종료되지만 BIND 명령은 바인드 중에 발생한 처음 100개의 오류를 리턴합니다.

REBIND는 DB2 Connect에서 지원됩니다.

REBIND를 다른 사용자가 사용 중인 패키지에서 실행하면 리바인드 중에 배타적 잠금이 SYSCAT.PACKAGES 시스템 카탈로그 테이블의 패키지 레코드에서 유지되기 때문에 다른 사용자의 논리적 작업 단위가 종료될 때까지 리바인드가 수행되지 않습니다.

REBIND가 실행되면 데이터베이스 관리 프로그램은 SYSCAT.PACKAGES 시스템 카탈로그 테이블에 저장된 SQL문에서 패키지를 재작성합니다. 패키지 번호와 작성자가 동일한 버전이 여러 개인 경우 한 번에 한 개의 버전만 바인드됩니다. SQL_VERSION_OPT 리바인드 옵션을 사용하도록 지정하지 않으면 VERSION의 디폴트는 ""입니다. 리바인드 요청에 지정된 이름 및 작성자에 일치하는 이름 및 작성자의 패키지가 하나만 있는 경우에도, 해당 VERSION이 명시적으로 또는 내재적으로 지정된 VERSION에 일치하지 않으면 리바인드되지 않습니다.

REBIND에 오류가 발생하면 처리가 중지되고 오류 메시지가 리턴됩니다.

SQL_EXPLSNAP_OPT 또는 SQL_EXPLAIN_OPT가 YES 또는 ALL로 설정된 경우(카탈로그에서 EXPLAIN_SNAPSHOT 및 EXPLAIN_MODE 컬럼 확인) REBIND 중에 Explain 테이블이 채워집니다. 사용된 Explain 테이블은 원래 바인더의 테이블이 아니라 REBIND 리퀘스터의 테이블입니다. 리바인드 옵션 유형 및 값은 sql.h에 정의됩니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 89 장 sqlbctcq - 테이블 스페이스 컨테이너 쿼리 닫기

테이블 스페이스 컨테이너 쿼리 요청을 종료하고 관련 자원을 사용 가능하게 합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbctcq (
   struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgctcq (
   struct sqlca * pSqlca);
```

sqlbctcq API 매개변수

pSqlca

출력. sqlca 구조의 포인터

제 90 장 sqlbctsq - 테이블 스페이스 쿼리 닫기

테이블 스페이스 쿼리 요청을 종료하고 관련 자원을 사용 가능하게 합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm
- load

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbctsq (
   struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgctsq (
   struct sqlca * pSqlca);
```

sqlbctsq API 매개변수

pSqlca

출력. sqlca 구조의 포인터

제 91 장 sqlbftcq - 테이블 스페이스 컨테이너의 행에 대한 쿼리 데이터 페치

지정된 수의 테이블 스페이스 컨테이너 쿼리 데이터의 행을 페치하며 각 행은 컨테이너의 데이터로 구성됩니다.

범위

파티션된 데이터베이스 환경에서 현재 데이터베이스 파티션의 테이블 스페이스만 나열됩니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbftcq (
   struct sqlca * pSqlca,
   sqluint32 MaxContainers,
   struct SQLB_TBSCONTQRY_DATA * pContainerData,
   sqluint32 * pNumContainers);

SQL_API_RC SQL_API_FN
  sqlgftcq (
   struct sqlca * pSqlca,
   sqluint32 MaxContainers,
   struct SQLB_TBSCONTQRY_DATA * pContainerData,
   sqluint32 * pNumContainers);
```

sqlbftcq API 매개변수

pSqlca

출력. sqlca 구조의 포인터

MaxContainers

입력. 사용자가 할당한 출력 영역의 최대 수의 데이터 행(pContainerData로 지시)만 보유할 수 있습니다.

pContainerData

출력. 쿼리 데이터의 구조인 출력 영역의 포인터. 이 구조에 대한 자세한 정보는 SQLB-TBSCONTQRY-DATA를 참조하십시오. 이 API 호출자는 이 구조의 MaxContainers에 스페이스를 할당하고 이 스페이스를 지시하도록 pContainerData를 설정해야 합니다. API는 이 스페이스를 사용하여 테이블 스페이스 컨테이너 데이터를 리턴합니다.

pNumContainers

출력. 리턴된 출력 행 수

사용 시 참고사항

사용자는 pContainerData 매개변수로 지시된 메모리를 할당하고 사용 가능하게 해야 합니다. 이 API는 sqlbotcq 호출이 성공한 뒤에만 사용할 수 있습니다. sqlbotcq에서 생성된 목록을 페치하기 위해 반복적으로 호출할 수 있습니다.

제 92 장 sqlbftpq - 테이블 스페이스의 행에 대해 쿼리 데이터 페치

지정된 수의 테이블 스페이스 쿼리 데이터의 행을 페치하며 각 행은 테이블 스페이스의 데이터로 구성됩니다.

범위

파티션된 데이터베이스 환경에서 현재 데이터베이스 파티션의 테이블 스페이스만 나열됩니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm
- load

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbftpq (
   struct sqlca * pSqlca,
   sqluint32 MaxTablespaces,
   struct SQLB_TBSPQRY_DATA * pTablespaceData,
   sqluint32 * pNumTablespaces);

SQL_API_RC SQL_API_FN
  sqlgftpq (
   struct sqlca * pSqlca,
   sqluint32 MaxTablespaces,
   struct SQLB_TBSPQRY_DATA * pTablespaceData,
   sqluint32 * pNumTablespaces);
```

sqlbftpq API 매개변수

pSqlca

출력. sqlca 구조의 포인터

MaxTablespaces

입력. 사용자가 할당한 출력 영역의 최대 수의 데이터 행(pTablespaceData로 지시)만 보유할 수 있습니다.

pTablespaceData

입력 및 출력. 쿼리 데이터의 구조인 출력 영역의 포인터. 이 구조에 대한 자세한 정보는 SQLB-TBSPQRY-DATA를 참조하십시오. 이 API의 호출자는 다음을 수행해야 합니다.

- 이 구조의 MaxTablespaces에 스페이스를 할당합니다.
- 구조를 초기화합니다.
- 첫 번째 구조의 TBSPQVER을 SQLB_TBSPQRY_DATA_ID로 설정합니다.
- 이 스페이스를 지시하도록 pTablespaceData를 설정하십시오. API는 이 스페이스를 사용하여 테이블 스페이스 데이터를 리턴합니다.

pNumTablespaces

출력. 리턴된 출력 행 수

사용 시 참고사항

사용자는 pTablespaceData 매개변수로 지시된 메모리를 할당하고 사용 가능하게 해야 합니다. 이 API는 sqlbotsq 호출이 성공한 뒤에만 사용할 수 있습니다. sqlbotsq에서 생성된 목록을 페치하기 위해 반복적으로 호출할 수 있습니다.

제 93 장 sqlbgtss - 테이블 스페이스 사용 통계 가져오기

테이블 스페이스의 스페이스 사용량에 대한 정보를 제공합니다.

범위

파티션된 데이터베이스 환경에서 현재 데이터베이스 파티션의 테이블 스페이스만 나열됩니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm
- load

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbgtss (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   struct SQLB_TBS_STATS * pTablespaceStats);

SQL_API_RC SQL_API_FN
  sqlggtss (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   struct SQLB_TBS_STATS * pTablespaceStats);
```

sqlbgtss API 매개변수

pSqlca

출력. sqlca 구조의 포인터

TablespaceId

입력. 쿼리되는 단일 테이블 스페이스 ID

pTablespaceStats

출력. 사용자가 할당한 SQLB_TBS_STATS 구조의 포인터. 테이블 스페이스 정보가 이 구조에서 리턴됩니다.

사용 시 참고사항

리턴되는 필드 및 해당 의미에 대한 정보는 SQLB-TBS-STATS를 참조하십시오.

제 94 장 sqlbmtsq - 모든 테이블 스페이스에 대해 쿼리 데이터 가져오기

테이블 스페이스 쿼리 데이터에 대해 단일 호출 인터페이스를 제공합니다. 데이터베이스의 모든 테이블 스페이스에 대한 쿼리 데이터가 배열에 리턴됩니다.

범위

파티션된 데이터베이스 환경에서 현재 데이터베이스 파티션의 테이블 스페이스만 나열됩니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm
- load

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbmtsq (
   struct sqlca * pSqlca,
   sqluint32 * pNumTablespaces,
   struct SQLB_TBSPQRY_DATA *** pppTablespaceData,
   sqluint32 reserved1,
   sqluint32 reserved2);

SQL_API_RC SQL_API_FN
  sqlgmtsq (
   struct sqlca * pSqlca,
   sqluint32 * pNumTablespaces,
   struct SQLB_TBSPQRY_DATA *** pppTablespaceData,
   sqluint32 reserved1,
   sqluint32 reserved2);
```

sqlbmtsq API 매개변수

pSqlca

출력. sqlca 구조의 포인터

pNumTablespaces

출력. 연결된 데이터베이스의 총 테이블 스페이스 수.

pppTablespaceData

출력. 호출자는 포인터 주소와 함께 API를 제공합니다. 테이블 스페이스 쿼리 데이터의 스페이스는 API로 할당되고 해당 스페이스에 대한 포인터가 호출자에 리턴됩니다. 호출 리턴 시에 포인터는 완벽한 세트의 테이블 스페이스 쿼리 데이터의 SQLB_TBSPQRY_DATA 포인터 배열을 지시합니다.

reserved1

입력. 항상 SQLB_RESERVED1.

reserved2

입력. 항상 SQLB_RESERVED2.

사용 시 참고사항

이 API는 다음과 같은 낮은 레벨의 서비스를 사용하여

- sqlbotsq
- sqlbftpq
- sqlbctsq

한 번에 모든 테이블 스페이스 쿼리 데이터를 가져옵니다.

사용 가능한 메모리가 충분한 경우 이 함수는 테이블 스페이스 수 및 테이블 스페이스 쿼리 데이터의 메모리 위치 포인터를 리턴합니다. 사용자는 sqlefmem 호출로 이 메모리를 사용 가능하게 해야 합니다.

사용 가능한 메모리가 부족한 경우 이 함수는 단순히 테이블 스페이스 수를 리턴하고 메모리는 할당되지 않습니다. 이 경우 sqlbotsq, sqlbftpq 및 sqlbctsq를 사용하여 한 번에 전체 목록보다 적게 페치하십시오.

제 95 장 sqlbotcq - 테이블 스페이스 컨테이너 쿼리 열기

테이블 스페이스 컨테이너 쿼리 조작용으로 준비되어 현재 테이블 스페이스에 있는 컨테이너 수를 리턴합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbotcq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   sqluint32 * pNumContainers);

SQL_API_RC SQL_API_FN
  sqlgotcq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   sqluint32 * pNumContainers);
```

sqlbotcq API 매개변수

pSqlca

출력. sqlca 구조의 포인터

TablespaceId

입력. 컨테이너 데이터에 필요한 테이블 스페이스 ID. 특수한 ID인 SQLB_ALL_TABLESPACES(sqlutil.h에서)를 지정하면 전체 데이터베이스의 전체 컨테이너 목록이 작성됩니다.

pNumContainers

출력. 지정한 테이블 스페이스의 컨테이너 수.

사용 시 참고사항

일반적으로 이 API 뒤에는 하나 이상의 sqlbftcq 호출이 발생하며 그 뒤에 sqlbctcq 호출이 발생합니다.

응용프로그램은 다음 API를 사용하여 테이블 스페이스가 사용 중인 컨테이너에 대한 정보를 페치할 수 있습니다.

- sqlbtcq

  전체 컨테이너 정보 목록을 페치합니다. API는 모든 컨테이너에 대한 정보를 보유하는 데 필요한 스페이스를 할당하고 이 정보에 포인터를 리턴합니다. 이 API를 사용하여 특정 정보에 대한 컨테이너 목록을 스캔하십시오. 이 API를 사용하는 것은 아래 세 개의 API(sqlbotcq, sqlbftcq, sqlbctcq)를 호출하는 것과 동일합니다. 단지, 이 API는 출력 정보에 대해 자동으로 메모리를 할당하는 점만 다릅니다. 이 API에 대한 호출 뒤에는 sqlefmem을 호출하여 메모리를 사용 가능하도록 해야 합니다.

- sqlbotcq
- sqlbftcq
- sqlbctcq

  SQL 커서와 같이 이 세 개의 API 함수도 OPEN/FETCH/CLOSE 패러다임을 사용합니다. 호출자는 페치에 대해 출력 영역을 제공해야 합니다. SQL 커서와는 다르게 한 번에 한 개의 테이블 스페이스 컨테이너만 사용할 수 있습니다. 이 API 세트를 사용하여 특정 정보에 대한 테이블 스페이스 컨테이너 목록을 스캔하십시오. 이 API를 사용하여 응용프로그램의 메모리 요구사항을 제어할 수 있습니다(sqlbtcq와 비교)

sqlbotcq가 호출되면 현재 컨테이너 정보의 스냅샷이 에이전트 서비스 응용프로그램에서 형식화됩니다. 응용프로그램이 두 번째 테이블 스페이스 컨테이너 쿼리 호출을 발행하면(sqlbtcq 또는 sqlbotcq) 이 스냅샷은 새로 고쳐진 정보로 교체됩니다.

잠금이 수행되지 않아, 버퍼의 정보는 스냅샷 생성 이후에 다른 응용프로그램에서 작성된 변경사항을 반영하지 않을 수 있습니다. 정보는 트랜잭션의 파트는 아닙니다.

테이블 스페이스 쿼리에 대한 한 개의 스냅샷 버퍼가 있고 테이블 스페이스 컨테이너 쿼리에 대한 다른 버퍼가 있습니다. 이 두 버퍼는 서로 독립적입니다.

제 96 장 sqlbotsq - 테이블 스페이스 쿼리 열기

테이블 스페이스 쿼리 조작용으로 준비되어 현재 데이터베이스에 있는 테이블 스페이스 수를 리턴합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm
- load

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbotsq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceQueryOptions,
   sqluint32 * pNumTablespaces);

SQL_API_RC SQL_API_FN
  sqlgotsq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceQueryOptions,
   sqluint32 * pNumTablespaces);
```

sqlbotsq API 매개변수

pSqlca

출력. sqlca 구조의 포인터

TablespaceQueryOptions

입력. 처리할 테이블 스페이스를 표시합니다. 유효한 값(sqlutil에 정의)은 다음과 같습니다.

SQLB_OPEN_TBS_ALL
데이터베이스의 모든 테이블 스페이스를 처리합니다.

SQLB_OPEN_TBS_RESTORE
사용자의 에이전트가 리스토어 중인 테이블 스페이스만 처리합니다.

pNumTablespaces
출력. 연결된 데이터베이스에 있는 테이블 스페이스 수.

사용 시 참고사항

일반적으로 이 API 뒤에는 하나 이상의 sqlbftpq 호출이 발생하며 그 뒤에 sqlbctsq 호출이 발생합니다.

응용프로그램은 다음 API를 사용하여 현재 정의된 테이블 스페이스 정보를 페치할 수 있습니다.

- sqlbstpq

  지정된 테이블 스페이스에 대한 정보를 페치합니다. 한 개의 테이블 스페이스 항목만 리턴됩니다(호출자가 제공한 스페이스로). 테이블 스페이스 ID를 알고 해당 테이블 스페이스에 대해서만 정보가 필요한 경우에 이 API를 사용하십시오.

- sqlbmtsq

  모든 테이블 스페이스에 대한 정보를 페치합니다. API는 모든 테이블 스페이스에 대한 정보를 보유하는 데 필요한 스페이스를 할당하고 이 정보의 포인터를 리턴합니다. 이 API를 사용하여 특정 정보를 검색할 때 테이블 스페이스 목록을 스캔하십시오. 이 API를 사용하는 것은 아래 세 개의 API를 호출하는 것과 동일합니다. 단지, 이 API는 출력 정보에 대해 자동으로 메모리를 할당하는 점만 다릅니다. 이 API에 대한 호출 뒤에는 sqlefmem을 호출하여 메모리를 사용 가능하도록 해야 합니다.

- sqlbotsq
- sqlbftpq
- sqlbctsq

  SQL 커서와 같이 이 세 개의 API 함수도 OPEN/FETCH/CLOSE 패러다임을 사용합니다. 호출자는 페치에 대해 출력 영역을 제공해야 합니다. SQL 커서와는 다르게 한 번에 한 개의 테이블 스페이스 쿼리만 사용됩니다. 이 API 세트를 사용하여 특정 정보를 검색할 때 테이블 스페이스 목록을 스캔하십시오. 이 API 세트를 사용하여 응용프로그램의 메모리 요구사항을 제어할 수 있습니다(sqlbmtsq와 비교)

sqlbotsq가 호출되면 현재 테이블 스페이스 정보의 스냅샷이 에이전트 서비스 응용프로그램에 버퍼 지정됩니다. 응용프로그램이 두 번째 테이블 스페이스 쿼리 호출을 발행하면(sqlbmtsq 또는 sqlbotsq) 이 스냅샷이 새로 고쳐진 정보로 교체됩니다.

잠금이 수행되지 않아, 버퍼의 정보가 다른 응용프로그램에서 작성된 더 최근의 변경사항을 반영하지 않을 수 있습니다. 정보는 트랜잭션의 파트는 아닙니다.

테이블 스페이스 쿼리에 대한 한 개의 스냅샷 버퍼가 있고 테이블 스페이스 컨테이너 쿼리에 대한 다른 버퍼가 있습니다. 이 두 버퍼는 서로 독립적입니다.

제 97 장 sqlbstpq - 단일 테이블 스페이스에 대한 정보 가져오기

현재 정의된 단일 테이블 스페이스에 대한 정보를 검색합니다.

범위

파티션된 데이터베이스 환경에서 현재 데이터베이스 파티션의 테이블 스페이스만 나열됩니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm
- load

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbstpq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   struct SQLB_TBSPQRY_DATA * pTablespaceData,
   sqluint32 reserved);

SQL_API_RC SQL_API_FN
  sqlgstpq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   struct SQLB_TBSPQRY_DATA * pTablespaceData,
   sqluint32 reserved);
```

sqlbstpq API 매개변수

pSqlca

출력. sqlca 구조의 포인터

TablespaceId

입력. 쿼리되는 테이블 스페이스 ID

pTablespaceData

입력 및 출력. 리턴 시에 테이블 스페이스 정보가 배치되는 사용자가 제공하는 SQLB_TBSPQRY_DATA 구조의 포인터. 이 API의 호출자는 구조를 초기화하고 TBSPQVER를 SQLB_TBSPQRY_DATA_ID로(sqlutil에서) 설정해야 합니다.

예약 입력. 항상 SQLB_RESERVED1.

사용 시 참고사항

이 API는 쿼리되는 테이블 스페이스 ID를 알고 있는 경우에 단일 테이블 스페이스에 대한 정보를 검색합니다. 이 API는 훨씬 더 비싼 API의 OPEN TABLESPACE QUERY, FETCH 및 CLOSE 조합의 대체를 제공하며 테이블 스페이스 ID를 미리 알고 있는 경우에는 이를 사용하여 원하는 테이블 스페이스를 스캔해야 합니다. 테이블 스페이스 ID는 시스템 카탈로그에서 찾을 수 있습니다. 에이전트 스냅샷이 수행되지 않습니다. 리턴할 항목이 하나만 있으므로 직접 리턴됩니다.

제 98 장 sqlbstsc - 테이블 스페이스 컨테이너 설정

이 API는 경로 재지정된 리스토어의 프로비전이 용이하도록 합니다. 이 경우 사용자가 데이터베이스를 리스토어하고, 다른 운영 체제 스토리지 컨테이너 세트가 바람직하거나 필요합니다. 테이블 스페이스가 스토리지 정의 보류 중 상태에 있거나 스토리지 정의 허용 상태에 있을 때 이 API를 사용하십시오. 이 상태는 데이터베이스 페이지 리스토어 바로 이전에, 리스토어 조작 중에 가능합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbstsc (
   struct sqlca * pSqlca,
   sqluint32 SetContainerOptions,
   sqluint32 TablespaceId,
   sqluint32 NumContainers,
   struct SQLB_TBSCONTQRY_DATA * pContainerData);

SQL_API_RC SQL_API_FN
  sqlgstsc (
   struct sqlca * pSqlca,
   sqluint32 SetContainerOptions,
   sqluint32 TablespaceId,
   sqluint32 NumContainers,
   struct SQLB_TBSCONTQRY_DATA * pContainerData);
```

sqlbstsc API 매개변수

pSqlca

출력. sqlca 구조의 포인터

SetContainerOptions

입력. 추가 옵션을 지정하려면 이 필드를 사용하십시오. 유효한 값(sqlutil에 정의)은 다음과 같습니다.

SQLB_SET_CONT_INIT_STATE
롤 포워드를 수행할 때 테이블 스페이스 변경 조작을 재실행합니다.

SQLB_SET_CONT_FINAL_STATE
롤 포워드를 수행할 때 테이블 스페이스 변경 조작을 무시합니다.

TablespaceId
입력. 변경되는 테이블 스페이스의 ID

NumContainers
입력. pContainerData가 지시하는 구조가 보유하는 행 수. pContainerData에 대해 NULL 포인터와 함께 0 값이 제공되면 테이블 스페이스가 자동 스토리지로 관리됨을 표시합니다.

pContainerData
입력. 컨테이너 스펙. SQLB_TBSCONTQRY_DATA 구조가 사용되어도, contType, totalPages, name 및 nameLen(C가 아닌 다른 언어의 경우) 필드만 사용됩니다. 다른 모든 필드는 무시됩니다. NumContainers에 대해 NULL 값과 함께 0 값이 제공되면 테이블 스페이스가 자동 스토리지로 관리됨을 표시합니다. 이 옵션은 컨테이너를 재정의하여 기존 스토리지 경로에서 기존의 자동 스토리지 가능 테이블 스페이스에 대해 더 나은 스트라이핑을 제공하기 위해 사용할 수도 있습니다.

주: 테이블 스페이스는 리스토어되는 동안 오프라인 상태가 됩니다.

사용 시 참고사항

이 API는 db2Restore와 함께 사용됩니다.

데이터베이스 또는 하나 이상의 테이블 스페이스 백업은 백업하는 테이블 스페이스가 사용 중인 모든 테이블 스페이스 컨테이너 레코드를 보존합니다. 리스토어 중, 백업에 나열된 모든 컨테이너를 점검하여 현재 존재하며 액세스 가능한지 확인합니다. 어떤 이유로 하나 이상의 컨테이너에 액세스할 수 없는 경우 리스토어가 실패합니다. 이와 같은 경우에 리스토어를 허용하기 위해, 리스토어 중에 테이블 스페이스 컨테이너의 경로 재지정이 지원됩니다. 이러한 지원에는 테이블 스페이스 컨테이너 추가, 변경 또는 제거가 포함됩니다. 사용자가 해당 컨테이너를 추가, 변경 또는 제거할 수 있는 것이 이 API입니다.

이 API의 일반적인 사용으로 다음의 조치 시퀀스가 있습니다.

1. CallerAction을 DB2RESTORE_RESTORE_STORDEF로 설정하여 db2Restore를 호출합니다. 리스토어 유틸리티는 컨테이너 중 일부에 액세스할 수 없음을 표시하는 sqlcode를 리턴합니다.
2. SetContainerOptions 매개변수를 SQLB_SET_CONT_FINAL_STATE로 설정하여 sqlbstsc를 호출하여 테이블 스페이스 컨테이너 정의를 설정합니다.

3. CallerAction을 DB2RESTORE_CONTINUE로 설정하여 db2Restore를 두 번 호출합니다.

위의 시퀀스는 리스토어에서 새 테이블 스페이스 컨테이너 정의를 사용하도록 허용하고 리스토어 완료 후 db2Rollforward가 호출될 때 로그에서 테이블 스페이스 컨테이너 추가 조작을 무시합니다.

이 API의 사용자는 컨테이너 목록을 설정할 때 리스토어 또는 롤백 조작이 원래의 모든 데이터를 이 새로운 컨테이너로 교체할 수 있도록 충분한 디스크 스페이스가 있어야 한다는 점에 유의해야 합니다. 충분한 스페이스가 없으면, 그러한 테이블 스페이스는 충분한 디스크 스페이스가 사용 가능할 때까지 복구 보류 상태로 남습니다. 신중한 데이터베이스 관리자는 일반적인 기준에 따라 디스크 이용 레코드를 유지합니다. 그러면, 리스토어 또는 롤 포워드 조작이 필요할 때 필요한 디스크 스페이스를 알 수 있습니다.

이 API를 사용하여 테이블 스페이스에 대한 자동 스토리지가 가능하도록 하면 모든 현재 컨테이너가 데이터베이스에 제공된 스토리지 경로를 사용하도록 재정의됩니다.

기존의 SMS(system-managed) 테이블 스페이스는 자동 스토리지를 사용하도록 변환할 수 없습니다.

SetContainerOptions는 테이블 스페이스가 자동 스토리지를 사용하도록 변환될 때 (NumContainers가 0이고 pContainerData가 NULL임) 무시됩니다.

SET TABLESPACE CONTAINERS 문의 USING AUTOMATIC STORAGE 옵션을 사용하는 다중 파티션 환경에서의 테이블 스페이스의 경로 재지정된 리스토어만 리스토어되는 파티션에서 테이블 스페이스를 자동 스토리지로 변환합니다. 다른 데이터베이스 파티션의 컨테이너는 재정의되지 않습니다.

주: 파티션 중 단 하나에서만 테이블 스페이스를 경로 재지정된 리스토어 조작의 일부로 자동 스토리지로 변환하면 테이블 스페이스 정의에서 불일치가 발생합니다. 새 데이터베이스 파티션을 시스템이나 데이터베이스 파티션 그룹에 추가할 때 예기치 않은 결과가 발생할 수도 있습니다. 예를 들어, 모든 데이터베이스 파티션이 SET TABLESPACE CONTAINERS 명령의 USING AUTOMATIC STORAGE 옵션을 사용하여 경로 재지정된 리스토어를 따른 경우, 테이블 스페이스는 모든 데이터베이스 파티션에서 자동 스토리지로 변환됩니다. 나중에 다른 데이터베이스 파티션을 추가하면 다른 데이터베이스 파티션에서 발견된 것과 같은 테이블 스페이스 정의를 갖습니다.

제 99 장 sqlbtcq - 모든 테이블 스페이스 컨테이너에 대한 쿼리 데이터 가져오기

테이블 스페이스 컨테이너 쿼리 데이터에 대한 단일 호출 인터페이스를 제공합니다. 테이블 스페이스의 모든 컨테이너에 대한 쿼리 데이터 또는 모든 테이블 스페이스의 모든 컨테이너에 대한 쿼리 데이터가 배열에 리턴됩니다.

범위

파티션된 데이터베이스 환경에서 현재 데이터베이스 파티션의 테이블 스페이스만 나열됩니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- sysmon
- dbadm

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlbtcq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   sqluint32 * pNumContainers,
   struct SQLB_TBSCONTQRY_DATA ** ppContainerData);

SQL_API_RC SQL_API_FN
  sqlgtcq (
   struct sqlca * pSqlca,
   sqluint32 TablespaceId,
   sqluint32 * pNumContainers,
   struct SQLB_TBSCONTQRY_DATA ** ppContainerData);
```

sqlbtcq API 매개변수

pSqlca

출력. sqlca 구조의 포인터

TablespaceId

입력. 컨테이너 데이터가 필요한 테이블 스페이스의 ID 또는 전체 데이터베이스를 위한 모든 컨테이너의 목록을 작성하는 특수 ID인 SQLB_ALL_TABLESPACES(sqlutil에 정의)

pNumContainers

출력. 테이블 스페이스에 있는 컨테이너 수.

ppContainerData

출력. 호출자는 SQLB_TBSCONTQRY_DATA 구조 포인터 주소가 포함된 API를 제공합니다. API로 테이블 스페이스 컨테이너 쿼리 데이터에 대한 스페이스가 할당되고 해당 스페이스의 포인터가 호출자에게 리턴됩니다. 호출에서 리턴 시 SQLB_TBSCONTQRY_DATA 구조에 대한 포인터는 테이블 스페이스 컨테이너 쿼리 데이터의 전체 세트를 지시합니다.

사용 시 참고사항

이 API는 다음과 같은 낮은 레벨의 서비스를 사용하여

- sqlbotcq
- sqlbftcq
- sqlbctcq

한 번에 모든 테이블 스페이스 컨테이너 쿼리 데이터를 가져옵니다.

사용 가능한 메모리가 충분한 경우 이 함수는 컨테이너 수 및 테이블 스페이스 컨테이너 쿼리 데이터의 메모리 위치 포인터를 리턴합니다. 사용자는 sqlefmem 호출로 이 메모리를 사용 가능하게 해야 합니다. 사용 가능한 메모리가 부족한 경우 이 함수는 단순히 컨테이너 수를 리턴하고 메모리는 할당되지 않습니다. 이 경우 sqlbotcq, sqlbftcq 및 sqlbctcq를 사용하여 한 번에 전체 목록보다 적게 페치하십시오.

제 100 장 sqlcspqy - DRDA 인다우트(Indoubt) 트랜잭션 나열

동기점 관리 프로그램 상대 연결 중간에 인다우트(Indoubt)인 트랜잭션 목록을 제공합니다. 이 API는 더 이상 사용되지 않습니다. 'db2SpmListIndTrans API - SPM 인다우트(Indoubt) 트랜잭션 목록'을 참조하십시오.

권한 부여

없음

필수 연결

인스턴스

API 내장 파일

```
sqlxa.h
```

API 및 데이터 구조 구문

```
extern int SQL_API_FN sqlcspqy(SQLCSPQY_INDOUBT     **indoubt_data,
                               sqlint32 *indoubt_count,
                               struct sqlca *sqlca);
```

sqlcspqy API 매개변수

indoubt_data
출력. 리턴된 배열의 포인터

indoubt_count
출력. 리턴된 배열의 요소 수

pSqlca
출력. sqlca 구조의 포인터

사용 시 참고사항

분산 작업 단위(DUOW)에서 코디네이터와 구성원(participant) 사이의 통신이 단절되면 DRDA 인다우트(Indoubt) 트랜잭션이 발생합니다.

분산 작업 단위(DUOW)를 사용하여 사용자나 응용프로그램은 단일 작업 단위(UOW) 내의 여러 위치에서 데이터를 읽고 갱신할 수 있습니다. 그러한 작업을 위해서는 2단계의 커미트가 필요합니다.

첫 번째 단계에서는 모든 구성원(participant)이 커미트를 준비하도록 요청됩니다. 두 번째 단계에서는 트랜잭션을 커미트하거나 롤백합니다. 코디네이터나 구성원(participant)이 첫 번째 단계 이후에 사용 불가능해진 경우 분산 트랜잭션이 인다우트(Indoubt)입니다.

LIST DRDA INDOUBT TRANSACTIONS 명령을 실행하기 전에 응용프로그램 프로세스를 동기점 관리 프로그램(SPM) 인스턴스에 연결해야 합니다. spm_name 데이터베이스 관리 프로그램 구성 매개변수를 CONNECT 문의 dbalias로 사용하십시오.

제 101 장 sqle_activate_db - 데이터베이스 활성화

데이터베이스에 연결할 수 있고 응용프로그램에서 사용할 수 있도록 지정한 데이터베이스를 활성화하고 모든 필요한 데이터베이스 서비스를 시작합니다.

범위

이 API는 모든 데이터베이스 파티션 서버에서 지정한 데이터베이스를 활성화합니다. 하나 이상의 이 데이터베이스 파티션 서버에서 데이터베이스 활성화 중에 오류가 발생하면 경고가 리턴됩니다. 데이터베이스는 API가 성공한 모든 데이터베이스 파티션 서버에서 활성화된 채로 유지됩니다.

주: 오류가 발생한 파티션이 코디네이터 파티션이거나 카탈로그 파티션인 경우 API는 음수의 SQL 코드를 리턴하고 모든 데이터베이스 파티션 서버에서 데이터베이스가 활성화되지 않습니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

없음. ACTIVATE DATABASE를 호출하는 응용프로그램에는 기존 데이터베이스 연결이 포함될 수 없습니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqle_activate_db (
        char * pDbAlias,
        char * pUserName,
        char * pPassword,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlg_activate_db (
        unsigned short DbAliasLen,
        unsigned short UserNameLen,
```

```
unsigned short PasswordLen,
char * pDbAlias,
char * pUserName,
char * pPassword,
void * pReserved,
struct sqlca * pSqlca);
```

sqle_activate_db API 매개변수

pDbAlias

입력. 데이터베이스 별명 이름의 포인터

pUserName

입력. 데이터베이스를 시작하는 사용자 ID의 포인터. NULL일 수 있습니다.

pPassword

입력. 사용자 이름 암호의 포인터. NULL일 수는 있지만 사용자 이름을 지정하는 경우 지정해야 합니다.

pReserved

나중에 사용하기 위해 예약됨

pSqlca

출력. sqlca 구조의 포인터

sqlg_activate_db API 특정 매개변수

DbAliasLen

입력. 데이터베이스 별명 이름 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

UserNameLen

입력. 사용자 이름 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 사용자 이름을 제공하지 않은 경우 영(0)으로 설정하십시오.

PasswordLen

입력. 암호 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 암호를 제공하지 않은 경우에는 영(0)으로 설정하십시오.

사용 시 참고사항

데이터베이스가 시작되지 않고 응용프로그램에서 DB2 CONNECT TO(또는 내재적 연결)가 발생한 경우 응용프로그램은 데이터베이스 관리 프로그램이 필요한 데이터베이스를 시작할 때까지 대기해야 합니다. 이런 경우 이 첫 번째 응용프로그램은 작업을 수행하기 전에 데이터베이스 초기화를 위해 시간을 소비합니다. 그러나 첫 번째 응용프로그램이 데이터베이스를 시작하면 다른 응용프로그램은 단순히 연결하여 사용할 수 있습니다.

데이터베이스 관리자는 ACTIVATE DATABASE를 사용하여 선택한 데이터베이스를 시작할 수 있습니다. 그러면 모든 응용프로그램이 데이터베이스 초기화에 시간을 소비하지 않아도 됩니다.

ACTIVATE DATABASE로 초기화된 데이터베이스는 sqle_deactivate_db 또는 db2InstanceStop으로 종료할 수 있습니다. 활성화된 데이터베이스 목록을 가져오려면 db2GetSnapshot을 호출하십시오.

DB2 CONNECT TO(또는 내재적 연결)로 데이터베이스가 시작되고 동일한 해당 데이터베이스에 대해 연이어 ACTIVATE DATABASE가 실행된 경우 해당 데이터베이스를 종료하려면 DEACTIVATE DATABASE를 사용해야 합니다.

재시작해야 하는 데이터베이스에 대해 작업할 때(예를 들어 데이터베이스가 불일치 상태인 경우) ACTIVATE DATABASE는 DB2 CONNECT TO(또는 내재적 연결)와 유사하게 작동합니다. 데이터베이스는 ACTIVATE DATABASE로 초기화되기 전에 재시작됩니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 102 장 sqle_deactivate_db - 데이터베이스 비활성화

지정한 데이터베이스를 중지합니다.

범위

파티션된 데이터베이스 환경에서 이 API는 모든 데이터베이스 파티션 서버에서 지정한 데이터베이스를 비활성화합니다. 이 데이터베이스 파티션 서버 중 하나 이상에 오류가 발생하면 경고가 리턴됩니다. 데이터베이스는 일부 데이터베이스 파티션 서버에서 제대로 비활성화되지만 오류가 발생한 데이터베이스 파티션 서버에서는 활성화되어 있을 수도 있습니다.

주: 오류가 발생한 파티션이 코디네이터 파티션 또는 카탈로그 파티션인 경우 API는 음수의 sqlcode를 리턴하고 비활성화된 어떤 데이터베이스 파티션 서버에서도 데이터베이스가 다시 활성화되지 않습니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

없음. DEACTIVATE DATABASE를 호출하는 응용프로그램에는 기존 데이터베이스 연결이 포함될 수 없습니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqle_deactivate_db (
        char * pDbAlias,
        char * pUserName,
        char * pPassword,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlg_deactivate_db (
        unsigned short DbAliasLen,
        unsigned short UserNameLen,
```

```
unsigned short PasswordLen,
char * pDbAlias,
char * pUserName,
char * pPassword,
void * pReserved,
struct sqlca * pSqlca);
```

sqle_deactivate_db API 매개변수

pDbAlias

입력. 데이터베이스 별명 이름의 포인터

pUserName

입력. 데이터베이스를 중지하는 사용자 ID의 포인터. NULL일 수 있습니다.

pPassword

입력. 사용자 이름 암호의 포인터. NULL일 수는 있지만 사용자 이름을 지정하는 경우 지정해야 합니다.

pReserved

나중에 사용하기 위해 예약됨

pSqlca

출력. sqlca 구조의 포인터

sqlg_deactivate_db API 특정 매개변수

DbAliasLen

입력. 데이터베이스 별명 이름 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

UserNameLen

입력. 사용자 이름 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 사용자 이름을 제공하지 않은 경우 영(0)으로 설정하십시오.

PasswordLen

입력. 암호 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 암호를 제공하지 않은 경우에는 영(0)으로 설정하십시오.

사용 시 참고사항

ACTIVATE DATABASE로 초기화되는 데이터베이스는 DEACTIVATE DATABASE로만 종료할 수 있습니다. db2InstanceStop은 데이터베이스 관리 프로그램을 중지하기 전에 자동으로 모든 활성화된 데이터베이스를 중지합니다. 데이터베이스가 ACTIVATE DATABASE로 초기화된 경우 마지막 DB2 CONNECT RESET 문(카운터는 0)은 데이터베이스를 종료하지 않습니다. DEACTIVATE DATABASE를 사용해야 합니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 103 장 sqleaddn - 파티션된 데이터베이스 환경에 데이터베이스 파티션 추가

데이터베이스 파티션 서버에 데이터베이스 파티션을 추가합니다.

범위

이 API는 실행된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleaddn (
        void * pAddNodeOptions,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgaddn (
        unsigned short addnOptionsLen,
        struct sqlca * pSqlca,
        void * pAddNodeOptions);
```

sqleaddn API 매개변수

+pAddNodeOptions

입력. 선택적 sqle_addn_options 구조의 포인터. 이 구조를 사용하여 작성한 모든 데이터베이스 파티션의 시스템 임시 테이블 스페이스 정의가 있는 경우 해당 소스 데이터베이스 파티션 서버를 지정합니다. 지정하지 않으면(즉, NULL 포인터를 지정하면) 시스템 임시 테이블 스페이스 정의는 카탈로그 파티션의 정의와 동일하게 됩니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgaddn API 특정 매개변수

addnOptionsLen

입력. 선택적 sqle_addn_options 구조 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

이 API는 데이터베이스 파티션 서버가 한 개의 데이터베이스를 포함하며 파티션 추가 조작 시에 데이터베이스가 카탈로그되지 않은 환경에 추가되는 경우에만 사용해야 합니다. 이 상황에서는 데이터베이스가 카탈로그되지 않기 때문에 파티션 추가 조작이 데이터베이스를 인식하지 못하고 새 데이터베이스 파티션 서버의 데이터베이스에 대해 데이터베이스 파티션을 작성하지 않습니다. 새 데이터베이스 파티션 서버에서 데이터베이스 파티션 연결 시도는 오류를 초래합니다. 데이터베이스는 sqleaddn API를 사용하여 새 데이터베이스 파티션 서버에서 데이터베이스에 대한 데이터베이스 파티션을 작성하기 전에 우선 카탈로그해야 합니다.

이 API는 둘 이상의 데이터베이스가 있고 이 데이터베이스 중 하나 이상이 파티션 추가 조작 시에 카탈로그된 경우 사용하면 안됩니다. 이 상황에서는 sqlecran API를 사용하여 파티션 추가 조작 시에 카탈로그되지 않은 각 데이터베이스에 대해 데이터베이스 파티션을 작성하십시오. 카탈로그 해제된 각 데이터베이스는 sqlecran API를 사용하여 새 데이터베이스 파티션 서버에서 데이터베이스에 대해 데이터베이스 파티션을 작성하기 전에 우선 카탈로그해야 합니다.

새 데이터베이스 파티션을 추가하기 전에 작성해야 하는 컨테이너에 필요한 스토리지가 충분한지 확인하십시오.

노드 추가 조작은 인스턴스에 있는 모든 데이터베이스의 새 데이터베이스 파티션 서버에 비어 있는 데이터베이스 파티션을 작성합니다. 새 데이터베이스 파티션의 구성 매개변수는 디폴트값으로 설정됩니다.

주: 카탈로그 해제된 데이터베이스는 새 데이터베이스 파티션을 추가할 때 인식되지 않습니다. 카탈로그 해제된 데이터베이스는 새 데이터베이스 파티션에 표시되지 않습니다. 새 데이터베이스 파티션에서 데이터베이스에 연결하려고 하면 오류 메시지 SQL1013N이 리턴됩니다.

로컬에서 데이터베이스 파티션 작성 시 노드 추가 조작이 실패하면 제거 단계가 되고 이 경우 작성된 모든 데이터베이스가 로컬에서 삭제됩니다. 즉, 데이터베이스 파티션은 추가 중인 데이터베이스 파티션 서버(즉, 로컬 데이터베이스 파티션 서버)에서만 제거됩

니다. 기존 데이터베이스 파티션은 모든 다른 데이터베이스 파티션 서버에서 이전과 동일하게 유지됩니다. 이 상황이 실패하면 추가적인 제거는 수행되지 않으며 오류가 리턴됩니다.

새 데이터베이스 파티션 서버의 데이터베이스 파티션은 ALTER DATABASE PARTITION GROUP 문을 사용하여 데이터베이스 파티션 서버를 데이터베이스 파티션 그룹에 추가해야 사용자 데이터를 포함하는 데 사용할 수 있습니다.

이 API는 데이터베이스 작성 또는 데이터베이스 삭제 조작이 진행 중인 경우 실패합니다. API는 조작이 완료될 때 다시 호출할 수 있습니다.

데이터베이스의 자동 스토리지 사용 가능 여부를 판별하기 위해 sqleaddn API가 인스턴스의 각 데이터베이스에 대해 카탈로그 파티션과 통신합니다. 자동 스토리지가 사용 가능한 경우 스토리지 경로 정의는 통신 중에 검색됩니다. 이와 유사하게 시스템 임시 테이블 스페이스가 데이터베이스 파티션과 같이 작성되는 경우 tsqleaddn API는 테이블 스페이스 정의를 검색하기 위해 파티션된 데이터베이스 환경의 다른 데이터베이스 파티션 서버와 통신해야 합니다. **start_stop_time** 데이터베이스 관리 프로그램 구성 매개변수를 사용하여 다른 데이터베이스 파티션 서버가 자동 스토리지 및 테이블 스페이스 정의에 응답해야 하는 시간을 분 단위로 지정합니다. 이 시간이 초과되면 API가 실패합니다. **start_stop_time** 값을 증가시키고 API를 다시 호출하십시오.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 104 장 sqleatcp - 인스턴스에 접속 및 암호 변경

응용프로그램이 인스턴스 레벨 함수(예: CREATE DATABASE 및 FORCE APPLICATION)가 실행되는 노드를 지정할 수 있도록 합니다. 이 노드는 현재 인스턴스(DB2INSTANCE 환경 변수 값으로 정의), 동일한 워크스테이션에 있는 다른 인스턴스 또는 리모트 워크스테이션에 있는 인스턴스일 수 있습니다. 지정한 노드에 대해 논리적 인스턴스 접속을 설정하고 노드에 물리적으로 통신이 연결되어 있지 않은 경우에는 물리적 통신 연결을 시작하십시오.

주: 이 API는 접속 중인 인스턴스에 대해 사용자 암호를 선택적으로 변경할 수 있도록 하여 sqleatin API 함수를 확장합니다. DB2 데이터베이스 시스템은 AIX, Linux 및 Windows 운영 체제에서 암호 변경을 지원합니다.

권한 부여

없음

필수 연결

이 API는 인스턴스 접속을 설정합니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleatcp (
        char * pNodeName,
        char * pUserName,
        char * pPassword,
        char * pNewPassword,
        struct sqlca * pSqlca);
```

sqleatcp API 매개변수

pNodeName

입력. 사용자가 접속하려는 인스턴스 별명이 포함된 문자열. 이 인스턴스에는 로컬 노드 디렉토리에 일치하는 항목이 있어야 합니다. 유일한 예외는 로컬 인스턴스(DB2INSTANCE 환경 변수로 지정)이며 접속된 오브젝트로 지정할 수는 있지만 노드 디렉토리에서 노드 이름으로 사용할 수는 없습니다. NULL도 가능합니다.

pUserName

입력. 접속이 인증되는 사용자 이름이 포함된 문자열. NULL도 가능합니다.

pPassword

입력. 지정한 사용자 이름의 암호가 포함된 문자열. NULL도 가능합니다.

pNewPassword

입력. 지정한 사용자 이름의 새 암호가 포함된 문자열. 암호를 변경할 필요가 없으면 NULL로 설정하십시오.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

노드 디렉토리의 노드 이름은 인스턴스 별명으로 사용될 수 있습니다.

접속 요청이 성공하면 sqlca의 sqlerrmc 필드에 16진수 FF로 구분된 9개의 토큰이 포함됩니다(CONNECT 요청 성공 시에 리턴되는 토큰과 유사)

1. 응용프로그램 서버(AS)의 국가/지역 코드
2. 응용프로그램 서버(AS)의 코드 페이지
3. 권한 부여 ID
4. 노드 이름(API에 설정)
5. 서버의 플랫폼 유형 식별
6. 서버에서 시작된 에이전트의 에이전트 ID
7. 에이전트 인덱스
8. 서버의 노드 번호
9. 서버가 파티션된 데이터베이스 서버인 경우 데이터베이스 파티션 수

노드 이름이 길이가 0인 문자열 또는 NULL인 경우 접속의 현재 상태에 대한 정보가 리턴됩니다. 접속이 없는 경우에는 sqlcode 1427이 리턴됩니다. 그렇지 않으면 접속에 대한 정보가 sqlca의 sqlerrmc 필드에 리턴됩니다(위에서 설명한 대로)

접속된 인스턴스 레벨의 API가 DB2INSTANCE 환경 변수로 지정된 현재 인스턴스에 대해 실행됩니다.

특정 함수(예: db2start, db2stop 및 모든 디렉토리 서비스)는 리모트로 실행되지 않습니다. 즉, DB2INSTANCE 환경 변수 값으로 정의된 로컬 인스턴스 환경에만 영향을 줍니다.

접속되어 있고 API가 노드 이름으로 발행된 경우 현재 접속이 삭제되고 새 노드의 접속이 시도됩니다.

사용자 이름 및 암호가 인증되고 암호가 변경되는 위치는 목표 인스턴스의 인증 유형에 따라 다릅니다.

접속이 작성되는 노드는 sqlesetc API 호출로도 지정할 수 있습니다.

REXX API 구문

REXX에서 이 API를 직접 호출할 수는 없습니다. 그러나 REXX 프로그래머가 DB2 명령행 처리기를 호출하여 ATTACH 명령이 실행되도록 이 함수를 사용할 수는 있습니다.

제 105 장 sqleatin - 인스턴스에 접속

응용프로그램이 인스턴스 레벨 함수(예: CREATE DATABASE 및 FORCE APPLICATION)가 실행되는 노드를 지정할 수 있습니다. 이 노드는 현재 인스턴스(DB2INSTANCE 환경 변수 값으로 정의), 동일한 워크스테이션에 있는 다른 인스턴스 또는 리모트 워크스테이션에 있는 인스턴스일 수 있습니다. 지정한 노드에 논리적 인스턴스 접속을 설정하고 노드에 물리적으로 통신이 연결되어 있지 않은 경우에는 물리적 통신 연결을 시작하십시오.

주: 암호를 변경해야 하는 경우에는 sqleatin API 대신 sqleatcp API를 사용하십시오.

권한 부여

없음

필수 연결

이 API는 인스턴스 접속을 설정합니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleatin (
        char * pNodeName,
        char * pUserName,
        char * pPassword,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgatin (
        unsigned short PasswordLen,
        unsigned short UserNameLen,
        unsigned short NodeNameLen,
        struct sqlca * pSqlca,
        char * pPassword,
        char * pUserName,
        char * pNodeName);
```

sqleatin API 매개변수

pNodeName

입력. 사용자가 접속하려는 인스턴스 별명이 포함된 문자열. 이 인스턴스에는 로컬 노드 디렉토리에 일치하는 항목이 있어야 합니다. 유일한 예외는 로컬 인스

턴스(DB2INSTANCE 환경 변수로 지정)이며 접속된 오브젝트로 지정할 수는 있지만 노드 디렉토리에서 노드 이름으로 사용할 수는 없습니다. NULL일 수 있습니다.

pUserName

입력. 접속이 인증되는 사용자 이름이 포함된 문자열. NULL일 수 있습니다.

pPassword

입력. 지정한 사용자 이름의 암호가 포함된 문자열. NULL일 수 있습니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgatin API 특정 매개변수

PasswordLen

입력. 암호 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 암호를 제공하지 않은 경우에는 영(0)으로 설정하십시오.

UserNameLen

입력. 사용자 이름 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 사용자 이름을 제공하지 않은 경우 영(0)으로 설정하십시오.

NodeNameLen

입력. 노드 이름 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 노드 이름을 제공하지 않은 경우 영(0)으로 설정하십시오.

사용 시 참고사항

주: 노드 디렉토리의 노드 이름은 인스턴스 별명으로 사용될 수 있습니다.

접속 요청이 성공하면 sqlca의 sqlerrmc 필드에 16진수 FF로 구분된 9개의 토큰이 포함됩니다(CONNECT 요청 성공 시에 리턴되는 토큰과 유사).

1. 응용프로그램 서버(AS)의 국가/지역 코드
2. 응용프로그램 서버(AS)의 코드 페이지
3. 권한 부여 ID
4. 노드 이름(API에 설정)
5. 서버의 플랫폼 유형 식별
6. 서버에서 시작된 에이전트의 에이전트 ID
7. 에이전트 인덱스
8. 서버의 노드 번호
9. 서버가 파티션된 데이터베이스 서버인 경우 데이터베이스 파티션 수

노드 이름이 길이가 0인 문자열 또는 NULL인 경우 접속의 현재 상태에 대한 정보가 리턴됩니다. 접속이 없는 경우에는 sqlcode 1427이 리턴됩니다. 그렇지 않으면 접속에 대한 정보가 sqlca의 sqlerrmc 필드에 리턴됩니다(위에서 설명한 대로)

접속이 없는 경우에는 작성된 인스턴스 레벨의 API가 DB2INSTANCE 환경 변수로 지정된 현재 인스턴스에 대해 실행됩니다.

특정 함수(예: db2start, db2stop 및 모든 디렉토리 서비스)는 리모트로 실행되지 않습니다. 즉, DB2INSTANCE 환경 변수 값으로 정의된 로컬 인스턴스 환경에만 영향을 줍니다.

접속되어 있고 API가 노드 이름으로 발행된 경우 현재 접속이 삭제되고 새 노드의 접속이 시도됩니다.

사용자 이름 및 암호가 인증되는 위치는 목표 인스턴스의 인증 유형에 따라 다릅니다.

접속된 노드는 sqlesetc API 호출로도 지정할 수 있습니다.

REXX API 구문

```
ATTACH [TO nodename [USER username USING password]]
```

REXX API 매개변수

nodename

사용자가 접속하려는 인스턴스의 별명. 이 인스턴스에는 로컬 노드 디렉토리에 일치하는 항목이 있어야 합니다. 유일한 예외는 로컬 인스턴스(DB2INSTANCE 환경 변수로 지정)이며 접속된 오브젝트로 지정할 수는 있지만 노드 디렉토리에서 노드 이름으로 사용할 수는 없습니다.

username

사용자가 인스턴스에 접속되는 이름

password

사용자 이름을 인증하는 데 사용되는 암호

제 106 장 sqlecadb - 시스템 데이터베이스 디렉토리에 데이터베이스 카탈로그

시스템 데이터베이스 디렉토리에 데이터베이스 위치 정보를 저장합니다. 데이터베이스는 로컬 워크스테이션 또는 데이터베이스 파티션 서버에 위치할 수 있습니다.

범위

이 API는 시스템 데이터베이스 디렉토리에 영향을 줍니다. 파티션된 데이터베이스 환경에서 로컬 데이터베이스를 시스템 데이터베이스 디렉토리에 카탈로깅할 때 데이터베이스가 있는 데이터베이스 파티션 서버에서 이 API를 호출해야 합니다.

권한 부여

다음 중 하나가 필요합니다.

- SYSADM
- SYSCTRL

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlecadb (
        _SQLOLDCHAR * pDbName,
        _SQLOLDCHAR * pDbAlias,
        unsigned char Type,
        _SQLOLDCHAR * pNodeName,
        _SQLOLDCHAR * pPath,
        _SQLOLDCHAR * pComment,
        unsigned short Authentication,
        _SQLOLDCHAR * pPrincipal,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcadb (
        unsigned short PrinLen,
        unsigned short CommentLen,
        unsigned short PathLen,
        unsigned short NodeNameLen,
        unsigned short DbAliasLen,
        unsigned short DbNameLen,
```

```
struct sqlca * pSqlca,
_SQLOLDCHAR * pPrinName,
unsigned short Authentication,
_SQLOLDCHAR * pComment,
_SQLOLDCHAR * pPath,
_SQLOLDCHAR * pNodeName,
unsigned char Type,
_SQLOLDCHAR * pDbAlias,
_SQLOLDCHAR * pDbName);
```

sqlecadb API 매개변수

pDbName

입력. 데이터베이스 이름이 포함된 문자열

pDbAlias

입력. 데이터베이스 별명이 포함된 문자열

유형 입력. 데이터베이스가 간접, 리모트인지 또는 DCE를 통해 카탈로그되었는지를 지정하는 단일 문자. 올바른 값(sqlenv.h에 정의)은 다음과 같습니다.

SQL_INDIRECT

데이터베이스가 이 인스턴스에 있도록 지정합니다.

SQL_REMOTE

데이터베이스가 다른 인스턴스에 있도록 지정합니다.

SQL_DCE

데이터베이스가 DCE를 통해 카탈로그되도록 지정합니다.

pNodeName

입력. 데이터베이스가 있는 데이터베이스 파티션 이름이 포함된 문자열. NULL일 수 있습니다.

주: pPath와 **pNodeName** 모두 지정하지 않으면 데이터베이스는 로컬로 간주되며 데이터베이스 위치는 데이터베이스 관리 프로그램 구성 매개변수 **dftdbpath**에 지정된 위치로 간주됩니다.

pPath

입력. Linux 및 UNIX 시스템에서 문자열은 카탈로그 중인 데이터베이스가 있는 경로 이름을 지정합니다. 최대 길이는 215자입니다.

Windows 운영 체제에서 이 문자열은 카탈로그하는 데이터베이스가 있는 드라이브 이름을 지정합니다.

NULL 포인터를 제공하는 경우 디폴트 데이터베이스 경로는 데이터베이스 관리 프로그램 구성 매개변수인 **dftdbpath**에서 지정한 경로로 간주됩니다.

pComment

입력. 데이터베이스의 선택적 설명이 포함된 문자열. 널(NULL) 문자열은 주석이 없음을 나타냅니다. 주석 문자열의 최대 길이는 30자입니다.

인증

입력. 데이터베이스에 대해 지정된 인증 유형을 포함합니다. Authentication은 요청한 사용자를 검증하는 프로세스입니다. 데이터베이스 오브젝트 액세스는 사용자의 인증에 따라 다릅니다. 올바른 값(sqlenv.h)은 다음과 같습니다.

SQL_AUTHENTICATION_SERVER
목표 데이터베이스를 포함하는 데이터베이스 파티션 서버에서 인증이 발생하도록 지정합니다.

SQL_AUTHENTICATION_CLIENT
응용프로그램이 호출되는 데이터베이스 파티션 서버에서 인증이 발생하도록 지정합니다.

SQL_AUTHENTICATION_KERBEROS
인증에서 Kerberos 보안 메커니즘을 사용하도록 지정합니다.

SQL_AUTHENTICATION_NOT_SPECIFIED
인증이 지정되지 않음

SQL_AUTHENTICATION_SVR_ENCRYPT
목표 데이터베이스를 포함하는 데이터베이스 파티션 서버에서 인증이 수행되고 인증 암호가 암호화되도록 지정합니다.

SQL_AUTHENTICATION_DATAENC
목표 데이터베이스를 포함하는 데이터베이스 파티션 서버에서 인증이 발생하도록 지정하며 해당 연결은 데이터 암호화를 사용해야 합니다.

SQL_AUTHENTICATION_GSSPLUGIN
인증에서 외부 GSS API 기반 플러그인 보안 메커니즘을 사용하도록 지정합니다.

SQL_AUTHENTICATION_SVRENC_AESO
목표 데이터베이스를 포함하는 데이터베이스 파티션 서버에서 인증이 수행되도록 지정하며 AES(Advanced Encryption Standard) 암호화 알고리즘으로 인증 사용자 ID 및 암호가 암호화되도록 지정합니다.

DB2 버전 1 서버에 있는 데이터베이스를 카탈로깅하는 경우를 제외하고는 이 매개변수를 SQL_AUTHENTICATION_NOT_SPECIFIED로 설정할 수 있습니다.

데이터베이스 카탈로그에 인증 유형을 지정하면 연결 시에 성능이 향상됩니다.

pPrincipal
입력. 데이터베이스가 있는 DB2 서버의 핵심부 이름이 포함된 문자열. 이 값은 **authentication**이 SQL_AUTHENTICATION_KERBEROS인 경우에만 지정해야 합니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgcadb API 특정 매개변수

PrinLen

입력. 핵심부 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 핵심부가 없는 경우에는 영(0)으로 설정하십시오. 이 값은 **authentication**이 SQL_AUTHENTICATION_KERBEROS로 지정된 경우에만 0이 아니어야 합니다.

CommentLen

입력. 주석 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 주석을 입력하지 않은 경우에는 0으로 설정하십시오.

PathLen

입력. 로컬 데이터베이스 디렉토리의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 경로를 입력하지 않은 경우에는 0으로 설정하십시오.

NodeNameLen

입력. 노드 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 노드 이름을 입력하지 않은 경우에는 0으로 설정하십시오.

DbAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

DbNameLen

입력. 데이터베이스 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

pPrinName

입력. 데이터베이스가 있는 DB2 서버의 핵심부 이름이 포함된 문자열. 이 값은 **authentication**이 SQL_AUTHENTICATION_KERBEROS인 경우에만 지정해야 합니다.

사용 시 참고사항

CATALOG DATABASE를 사용하여 로컬이나 리모트 노드에 있는 데이터베이스를 카탈로그하고 이전에 카탈로그 해제된 데이터베이스를 재카탈로그하거나 데이터베이스 하나에 대해 여러 개의 별명을 유지하십시오(데이터베이스 위치에 상관 없이)

DB2는 데이터베이스가 작성될 때 자동으로 데이터베이스를 카탈로그합니다. 로컬 데이터베이스 디렉토리의 데이터베이스 항목 및 시스템 데이터베이스 디렉토리의 다른 항목

을 카탈로그합니다. 리모트 클라이언트(또는 동일한 머신의 다른 인스턴스에서 실행되는 클라이언트)에서 데이터베이스가 작성된 경우 클라이언트 인스턴스에서도 항목이 시스템 데이터베이스 디렉토리에 작성됩니다.

현재 인스턴스에서 작성된 데이터베이스(**DB2INSTANCE** 환경 변수 값으로 정의)는 간접으로 카탈로그됩니다. 다른 인스턴스에서 작성된 데이터베이스는 리모트로 카탈로그됩니다(실제로 동일한 머신에 있는 경우에도)

CATALOG DATABASE는 시스템 데이터베이스 디렉토리가 없는 경우 자동으로 이를 작성합니다. 시스템 데이터베이스 디렉토리는 사용 중인 데이터베이스 관리 프로그램 인스턴스가 포함된 경로에 저장됩니다. 시스템 데이터베이스 디렉토리는 데이터베이스 밖에서 유지보수됩니다. 디렉토리의 각 항목에는 다음이 포함됩니다.

- 별명
- 인증 유형
- 주석
- 데이터베이스
- 항목 유형
- 로컬 데이터베이스 디렉토리(로컬 데이터베이스 카탈로깅 시)
- 노드 이름(리모트 데이터베이스 카탈로깅 시)
- 릴리스 정보

SQL_INDIRECT로 설정된 **type** 매개변수를 사용하여 데이터베이스가 카탈로그된 경우 제공된 **authentication** 매개변수 값이 무시되고 디렉토리에서의 인증은 SQL_AUTHENTICATION_NOT_SPECIFIED로 설정됩니다.

디렉토리 캐싱이 사용 가능한 경우, 데이터베이스, 노드 및 DCS 디렉토리 파일이 메모리에 캐시됩니다. 응용프로그램의 디렉토리 캐시가 첫 번째 디렉토리 검색 시에 작성됩니다. 캐시는 응용프로그램이 임의의 디렉토리 파일을 수정하는 경우에만 새로 고쳐지기 때문에 다른 응용프로그램이 작성한 디렉토리 변경은 응용프로그램을 재시작해야 적용됩니다. DB2 공유 캐시를 새로 고치려면(서버에만 해당) 데이터베이스 관리 프로그램를 중지(db2stop)한 다음 재시작(db2start)해야 합니다. 다른 응용프로그램에 대해 디렉토리 캐시를 새로 고치려면 응용프로그램을 중지한 다음 재시작하십시오.

REXX API 구문

```
CATALOG DATABASE dbname [AS alias] [ON path|AT NODE nodename]
[AUTHENTICATION authentication] [WITH "comment"]
CATALOG GLOBAL DATABASE db_global_name AS alias
USING DIRECTORY {DCE} [WITH "comment"]
```

REXX API 매개변수

dbname

카탈로그되는 데이터베이스 이름

alias 데이터베이스의 대체 이름. 별명을 지정하지 않으면 데이터베이스 이름이 별명으로 사용됩니다.

path 카탈로그 중인 데이터베이스가 있는 경로

nodename

카탈로그 중인 데이터베이스가 있는 리모트 워크스테이션 이름.

주: path와 **nodename** 모두 지정하지 않으면 데이터베이스는 로컬로 간주되며 데이터베이스 위치는 데이터베이스 관리 프로그램 구성 매개변수 **dftdbpath**에 지정된 위치로 간주됩니다.

authentication

인증이 수행되어야 하는 위치. 가능한 값은 다음과 같습니다.

SERVER

인증은 목표 데이터베이스를 포함하는 데이터베이스 파티션 서버에서 수행됩니다. 이는 디폴트값입니다.

CLIENT

인증은 응용프로그램이 호출된 데이터베이스 파티션 서버에서 수행됩니다.

KERBEROS

인증에서 Kerberos 보안 메커니즘을 사용하도록 지정합니다.

NOT_SPECIFIED

인증이 지정되지 않음

SVR_ENCRYPT

목표 데이터베이스를 포함하는 데이터베이스 파티션 서버에서 인증이 수행되고 인증 사용자 ID 및 암호가 암호화되도록 지정합니다.

DATAENC

목표 데이터베이스를 포함하는 데이터베이스 파티션 서버에서 인증이 발생하도록 지정하며 해당 연결은 데이터 암호화를 사용해야 합니다.

GSSPLUGIN

인증에서 외부 GSS API 기반 플러그인 보안 메커니즘을 사용하도록 지정합니다.

SQL_AUTHENTICATION_SVRENC_AESO

목표 데이터베이스를 포함하는 데이터베이스 파티션 서버에서 인증이 수행되도록 지정하며 AES 암호화 알고리즘으로 인증 사용자 ID 및 암호가 암호화되도록 지정합니다.

comment

데이터베이스 또는 시스템 데이터베이스 디렉토리의 데이터베이스 항목에 대해 설명합니다. 주석 문자열의 최대 길이는 30자입니다. 캐리지 리턴 또는 줄 바꾸기 문자는 허용되지 않습니다. 주석 텍스트는 큰따옴표로 묶어야 합니다.

db_global_name

DCE 이름 스페이스에서 데이터베이스를 고유하게 식별하는 완전한 이름

DCE 사용 중인 전역 디렉토리 서비스.

REXX 예

```
call SQLDBS 'CATALOG GLOBAL DATABASE /.../cell1/subsys/database/DB3
AS dbtest USING DIRECTORY DCE WITH "Sample Database"'
```

제 107 장 sqlecran - 데이터베이스 파티션 서버에 데이터베이스 작성

API를 호출하는 데이터베이스 파티션 서버에서만 데이터베이스를 작성합니다. 이 API는 일반적인 용도로는 사용되지 않습니다. 예를 들어 데이터베이스 파티션 서버의 데이터베이스 파티션이 손상되어 재작성해야 하는 경우에 db2Restore와 같이 사용해야 합니다. 이 API를 잘못 사용하면 시스템에서 불일치가 발생하기 때문에 주의해서 사용해야 합니다.

주: 이 API를 사용하여 손상되어 삭제된 데이터베이스 파티션을 재작성하는 경우 이 데이터베이스 파티션 서버의 데이터베이스는 리스토어 보류 상태가 됩니다. 데이터베이스 파티션을 재작성한 후에 데이터베이스는 이 데이터베이스 파티션 서버에서 바로 리스토어해야 합니다.

범위

이 API는 호출된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

인스턴스. 다른 데이터베이스 파티션 서버에서 데이터베이스를 작성하려면 우선 해당 데이터베이스 파티션 서버에 접속해야 합니다. 데이터베이스 연결은 처리 중에 이 API에서 임시로 작성됩니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlecran (
        char * pDbName,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcran (
        unsigned short reservedLen,
```

```
unsigned short dbNameLen,
struct sqlca * pSqlca,
void * pReserved,
char * pDbName);
```

sqlecran API 매개변수

pDbName

입력. 작성할 데이터베이스 이름이 포함된 문자열. NULL이면 안됩니다.

pReserved

입력. 영(0)을 가리키거나 널(NULL)로 설정되는 여분의 포인터. 나중에 사용하기 위해 예약됨

pSqlca

출력. sqlca 구조의 포인터

sqlgcran API 특정 매개변수

reservedLen

입력. pReserved 길이로 예약되었습니다.

dbNameLen

입력. 데이터베이스 이름 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

데이터베이스가 제대로 작성되면 리스토어 보류 상태가 됩니다. 데이터베이스는 사용하기 전에 이 데이터베이스 파티션 서버에서 리스토어해야 합니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 108 장 sqlecrea - 데이터베이스 작성

선택적인 사용자 정의 조합 시퀀스가 포함된 새 데이터베이스를 초기화하고 세 개의 초기 테이블 스페이스를 작성하며 시스템 테이블을 작성하고 복구 로그를 할당합니다.

범위

파티션된 데이터베이스 환경에서 이 API는 db2nodes.cfg 파일에 표시된 모든 데이터베이스 파티션 서버에 영향을 줍니다.

이 API가 호출된 데이터베이스 파티션 서버가 새 데이터베이스의 카탈로그 파티션이 됩니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

인스턴스. 다른(리모트) 노드에 데이터베이스를 작성하려면 우선 해당 노드에 접속해야 합니다. 데이터베이스 연결은 처리 중에 이 API에서 임시로 작성됩니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlecrea (
        char * pDbName,
        char * pLocalDbAlias,
        char * pPath,
        struct sqledbdesc * pDbDescriptor,
        SQLEDBTERRITORYINFO * pTerritoryInfo,
        char Reserved2,
        void * pDbDescriptorExt,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcrea (
        unsigned short PathLen,
        unsigned short LocalDbAliasLen,
        unsigned short DbNameLen,
        struct sqlca * pSqlca,
        void * pReserved1,
        unsigned short Reserved2,
```

```
SQLEDBTERRITORYINFO * pTerritoryInfo,
struct sqledbdesc * pDbDescriptor,
char * pPath,
char * pLocalDbAlias,
char * pDbName);
```

sqlecrea API 매개변수

pDbName

입력. 데이터베이스 이름이 포함된 문자열. 시스템 데이터베이스 디렉토리에서 카탈로그되는 데이터베이스 이름입니다. 데이터베이스가 서버의 시스템 데이터베이스 디렉토리에서 제대로 작성되면 데이터베이스 이름과 동일한 데이터베이스 별명으로 시스템 데이터베이스 디렉토리에서 자동으로 카탈로그됩니다. NULL이면 안됩니다.

pLocalDbAlias

입력. 클라이언트 시스템 데이터베이스 디렉토리에 배치되는 별명이 포함된 문자열. NULL일 수 있습니다. 로컬 별명을 지정하지 않으면 데이터베이스 이름이 디폴트입니다.

pPath 입력. Linux 및 UNIX 시스템에서 데이터베이스를 작성하는 경로를 지정합니다. 경로를 지정하지 않으면 데이터베이스 관리 프로그램 구성 파일(dftdbpath 매개변수)에 지정된 디폴트 데이터베이스 경로에 데이터베이스가 작성됩니다. Windows 운영 체제에서 데이터베이스가 작성되는 드라이브 이름을 지정합니다. NULL일 수 있습니다.

주: 파티션된 데이터베이스 환경의 경우 NFS 마운트 디렉토리에 데이터베이스를 작성하면 안됩니다. 경로를 지정하지 않는 경우에는 dftdbpath 데이터베이스 관리 프로그램 구성 매개변수를 NFS 마운트 경로로 설정하지 마십시오(예: UNIX 기반 시스템에서 인스턴스 소유자의 $HOME 디렉토리를 지정하지 않아야 함). 파티션된 데이터베이스 환경에서 이 API에 지정된 경로는 상대 경로일 수 없습니다.

pDbDescriptor

입력. 데이터베이스 작성 시에 사용되는 데이터베이스 설명 블록의 포인터. 데이터베이스의 구성 파일에 영구적으로 저장되는 값을 제공하는 데 데이터베이스 설명 블록을 사용할 수 있습니다.

입력한 값은 조합 시퀀스, 데이터베이스 주석 또는 테이블 스페이스 정의입니다. 값을 하나도 제공하지 않으려는 경우 입력한 값은 NULL일 수 있습니다. 이 매개변수를 사용하여 입력할 수 있는 값에 대한 정보는 SQLEDBDESC 데이터 구조 주제를 참조하십시오.

pTerritoryInfo

입력. 데이터베이스의 로케일 및 코드 세트를 포함하는 sqledbterritoryinfo 구조의 포인터. NULL일 수 있습니다. 데이터베이스의 디폴트 코드 세트는

UTF-8(유니코드)입니다. 데이터베이스에 특정 코드 세트 및 지역이 필요한 경우 sqledbterritoryinfo 구조를 사용하여 원하는 코드 세트와 지역을 지정해야 합니다. 이 필드가 NULL인 경우에는 다음 중 하나가 데이터베이스의 조합 값으로 허용됩니다(sqlcode 1083): NULL, SQL_CS_SYSTEM, SQL_CS_IDENTITY_16BIT, SQL_CS_UCA400_NO, SQL_CS_UCA400_LTH, SQL_CS_UCA400_LSK 또는 SQL_CS_UNICODE.

Reserved2

입력. 나중에 사용하기 위해 예약됨

pDbDescriptorExt

입력. 이 매개변수는 데이터베이스 작성 시에 사용되는 확장된 데이터베이스 설명 블록(sqledbdescext)입니다. 확장된 데이터베이스 설명 블록은 데이터베이스의 자동 스토리지를 제어하고 데이터베이스의 디폴트 페이지 *크기*를 선택하며 소개된 새 테이블 스페이스 속성 값을 지정합니다. 널(NULL) 또는 영(0)으로 설정된 경우 데이터베이스에 대해 디폴트 페이지 *크기*인 4096바이트가 선택되고 자동 스토리지가 사용 가능해집니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgcrea API 특정 매개변수

PathLen

입력. 경로 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 경로가 없는 경우에는 영(0)으로 설정하십시오.

LocalDbALiasLen

입력. 로컬 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 로컬 별명이 없는 경우 영(0)으로 설정하십시오.

DbNameLen

입력. 데이터베이스 이름 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

데이터베이스 작성:

- 지정한 서브디렉토리에 데이터베이스를 작성합니다. 파티션된 데이터베이스 환경에서 db2nodes.cfg에 표시된 모든 데이터베이스 파티션 서버에서 데이터베이스를 작성하고 각 데이터베이스 파티션 서버의 지정한 서브디렉토리에 $DB2INSTANCE/NODExxxx 디렉토리를 작성합니다(여기서, xxxx는 로컬 데이터베이스 파티션 서버 번호). 단일 파티션 환경에서는 지정한 서브디렉토리에 $DB2INSTANCE/NODE0000 디렉토리를 작성합니다.
- 시스템 카탈로그 테이블 및 복구 로그를 작성합니다.

- 다음 데이터베이스 디렉토리에 데이터베이스를 카탈로그합니다.
  - pPath로 표시한 경로에서 서버의 로컬 데이터베이스 디렉토리 또는, 경로를 지정하지 않은 경우에는 데이터베이스 관리 프로그램 시스템 구성 파일에 정의된 디폴트 데이터베이스 경로. 로컬 데이터베이스 디렉토리는 데이터베이스가 포함된 각 파일 시스템에 상주합니다.
  - 접속된 인스턴스에 대해 서버의 시스템 데이터베이스 디렉토리. 결과로 작성되는 디렉토리 항목에는 데이터베이스 이름 및 데이터베이스 별명이 포함됩니다.

    리모트 클라이언트에서 API가 호출된 경우 클라이언트의 시스템 데이터베이스 디렉토리도 데이터베이스 이름 및 별명으로 갱신됩니다.
- 둘 다 없는 경우에는 시스템 또는 로컬 데이터베이스 디렉토리를 작성합니다. 지정한 경우 주석과 코드 세트 값이 두 디렉토리에 배치됩니다.
- 지정한 코드 세트, 지역 및 조합 시퀀스를 저장합니다. 조합 시퀀스가 고유 가중치로 구성된 경우 또는 조합 시퀀스가 식별 시퀀스인 경우에는 플래그가 데이터베이스 구성 파일에 설정됩니다.
- 소유자가 SYSIBM인 SYSCAT, SYSFUN, SYSIBM 및 SYSSTAT schemata를 작성합니다. 이 API가 호출된 데이터베이스 파티션 서버가 새 데이터베이스의 카탈로그 파티션이 됩니다. IBMDEFAULTGROUP 및 IBMCATGROUP의 두 데이터베이스 파티션 그룹이 자동으로 작성됩니다.
- 이전에 정의된 데이터베이스 관리 프로그램 바인드 파일이 데이터베이스에 바인드됩니다(해당 파일은 db2ubind.lst에 표시). 이 파일 중 하나 이상이 제대로 바인드되지 않으면 sqlecrea가 SQLCA에 경고를 리턴하고 실패한 바인드에 대한 정보를 제공합니다. 바인드가 실패한 경우 사용자는 정정 조치를 수행할 수 있으며 수동으로 실패한 파일을 바인드할 수 있습니다. 어느 경우에든 데이터베이스가 작성됩니다. NULLID 스키마는 RESTRICTIVE 옵션을 선택하지 않은 경우 PUBLIC에 부여된 CREATEIN 특권으로 바인드를 수행할 때 내재적으로 작성됩니다.
- SYSCATSPACE, TEMPSPACE1 및 USERSPACE1 테이블 스페이스가 작성됩니다. SYSCATSPACE 테이블 스페이스는 카탈로그 파티션에서만 작성됩니다. 모든 데이터베이스 파티션에는 동일한 테이블 스페이스 정의가 있습니다.
- 다음을 권한 부여합니다.
  - 데이터베이스 작성자에게 DBADM, CONNECT, CREATETAB, BINDADD, CREATE_NOT_FENCED, IMPLICIT_SCHEMA 및 LOAD 권한 부여
  - PUBLIC에 CONNECT, CREATETAB, BINDADD 및 IMPLICIT_SCHEMA 권한 부여
  - PUBLIC에 USERSPACE1 테이블 스페이스의 USE 특권 부여
  - PUBLIC에 각 시스템 카탈로그의 SELECT 특권 부여
  - 성공한 각 바운드 유틸리티에 대해 PUBLIC에 BIND 및 EXECUTE 특권 부여

– PUBLIC에 SYSFUN 스키마의 모든 함수에 EXECUTE WITH GRANT 특권 부여
– PUBLIC에 SYSIBM 스키마의 모든 프로시저에 EXECUTE 특권 부여

주: RESTRICTIVE 옵션이 있는 경우 RESTRICT_ACCESS 데이터베이스 구성 매개변수가 YES로 설정되고, 특권 또는 권한이 자동으로 PUBLIC에 부여되지 않습니다. 자세한 정보는 CREATE DATABASE 명령의 RESTRICTIVE 옵션을 참조하십시오.

dbadm 권한을 사용하여 다른 사용자나 PUBLIC에 해당 특권 부여 및 권한 취소할 수도 있습니다. 데이터베이스에 대해 sysadm 또는 dbadm 권한이 있는 다른 관리자가 이 특권을 권한 취소하는 경우 데이터베이스 작성자는 이에 상관 없이 해당 권한을 유지합니다.

파티션된 데이터베이스 환경에서 데이터베이스 관리 프로그램은 모든 데이터베이스 파티션 서버에서 지정한 경로나 디폴트 경로에 $DB2INSTANCE/NODExxxx 서브디렉토리를 작성합니다. xxxx는 db2nodes.cfg 파일에 정의된 노드 번호입니다(즉, 노드 0은 NODE0000). SQL00001 - SQLnnnnn 서브디렉토리는 이 경로에 상주합니다. 그러면 다른 데이터베이스 파티션 서버에 관련된 데이터베이스 오브젝트는 다른 디렉토리에 저장됩니다(지정한 경로 또는 디폴트 경로의 서브디렉토리 $DB2INSTANCE를 모든 데이터베이스 파티션 서버에서 공유하는 경우도).

Windows 및 AIX 운영 체제에서 코드 세트 이름 길이는 최대 9자로 제한됩니다. 예를 들어 코드 세트 이름은 ISO8859-15가 아니라 ISO885915로 지정하십시오.

sqlecrea API는 데이터베이스 디스크립터 블록(SQLEDBDESC)이라는 데이터 구조를 승인합니다. 이 구조 내에 자체 조합 시퀀스를 정의할 수 있습니다.

주: 단일 바이트 데이터베이스에 대해서는 자체 조합 시퀀스만 정의할 수 있습니다.

데이터베이스에 조합 시퀀스를 지정하려면 다음을 수행하십시오.

- 원하는 SQLEDBDESC 구조를 전달하거나
- NULL 포인터를 전달하십시오. 운영 체제의 조합 시퀀스(현재 로케일 코드 및 코드 페이지를 바탕으로)가 사용됩니다. 이는 SQL_CS_SYSTEM(0)과 동일한 SQLDBCSS를 지정하는 것과 동일합니다.

응용프로그램이 이미 데이터베이스에 연결되어 있으면 CREATE DATABASE 명령 실행이 실패합니다.

데이터베이스 설명 블록 구조가 제대로 설정되어 있지 않으면 오류 메시지가 리턴됩니다.

데이터베이스 설명 블록의 가장 특별한 값은 기호 값인 SQLE_DBDESC_2로 설정해야 합니다(sqlenv에 정의). 호스트 언어 내장 파일에서 다음 샘플 사용자 정의 조합 시퀀스가 사용 가능합니다.

sqle819a
데이터베이스의 코드 페이지가 819(ISO Latin/1)인 경우 이 시퀀스를 사용하면 호스트 CCSID 500(EBCDIC 국제)에 따라 정렬이 수행됩니다.

sqle819b
데이터베이스의 코드 페이지가 819(ISO Latin/1)인 경우 이 시퀀스를 사용하면 호스트 CCSID 037(EBCDIC 미국 영어)에 따라 정렬이 수행됩니다.

sqle850a
데이터베이스의 코드 페이지가 850(ASCII Latin/1)인 경우 이 시퀀스를 사용하면 호스트 CCSID 500(EBCDIC 국제)에 따라 정렬이 수행됩니다.

sqle850b
데이터베이스의 코드 페이지가 850(ASCII Latin/1)인 경우 이 시퀀스를 사용하면 호스트 CCSID 037(EBCDIC 미국 영어)에 따라 정렬이 수행됩니다.

sqle932a
데이터베이스의 코드 페이지가 932(ASCII 일본어)인 경우 이 시퀀스를 사용하면 호스트 CCSID 5035(EBCDIC 일본어)에 따라 정렬이 수행됩니다.

sqle932b
데이터베이스의 코드 페이지가 932(ASCII 일본어)인 경우 이 시퀀스를 사용하면 호스트 CCSID 5026(EBCDIC 일본어)에 따라 정렬이 수행됩니다.

데이터베이스 작성 중에 지정된 조합 시퀀스는 이후에 변경할 수 없습니다. 이는 문자열이 비교되는 방법을 판별합니다. 이는 인덱스 구조 및 쿼리 결과에도 영향을 줍니다. 유니코드 데이터베이스에서 카탈로그 테이블 및 뷰는 데이터베이스 작성 호출에 지정된 조합에 상관 없이 항상 IDENTITY 조합으로 작성됩니다. 유니코드가 아닌 데이터베이스에서 카탈로그 테이블과 뷰는 데이터베이스 조합으로 작성됩니다.

sqlecadb를 사용하여 새 데이터베이스에 대해 다른 별명 이름을 정의하십시오.

특별히 명시하지 않으면 데이터베이스 작성 프로세스 중에 디폴트로 구성 어드바이저가 호출됩니다.

REXX API 구문

```
CREATE DATABASE dbname [ON path] [ALIAS dbalias]
[USING CODESET codeset TERRITORY territory]
[COLLATE USING {SYSTEM | IDENTITY | USER :udcs}]
[NUMSEGS numsegs] [DFT_EXTENT_SZ dft_extentsize]
[CATALOG TABLESPACE <tablespace_definition>]
[USER TABLESPACE <tablespace_definition>]
[TEMPORARY TABLESPACE <tablespace_definition>]
[WITH comment]
```

```
Where <tablespace_definition> stands for:
MANAGED BY {
SYSTEM USING :SMS_string |
DATABASE USING :DMS_string }
[ EXTENTSIZE number_of_pages ]
[ PREFETCHSIZE number_of_pages ]
[ OVERHEAD number_of_milliseconds ]
[ TRANSFERRATE number_of_milliseconds ]
```

REXX API 매개변수

dbname

데이터베이스 이름

dbalias

데이터베이스 별명

경로 데이터베이스를 작성하는 경로. 경로를 지정하지 않으면 데이터베이스 관리 프로그램 구성 파일(dftdbpath 구성 매개변수)에 지정된 디폴트 데이터베이스 경로에 데이터베이스가 작성됩니다.

주: 파티션된 데이터베이스 환경의 경우 NFS 마운트 디렉토리에 데이터베이스를 작성하면 안됩니다. 경로를 지정하지 않는 경우에는 dftdbpath 데이터베이스 관리 프로그램 구성 매개변수를 NFS 마운트 경로로 설정하지 마십시오(예: UNIX 기반 시스템에서 인스턴스 소유자의 $HOME 디렉토리를 지정하지 않아야 함). 파티션된 데이터베이스 환경에서 이 API에 지정된 경로는 상대 경로일 수 없습니다.

codeset

데이터베이스에 입력하는 데이터에 사용되는 코드 세트.

territory

데이터베이스에 입력하는 데이터에 사용되는 지역 코드(로케일)

SYSTEM

유니코드가 아닌 데이터베이스의 경우 이 옵션은 데이터베이스 지역을 기반으로 하는 조합 시퀀스가 포함된 디폴트 옵션입니다. 유니코드 데이터베이스의 경우 이 옵션은 IDENTITY 옵션과 동등합니다.

IDENTITY

문자열이 바이트별로 비교되는 조합 시퀀스를 식별합니다. 유니코드 데이터베이스의 경우 디폴트입니다.

USER udcs

조합 시퀀스는 조합 시퀀스를 정의하는 256바이트 문자열이 포함된 호스트 변수에서 응용프로그램을 호출하여 지정합니다.

numsegs

작성되어 모든 디폴트 SMS 테이블 스페이스에 대한 데이터베이스 테이블 파일을 저장하는 데 사용되는 디렉토리 수(테이블 스페이스 컨테이너)

dft_extentsize

데이터베이스에서 테이블 스페이스의 디폴트 Extent 크기를 지정합니다.

SMS_string

테이블 스페이스에 속하고 테이블 스페이스 데이터가 저장되는 하나 이상의 컨테이너를 식별하는 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다. 각 디렉토리 이름 길이는 254바이트를 초과할 수 없습니다.

XXX.0

지정한 디렉토리 수

XXX.1

SMS 테이블 스페이스의 첫 번째 디렉토리 이름

XXX.2

SMS 테이블 스페이스의 두 번째 디렉토리 이름

XXX.3

기타

DMS_string

테이블 스페이스에 속하고 테이블 스페이스 데이터가 저장되는 하나 이상의 컨테이너, 컨테이너 크기(4KB 페이지 수로 지정) 및 유형(파일 또는 디바이스)을 식별하는 복합 REXX 호스트 변수. 지정된 디바이스(파일이 아닌)는 이미 있어야 합니다. 다음에서 XXX는 호스트 변수 이름을 나타냅니다. 각 컨테이너 이름 길이는 254바이트를 초과할 수 없습니다.

XXX.0

REXX 호스트 변수의 문자열 수(첫 번째 레벨 요소 수)

XXX.1.1

첫 번째 컨테이너 유형(파일 또는 디바이스)

XXX.1.2

첫 번째 파일 이름 또는 디바이스 이름

XXX.1.3

첫 번째 컨테이너 크기(페이지 단위)

XXX.2.1

두 번째 컨테이너 유형(파일 또는 디바이스)

XXX.2.2

두 번째 파일 이름 또는 디바이스 이름

XXX.2.3

두 번째 컨테이너 *크기*(페이지 단위)

XXX.3.1

기타

EXTENTSIZE number_of_pages

다음 컨테이너로 건너뛰기 전에 컨테이너에 기록되는 4KB 페이지 수.

PREFETCHSIZE number_of_pages

데이터 프리페칭이 수행될 때 테이블에서 읽는 4KB 페이지의 수.

OVERHEAD number_of_milliseconds

입출력 제어기 오버헤드, 디스크 탐색 및 대기 시간(밀리초)을 지정하는 수.

TRANSFERRATE number_of_milliseconds

한 개의 4KB 페이지를 메모리로 읽는 시간(밀리초)을 지정하는 수.

comment

데이터베이스 또는 시스템 디렉토리의 데이터베이스 항목 설명. 주석에는 캐리지 리턴이나 줄 바꾸기 문자를 사용하지 마십시오. 주석은 큰따옴표로 묶어야 합니다. 최대 *크기*는 30자입니다.

제 109 장 sqlectnd - 노드 디렉토리의 항목 카탈로그

해당 인스턴스 액세스에 사용한 통신 프로토콜을 기반으로 DB2 서버 인스턴스 위치에 대한 정보를 노드 디렉토리에 저장합니다. 정보는 데이터베이스 연결 또는 응용프로그램과 서버 인스턴스 사이의 접속에 필요합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlectnd (
        struct sqle_node_struct * pNodeInfo,
        void * pProtocolInfo,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgctnd (
        struct sqlca * pSqlca,
        struct sqle_node_struct * pNodeInfo,
        void * pProtocolInfo);
```

sqlectnd API 매개변수

pNodeInfo

입력. 노드 디렉토리 구조의 포인터

pProtocolInfo

입력. 프로토콜 구조의 포인터

- SQLE-NODE-LOCAL
- SQLE-NODE-NPIPE
- SQLE-NODE-TCPIP

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

DB2는 노드 디렉토리가 없는 경우 이 API에 대한 첫 번째 호출 시에 노드 디렉토리를 작성합니다. Windows 운영 체제에서 노드 디렉토리는 사용 중인 인스턴스 디렉토리에 저장됩니다. UNIX 기반 시스템에서는 DB2 설치 디렉토리(예: sqllib)에 저장됩니다.

디렉토리 캐싱이 사용 가능한 경우, 데이터베이스, 노드 및 DCS 디렉토리 파일이 메모리에 캐시됩니다. 응용프로그램의 디렉토리 캐시가 첫 번째 디렉토리 검색 시에 작성됩니다. 캐시는 응용프로그램이 임의의 디렉토리 파일을 수정하는 경우에만 새로 고쳐지기 때문에 다른 응용프로그램이 작성한 디렉토리 변경은 응용프로그램을 재시작해야 적용됩니다. DB2 공유 캐시를 새로 고치려면(서버 전용) 데이터베이스 관리 프로그램을 중지(db2stop 명령)한 다음 재시작(db2start 명령)하십시오. 다른 응용프로그램에 대해 디렉토리 캐시를 새로 고치려면 응용프로그램을 중지한 다음 재시작하십시오.

REXX API 구문, 옵션 1

```
CATALOG LOCAL NODE nodename INSTANCE instance_name [WITH comment]
```

REXX API 매개변수, 옵션 1

nodename

카탈로그되는 노드 별명

instance_name

카탈로그되는 인스턴스 이름

comment

이 노드 디렉토리 항목에 연관된 선택적 설명. 주석에는 CR/LF 문자를 포함하지 마십시오. 최대 길이는 30자입니다. 주석 텍스트는 큰따옴표로 묶어야 합니다.

REXX API 구문, 옵션 2

```
CATALOG NPIPE NODE nodename REMOTE computer_name INSTANCE instance_name
```

REXX API 매개변수, 옵션 2

nodename

카탈로그되는 노드 별명

computer_name

목표 데이터베이스가 있는 노드의 컴퓨터 이름

instance_name

카탈로그되는 인스턴스 이름

REXX API 구문, 옵션 3

```
CATALOG TCPIP NODE nodename REMOTE hostname SERVER servicename
[WITH comment]
```

REXX API 매개변수, 옵션 3

nodename

카탈로그되는 노드 별명

hostname

목표 데이터베이스가 있는 노드의 호스트 이름 또는 IPv4 주소나 IPv6 주소

servicename

리모트 노드의 데이터베이스 관리 프로그램 인스턴스의 서비스 이름 또는 해당 서비스 이름에 연관된 포트 번호

comment

이 노드 디렉토리 항목에 연관된 선택적 설명. 주석에는 CR/LF 문자를 포함하지 마십시오. 최대 길이는 30자입니다. 주석 텍스트는 큰따옴표로 묶어야 합니다.

REXX API 구문, 옵션 4

```
CATALOG TCPIP4 NODE nodename REMOTE hostname SERVER servicename
[WITH comment]
```

REXX API 매개변수, 옵션 4

nodename

카탈로그되는 노드 별명

hostname

목표 데이터베이스가 있는 노드의 호스트 이름 또는 IPv4 주소나 IPv6 주소

servicename

리모트 노드의 데이터베이스 관리 프로그램 인스턴스의 서비스 이름 또는 해당 서비스 이름에 연관된 포트 번호

comment

이 노드 디렉토리 항목에 연관된 선택적 설명. 주석에는 CR/LF 문자를 포함하지 마십시오. 최대 길이는 30자입니다. 주석 텍스트는 큰따옴표로 묶어야 합니다.

REXX API 구문, 옵션 5

```
CATALOG TCPIP6 NODE nodename REMOTE hostname SERVER servicename
[WITH comment]
```

REXX API 매개변수, 옵션 5

nodename

카탈로그되는 노드 별명

hostname

목표 데이터베이스가 있는 노드의 호스트 이름 또는 IPv4 주소나 IPv6 주소

servicename

리모트 노드의 데이터베이스 관리 프로그램 인스턴스의 서비스 이름 또는 해당 서비스 이름에 연관된 포트 번호

comment

이 노드 디렉토리 항목에 연관된 선택적 설명. 주석에는 CR/LF 문자를 포함하지 마십시오. 최대 길이는 30자입니다. 주석 텍스트는 큰따옴표로 묶어야 합니다.

제 110 장 sqledcgd - 시스템 또는 로컬 데이터베이스 디렉토리에서 데이터베이스 주석 변경

시스템 데이터베이스 디렉토리 또는 로컬 데이터베이스 디렉토리에서 데이터베이스 주석을 변경합니다. 새 주석 텍스트는 현재 주석에 연관된 텍스트를 대체할 수 있습니다.

범위

이 API는 발행된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqledcgd (
        _SQLOLDCHAR * pDbAlias,
        _SQLOLDCHAR * pPath,
        _SQLOLDCHAR * pComment,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdcgd (
        unsigned short CommentLen,
        unsigned short PathLen,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pComment,
        _SQLOLDCHAR * pPath,
        _SQLOLDCHAR * pDbAlias);
```

sqledcgd API 매개변수

pDbAlias

입력. 데이터베이스 별명이 포함된 문자열. 이 문자열은 시스템 데이터베이스 디렉토리에서 카탈로그된 이름 또는 경로를 지정한 경우 로컬 데이터베이스 디렉토리에서 카탈로그된 이름입니다.

pPath 입력. 로컬 데이터베이스 디렉토리가 있는 경로가 포함된 문자열. 지정한 경로가 NULL 포인터인 경우 시스템 데이터베이스 디렉토리가 사용됩니다.

주석은 로컬 데이터베이스 디렉토리 또는 API가 실행되는 데이터베이스 파티션 서버의 시스템 데이터베이스 디렉토리에서만 변경됩니다. 모든 데이터베이스 파티션 서버에서 데이터베이스 주석을 변경하려면 모든 데이터베이스 파티션 서버에서 API를 실행하십시오.

pComment

입력. 데이터베이스의 선택적 설명이 포함된 문자열. 널(NULL) 문자열은 주석이 없음을 나타냅니다. 기존 데이터베이스 주석에 변경사항이 없음을 나타내기도 합니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgdcgd API 특정 매개변수

CommentLen

입력. 주석 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 주석이 없는 경우에는 영(0)으로 설정하십시오.

PathLen

입력. 경로 매개변수의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 경로가 없는 경우에는 영(0)으로 설정하십시오.

DbAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

새 주석 텍스트는 기존 텍스트를 교체합니다. 정보를 추가하려면 이전 주석 텍스트를 입력하고 그 뒤에 새 텍스트를 입력하십시오.

데이터베이스 별명에 연관된 항목의 주석만 수정됩니다. 데이터베이스 이름은 동일하지만 별명은 다른 기타 항목에는 영향을 주지 않습니다.

경로를 지정한 경우 로컬 데이터베이스 디렉토리에서 데이터베이스 별명을 카탈로그해야 합니다. 경로를 지정하지 않으면 시스템 데이터베이스 디렉토리에서 데이터베이스 별명을 카탈로그해야 합니다.

REXX API 구문

```
CHANGE DATABASE database_alias COMMENT [ON path] WITH comment
```

REXX API 매개변수

database_alias

주석을 변경하는 데이터베이스 별명

시스템 데이터베이스 디렉토리에서 주석을 변경하려면 데이터베이스 별명을 지정해야 합니다.

데이터베이스가 있는 경로를 지정한 경우(경로 매개변수 사용) 데이터베이스의 이름(별명이 아님)을 입력하십시오. 이 방법을 사용하여 로컬 데이터베이스 디렉토리에서 주석을 변경하십시오.

path 데이터베이스가 있는 경로

comment

시스템 데이터베이스 디렉토리 또는 로컬 데이터베이스 디렉토리에서 항목을 설명하십시오. 카탈로그된 데이터베이스를 설명하는 데 도움이 되는 모든 주석을 입력할 수 있습니다. 주석 문자열의 최대 길이는 30자입니다. 캐리지 리턴 또는 줄 바꾸기 문자를 사용할 수 없습니다. 주석 텍스트는 큰따옴표로 묶어야 합니다.

제 111 장 sqledpan - 데이터베이스 파티션 서버에서 데이터베이스 삭제

지정한 데이터베이스 파티션 서버에서 데이터베이스를 삭제합니다. 파티션된 데이터베이스 환경에서만 실행할 수 있습니다.

범위

이 API는 호출된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음. 호출 지속 기간 동안 인스턴스가 접속됩니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqledpan (
        char * pDbAlias,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdpan (
        unsigned short Reserved1,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        void * pReserved2,
        char * pDbAlias);
```

sqledpan API 매개변수

pDbAlias

입력. 삭제할 데이터베이스 별명이 포함된 문자열. 이 이름은 시스템 데이터베이스 디렉토리에서 실제 데이터베이스 이름을 참조하는 데 사용됩니다.

pReserved

예약됨. NULL이어야 합니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgdpan API 특정 매개변수

Reserved1

나중에 사용하기 위해 예약됨

DbAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

pReserved2

영(0)을 가리키거나 널(NULL)로 설정되는 여분의 포인터. 나중에 사용하기 위해 예약됨

사용 시 참고사항

이 API를 잘못 사용하면 시스템에서 불일치가 발생하기 때문에 주의해서 사용해야 합니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 112 장 sqledrpd - 데이터베이스 삭제

데이터베이스 컨텐츠 및 데이터베이스의 모든 로그 파일을 삭제하고 데이터베이스 카탈로그를 해제하며 데이터베이스 서브디렉토리를 삭제합니다.

범위

디폴트로 이 API는 db2nodes.cfg 파일에 나열된 모든 데이터베이스 파티션 서버에 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

인스턴스. 리모트 데이터베이스를 삭제하기 전에 ATTACH를 호출할 필요는 없습니다. 데이터베이스가 리모트로 카탈로그된 경우 리모트 노드의 인스턴스 접속이 호출 지속 기간에 대해 설정됩니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqledrpd (
        _SQLOLDCHAR * pDbAlias,
        _SQLOLDCHAR * pReserved2,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrpd (
        unsigned short Reserved1,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pReserved2,
        _SQLOLDCHAR * pDbAlias);
```

sqledrpd API 매개변수

pDbAlias

입력. 삭제할 데이터베이스 별명이 포함된 문자열. 이 이름은 시스템 데이터베이스 디렉토리에서 실제 데이터베이스 이름을 참조하는 데 사용됩니다.

pReserved2

영(0)을 가리키거나 널(NULL)로 설정되는 여분의 포인터. 나중에 사용하기 위해 예약됨

pSqlca

출력. sqlca 구조의 포인터

sqlgdrpd API 특정 매개변수

Reserved1

나중에 사용하기 위해 예약됨

DbAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

sqledrpd API는 모든 사용자 데이터 및 로그 파일을 삭제합니다. 리스토어 조작 후에 롤 포워드 복구에 로그 파일이 필요한 경우 이 API를 호출하기 전에 이 파일을 저장해야 합니다.

데이터베이스는 사용 중이면 안됩니다. 데이터베이스를 삭제하기 전에 데이터베이스의 모든 사용자 연결을 끊어야 합니다.

삭제하기 위해서는 데이터베이스가 시스템 데이터베이스 디렉토리에서 카탈로그되어야 합니다. 지정한 데이터베이스 별명만 시스템 데이터베이스 디렉토리에서 제거됩니다. 데이터베이스 이름이 동일한 다른 별명이 있는 경우에는 해당 항목은 유지됩니다. 삭제 중인 데이터베이스가 로컬 데이터베이스 디렉토리의 마지막 항목인 경우 로컬 데이터베이스 디렉토리가 자동으로 삭제됩니다.

이 API가 리모트 클라이언트(또는 동일한 머신의 다른 인스턴스)에서 호출되는 경우 지정한 별명이 클라이언트의 시스템 데이터베이스 디렉토리에서 제거됩니다. 해당 데이터베이스 이름은 서버의 시스템 데이터베이스 디렉토리에서 제거됩니다.

REXX API 구문

```
DROP DATABASE dbalias
```

REXX API 매개변수

dbalias

삭제할 데이터베이스 별명

제 113 장 sqledrpn - 데이터베이스 파티션 서버 삭제 가능 여부 점검

데이터베이스에서 데이터베이스 파티션 서버를 사용 중인지 확인합니다. 데이터베이스 파티션 서버가 삭제 가능한지를 나타내는 메시지가 리턴됩니다.

범위

이 API는 발행된 데이터베이스 파티션 서버에만 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqledrpn (
        unsigned short Action,
        void * pReserved,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrpn (
        unsigned short Reserved1,
        struct sqlca * pSqlca,
        void * pReserved2,
        unsigned short Action);
```

sqledrpn API 매개변수

Action

요청된 조치. 유효한 값은 SQL_DROPNODE_VERIFY입니다.

pReserved

예약됨. NULL이어야 합니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgdrpn API 특정 매개변수

Reserved1

pReserved2 길이로 예약되었습니다.

pReserved2

NULL로 설정되거나 0을 지시하는 여분의 포인터. 나중에 사용되도록 예약되어 있습니다.

사용 시 참고사항

데이터베이스 파티션 서버가 사용되고 있지 않음을 나타내는 메시지가 리턴되면 db2stop 명령을 DROP NODENUM과 같이 사용하여 db2nodes.cfg 파일에서 데이터베이스 파티션 서버 항목을 제거하십시오. 그러면 파티션된 데이터베이스 환경에서 데이터베이스 파티션 서버가 제거됩니다.

데이터베이스 파티션 서버가 사용 중임을 나타내는 메시지가 리턴되면 다음 조치를 수행해야 합니다.

1. 삭제할 데이터베이스 파티션 서버가 인스턴스의 각 데이터베이스에 대해 데이터베이스 파티션을 갖게 됩니다. 이 데이터베이스 파티션 중 하나라도 데이터를 포함한 경우 이 데이터베이스 파티션을 사용하여 데이터베이스 파티션 그룹에 다시 분배하십시오. 삭제 중이지 않은 데이터베이스 파티션 서버에 있는 데이터베이스 파티션의 데이터를 이동하기 위해 데이터베이스 파티션 그룹을 다시 분배하십시오.
2. 데이터베이스 파티션 그룹을 재분배하고 나면 데이터베이스 파티션을 사용하고 있는 모든 데이터베이스 파티션 그룹에서 데이터베이스 파티션을 삭제하십시오. 데이터베이스 파티션 그룹에서 데이터베이스 파티션을 제거하려면 sqludrdt API 또는 ALTER DATABASE PARTITION GROUP 문을 사용할 수 있습니다.
3. 데이터베이스 파티션 서버에 정의된 이벤트 모니터를 삭제하십시오.
4. sqledrpn을 재실행하여 데이터베이스 파티션 서버의 데이터베이스 파티션이 더 이상 사용되지 않도록 하십시오.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 114 장 sqledtin - 인스턴스에서 접속 해제

논리적인 인스턴스 접속을 제거하고 이 계층을 사용하는 다른 논리적 연결이 더 이상 없는 경우 물리적 통신 연결을 종료합니다.

권한 부여

없음

필수 연결

없음. 기존 인스턴스 접속을 제거합니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqledtin (
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdtin (
        struct sqlca * pSqlca);
```

sqledtin API 매개변수

pSqlca

출력. sqlca 구조의 포인터

REXX API 구문

```
DETACH
```

제 115 장 sqlefmem - sqlbtcq 및 sqlbmtsq API로 할당된 메모리 사용 가능화

호출자를 위해 DB2 API가 할당한 메모리를 사용 가능하게 합니다. sqlbtcq 및 sqlbmtsq API와 같이 사용되어야 합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlefmem (
        struct sqlca * pSqlca,
        void * pBuffer);

SQL_API_RC SQL_API_FN
  sqlgfmem (
        struct sqlca * pSqlca,
        void * pBuffer);
```

sqlefmem API 매개변수

pSqlca

출력. sqlca 구조의 포인터

pBuffer

입력. 사용 가능해지는 메모리의 포인터

제 116 장 sqlefrce - 사용자 및 응용프로그램에서 시스템 강제 중단

강제로 로컬 또는 리모트 사용자 또는 응용프로그램이 시스템을 해제하도록 하여 서버에서 유지보수를 할 수 있도록 합니다. 주의: 인터럽트할 수 없는 조작(예: 데이터베이스 리스토어)이 강제 중단되면 데이터베이스가 사용 가능해 지기 전에 조작을 제대로 다시 실행해야 합니다.

범위

이 API는 db2nodes.cfg 파일에 나열된 모든 데이터베이스 파티션 서버에 영향을 줍니다.

파티션된 데이터베이스 환경에서는 이 API는 강제 중단 중인 응용프로그램의 코디네이터 파티션에서 발행할 필요가 없습니다. 이 API는 파티션된 데이터베이스 환경의 모든 데이터베이스 파티션 서버에서 발행할 수 있습니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint

필수 연결

인스턴스. 리모트 서버에서 강제로 사용자를 해제하려면 우선 해당 서버에 연결해야 합니다. 연결되어 있지 않은 경우 이 API는 로컬로 실행됩니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlefrce (
        sqlint32 NumAgentIds,
        sqluint32 * pAgentIds,
        unsigned short ForceMode,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgfrce (
```

```
struct sqlca * pSqlca,
unsigned short ForceMode,
sqluint32 * pAgentIds,
sqlint32 NumAgentIds);
```

sqlefrce API 매개변수

NumAgentIds

입력. 종료되는 총 사용자 수를 나타내는 정수. 이 수는 에이전트 ID 배열의 요소 수와 동일해야 합니다.

이 매개변수를 SQL_ALL_USERS로 설정하는 경우(sqlenv에 정의) 데이터베이스 연결 또는 인스턴스 접속된 모든 응용프로그램이 강제로 중단됩니다. 영(0)으로 설정하면 오류가 리턴됩니다.

pAgentIds

입력. 부호없는 긴 정수 배열의 포인터. 각 항목은 해당 데이터베이스 사용자의 에이전트 ID를 설명합니다.

ForceMode

입력. sqlefrce API의 작업 모드를 지정하는 정수. 비동기 모드만 지원됩니다. 즉, API는 리턴 전에 지정한 모든 사용자가 종료될 때까지 대기하지 않습니다. API가 제대로 발행되거나 오류가 발생한 즉시 리턴합니다. 따라서 응용프로그램 호출 강제 중단이 완료되고 지정한 사용자가 종료되는 시간 간격은 짧습니다.

이 매개변수는 SQL_ASYNCH에 설정되어야 합니다(sqlenv에 정의)

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

데이터베이스 관리 프로그램은 연속된 연산이 db2start를 사용하지 않고도 처리될 수 있도록 계속 활성화되어 있습니다.

데이터베이스 무결성을 보존하기 위해 유휴 중이거나 인터럽트할 수 있는 데이터베이스 연산을 실행하는 사용자만 강제로 종료할 수 있습니다.

force 명령을 실행한 후에도 데이터베이스는 연결 요청을 계속 승인합니다. 모든 사용자를 강제로 해제하려면 추가적인 강제 실행이 필요할 수도 있습니다. 강제로 종료되는 사용자의 에이전트 ID를 수집하는 데 데이터베이스 시스템 모니터 기능이 사용됩니다.

강제 실행 모드가 SQL_ASYNCH(허용된 유일한 값)로 설정되면 API는 즉시 호출 응용프로그램으로 리턴됩니다.

강제 실행되는 에이전트 ID 배열에서 최소 유효성 확인이 수행됩니다. 포인터가 지정된 총 요소 수가 포함된 배열을 지시하는지 확인해야 합니다. NumAgentIds가 SQL_ALL_USERS로 설정된 경우에는 배열이 무시됩니다.

사용자가 강제로 종료되는 경우 작업 단위(UOW) 롤백이 수행되어 데이터베이스 일관성이 유지됩니다.

강제 종료할 수 있는 모든 사용자는 강제로 종료됩니다. 지정된 하나 이상의 에이전트 ID를 찾을 수 없는 경우에는 sqlca 구조의 sqlcode가 1230으로 설정됩니다. 예를 들어 에이전트 ID가 수집되고 sqlefrece가 호출되는 중간에 사용자가 로그오프한 경우에는 에이전트 ID를 찾을 수 없습니다. 이 API를 호출하는 사용자는 강제로 종료되지 않습니다.

에이전트 ID는 재사용되고 데이터베이스 시스템 모니터로 수집된 후에 응용프로그램을 강제 중단하는 경우에 사용됩니다. 사용자가 로그오프하고 다른 사용자가 로그온하여 이 재사용 프로세스를 통해 동일한 에이전트 ID를 획득하면 결과적으로 잘못된 사용자가 강제로 중단될 수도 있습니다.

REXX API 구문

```
FORCE APPLICATION {ALL | :agentidarray} [MODE ASYNC]
```

REXX API 매개변수

ALL 모든 응용프로그램의 연결이 끊어집니다. 여기에는 데이터베이스 연결을 포함하는 응용프로그램 및 인스턴스가 접속된 응용프로그램이 포함됩니다.

agentidarray

종료되는 에이전트 ID 목록이 포함된 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름입니다.

- **XXX.0**

  종료되는 에이전트 수

- **XXX.1**

  첫 번째 에이전트 ID

- **XXX.2**

  두 번째 에이전트 ID

- **XXX.3**

  기타

ASYNC

리턴 전에 지정한 응용프로그램이 모두 종료될 때까지 sqlefrce는 대기하지 않는 것이 현재 지원되는 유일한 모드입니다.

제 117 장 sqlegdad - 데이터베이스 연결 서비스(DCS) 디렉토리에서 데이터베이스 카탈로그

리모트 데이터베이스에 대한 정보를 데이터베이스 연결 서비스(DCS) 디렉토리에 저장합니다. 이 데이터베이스에는 DB2 Connect와 같은 응용프로그램 리퀘스터(AR)를 사용하여 액세스합니다. 시스템 데이터베이스 디렉토리의 데이터베이스 이름과 일치하는 데이터베이스 이름의 DCS 디렉토리 항목을 사용하여 SQL 요청을 데이터베이스가 있는 리모트 서버로 포워드하도록 지정한 AR을 호출합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlegdad (
        struct sql_dir_entry * pDCSDirEntry,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdad (
        struct sqlca * pSqlca,
        struct sql_dir_entry * pDCSDirEntry);
```

sqlegdad API 매개변수

pDCSDirEntry

입력. sql_dir_entry(데이터베이스 연결 서비스 디렉토리) 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

DB2 Connect 프로그램은 다음과 같은 DRDA 응용프로그램 서버(AS) 연결을 제공합니다.

- System/370™ 및 System/390® 아키텍처 호스트 컴퓨터에서 OS/390용 DB2
- System/370 및 System/390 아키텍처 호스트 컴퓨터에서에서 VM 및 VSE 데이터베이스용 DB2
- Application System/400® (AS/400®) 호스트 컴퓨터에서 OS/400 데이터베이스

데이터베이스 관리 프로그램은 데이터베이스 연결 서비스 디렉토리가 없는 경우 이를 작성합니다. 이 디렉토리는 사용 중인 데이터베이스 관리 프로그램 인스턴스를 포함하는 경로에 저장됩니다. DCS 디렉토리는 데이터베이스 밖에서 유지보수됩니다.

데이터베이스도 시스템 데이터베이스 디렉토리에서 리모트 데이터베이스로 카탈로그되어야 합니다.

주: 디렉토리 캐싱이 사용 가능한 경우, 데이터베이스, 노드 및 DCS 디렉토리 파일이 메모리에 캐시됩니다. 응용프로그램의 디렉토리 캐시가 첫 번째 디렉토리 검색 시에 작성됩니다. 캐시는 응용프로그램이 임의의 디렉토리 파일을 수정하는 경우에만 새로 고쳐지기 때문에 다른 응용프로그램이 작성한 디렉토리 변경은 응용프로그램을 재시작해야 적용됩니다. DB2 공유 캐시를 새로 고치려면(서버 전용) 데이터베이스 관리 프로그램을 중지(db2stop)한 다음 재시작(db2start)하십시오. 다른 응용프로그램에 대해 디렉토리 캐시를 새로 고치려면 응용프로그램을 중지한 다음 재시작하십시오.

REXX API 구문

```
CATALOG DCS DATABASE dbname [AS target_dbname]
[AR arname] [PARMS parms] [WITH comment]
```

REXX API 매개변수

dbname
추가하려는 디렉토리 항목의 로컬 데이터베이스 이름

target_dbname
목표 데이터베이스 이름

arname
응용프로그램 클라이언트 이름

parms 매개변수 문자열. 지정한 경우 문자열은 큰따옴표로 묶어야 합니다.

comment
항목에 연관된 설명. 최대 길이는 30자입니다. 주석은 큰따옴표로 묶으십시오.

제 118 장 sqlegdcl - 데이터베이스 연결 서비스(DCS) 디렉토리 스캔 종료

sqlegdsc API로 할당된 자원을 사용 가능하게 합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlegdcl (
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdcl (
        struct sqlca * pSqlca);
```

sqlegdcl API 매개변수

pSqlca

출력. sqlca 구조의 포인터

REXX API 구문

```
CLOSE DCS DIRECTORY
```

제 119 장 sqlegdel - 데이터베이스 연결 서비스(DCS) 디렉토리에서 데이터베이스 카탈로그 해제

데이터베이스 연결 서비스(DCS) 디렉토리에서 항목을 삭제합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlegdel (
        struct sql_dir_entry * pDCSDirEntry,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdel (
        struct sqlca * pSqlca,
        struct sql_dir_entry * pDCSDirEntry);
```

sqlegdel API 매개변수

pDCSDirEntry

입/출력(I/O). 데이터베이스 연결 서비스 디렉토리 구조의 포인터. 이 구조의 ldb 필드에 삭제하려는 데이터베이스의 로컬 이름을 입력하십시오. 일치하는 로컬 데이터베이스 이름의 DCS 디렉토리 항목이 삭제 전에 이 구조로 복사됩니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

DCS 데이터베이스는 sqleuncd API를 사용하여 카탈로그 해제 가능한 리모트 데이터베이스로 시스템 데이터베이스 디렉토리에 카탈로그됩니다.

DCS 디렉토리에 데이터베이스를 재카탈로그하려면 sqlegdad API를 사용하십시오.

노드에 카탈로그된 DCS 데이터베이스를 나열하려면 sqlegdsc, sqlegdgt 및 sqlegdcl API를 사용하십시오.

dir_cache 구성 매개변수를 사용하여 디렉토리 캐싱이 사용 가능한 경우, 데이터베이스, 노드 및 DCS 디렉토리 파일이 메모리에 캐시됩니다. 응용프로그램의 디렉토리 캐시가 첫 번째 디렉토리 검색 시에 작성됩니다. 캐시는 응용프로그램이 임의의 디렉토리 파일을 수정하는 경우에만 새로 고쳐지기 때문에 다른 응용프로그램이 작성한 디렉토리 변경은 응용프로그램을 재시작해야 적용됩니다. DB2 공유 캐시를 새로 고치려면(서버 전용) 데이터베이스 관리 프로그램을 중지(db2stop)한 다음 재시작(db2start)하십시오. 다른 응용프로그램에 대해 디렉토리 캐시를 새로 고치려면 해당 응용프로그램을 중지한 다음 재시작하십시오.

REXX API 구문

```
UNCATALOG DCS DATABASE dbname [USING :value]
```

REXX API 매개변수

dbname

삭제하려는 디렉토리 항목의 로컬 데이터베이스 이름

값 디렉토리 항목 정보가 리턴되는 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다. 이름을 지정하지 않으면 이름으로 SQLGWINF가 사용됩니다.

XXX.0

변수의 요소 수(항상 7)

XXX.1

RELEASE

XXX.2

LDB

XXX.3

TDB

XXX.4

AR

XXX.5

PARMS

XXX.6

COMMENT

XXX.7

RESERVED

제 120 장 sqlegdge - 데이터베이스 연결 서비스(DCS) 디렉토리에서 특정 항목 가져오기

데이터베이스 연결 서비스(DCS) 디렉토리에서 특정 항목에 대한 정보를 리턴합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlegdge (
        struct sql_dir_entry * pDCSDirEntry,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdge (
        struct sqlca * pSqlca,
        struct sql_dir_entry * pDCSDirEntry);
```

sqlegdge API 매개변수

pDCSDirEntry

입/출력(I/O). 데이터베이스 연결 서비스 디렉토리 구조의 포인터. 이 구조의 ldb 필드에 DCS 디렉토리 항목을 검색하려는 로컬 데이터베이스 이름을 입력하십시오.

구조의 나머지 필드는 이 API 리턴 시에 입력됩니다.

pSqlca

출력. sqlca 구조의 포인터

REXX API 구문

```
GET DCS DIRECTORY ENTRY FOR DATABASE dbname [USING :value]
```

REXX API 매개변수

dbname

확보하려는 디렉토리 항목의 로컬 데이터베이스 이름을 지정합니다.

값 디렉토리 항목 정보가 리턴되는 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다. 이름을 지정하지 않으면 이름으로 SQLGWINF가 사용됩니다.

XXX.0

변수의 요소 수(항상 7)

XXX.1

RELEASE

XXX.2

LDB

XXX.3

TDB

XXX.4

AR

XXX.5

PARMS

XXX.6

COMMENT

XXX.7

RESERVED.

제 121 장 sqlegdgt - 데이터베이스 연결 서비스(DCS) 디렉토리 항목 가져오기

데이터베이스 연결 서비스(DCS) 디렉토리 항목 사본을 응용프로그램에서 제공하는 버퍼로 전송합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlegdgt (
        short * pNumEntries,
        struct sql_dir_entry * pDCSDirEntries,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdgt (
        struct sqlca * pSqlca,
        short * pNumEntries,
        struct sql_dir_entry * pDCSDirEntries);
```

sqlegdgt API 매개변수

pNumEntries

입/출력(I/O). 호출자의 버퍼에 복사되는 항목 수를 나타내는 짧은 정수의 포인터. 실제로 복사되는 항목 수가 리턴됩니다.

pDCSDirEntries

출력. 수집된 DCS 디렉토리 항목이 API 호출 리턴 시에 보유되는 버퍼의 포인터. 버퍼는 pNumEntries 매개변수에 지정된 항목 수를 보유할 만큼 커야 합니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

항목 계수를 리턴하는 sqlegdsc API는 GET DCS DIRECTORY ENTRIES 발행 전에 호출되어야 합니다.

모든 항목이 호출자에 복사되면 데이터베이스 연결 서비스 디렉토리 스캔이 자동으로 닫히고 모든 자원이 릴리스됩니다.

항목이 남아 있는 경우 이 API에 대한 후속 호출을 작성하거나 CLOSE DCS DIRECTORY SCAN을 호출하여 시스템 자원을 릴리스해야 합니다.

REXX API 구문

```
GET DCS DIRECTORY ENTRY [USING :value]
```

REXX API 매개변수

값 디렉토리 항목 정보가 리턴되는 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다. 이름을 지정하지 않으면 이름으로 SQLGWINF가 사용됩니다.

XXX.0
변수의 요소 수(항상 7)

XXX.1
RELEASE

XXX.2
LDB

XXX.3
TDB

XXX.4
AR

XXX.5
PARMS

XXX.6
COMMENT

XXX.7
RESERVED

제 122 장 sqlegdsc - 데이터베이스 연결 서비스(DCS) 디렉토리 스캔 시작

데이터베이스 연결 서비스 디렉토리 항목 사본을 메모리에 저장하고 항목 수를 리턴합니다. 이는 디렉토리를 연 시점의 디렉토리 스냅샷입니다.

이 API를 호출한 후에 디렉토리 자체가 변경된 경우에는 사본이 갱신되지 않습니다. sqlegdgt API 및 sqlegdcl API를 사용하여 이 API 호출에 연관된 자원을 릴리스하십시오.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlegdsc (
        short * pNumEntries,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggdsc (
        struct sqlca * pSqlca,
        short * pNumEntries);
```

sqlegdsc API 매개변수

pNumEntries

출력. 디렉토리 항목 수가 리턴되는 2바이트 영역의 주소

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

스캔의 호출자는 리턴된 값인 pNumEntries를 사용하여 항목 수신에 충분한 메모리를 할당합니다. 사본이 이미 있는 상태에서 스캔 호출을 수신하면 이전의 사본이 릴리스되고 새 사본이 수집됩니다.

REXX API 구문

OPEN DCS DIRECTORY

제 123 장 sqlegins - 현재 인스턴스 가져오기

DB2INSTANCE 환경 변수 값을 리턴합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlegins (
        _SQLOLDCHAR * pInstance,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlggins (
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pInstance);
```

sqlegins API 매개변수

pInstance

출력. 데이터베이스 관리 프로그램 인스턴스 이름이 배치되는 문자열 버퍼의 포인터. 이 버퍼 길이는 널(NULL) 종료 문자를 위한 1바이트를 포함하여 최소 9바이트여야 합니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

DB2INSTANCE 환경 변수 값은 사용자가 연결된 인스턴스가 아니어도 됩니다.

사용자가 현재 접속된 인스턴스를 식별하려면 널(NULL) 인수를 포함하여 `sqleatin - Attach`를 호출하십시오(sqlca 구조는 제외)

REXX API 구문

```
GET INSTANCE INTO :instance
```

REXX API 매개변수

인스턴스

데이터베이스 관리 프로그램 인스턴스 이름이 배치되는 REXX 호스트 변수

제 124 장 sqleintr - 응용프로그램 요청 인터럽트

요청을 중지합니다. 이 API는 응용프로그램의 제어 중단 신호 핸들러에서 호출됩니다. 제어 중단 신호 핸들러는 sqleisig - 신호 핸들러 설치에서 설치되는 디폴트이거나 프로그래머가 제공하고 적절한 운영 체제 호출을 사용하여 설치되는 루틴일 수 있습니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_INTR
        sqleintr ( void );

SQL_API_RC SQL_API_FN
  sqlgintr (
        void);
```

sqleintr API 매개변수

없음

사용 시 참고사항

인터럽트 핸들러에서는 sqleintr을 제외한 데이터베이스 관리 프로그램 API는 호출되면 안됩니다. 그러나 시스템에서 이를 금지하지는 않습니다.

커미트 또는 롤백 상태의 모든 데이터베이스 트랜잭션은 인터럽트할 수 없습니다.

인터럽트된 데이터베이스 관리 프로그램 요청은 인터럽트되었음을 나타내는 코드를 리턴합니다.

다음 표에서는 기타 API에 대한 인터럽트 조작의 영향을 요약하여 나타냅니다.

표 9. INTERRUPT 조치

데이터베이스 활동	조치
BACKUP	유틸리티가 취소됩니다. 미디어의 데이터가 완벽하지 않을 수도 있습니다.
BIND	바인드가 취소됩니다. 패키지 작성이 롤백됩니다.

표 9. INTERRUPT 조치 (계속)

데이터베이스 활동	조치
COMMIT	없음. COMMIT가 완료됩니다.
CREATE DATABASE/CREATE DATABASE AT NODE/ADD NODE/DROP NODE VERIFY	특정 시점에서는 이 API를 인터럽트할 수 없습니다. 이 시점 이전에 인터럽트 호출이 수신되면 데이터베이스가 작성되지 않습니다. 이 시점 이후에 인터럽트 호출이 수신되면 무시됩니다.
DROP DATABASE/DROP DATABASE AT NODE	없음. API가 완료됩니다.
EXPORT/IMPORT/ RUNSTATS	유틸리티가 취소됩니다. 데이터베이스 갱신이 롤백됩니다.
FORCE APPLICATION	없음. FORCE APPLICATION이 완료됩니다.
LOAD	유틸리티가 취소됩니다. 테이블의 데이터가 완벽하지 않을 수도 있습니다.
PREP	프리컴파일이 취소됩니다. 패키지 작성이 롤백됩니다.
REORGANIZE TABLE	복사가 완료될 때까지 인터럽트가 지연됩니다. 다음에 테이블에 액세스할 때 인덱스의 재작성이 다시 시작됩니다.
RESTORE	유틸리티가 취소됩니다. DROP DATABASE가 수행됩니다. 테이블 스페이스 레벨 리스토어에는 적용되지 않습니다.
ROLLBACK	없음. ROLLBACK이 완료됩니다.
디렉토리 서비스	디렉토리는 계속 일관성 있는 상태를 유지합니다. 유틸리티 함수가 수행되거나 수행되지 않습니다.
SQL 데이터 정의 명령문	데이터베이스 트랜잭션이 SQL문 호출 전에 있던 상태로 설정됩니다.
기타 SQL문	데이터베이스 트랜잭션이 SQL문 호출 전에 있던 상태로 설정됩니다.

REXX API 구문

```
INTERRUPT
```

예:

```
call SQLDBS 'INTERRUPT'
```

제 125 장 sqleisig - 신호 핸들러 설치

디폴트 인터럽트(일반적으로 Control-C 및/또는 Control-Break) 신호 핸들러를 설치합니다. 이 디폴트 핸들러가 인터럽트 신호를 발견하면 신호를 재설정하고 sqleintr을 호출합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleisig (
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgisig (
        struct sqlca * pSqlca);
```

sqleisig API 매개변수

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

응용프로그램에 신호 핸들러가 없고 인터럽트를 수신하면 응용프로그램이 종료됩니다. 이 API는 단순한 신호 조절을 제공하고 응용프로그램에 확장된 인터럽트 조절에 대한 요구사항이 없는 경우에 사용 가능합니다.

인터럽트 신호 핸들러가 제대로 작동하려면 이에 대해 API가 호출되어야 합니다.

응용프로그램에 좀 더 정교한 인터럽트 조절 스킴이 필요한 경우 sqleintr API를 호출할 수 있는 신호 조절 루틴을 개발할 수 있습니다. 운영 체제 호출 또는 언어 고유의 라이브러리 신호 기능을 사용하십시오. sqleintr API가 사용자 정의된 신호 핸들러에서 수행되는 유일한 데이터베이스 관리 프로그램 연산이어야 합니다. 이전에 설치된 신호 핸들러가 제대로 작동하도록 모든 운영 체제 프로그래밍 기술 및 연습을 사용하십시오.

REXX API 구문

INSTALL SIGNAL HANDLER

제 126 장 sqlemgdb - 이전 버전의 DB2 데이터베이스를 현재 버전으로 이주

이전(버전 8 이상) 버전의 DB2 데이터베이스를 현재 릴리스로 변환합니다. sqlemgdb 및 sqlgmgdb API는 더 이상 사용되지 않으며 추후 릴리스에서 제공되지 않습니다. 대신 새 db2DatabaseUpgrade API를 사용해야 합니다.

권한 부여

SYSADM

필수 연결

이 API는 데이터베이스 연결을 작성합니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlemgdb (
        _SQLOLDCHAR * pDbAlias,
        _SQLOLDCHAR * pUserName,
        _SQLOLDCHAR * pPassword,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgmgdb (
        unsigned short PasswordLen,
        unsigned short UserNameLen,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pPassword,
        _SQLOLDCHAR * pUserName,
        _SQLOLDCHAR * pDbAlias);
```

sqlemgdb API 매개변수

pDbAlias

입력. 시스템 데이터베이스 디렉토리에 카탈로그된 데이터베이스 별명이 포함된 문자열

pUserName

입력. 응용프로그램의 사용자 이름이 포함된 문자열. NULL도 가능합니다.

pPassword

입력. 입력한 사용자 이름(있는 경우)의 암호가 포함된 문자열. NULL도 가능합니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgmgdb API 특정 매개변수

PasswordLen

입력. 암호 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 암호를 제공하지 않은 경우 영(0)으로 설정하십시오.

UserNameLen

입력. 사용자 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 사용자 이름을 제공하지 않은 경우 영(0)으로 설정하십시오.

DbAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

이 API는 데이터베이스를 새 버전으로 이주만 하며 이주된 데이터베이스를 해당 이전 버전으로 변환하는 데는 사용할 수 없습니다.

데이터베이스는 이주 전에 카탈로그해야 합니다.

REXX API 구문

```
MIGRATE DATABASE dbalias [USER username USING password]
```

REXX API 매개변수

dbalias

이주할 데이터베이스 별명

username

데이터베이스가 재시작되는 사용자 이름

password

사용자 이름을 인증하는 데 사용되는 암호

제 127 장 sqlencls - 노드 디렉토리 스캔 종료

sqlenops API로 할당된 자원을 사용 가능하게 합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlencls (
        unsigned short Handle,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgncls (
        unsigned short Handle,
        struct sqlca * pSqlca);
```

sqlencls API 매개변수

Handle

입력. 관련 OPEN NODE DIRECTORY SCAN API에서 리턴된 ID

pSqlca

출력. sqlca 구조의 포인터

REXX API 구문

```
CLOSE NODE DIRECTORY :scanid
```

REXX API 매개변수

scanid OPEN NODE DIRECTORY SCAN API에서 리턴된 scanid를 포함하는 호스트 변수

제 128 장 sqlengne - 다음 노드 디렉토리 항목 가져오기

sqlenops - 노드 디렉토리 스캔 열기가 호출된 뒤에 노드에서 다음 항목이 리턴됩니다. 이 API에 대한 후속 호출로 추가 항목이 리턴됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlengne (
        unsigned short Handle,
        struct sqleninfo ** ppNodeDirEntry,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgngne (
        unsigned short Handle,
        struct sqleninfo ** ppNodeDirEntry,
        struct sqlca * pSqlca);
```

sqlengne API 매개변수

Handle

입력. sqlenops - 노드 디렉토리 스캔 열기에서 리턴된 ID

ppNodeDirEntry

출력. sqleninfo 구조 포인터의 주소. 이 API 호출자는 단지 포인터인 구조에 대해 메모리를 제공할 필요가 없습니다. API에서 리턴될 때 포인터는 sqlenops - 노드 디렉토리 스캔 열기에서 할당된 노드 디렉토리의 사본에 있는 다음 노드 디렉토리 항목을 지시합니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

노드 디렉토리 항목 정보 버퍼의 모든 필드 오른쪽은 공백으로 채워집니다.

sqlca의 sqlcode 값은 이 API가 호출될 때 더 이상 스캔할 항목이 없는 경우에 1014로 설정됩니다.

전체 디렉토리는 이 API pNumEntries times를 호출하여 스캔할 수 있습니다.

REXX API 구문

```
GET NODE DIRECTORY ENTRY :scanid [USING :value]
```

REXX API 매개변수

scanid OPEN NODE DIRECTORY SCAN API에서 리턴된 ID를 포함하는 REXX 호스트 변수

값 노드 항목 정보가 리턴되는 복합 REXX 호스트 변수. 이름을 지정하지 않으면 이름으로 SQLENINFO가 사용됩니다. 다음에서 XXX는 호스트 변수 이름을 나타냅니다(해당 필드 이름은 API에서 리턴된 구조에서 가져옴)

XXX.0
변수의 요소 수(항상 16)

XXX.1
NODENAME

XXX.2
LOCALLU

XXX.3
PARTNERLU

XXX.4
MODE

XXX.5
COMMENT

XXX.6
RESERVED

XXX.7
PROTOCOL(프로토콜 유형)

XXX.9
RESERVED

XXX.10
SYMDESTNAME(기호 목적지 이름)

XXX.11
SECURITY(보안 유형)

XXX.12

HOSTNAME

XXX.13

SERVICENAME

XXX.14

FILESERVER

XXX.15

OBJECTNAME

XXX.16

INSTANCE(로컬 인스턴스 이름)

제 129 장 sqlenops - 노드 디렉토리 스캔 시작

노드 디렉토리 사본을 메모리에 저장하고 항목 수를 리턴합니다. 이는 디렉토리를 연 시점의 디렉토리 스냅샷입니다. 이 사본은 디렉토리 자체가 이후에 변경되어도 갱신되지 않습니다.

sqlengne API를 호출하여 노드 디렉토리에서 진행하고 노드 항목에 대한 정보를 평가하십시오. sqlencls API를 호출하여 스캔을 닫으십시오. 그러면 메모리에서 디렉토리 사본이 제거됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlenops (
        unsigned short * pHandle,
        unsigned short * pNumEntries,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgnops (
        unsigned short * pHandle,
        unsigned short * pNumEntries,
        struct sqlca * pSqlca);
```

sqlenops API 매개변수

pHandle

출력. 이 API에서 리턴된 ID. 이 ID는 sqlengne API 및 sqlencls API로 전달되어야 합니다.

pNumEntries

출력. 디렉토리 항목 수가 리턴되는 2바이트 영역의 주소

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

이 API가 할당한 스토리지는 sqlencls - 노드 디렉토리 스캔 닫기를 호출하여 사용 가능해집니다.

노드 디렉토리에 대해 다중 노드 디렉토리 스캔이 발행될 수 있습니다. 그러나 결과는 동일하지 않을 수 있습니다. 디렉토리는 열 때마다 달라질 수 있습니다.

프로세스별로 최대 8개의 노드 디렉토리 스캔이 가능합니다.

REXX API 구문

```
OPEN NODE DIRECTORY USING :value
```

REXX API 매개변수

값 노드 디렉토리 정보가 리턴되는 복합 REXX 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0
변수의 요소 수(항상 2)

XXX.1
scanid 번호가 포함된 REXX 호스트 변수를 지정합니다.

XXX.2
디렉토리에 포함된 항목 수

제 130 장 sqleqryc - 클라이언트 연결 설정 쿼리

응용프로그램 프로세스의 현재 연결 설정을 리턴합니다. sqle_conn_setting 데이터 구조가 연결 설정 유형 및 값으로 채워집니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleqryc (
        struct sqle_conn_setting * pConnectionSettings,
        unsigned short NumSettings,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgqryc (
        struct sqle_conn_setting * pConnectionSettings,
        unsigned short NumSettings,
        struct sqlca * pSqlca);
```

sqleqryc API 매개변수

pConnectionSettings

입/출력(I/O). 연결 설정 유형 및 값을 지정하는 sqle_conn_setting 구조의 포인터. NumSettings 연결 설정 구조 배열을 정의하고 가능한 5개의 연결 설정 옵션 중 하나를 표시하기 위해 이 배열에 각 요소의 유형 필드를 설정합니다. 리턴 시에 각 요소의 값 필드에 지정한 옵션의 현재 설정이 포함됩니다.

NumSettings

입력. 리턴되는 연결 옵션 값 수를 나타내는 정수(0 - 7)

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

응용프로그램 프로세스의 연결 설정은 실행 중에 언제든지 쿼리할 수 있습니다.

QUERY CLIENT가 성공한 경우 sqle_conn_setting 구조의 필드에는 응용프로그램 프로세스의 현재 연결 설정이 포함됩니다. SET CLIENT가 호출된 경우가 없는 경우에는 SQL문이 이미 처리된 경우에만 설정에 프리컴파일 옵션 값이 포함됩니다. 이 외의 경우에는 프리컴파일 옵션의 디폴트값이 포함됩니다.

REXX API 구문

```
QUERY CLIENT INTO :output
```

REXX API 매개변수

output

응용프로그램 프로세스의 현재 연결 설정에 대한 정보가 포함된 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다.

XXX.1

CONNECTION 유형의 현재 연결 설정

XXX.2

SQLRULES의 현재 연결 설정

XXX.3

COMMIT가 발행될 때 릴리스되는 연결을 나타내는 현재 연결 설정

XXX.4

SYNCPOINT 옵션의 현재 연결 설정. SYNCPOINT 옵션은 무시되고 이전 버전과의 호환성을 위해서만 사용됩니다. 트랜잭션 관리 프로그램을 사용하여 2단계의 커미트 시맨틱을 강제로 실행해야 하는지, 데이터베이스 관리 프로그램이 단일 트랜잭션에서 여러 데이터베이스에 액세스할 때 한 개의 데이터베이스만 갱신 중인지 확인해야 하는지 또는 이 옵션을 둘 다 사용하지 않아야 하는지를 나타냅니다.

XXX.6

지연된 PREPARE에 대한 현재 연결 설정

제 131 장 sqleqryi - 클라이언트 정보 쿼리

sqle_client_info 데이터 구조의 필드를 채워서 기존 클라이언트 정보를 리턴합니다. 이 API는 데이터베이스 별명 스펙을 허용하므로 응용프로그램은 특정 연결에 연관된 클라이언트 정보를 쿼리할 수 있습니다. sqleseti API가 이전에 값을 작성하지 않은 경우 널(NULL)을 리턴합니다.

특정 연결이 요청되면 이 API는 해당 연결의 최신 값을 리턴합니다. 모든 연결이 지정된 경우 API는 모든 연결에 연관된 값을 리턴합니다. 즉, sqleseti(모든 연결 정의)의 마지막 연결로 전달된 값.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleqryi (
        unsigned short DbAliasLen,
        char * pDbAlias,
        unsigned short NumItems,
        struct sqle_client_info* pClient_Info,
        struct sqlca * pSqlca);
```

sqleqryi API 매개변수

DbAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 값이 영(0)보다 큰 경우 pDbAlias는 별명 이름을 지시해야 합니다. 이 별명에 대한 sqleseti의 마지막 호출(또는 0 길이 별명을 지정하는 sqleseti에 대한 호출)과 연관된 설정을 리턴합니다. 영(0)이 지정된 경우 0 길이 별명을 지정한 sqleseti에 대한 마지막 호출과 연관된 설정을 리턴합니다.

pDbAlias

입력. 데이터베이스 별명이 포함된 문자열의 포인터

NumItems

입력. 수정 중인 항목 수. 최소값은 1입니다.

pClient_Info

입력. 리턴할 값을 나타내는 유형 필드 및 리턴된 값의 포인터 각각을 포함하는 sqle_client_info 구조 배열의 포인터. 지시된 영역에는 요청 중인 값을 수용할 만한 공간이 있어야 합니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

실행 중에는 항상 설정을 쿼리할 수 있습니다. API 호출이 성공하면 현재 설정이 지정된 영역에 리턴됩니다. sqleseti API 호출로 설정되지 않은 영(0) 길이 및 널(null)로 끝나는 문자열(₩0)을 리턴합니다.

제 132 장 sqlesact - 어카운팅 문자열 설정

응용프로그램의 다음 연결 요청으로 DRDA 서버에 송신되는 어카운팅 정보를 제공합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlesact (
        char * pAccountingString,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgsact (
        unsigned short AccountingStringLen,
        char * pAccountingString,
        struct sqlca * pSqlca);
```

sqlesact API 매개변수

pAccountingString
입력. 어카운팅 데이터가 포함된 문자열

pSqlca
출력. sqlca 구조의 포인터

sqlgsact API 특정 매개변수

AccountingStringLen
입력. 어카운팅 문자열의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

연결 요청으로 어카운팅 데이터를 송신하기 위해 응용프로그램은 데이터베이스에 연결하기 전에 이 API를 호출해야 합니다. API를 다시 호출하여 다른 데이터베이스에 연결하기 전에 어카운팅 문자열을 변경할 수 있습니다. 그 이외의 경우에는 응용프로그램

이 종료될 때까지 값이 계속 유지됩니다. 어카운팅 문자열 길이는 최대 SQL_ACCOUNT_STR_SZ(sqlenv에 정의) 바이트이며 더 긴 문자열은 절단됩니다. DRDA 서버로 전송될 때 어카운팅 문자열이 제대로 변환되도록 하려면 A - Z, 0 - 9 및 밑줄(_) 문자만 사용하십시오.

제 133 장 sqlesdeg - 최대 런타임 파티션 내 병렬 처리 레벨 또는 SQL문의 등급 설정

지정된 활성 응용프로그램을 위한 SQL문 실행에 대한 파티션 내 병렬 처리의 최대 런타임 등급을 설정합니다. CREATE INDEX문 실행 병렬 처리에는 아무런 영향도 없습니다.

범위

이 API는 db2nodes.cfg 파일에 나열된 모든 데이터베이스 파티션 서버에 영향을 줍니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

인스턴스. 리모트 서버에서 병렬 처리의 최대 런타임 등급을 변경하려면 우선 해당 서버에 연결해야 합니다. 전혀 접속되지 않은 경우에는 SET RUNTIME DEGREE 문이 실패합니다.

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlesdeg (
        sqlint32 NumAgentIds,
        sqluint32 * pAgentIds,
        sqlint32 Degree,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgsdeg (
        struct sqlca * pSqlca,
        sqlint32 Degree,
        sqluint32 * pAgentIds,
        sqlint32 NumAgentIds);
```

sqlesdeg API 매개변수

NumAgentIds

입력. 새 등급 값이 적용되는 총 활성 응용프로그램 수를 나타내는 정수. 이 수는 에이전트 ID 배열의 요소 수와 동일해야 합니다.

이 매개변수가 SQL_ALL_USERS로 설정되는 경우(sqlenv에 정의) 새 등급이 모든 활성 응용프로그램에 적용됩니다. 영(0)으로 설정하면 오류가 리턴됩니다.

pAgentIds

입력. 부호없는 긴 정수 배열의 포인터. 각 항목은 해당 응용프로그램의 에이전트 ID를 설명합니다. 활성 응용프로그램의 에이전트 ID를 나열하려면 db2GetSnapshot API를 사용하십시오.

Degree

입력. 병렬 처리의 최대 런타임 등급의 새 값. 값은 1 - 32767 범위 내에 있어야 합니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

데이터베이스 시스템 모니터 기능이 에이전트 ID 및 활성 응용프로그램의 등급을 수집하는 데 사용됩니다.

에이전트 ID 배열에서는 최소한의 유효성 확인이 수행됩니다. 포인터가 지정된 총 요소 수가 포함된 배열을 지시하도록 확인해야 합니다. NumAgentIds가 SQL_ALL_USERS로 설정된 경우에는 배열이 무시됩니다.

지정된 하나 이상의 에이전트 ID를 찾을 수 없는 경우에는 알 수 없는 에이전트 ID는 무시되고 함수가 계속됩니다. 오류가 리턴되지 않습니다. 예를 들어, 에이전트 ID가 수집되고 API가 호출되는 사이에 사용자가 로그오프하면 에이전트 ID를 찾을 수 없습니다.

에이전트 ID는 재사용되고 데이터베이스 시스템 모니터로 수집된 후에 응용프로그램의 병렬 처리 등급을 변경하는 데 사용됩니다. 사용자가 로그오프하고 다른 사용자가 로그온하여 이 재사용 프로세스를 통해 동일한 에이전트 ID를 획득하면 결과적으로 병렬 처리의 새 등급이 잘못된 사용자에 대해 수정됩니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 134 장 sqlesetc - 클라이언트 연결 설정값 설정

응용프로그램의 연결 설정을 지정합니다. sqle_conn_setting 데이터 구조를 사용하여 연결 설정 유형 및 값을 지정하십시오.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlesetc (
        struct sqle_conn_setting * pConnectionSettings,
        unsigned short NumSettings,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgsetc (
        struct sqle_conn_setting * pConnectionSettings,
        unsigned short NumSettings,
        struct sqlca * pSqlca);
```

sqlesetc API 매개변수

pConnectionSettings

입력. 연결 설정 유형 및 값을 지정하는 sqle_conn_setting 구조의 포인터. NumSettings sqle_conn_setting 구조 배열을 할당하십시오. 설정할 연결 옵션을 표시하기 위해 이 배열에 각 요소의 유형 필드를 설정하십시오. 옵션에 대해 필요한 값으로 값 필드를 설정하십시오.

NumSettings

입력. 설정할 연결 옵션 값 수를 나타내는 정수(0 - 7)

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

API가 성공하면 후속 작업 단위(UOW)의 연결에서 지정한 연결 설정값을 사용합니다. 이 API가 실패하면 연결 설정이 변경되지 않습니다.

응용프로그램의 연결 설정은 기존 연결이 없는 경우에만 변경 가능합니다(예를 들어, 연결 작성 이전 또는 RELEASE ALL 및 COMMIT 이후).

SET CLIENT API가 제대로 실행되면 연결 설정이 고정되어 SET CLIENT API를 다시 실행해서만 변경할 수 있습니다. 응용프로그램 모듈의 해당하는 모든 프리컴파일된 옵션이 겹쳐쓰기 됩니다.

REXX API 구문

```
SET CLIENT USING :values
```

REXX API 매개변수

값 응용프로그램 프로세스의 연결 설정이 포함된 복합 REXX 호스트 변수. 다음에서 XXX는 호스트 변수 이름을 나타냅니다.

XXX.0

작성하려는 연결 설정 수

XXX.1

CONNECTION 유형 설정 방법을 지정합니다. 유효한 값은 다음과 같습니다.

1 유형 1 CONNECT

2 유형 2 CONNECT

XXX.2

다음에 따라 SQLRULES를 설정하는 방법을 지정합니다.

- 유형 2 CONNECT는 DB2 규칙 또는 ISO/ANS SQL92를 기반으로 하는 표준(STD) 규칙을 사용하여 처리됩니다.
- 결과 세트에 응용프로그램이 LOB 컬럼 형식을 지정하는 방법에 따라 설정됩니다.

DB2

- SQL CONNECT 문이 현재 연결을 다른 설정된(*유휴*) 연결로 전환할 수 있도록 합니다.
- 이 디폴트 설정을 사용하여 응용프로그램은 첫 번째 페치 요청에서만 LOB 값 또는 LOB 로케이터가 검색되도록 지정할 수 있습니다. 후속 페치 요청에서는 동일한 LOB 컬럼 형식을 사용해야 합니다.

STD

- SQL CONNECT 문이 *새* 연결만 작성하도록 합니다. SQL SET CONNECTION 문을 사용하여 유휴 연결로 전환해야 합니다.
- 응용프로그램은 각 페치 요청에서 LOB 값과 LOB 로케이터 검색 사이에서 변경할 수 있습니다. 즉, 하나 이상의 LOB 컬럼이 포함된 커서는 BLOCKING 바인드 옵션 설정에 상관 없이 블록할 수 없습니다.

XXX.3

커미트 시에 데이터베이스의 연결 끊기 범위 설정 방법을 지정합니다. 유효한 값은 다음과 같습니다.

EXPLICIT

SQL RELEASE 문으로 표시된 연결만 끊기

CONDITIONAL

열려 있는 WITH HOLD 커서가 없는 연결만 끊기

AUTOMATIC

모든 연결 끊기

XXX.4

커미트 또는 롤백 중에 여러 데이터베이스 연결에서 조정 설정 방법을 지정합니다. 유효한 값은 다음과 같습니다.

TWOPHASE

트랜잭션 관리 프로그램(TM)을 사용하여 2단계 커미트로 조정합니다. SYNCPOINT 옵션은 무시되고 이전 버전과의 호환성을 위해서만 사용됩니다.

XXX.6

PREPARE문 실행 시기를 지정합니다. 유효한 값은 다음과 같습니다.

NO PREPARE문이 발행된 시점에서 실행됩니다.

YES PREPARE문이 해당 OPEN, DESCRIBE 또는 EXECUTE 문 발행 이전에는 실행되지 않습니다. 그러나 PREPARE INTO 문은 지연되지 않습니다.

ALL YES와 동일하며 PREPARE INTO문도 지연되는 점만 다릅니다.

제 135 장 sqleseti - 클라이언트 정보 설정

이미 연결되어 있는 경우 응용프로그램이 특정 연결과 연관된 클라이언트 정보를 설정하도록 합니다(sqle_client_info 데이터 구조에 필드를 설정하여)

TP 모니터 또는 3 티어 클라이언트/서버 응용프로그램 환경에서 클라이언트 대신 작동 중인 응용프로그램 서버(AS) 뿐만 아니라 클라이언트에 대한 정보도 확보해야 합니다. 이 API를 사용하여 응용프로그램 서버(AS)는 클라이언트의 사용자 ID, 워크스테이션 정보, 프로그램 정보 및 기타 어카운팅 정보를 DB2 서버에 전달할 수 있습니다. 그 이외의 경우에는 응용프로그램 서버(AS) 정보만 전달되고 해당 정보는 동일한 응용프로그램 서버(AS)를 사용하는 다수의 클라이언트 호출에 대해 동일합니다.

응용프로그램은 별명을 지정하지 않도록 선택할 수 있으며 이 경우 클라이언트 정보가 모든 기존 연결 뿐만 아니라 이후의 연결에도 설정됩니다. 이 API는 임의의 SQL이 실행되기 전 또는 커미트 또는 롤백 이후에 작업 단위(UOW) 이외에서만 변경되도록 허용합니다. 호출이 성공하는 경우 연결 값은 다음 기회에 전송되고 해당 연결에서 전송되는 다음 SQL 요청에서 그룹화됩니다. 성공적인 호출은 값이 승인되고 후속 연결에서 전파되는 것을 나타냅니다.

이 API는 데이터베이스에 연결하기 전에 값을 작성하는 데 사용하거나 연결된 후에 값을 설정하거나 수정하는 데 사용할 수 있습니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleseti (
        unsigned short DbAliasLen,
        char * pDbAlias,
        unsigned short NumItems,
        struct sqle_client_info* pClient_Info,
        struct sqlca * pSqlca);
```

sqleseti API 매개변수

DbAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수. 0보다 큰 값이 제공되면 **pDbAlias**는 별명 이름을 지시해야 하며 설정은 지정한 연결에만 영향을 줍니다. 영(0)으로 지정된 경우 설정은 기존 및 이후 연결 모두에 영향을 줍니다.

pDbAlias

입력. 데이터베이스 별명이 포함된 문자열의 포인터

NumItems

입력. 수정 중인 항목 수. 최소 값은 1입니다.

pClient_Info

입력. **NumItems** sqle_client_info 구조 배열의 포인터로, 각각은 설정할 값을 나타내는 유형 필드, 해당 값의 길이 및 새 값의 포인터를 포함합니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

별명 이름이 제공된 경우 별명의 연결은 이미 있어야 하고 해당 별명의 모든 연결이 변경사항을 상속합니다. 정보는 해당 별명의 연결이 끊어질 때까지 유지됩니다. 별명 이름이 제공되지 않으면 기존의 모든 연결 설정이 변경되고 이후의 모든 연결이 변경사항을 상속합니다. 프로그램이 종료될 때까지 정보가 유지됩니다.

필드 이름은 제공 가능한 정보 유형의 가이드라인입니다. 예를 들어, TP 모니터 응용프로그램은 SQL_CLIENT_INFO_APPLNAM 필드에 TP 모니터 트랜잭션 ID와 응용프로그램 이름을 같이 제공하도록 선택할 수 있습니다. 그러면 DB2 서버에서 더 나은 모니터링 및 어카운팅을 제공하며 여기서 DB2 트랜잭션 ID는 TP 모니터 트랜잭션 ID와 연관될 수 있습니다.

현재 이 API는 정보를 DB2 OS/390 버전 5 이상, DB2 UDB 버전 7 이상 및 DB2 i5/OS® V6R1 이상으로 전달합니다. 모든 정보가 DISPLAY THREAD 명령에 표시되며(어카운팅 문자열은 제외) 어카운팅 레코드에 모두 로깅됩니다.

API가 제공하는 데이터 값은 SQL 특수 레지스터에서도 액세스할 수 있습니다. 이 레지스터의 값은 데이터베이스 코드 페이지에 저장됩니다. 이 API에서 제공되는 데이터 값은 특수 레지스터에 저장되기 전에 데이터베이스 코드 페이지로 변환됩니다. 데이터베이스 코드 페이지로 변환한 후에 지원되는 최대 크기를 초과하는 모든 데이터 값은 서버에 저장되기 전에 절단됩니다. 절단된 이 값은 특수 레지스터로 리턴됩니다. 원래 데이터 값도 서버에 저장되며 데이터베이스 코드 페이지로 변환되지 않습니다. 변환되지 않는 값은 sqleqryi API를 호출하여 리턴할 수 있습니다.

연결 전에 CLI 프로그램에서 sqleseti API를 호출하면 작동하지 않습니다. 연결 후에 CLI 프로그램에서 sqleseti API를 호출하면 예상할 수 없는 동작이 발생할 수도 있습니다. 대신 해당 CLI 함수인 SQLSetConnectAttr() 또는 SQLSetEnvAttr()을 사용하는 것이 좋습니다.

제 136 장 sqleuncd - 시스템 데이터베이스 디렉토리에서 데이터베이스 카탈로그 해제

시스템 데이터베이스 디렉토리에서 항목을 삭제합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleuncd (
        _SQLOLDCHAR * pDbAlias,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlguncd (
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pDbAlias);
```

sqleuncd API 매개변수

pDbAlias

입력. 카탈로그 해제되는 데이터베이스 별명이 포함된 문자열

pSqlca

출력. sqlca 구조의 포인터

sqlguncd API 특정 매개변수

DbAliasLen

입력. 데이터베이스 별명 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

시스템 데이터베이스 디렉토리의 항목만 카탈로그 해제할 수 있습니다. 로컬 데이터베이스 디렉토리의 항목은 sqledrpd API를 사용하여 삭제할 수 있습니다.

재카탈로그하려면 sqlecadb API를 사용하십시오.

노드에서 카탈로그된 데이터베이스를 나열하려면 db2DbDirOpenScan, db2DbDirGetNextEntry 및 db2DbDirCloseScan API를 사용하십시오.

이전 서버와 통신할 때 사용되는 데이터베이스의 인증 유형은 우선 데이터베이스를 카탈로그 해제한 다음 다른 유형으로 다시 카탈로깅하여 변경할 수 있습니다.

dir_cache 구성 매개변수를 사용하여 디렉토리 캐싱이 사용 가능한 경우, 데이터베이스, 노드 및 DCS 디렉토리 파일이 메모리에 캐시됩니다. 응용프로그램의 디렉토리 캐시가 첫 번째 디렉토리 검색 시에 작성됩니다. 캐시는 응용프로그램이 임의의 디렉토리 파일을 수정하는 경우에만 새로 고쳐지기 때문에 다른 응용프로그램이 작성한 디렉토리 변경은 응용프로그램을 재시작해야 적용됩니다. DB2 공유 캐시를 새로 고치려면(서버 전용) 데이터베이스 관리 프로그램을 중지(db2stop)한 다음 재시작(db2start)하십시오. 다른 응용프로그램에 대해 디렉토리 캐시를 새로 고치려면 해당 응용프로그램을 중지한 다음 재시작하십시오.

REXX API 구문

```
UNCATALOG DATABASE dbname
```

REXX API 매개변수

dbname

카탈로그 해제되는 데이터베이스 별명

제 137 장 sqleuncn - 노드 디렉토리에서 항목 카탈로그 해제

노드 디렉토리에서 항목을 삭제합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl

필수 연결

없음

API 내장 파일

```
sqlenv.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleuncn (
        _SQLOLDCHAR * pNodeName,
        struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlguncn (
        unsigned short NodeNameLen,
        struct sqlca * pSqlca,
        _SQLOLDCHAR * pNodeName);
```

sqleuncn API 매개변수

pNodeName

입력. 카탈로그 해제되는 노드 이름이 포함된 문자열

pSqlca

출력. sqlca 구조의 포인터

sqlguncn API 특정 매개변수

NodeNameLen

입력. 노드 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

노드를 재카탈로그하려면 sqlectnd API를 사용하십시오.

카탈로그되는 노드를 나열하려면 db2DbDirOpenScan, db2DbDirGetNextEntry 및 db2DbDirCloseScan API를 사용하십시오.

dir_cache 구성 매개변수를 사용하여 디렉토리 캐싱이 사용 가능한 경우, 데이터베이스, 노드 및 DCS 디렉토리 파일이 메모리에 캐시됩니다. 응용프로그램의 디렉토리 캐시가 첫 번째 디렉토리 검색 시에 작성됩니다. 캐시는 응용프로그램이 임의의 디렉토리 파일을 수정하는 경우에만 새로 고쳐지기 때문에 다른 응용프로그램이 작성한 디렉토리 변경은 응용프로그램을 재시작해야 적용됩니다. DB2 공유 캐시를 새로 고치려면(서버 전용) 데이터베이스 관리 프로그램을 중지(db2stop)한 다음 재시작(db2start)하십시오. 다른 응용프로그램에 대해 디렉토리 캐시를 새로 고치려면 해당 응용프로그램을 중지한 다음 재시작하십시오.

REXX API 구문

```
UNCATALOG NODE nodename
```

REXX API 매개변수

nodename

카탈로그 해제되는 노드 이름

제 138 장 sqlgaddr - 변수 주소 가져오기

변수 주소를 다른 변수로 배치합니다. 이 API는 호스트 프로그래밍 언어인 FORTRAN 및 COBOL에서 사용되며 이 언어는 포인터 처리는 제공하지 않습니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlgaddr (
   char * pVariable,
   char ** ppOutputAddress);
```

sqlgaddr API 매개변수

pVariable

입력. 주소가 리턴되는 변수

ppOutputAddress

출력. 변수 주소가 리턴되는 4바이트의 영역.

제 139 장 sqlgdref - 주소 비참조

포인터로 정의한 버퍼에서 응용프로그램이 직접 액세스할 수 있는 변수로 데이터를 복사합니다. 이 API는 호스트 프로그래밍 언어인 FORTRAN 및 COBOL에서 사용되며 이 언어는 포인터 처리는 제공하지 않습니다. 이 API는 원하는 데이터에 대한 포인터를 리턴하는 API에서 결과를 가져오는 데 사용할 수 있습니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlgdref (
   unsigned int NumBytes,
   char * pTargetBuffer,
   char ** ppSourceBuffer);
```

sqlgdref API 매개변수

NumBytes
입력. 전송되는 바이트 수를 나타내는 정수.

pTargetBuffer
출력. 데이터가 이동되는 영역.

ppSourceBuffer
입력. 원하는 데이터가 포함된 영역의 포인터

제 140 장 sqlgmcpy - 임의 메모리 영역에서 다른 영역으로 데이터 복사

임의 메모리 영역에서 다른 영역으로 데이터를 복사합니다. 이 API는 메모리 블록 복사 기능을 제공하지 않는 호스트 프로그래밍 언어, FORTRAN 및 Cobol 언어에서 사용됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlgmcpy (
   void * pTargetBuffer,
   const void * pSource,
   sqluint32 NumBytes);
```

sqlgmcpy API 매개변수

pTargetBuffer
출력. 데이터가 이동되는 영역.

pSource
입력. 데이터를 이동한 영역.

NumBytes
입력. 전송되는 바이트 수를 나타내는 4바이트의 부호없는 정수

제 141 장 sqlogstt - SQLSTATE 메시지 가져오기

SQLSTATE 값과 연관된 메시지 텍스트를 검색합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlogstt (
        char * pBuffer,
        short BufferSize,
        short LineWidth,
        char * pSqlstate);

SQL_API_RC SQL_API_FN
  sqlggstt (
        short BufferSize,
        short LineWidth,
        char * pSqlstate,
        char * pBuffer);
```

sqlogstt API 매개변수

pBuffer

출력. 메시지 텍스트가 배치되는 문자열 버퍼의 포인터. 버퍼에 맞도록 메시지가 절단되는 경우에는 널(NULL) 문자열 종단자 문자를 절단 시에 사용할 수 있습니다.

BufferSize

입력. 검색된 메시지 텍스트를 보유하는 문자열 버퍼 크기(바이트)

LineWidth

입력. 각 메시지 텍스트 라인의 최대 라인 너비. 라인은 단어를 경계로 분리됩니다. 값이 0인 경우에는 메시지 텍스트가 라인 중단 없이 리턴됩니다.

pSqlstate

입력. 메시지 텍스트가 검색되는 SQLSTATE가 포함된 문자열. 이 필드는 영숫자로 5자리(특정 SQLSTATE) 또는 2자리(SQLSTATE 클래스, SQLSTATE

의 첫 번째 2자리)여야 합니다. 5자리가 전달되는 경우에는 이 필드가 NULL로 중단될 필요는 없지만 2자리가 전달되는 경우에는 NULL로 중단되어야 합니다.

사용 시 참고사항

호출별로 한 개의 메시지가 리턴됩니다.

LF/NULL 시퀀스는 각 메시지 끝에 위치합니다.

양수의 라인 너비를 지정하는 경우 LF/NULL 시퀀스가 단어 사이에 삽입되어 라인이 라인 너비를 초과하지 않습니다.

단어가 라인 너비보다 큰 경우에는 라인이 최대한의 문자로 채워진 다음 LF/NULL이 삽입되고 나머지 문자가 다음 라인에 표시됩니다.

리턴 코드

코드	메시지
+i	형식화된 메시지의 바이트 수를 나타내는 양수. 이 값이 호출자의 버퍼 크기 입력보다 큰 경우에 메시지가 잘립니다.
-1	함수의 메시지 포맷팅 서비스에 사용할 수 있는 메모리가 부족합니다. 요청된 메시지가 리턴되지 않습니다.
-2	SQLSTATE의 형식이 잘못 되었습니다. 영숫자여야 하며 길이는 2 또는 5자리여야 합니다.
-3	메시지 파일에 액세스할 수 없거나 해당 파일이 올바르지 않습니다.
-4	라인 너비가 영(0)보다 작습니다.
-5	유효하지 않은 sqlca, 잘못된 버퍼 주소 또는 버퍼 길이.

리턴 코드가 -1 또는 -3인 경우 메시지 버퍼에는 문제점에 대한 추가 정보가 포함됩니다.

REXX API 구문

```
GET MESSAGE FOR SQLSTATE sqlstate INTO :msg [LINEWIDTH width]
```

REXX API 매개변수

sqlstate
메시지 텍스트가 검색되는 SQLSTATE.

msg 메시지가 배치되는 REXX 변수

width 각 메시지 텍스트 라인의 최대 라인 너비. 라인은 단어를 경계로 분리됩니다. 값을 지정하지 않거나 이 매개변수를 0으로 설정하면 메시지 텍스트가 라인 중단 없이 리턴됩니다.

제 142 장 sqludrdt - 데이터베이스 파티션 그룹에서 데이터 재분배 API

데이터베이스 파티션 그룹에 있는 데이터베이스 파티션 간에 데이터를 재분배합니다. 현재 데이터 분산이 균등한지 여부에 관계없이 이를 지정할 수 있습니다. 재분배 알고리즘으로 현재 데이터 분산에 기초하여 이동할 데이터베이스 파티션을 선택합니다. 이 API는 REDISTRIBUTE DATABASE PARTITION GROUP 명령의 NOT ROLLFORWARD RECOVERABLE 옵션을 지원하지 않습니다.

이 API는 카탈로그 파티션에서만 호출할 수 있습니다. LIST DATABASE DIRECTORY 명령을 사용하여 각 데이터베이스에 대한 카탈로그 파티션인 데이터베이스 파티션 서버를 판별하십시오.

범위

이 API는 데이터베이스 파티션 그룹의 모든 데이터베이스 파티션에 영향을 줍니다.

권한 부여

다음 권한 중 하나여야 합니다.

- sysadm
- sysctrl
- dbadm

또한, 다음과 같은 권한 그룹 중 하나도 필요합니다.

- 재분배 중인 데이터베이스 파티션 그룹의 모든 테이블에 대한 DELETE, INSERT, SELECT 특권
- DATAACCESS 권한

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqludrdt (
   char * pNodeGroupName,
   char * pTargetPMapFileName,
   char * pDataDistFileName,
   SQL_PDB_NODE_TYPE * pAddList,
   unsigned short AddCount,
   SQL_PDB_NODE_TYPE * pDropList,
   unsigned short DropCount,
```

```
   unsigned char DataRedistOption,
   struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrdt (
   unsigned short NodeGroupNameLen,
   unsigned short TargetPMapFileNameLen,
   unsigned short DataDistFileNameLen,
   char * pNodeGroupName,
   char * pTargetPMapFileName,
   char * pDataDistFileName,
   SQL_PDB_NODE_TYPE * pAddList,
   unsigned short AddCount,
   SQL_PDB_NODE_TYPE * pDropList,
   unsigned short DropCount,
   unsigned char DataRedistOption,
   struct sqlca * pSqlca);
```

sqludrdt API 매개변수

pNodeGroupName

재분배하려는 데이터베이스 파티션 그룹의 이름

pTargetPMapFileName

대상 분산 맵을 포함하는 파일의 이름. 디렉토리 경로가 파일 이름의 일부로 지정되지 않은 경우에는 현재 디렉토리가 사용됩니다. 이 매개변수는 DataRedistOption 값이 T인 경우에 사용됩니다. 파일은 문자 형식이어야 하며 4096개의 항목(다중 파티션 데이터베이스 파티션 그룹의 경우) 또는 한 개의 항목(단일 파티션 데이터베이스 파티션 그룹의 경우)을 포함해야 합니다. 파일의 항목은 노드 수를 나타냅니다. 항목은 자유 형식으로 사용할 수 있습니다.

pDataDistFileName

입력 분산 정보를 포함하는 파일 이름. 디렉토리 경로가 파일 이름의 일부로 지정되지 않은 경우에는 현재 디렉토리가 사용됩니다. 이 매개변수는 DataRedistOption 값이 U인 경우에 사용됩니다. 파일은 문자 형식이어야 하며 4096의 양의 정수 항목을 포함해야 합니다. 파일의 각 항목은 해당 데이터베이스 파티션의 가중치를 표시해야 합니다. 4096값의 합계는 4,294,967,295 이하여야 합니다.

pAddList

데이터 재분배 중에 데이터베이스 파티션 그룹에 추가하려는 데이터베이스 파티션 목록. 목록의 항목은 SQL_PDB_NODE_TYPE 형식이어야 합니다.

AddCount

데이터베이스 파티션 그룹에 추가하는 데이터베이스 파티션 수.

pDropList

데이터 재분배 중에 데이터베이스 파티션 그룹에서 삭제하려는 데이터베이스 파티션 목록. 목록의 항목은 SQL_PDB_NODE_TYPE 형식이어야 합니다.

DropCount

데이터베이스 파티션 그룹에서 삭제하는 데이터베이스 파티션 수.

DataRedistOption

수행하려는 데이터 재분배 유형을 나타내는 단일 문자. 가능한 값은 다음과 같습니다.

U 균형있는 분산을 위해 데이터베이스 파티션 그룹을 재분배하도록 지정합니다. pDataDistFileName이 널(NULL)인 경우 현재 데이터 분산은 일정합니다(즉, 각 데이터베이스 파티션의 데이터 양이 동일함). pDataDistFileName 매개변수가 널(null)이 아닌 경우 이 파일의 값은 현재 데이터 분산을 나타내는 것으로 간주됩니다. DataRedistOption이 U인 경우 pTargetPMapFileName 매개변수는 널(NULL)이어야 합니다. 추가 목록에 지정된 데이터베이스 파티션이 추가되면 삭제 목록에 지정된 데이터베이스 파티션은 데이터베이스 파티션 그룹에서 삭제됩니다.

T pTargetPMapFileName 매개변수를 사용하여 데이터베이스 파티션 그룹을 재분배하도록 지정합니다. 이 옵션의 경우 pDataDistFileName, pAddList 및 pDropList 매개변수는 널(NULL)이어야 하며 AddCount 및 DropCount 매개변수는 둘 다 영(0)이어야 합니다.

C 실패한 재분배 조작이 계속되도록 지정합니다. 이 옵션의 경우 pTargetPMapFileName, pDataDistFileName, pAddList 및 pDropList 매개변수는 널(NULL)이어야 하며 AddCount 및 DropCount 매개변수는 둘 다 영(0)이어야 합니다.

R 실패한 재분배 조작이 롤백되도록 지정합니다. 이 옵션의 경우 pTargetPMapFileName, pDataDistFileName, pAddList 및 pDropList 매개변수는 널(NULL)이어야 하며 AddCount 및 DropCount 매개변수는 둘 다 영(0)이어야 합니다.

pSqlca

출력. sqlca 구조의 포인터

sqlgdrdt API 특정 매개변수

NodeGroupNameLen

데이터베이스 파티션 그룹 이름 길이.

TargetPMapFileNameLen

대상 분산 맵 파일의 이름 길이.

DataDistFileNameLen

데이터 분산 파일 이름 길이.

사용 시 참고사항

재분배 조작이 완료되면 메시지 파일이 다음에 기록됩니다.

- UNIX 기반 시스템에서 서브디렉토리 및 파일 이름에 대해 database-name.nodegroup-name.timestamp 형식을 사용하여 $HOME/sqllib/redist 디렉토리에 기록됩니다.
- Windows 운영 체제에서 서브디렉토리 및 파일 이름에 대해 database-name₩first-eight-characters-of-the-nodegroup-name₩date₩time 형식을 사용하여 $HOME₩sqllib₩redist₩ 디렉토리에 기록됩니다.

시간소인 값은 API가 호출된 시간입니다.

유틸리티 프로그램은 처리 중 간헐적인 COMMIT를 수행합니다.

데이터베이스 파티션 그룹에 데이터베이스 파티션을 추가하려면 ALTER DATABASE PARTITION GROUP문을 사용하십시오. 이 명령문이 데이터베이스 파티션과 연관된 테이블 스페이스에 대한 컨테이너를 정의할 수 있도록 합니다.

재분배가 일어난 테이블에 종속성을 갖는 모든 패키지는 무효화됩니다. 데이터베이스 파티션 그룹 재분배 조작이 완료된 후 이러한 패키지를 명시적으로 리바인드하는 것이 바람직합니다. 명시적으로 리바인드하면 유효하지 않은 패키지에 대한 첫 번째 SQL 요청을 실행할 때 초기 지연이 발생하지 않습니다. 재분배 메시지 파일에는 재분배가 일어난 모든 테이블 목록이 들어 있습니다.

데이터베이스 파티션 그룹 재분배 조작이 완료된 후에 db2Runstats API를 발행하여 통계도 갱신하는 것이 바람직합니다.

복제된 요약 테이블이나 DATA CAPTURE CHANGES 절에 정의된 테이블이 포함된 데이터베이스 파티션 그룹은 재분배할 수 없습니다.

데이터베이스 파티션 그룹에 기존의 선언된 임시 테이블이 있는 사용자 임시 테이블 스페이스가 있으면 재분배를 수행할 수 없습니다.

REXX API 구문

SQLDB2 인터페이스를 사용하여 이 API는 REXX에서 호출 가능합니다.

제 143 장 sqlugrpn - 행에 대해 데이터베이스 파티션 서버 번호 가져오기

DB2 9.7을 시작으로 이 API는 더 이상 사용되지 않습니다. db2GetRowPartNum(행에 대해 데이터베이스 파티션 서버 번호 가져오기) API를 사용하여 행에 대한 데이터베이스 파티션 번호 및 데이터베이스 파티션 서버 번호를 리턴하십시오. 이 API가 사용되는 경우 SQL2768N 메시지가 리턴됩니다.

분산 키 값을 바탕으로 데이터베이스 파티션 번호 및 데이터베이스 파티션 서버 번호를 리턴합니다. 응용프로그램은 이 정보를 사용하여 특정 테이블 행이 저장되는 데이터베이스 파티션 서버를 판별합니다.

파티셔닝 데이터 구조인 sqlupi가 이 API의 입력입니다. 구조는 sqlugtpi API로 리턴할 수 있습니다. 다른 입력은 해당 분산 키 값의 문자 표시입니다. 출력은 분산 전략 및 분산 맵의 해당 데이터베이스 파티션 서버 번호로 생성된 데이터베이스 파티션 번호입니다. 분산 맵 정보가 제공되지 않으면 데이터베이스 파티션 번호만 리턴됩니다. 이는 데이터 분산을 분석할 때 유용할 수 있습니다.

이 API를 호출할 때 데이터베이스 관리 프로그램은 실행 중일 필요는 없습니다.

범위

이 API는 db2nodes.cfg 파일의 데이터베이스 파티션 서버에서 호출해야 합니다. 클라이언트와 서버 사이의 코드 페이지 및 엔디안의 차이로 인해 데이터베이스 파티셔닝 정보에 오류가 발생할 수 있기 때문에 이 API는 클라이언트에서 호출하면 안됩니다.

권한 부여

없음

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlugrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_ctrycode,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
```

```
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);

SQL_API_RC SQL_API_FN
  sqlggrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_code,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);
```

sqlugrpn API 매개변수

num_ptrs

ptr_array의 포인터 수. 값은 part_info 매개변수에 지정한 값과 동일해야 합니다. 즉, part_info->sqld.

ptr_array

part_info에 지정한 분산 키의 각 파트에 해당하는 값의 문자 표시를 지시하는 포인터 배열. 널(NULL) 값이 필요한 경우 해당 포인터가 널(NULL)로 설정됩니다. 생성된 컬럼의 경우 이 함수는 행에 대해 값을 생성하지 않습니다. 사용자가 행의 올바른 파티셔닝을 위해 값을 제공해야 합니다.

ptr_lens

part_info에 지정된 파티셔닝 키의 각 부분에 해당하는 값에 대한 문자 표시 길이가 포함된 부호없는 정수의 배열.

territory_ctrycode

목표 데이터베이스의 국가/지역 코드. 이 값은 GET DATABASE CONFIGURATION 명령을 사용하여 데이터베이스 구성 파일에서 가져올 수도 있습니다.

codepage

목표 데이터베이스의 코드 페이지. 이 값은 GET DATABASE CONFIGURATION 명령을 사용하여 데이터베이스 구성 파일에서 가져올 수도 있습니다.

part_info

sqlupi 구조의 포인터

part_num

데이터베이스 파티션 번호를 저장하는 데 사용되는 2바이트의 부호있는 정수의 포인터

node_num

노드 번호를 저장하는 데 사용되는 SQL_PDB_NODE_TYPE 필드의 포인터. 포인터가 널(NULL)인 경우 노드 번호가 리턴되지 않습니다.

chklvl 입력 매개변수에서 수행되는 점검 레벨을 지정하는 부호없는 정수. 값을 영(0)으로 지정하는 경우 점검이 수행되지 않습니다. 0이 아닌 값을 지정하면 모든 입력 매개변수가 점검됩니다.

sqlca 출력. sqlca 구조의 포인터

dataformat

분산 키 값 표시를 지정합니다. 가능한 값은 다음과 같습니다.

SQL_CHARSTRING_FORMAT

모든 분산 키 값은 문자열로 표시됩니다. 이것은 디폴트값입니다.

SQL_IMPLIEDDECIMAL_FORMAT

내재된 소수점 위치는 컬럼 정의로 별됩니다. 예를 들어 컬럼 정의가 DECIMAL(8,2)인 경우 값 12345는 123.45로 처리됩니다.

SQL_PACKEDDECIMAL_FORMAT

모든 10진수 컬럼 분산 키 값은 10진수 형식으로 압축됩니다.

SQL_BINARYNUMERICS_FORMAT

모든 숫자 키 값은 빅 엔디안(big-endian) 2진 형식입니다.

pReserved1

나중에 사용하기 위해 예약됨

pReserved2

나중에 사용하기 위해 예약됨

사용 시 참고사항

운영 체제에서 지원되는 데이터 유형은 분산 키로 정의된 유형과 동일합니다.

주: CHAR, VARCHAR, GRAPHIC 및 VARGRAPHIC 데이터 유형은 이 API를 호출하기 전에 데이터베이스 코드 페이지로 변환해야 합니다.

숫자 및 날짜 시간 데이터 유형의 경우 문자 표시는 API가 호출되는 각 시스템의 코드 페이지에 있어야 합니다.

node_num이 널(null)인 경우 분산 맵을 입력해야 합니다. 즉, part_info 매개변수의 pmaplen 필드(part_info->pmaplen)는 2 또는 8192입니다. 그 외의 경우에는 SQLCODE

-6038이 리턴됩니다. 분산 키를 정의해야 합니다. 즉, part_info 매개변수의 sqld 필드(part_info->sqld)는 0보다 커야 합니다. 그 외의 경우에는 SQLCODE -2032가 리턴됩니다.

널(null) 값을 허용하지 않는 컬럼에 널(NULL) 값을 지정하면 SQLCODE -6039가 리턴됩니다.

입력 문자열의 모든 앞 공백 및 뒤 공백은 스트립되며 뒤 공백만 스트립될 수 있는 CHAR, VARCHAR, GRAPHIC 및 VARGRAPHIC 데이터 유형은 예외입니다.

제 144 장 sqlugtpi - 테이블 분산 정보 가져오기

DB2 9.7을 시작으로 이 API는 더 이상 사용되지 않습니다. 분산 정보를 가져오는 데 사용되는 경우 이 API는 SQL2768N을 리턴합니다. db2GetDistMap(분산 맵 가져오기) API를 사용하여 분산 정보를 리턴하십시오.

응용프로그램에서 테이블의 분산 정보를 가져올 수 있습니다. 분산 정보에는 분산 맵과 분산 키의 컬럼 정의가 포함됩니다. 이 API가 리턴한 정보는 sqlugrpn API로 전달되어 임의 행에 대한 데이터베이스 파티션 번호 및 테이블의 데이터베이스 파티션 서버 번호를 판별합니다.

이 API를 사용하려면 분산 정보가 요청된 테이블이 포함된 데이터베이스에 응용프로그램이 연결되어야 합니다.

범위

이 API는 db2nodes.cfg 파일에 정의된 모든 데이터베이스 파티션 서버에서 실행할 수 있습니다.

권한 부여

참조 중인 테이블의 경우 다음 중 하나 이상이 있어야 합니다.

- DATAACCESS 권한
- CONTROL 특권
- SELECT 특권

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqlugtpi (
   unsigned char * tablename,
   struct sqlupi * part_info,
   struct sqlca * sqlca);

SQL_API_RC SQL_API_FN
  sqlggtpi (
```

```
unsigned short tn_length,
unsigned char * tablename,
struct sqlupi * part_info,
struct sqlca * sqlca);
```

sqlugtpi API 매개변수

tablename

테이블의 완전한 이름

part_info

sqlupi 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

sqlggtpi API 특정 매개변수

tn_length

테이블 이름 길이가 포함된 2바이트의 부호없는 정수

제 145 장 sqluvqdp - 테이블에 대한 테이블 스페이스 Quiesce

테이블에 대한 테이블 스페이스를 Quiesce합니다. 유효한 Quiesce 모드는 공유, 갱신 가능 및 독점으로 세 가지입니다. Quiesce 함수에서 작성되는 세 가지 가능한 테이블 스페이스 상태가 있습니다.

- Quiesced: SHARE
- Quiesced: UPDATE
- Quiesced: EXCLUSIVE

범위

단일 파티션 데이터베이스 환경에서 이 API는 로드 지속 기간 동안 독점 모드로 로드 조작에 연관되는 모든 테이블 스페이스를 Quiesce합니다. 파티션된 데이터베이스 환경에서 이 API는 데이터베이스 파티션에서 로컬로 사용됩니다. 이 API는 로드가 수행되는 데이터베이스 파티션에 속한 테이블 스페이스 부분만 Quiesce합니다.

권한 부여

다음 중 하나가 필요합니다.

- sysadm
- sysctrl
- sysmaint
- dbadm
- load

필수 연결

데이터베이스

API 내장 파일

```
sqlutil.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqluvqdp (
   char * pTableName,
   sqlint32 QuiesceMode,
   void * pReserved,
   struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgvqdp (
```

```
unsigned short TableNameLen,
char * pTableName,
sqlint32 QuiesceMode,
void * pReserved,
struct sqlca * pSqlca);
```

sqluvqdp API 매개변수

pTableName

입력. 시스템 카탈로그에 사용되는 테이블 이름이 포함된 문자열. 이 이름은 마침표(.)로 구분되는 스키마와 테이블 이름이 포함된 두 파트의 이름입니다. 스키마를 제공하지 않으면 CURRENT SCHEMA가 사용됩니다.

테이블은 시스템 카탈로그 테이블이 될 수 없습니다. 이 필드는 필수입니다.

QuiesceMode

입력. Quiesce 모드를 지정합니다. 유효한 값(sqlutil에 정의)은 다음과 같습니다.

SQLU_QUIESCEMODE_SHARE
공유 모드용

SQLU_QUIESCEMODE_INTENT_UPDATE
갱신 모드용

SQLU_QUIESCEMODE_EXCLUSIVE
독점 모드용

SQLU_QUIESCEMODE_RESET
다음 중 하나가 적용되는 경우에 테이블 스페이스 상태를 일반으로 재설정:

- 호출자가 Quiesce를 소유
- "팬텀 Quiesce"를 작성하여 Quiesce 연결 끊기를 설정한 호출자

SQLU_QUIESCEMODE_RESET_OWNED
호출자가 Quiesce를 소유하는 경우에 테이블 스페이스 상태를 일반으로 재설정.

이 필드는 필수입니다.

pReserved

나중에 사용하기 위해 예약됨

pSqlca

출력. sqlca 구조의 포인터

sqlgvqdp API 특정 매개변수

TableNameLen

입력. 테이블 이름의 길이(바이트 단위)를 나타내는 2바이트의 부호없는 정수

사용 시 참고사항

선언된 임시 테이블에서는 이 API가 지원되지 않습니다.

Quiesce 상태 공유 요청을 수신하면 트랜잭션에서 테이블 스페이스에 대해 부분 공유를 요청하고 테이블에 대해 공유 잠금을 요청합니다. 트랜잭션이 잠금을 획득할 때 테이블 스페이스의 상태는 QUIESCED SHARE로 변경됩니다. 이 상태는 다른 사용자가 보유하는 상태와 충돌하지 않는 경우에만 Quiescer에게 권한 부여됩니다. 테이블 스페이스 상태가 지속될 수 있도록 해당 상태는 권한 부여 ID 및 데이터베이스 에이전트 ID와 같이 테이블 스페이스에 기록됩니다.

테이블에 대한 테이블 스페이스가 QUIESCED SHARE 상태인 동안 테이블을 변경할 수 없습니다. 테이블 및 테이블 스페이스에 대한 기타 공유 모드 요청이 허용됩니다. 트랜잭션이 커미트하거나 롤백할 때 잠금은 릴리스되지만, 테이블에 대한 테이블 스페이스는 상태가 명시적으로 재설정될 때까지 QUIESCED SHARE 상태로 남아 있습니다.

Quiesce 독점 요청을 작성하면 트랜잭션은 테이블 스페이스에 대해 강한 배타적 잠금을 요청하고 테이블에 대해서는 강한 배타적 잠금을 요청합니다. 트랜잭션이 잠금을 획득할 때, 테이블 스페이스의 상태는 QUIESCED EXCLUSIVE로 변경됩니다. 테이블 스페이스의 상태는 Quiescer의 데이터베이스 에이전트 ID 및 권한 부여 ID와 함께 테이블 스페이스 테이블에 기록됩니다. 테이블 스페이스가 강한 독점 모드로 보유되어 있으므로 이 테이블 스페이스에 대한 다른 액세스는 허용되지 않습니다. 그러나 Quiesce 함수(Quiescer)를 호출하는 사용자는 테이블 및 테이블 스페이스에 대해 독점 액세스합니다.

Quiesce 상태 갱신 요청을 작성하면 테이블 스페이스가 의도를 가진 독점(IX) 모드로 잠겨지고 테이블은 갱신(U) 모드로 잠겨집니다. Quiescer가 포함된 테이블 스페이스 상태가 테이블 스페이스에 기록됩니다.

지정된 시간에 한 테이블 스페이스에서 Quiescer를 5개까지 사용하도록 제한합니다. QUIESCED EXCLUSIVE는 다른 모든 상태와 호환되지 않고 QUIESCED UPDATE는 다른 QUIESCED UPDATE와 호환되지 않으며 5개의 Quiescer 제한에 도달하면 4개 이상의 QUIESCED SHARE 및 최대 한 개의 QUIESCED UPDATE가 있어야 합니다.

Quiescer는 테이블 스페이스의 상태를 덜 제한적인 상태에서 더 제한적인 상태(예: S에서 U로 또는 U에서 X로)로 업그레이드할 수 있습니다. 사용자가 이미 보유되어 있는 상태보다 더 낮은 상태를 요청하면 원래 상태가 리턴됩니다. 상태는 강등되지 않습니다.

테이블 스페이스의 Quiesce 상태는 SQLU_QUIESCEMODE_RESET을 사용하여 명시적으로 재설정되어야 합니다.

REXX API 구문

```
QUIESCE TABLESPACES FOR TABLE table_name
{SHARE | INTENT TO UPDATE | EXCLUSIVE | RESET}
```

REXX API 매개변수

table_name

시스템 카탈로그에 사용되는 테이블 이름. 이 이름은 마침표(.)로 구분되는 스키마와 테이블 이름이 포함된 두 파트의 이름입니다. 스키마를 제공하지 않으면 CURRENT SCHEMA가 사용됩니다.

제 6 부 REXX에서 DB2 API 호출

SQLDBS 루틴을 사용하여 다음 구문이 포함된 DB2 API를 호출하십시오.

```
CALL SQLDBS 'command string'
```

사용하려는 DB2 API가 SQLDBS 루틴을 사용하여 호출할 수 없는 경우에는 REXX 응용프로그램 내에서 DB2 명령행 처리기(CLP)를 호출하여 API를 호출할 수 있습니다. 그러나 DB2 CLP가 표준 출력 디바이스나 지정한 파일로 직접 출력하기 때문에 REXX 응용프로그램은 호출된 DB2 API에서 출력에 직접 액세스할 수 없고 호출한 API 성공 여부를 판별하기도 쉽지 않습니다. SQLDB2 API는 복합 REXX 변수, SQLCA를 설정하여 각 호출 후에 호출된 각 API 성공 또는 실패 시에 REXX 응용프로그램에 직접 피드백을 제공하는 DB2 CLP에 대한 인터페이스를 제공합니다.

SQLDB2 루틴을 사용하여 다음 구문을 사용하는 DB2 API를 호출할 수 있습니다.

```
CALL SQLDB2 'command string'
```

여기서 'command string'은 명령행 처리기(CLP)로 처리할 수 있는 문자열입니다.

SQLDB2를 사용하여 DB2 API를 호출하는 것은 다음과 같은 점을 제외하고는 CLP를 직접 호출하는 것과 동일합니다.

- CLP 실행 파일 호출은 SQLDB2 호출로 대체됩니다(모든 다른 CLP 옵션 및 매개변수도 동일한 방식으로 지정됨)
- REXX 복합 변수 SQLCA는 SQLDB2 호출 후에 설정되지만 CLP 실행 파일 호출 후에는 설정되지 않습니다.
- SQLDB2 호출 시에는 CLP의 디폴트 출력 표시가 해제로 설정되지만 CLP 실행 파일 호출 시에는 표시가 출력으로 설정됩니다. +o 또는 -o- 옵션을 SQLDB2로 전달하여 CLP의 출력 표시를 설정할 수 있습니다.

SQLDB2 호출 후에 설정되는 유일한 REXX 변수는 SQLCA이기 때문에 SQLCA 이외의 데이터는 리턴하지 않고 현재 SQLDBS 인터페이스를 통해 구현되지 않는 DB2 API를 호출하는 경우에만 이 루틴을 사용합니다. 따라서 SQLDB2에서는 다음 DB2 API만 지원됩니다.

- 데이터베이스 활성화
- 노드 추가
- DB2 버전 1 바인드(1) (2)
- DB2 버전 2 또는 5 바인드(1)
- 노드에서 데이터베이스 작성
- 노드에서 데이터베이스 삭제
- 노드 삭제 검증

- 데이터베이스 비활성화
- 등록 해제
- 로드(3)
- 쿼리 로드
- 프로그램 프리컴파일(1)
- 패키지 리바인드(1)
- 데이터베이스 파티션 그룹 재분배
- 등록
- 데이터베이스 관리 프로그램 시작
- 데이터베이스 관리 프로그램 중지

SQLDB2에서 지원되는 DB2 API에 대한 참고:

1. 이 명령에는 SQLDB2 인터페이스를 사용하는 CONNECT 문이 필요합니다. SQLDB2 인터페이스를 사용하는 연결은 SQLEXEC 인터페이스에 액세스할 수 없으며 SQLEXEC 인터페이스를 사용하는 연결은 SQLDB2 인터페이스에 액세스할 수 없습니다.
2. SQLDB2 인터페이스를 사용하여 Windows 기반 플랫폼에서 지원됩니다.
3. 로드 API를 위한 선택적 출력 매개변수인 `poLoadInfoOut`이 REXX에서 응용프로그램에 리턴되지 않습니다.

주: SQLDB2 루틴은 위에 표시된 DB2 API에서만 사용되도록 제공되지만 SQLDBS 루틴을 통해서 지원되지 않는 기타 DB2 API에도 사용할 수 있습니다. DB2 API는 REXX 응용프로그램 내에서 CLP를 통해 액세스할 수도 있습니다.

제 146 장 분리 레벨 변경

데이터베이스에 액세스하는 동안 DB2가 다른 프로세스에서 데이터를 분리하는 방법을 변경합니다. 이 API는 REXX 응용프로그램에서만 호출할 수 있습니다.

권한 부여

없음

필수 연결

없음

REXX API 구문

```
CHANGE SQLISL TO {RR|CS|UR|RS|NC}
```

REXX API 매개변수

RR 반복 읽기

CS 커서 안정성. 이는 디폴트값입니다.

UR 언커미트 읽기

RS 읽기 안정성

NC 커미트 안함

제 7 부 인다우트(Indoubt) 트랜잭션 관리 API

트랜잭션 관리 프로그램(TM)이 *재동기화* 조치를 수행하기를 기다리지 않고 인다우트(Indoubt) 트랜잭션을 쿼리, 커미트 및 롤백하는 것이 유용한 경우가 있습니다. 예를 들어, 통신 회선이 중단되어 인다우트(Indoubt) 트랜잭션이 필수 자원을 방해하는 경우에 이러한 상황이 발생할 수 있습니다.

데이터베이스 관리자와 같은 자원 소유자가 TM이 *재동기화* 조작을 수행할 때까지 대기할 수 없는 경우 인다우트(Indoubt) 트랜잭션에서 도구 기록기가 경험적 기능을 수행하도록 API 세트가 제공됩니다. 데이터베이스 관리 프로그램의 경우, 이러한 자원에는 테이블과 인덱스에 대한 잠금, 로그 스페이스 및 트랜잭션에 의해 사용된 스토리지 등이 있습니다. 또한 각 인다우트(Indoubt) 트랜잭션은 데이터베이스 관리 프로그램이 처리할 수 있는 최대 동시 트랜잭션 수에서 1씩 감소합니다.

경험적 API로 쿼리, 커미트 및 인다우트(Indoubt) 트랜잭션를 롤백할 수 있으며 로그 레코드를 제거하고 로그 페이지를 릴리스하여 경험적으로 커미트 또는 롤백된 트랜잭션을 취소할 수 있습니다.

경고: 경험적 API를 주의하여 사용하고 마지막 수단으로만 사용하십시오. TM에서 재동기화 이벤트를 실행해야 합니다. TM에 재동기화 조치를 시작하는 연산자 명령이 있는 경우 이를 사용해야 합니다. 사용자가 TM에서 재동기화를 수행하도록 기다릴 수 없는 경우에는 경험적 조치가 필요합니다.

이러한 조치를 수행하도록 지정된 방법은 없으나 다음과 같은 가이드라인이 도움이 될 수 있습니다.

- db2XaListIndTrans 함수를 사용하여 인다우트(Indoubt) 트랜잭션을 표시하십시오. 트랜잭션 상태 = 'P'(준비됨)이며 연결되어 있지 않습니다. *xid*의 부분 *gtrid*는 전역 트랜잭션에 참여하는 기타 자원 관리자(RM)의 트랜잭션 ID와 동일한 전역 트랜잭션 ID입니다.
- 응용프로그램 및 운영 환경 정보를 사용하여 기타 참여 RM을 식별하십시오.
- 트랜잭션 관리 프로그램이 CICS®이고 유일한 RM이 CICS 자원인 경우 경험적 롤백을 수행하십시오.
- 트랜잭션 관리 프로그램이 CICS가 아니면 인다우트(Indoubt) 트랜잭션과 동일한 *gtrid*가 있는 트랜잭션의 상태를 결정할 때 CICS를 사용하십시오.
- 최소 하나의 RM이 커미트되었거나 롤백되었을 경우 경험적 커미트 또는 롤백을 수행하십시오.
- 준비된 상태인 경우 경험적 롤백을 수행하십시오.
- 최소 하나의 RM이 사용 불가능한 경우 경험적 롤백을 수행하십시오.

트랜잭션 관리 프로그램이 사용 가능하고, RM으로 인해 두 번째 단계 또는 재동기화 초기 단계에서 인다우트(Indoubt) 트랜잭션을 사용할 수 없는 경우 DBA는 TM의 로그에서 기타 RM에 대해 수행된 조치를 판별한 후 동일한 조치를 수행해야 합니다. *gtrid*는 TM과 RM 간의 일치 키입니다.

경험적으로 커미트되거나 롤백된 트랜잭션으로 인해 로그가 가득차는 경우가 아니면 sqlxhfrg를 실행하지 마십시오. forget 함수는 이 인다우트(Indoubt) 트랜잭션에서 점유한 로그 스페이스를 사용 가능하게 합니다. 트랜잭션 관리 프로그램이 마침내 이 인다우트(Indoubt) 트랜잭션에 필요한 재동기화 조치를 수행하는 경우, 해당 RM에서 레코드를 찾을 수 없기 때문에 TM이 기타 RM을 커미트하거나 롤백하도록 잘못된 결정을 내릴 수 있습니다. 일반적으로 레코드가 누락된 것은 RM이 롤백되었음을 의미합니다.

제 147 장 db2XaGetInfo - 자원 관리 프로그램에 대한 정보 가져오기

xa_open 호출이 발생하면 특정 자원 관리 프로그램에 대한 정보를 추출합니다.

권한 부여

인스턴스 - SPM 이름 연결

필요한 연결

데이터베이스

API 내장 파일

```
sqlxa.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2XaGetInfo(db2Uint32 versionNumber,
               void * pParmStruct,
               struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2XaGetInfoStruct
{
   db2int32 iRmid;
   struct sqlca         oLastSqlca;
} db2XaGetInfoStruct;
```

db2XaGetInfo API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2XaGetInfoStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2XaGetInfoStruct 데이터 구조 매개변수

iRmid

입력. 정보가 필요한 자원 관리 프로그램을 지정합니다.

oLastSqlca

출력. 마지막 XA API 호출의 sqlca가 포함됩니다.

주: 마지막으로 실패한 XA API에서 작성된 sqlca만 검색할 수 있습니다.

제 148 장 db2XaListIndTrans - 인다우트 트랜잭션 목록

현재 연결된 데이터베이스에 대한 모든 인다우트(Indoubt) 트랜잭션 목록을 제공합니다.

범위

이 API는 발행된 데이터베이스 파티션에만 영향을 줍니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sqlxa.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  db2XaListIndTrans (
       db2Uint32 versionNumber,
       void * pParmStruct,
       struct sqlca * pSqlca);

typedef SQL_STRUCTURE db2XaListIndTransStruct
{
   db2XaRecoverStruct * piIndoubtData;
   db2Uint32 iIndoubtDataLen;
   db2Uint32 oNumIndoubtsReturned;
   db2Uint32 oNumIndoubtsTotal;
   db2Uint32 oReqBufferLen;
} db2XaListIndTransStruct;

typedef SQL_STRUCTURE db2XaRecoverStruct{
   sqluint32 timestamp;
   SQLXA_XID xid;
   char dbalias[SQLXA_DBNAME_SZ];
   char applid[SQLXA_APPLID_SZ];
   char sequence_no[SQLXA_SEQ_SZ];
   char auth_id[SQLXA_USERID_SZ];
   char log_full;
   char connected;
   char indoubt_status;
   char originator;
   char reserved[8];
   sqluint32 rmn;
   rm_entry rm_list[SQLXA_MAX_FedRM];
   } db2XaRecoverStruct;
```

```
typedef SQL_STRUCTURE rm_entry
{
  char name[SQLQG_MAX_SERVER_NAME_LEN];
  SQLXA_XID xid;
} rm_entry;
```

db2XaListIndTrans API 매개변수

versionNumber

입력. 두 번째 매개변수인 pParmStruct로 전달된 구조의 버전 및 릴리스 레벨을 지정합니다.

pParmStruct

입력. db2XaListIndTransStruct 구조의 포인터

pSqlca

출력. sqlca 구조의 포인터

db2XaListIndTransStruct 데이터 구조 매개변수

piIndoubtData

입력. 인다우트(Indoubt) 데이터가 리턴되는 응용프로그램 입력 버퍼의 포인터. 인다우트(Indoubt) 데이터는 db2XaRecoverStruct 형식입니다. 응용프로그램은 이 매개변수에서 제공되는 주소를 시작으로 db2XaRecoverStruct 구조 *크기*를 사용하여 인다우트(Indoubt) 트랜잭션의 목록을 트래버스합니다.

값이 NULL인 경우 DB2는 필요한 버퍼 *크기*를 계산하고 이 값을 oReqBufferLen으로 리턴합니다. oNumIndoubtsTotal에는 총 인다우트(Indoubt) 트랜잭션 수가 포함됩니다. 응용프로그램은 필요한 버퍼 *크기*를 할당하고 API를 다시 발행합니다.

iIndoubtDataLen

입력. piIndoubtData 매개변수로 지시된 버퍼 *크기*(바이트)

oNumIndoubtsReturned

출력. pIndoubtData에서 지정된 버퍼에 리턴되는 인다우트(Indoubt) 트랜잭션 레코드 수

oNumIndoubtsTotal

출력. API 호출 시에 사용 가능한 총 인다우트(Indoubt) 트랜잭션 레코드 수. piIndoubtData 버퍼가 너무 작아서 모든 레코드를 포함하지 못하는 경우 oNumIndoubtsTotal이 oNumIndoubtsReturned에 대한 총계보다 크게 됩니다. 응용프로그램은 모든 레코드를 확보하기 위해 API를 재발행합니다.

주: 이 수는 자동 또는 경험적 인다우트(Indoubt) 트랜잭션 재동기화 결과 또는 인다우트(Indoubt) 상태가 되는 다른 트랜잭션 결과로 API 호출 간에 변경될 수 있습니다.

oReqBufferLen

출력. API 호출 시에 모든 인다우트(Indoubt) 트랜잭션 레코드를 보유하는 데 필요한 버퍼 길이. 응용프로그램은 pIndoubtData를 NULL로 설정하고 API를 호출하여 필요한 버퍼 *크기*를 판별하는 데 이 값을 사용할 수 있습니다. 그러면 이 값을 사용하여 필요한 버퍼를 할당하고 API는 pIndoubtData를 할당된 버퍼 주소로 설정하여 발행할 수 있습니다.

주: 필요한 버퍼 *크기*는 자동 또는 경험적 인다우트(Indoubt) 트랜잭션 재동기화 결과 또는 인다우트(Indoubt) 상태가 되는 다른 트랜잭션 결과로 API 호출 간에 변경될 수 있습니다. 이를 위해 응용프로그램은 더 큰 버퍼를 어카운트에 할당할 수도 있습니다.

db2XaRecoverStruct 데이터 구조 매개변수

timestamp

출력. 트랜잭션이 인다우트(Indoubt) 상태된 시간을 지정합니다.

xid 출력. 트랜잭션 관리 프로그램이 전역 트랜잭션을 고유하게 식별하기 위해 지정하는 XA ID를 지정합니다.

dbalias

출력. 인다우트(Indoubt) 트랜잭션이 있는 데이터베이스 별명을 지정합니다.

applid

출력. 이 트랜잭션을 위해 데이터베이스 관리 프로그램이 지정하는 응용프로그램 ID를 지정합니다.

sequence_no

출력. applid 확장으로 데이터베이스 관리 프로그램이 지정한 시퀀스 번호를 지정합니다.

auth_id

출력. 트랜잭션을 실행한 사용자의 권한 부여 ID를 지정합니다.

log_full

출력. 이 트랜잭션이 로그 가득참 조건을 초래하는지 여부를 표시합니다. 가능한 값은 다음과 같습니다.

SQLXA_TRUE

이 인다우트(Indoubt) 트랜잭션으로 로그 가득참 조건이 발생합니다.

SQLXA_FALSE

이 인다우트(Indoubt) 트랜잭션은 로그 가득참 조건을 초래하지 않습니다.

connected

응용프로그램 연결 여부를 표시합니다.

CONNECTED의 가능한 값(sqlxa에 정의)은 다음과 같습니다.

SQLXA_TRUE

참. 트랜잭션은 정상적인 동기점 처리 중으로 2단계 커미트의 두 번째 단계를 대기 중입니다.

SQLXA_FALSE

거짓. 이전 실패로 인해 트랜잭션이 인다우트(Indoubt) 상태이고 현재 트랜잭션 관리 프로그램에서 재동기화를 대기 중입니다.

indoubt_status

출력. 이 인다우트(Indoubt) 트랜잭션 상태를 표시합니다. 가능한 값은 다음과 같습니다.

- SQLXA_TS_PREP

트랜잭션이 준비되었습니다. 연결된 매개변수를 사용하여 트랜잭션이 일반 커미트 처리의 두 번째 단계를 대기 중인지 또는 오류가 발생하여 트랜잭션 관리 프로그램을 재동기화해야 하는지 판별할 수 있습니다.

- SQLXA_TS_HCOM

트랜잭션이 경험적으로 커미트되었습니다.

- SQLXA_TS_HROL

트랜잭션이 경험적으로 롤백되었습니다.

- SQLXA_TS_MACK

트랜잭션이 파티션된 데이터베이스의 노드에서 커미트 승인이 누락되었습니다.

- SQLXA_TS_END

트랜잭션이 이 데이터베이스에서 종료되었습니다. 이 트랜잭션은 이후에 재활성화, 커미트 또는 롤백될 수도 있습니다. 트랜잭션 관리 프로그램에서 오류가 발생하여 트랜잭션이 완료되지 않을 수도 있습니다. 이 경우 이 트랜잭션은 잠금을 보유하여 다른 응용프로그램이 데이터에 액세스하지 못하도록 하기 때문에 이 트랜잭션에서 경험적 조치를 수행해야 합니다.

시작자 매개변수가 SQLXA_ORIG_FXA 값으로 설정된 경우 indoubt_status 매개변수의 유효한 값(include 디렉토리의 sqlxa.h에 정의)은 다음과 같습니다.

SQLXA_TS_MFCACK

트랜잭션이 하나 이상의 페더레이티드 데이터 소스에서 커미트 승인이 누락되었습니다.

SQLXA_TS_MFRACK

트랜잭션이 하나 이상의 페더레이티드 데이터 소스에서 롤백 승인이 누락되었습니다.

originator

인다우트(Indoubt) 트랜잭션의 원점을 식별합니다.

ORIGINATOR의 가능한 값(include 디렉토리의 sqlxa.h에 정의)은 다음과 같습니다.

SQLXA_ORIG_PE

MPP 환경에서 DB2로 시작된 트랜잭션.

SQLXA_ORIG_XA

XA로 시작된 트랜잭션.

SQLXA_ORIG_FXA

페더레이티드 2단계 커미트 프로세스의 두 번째 단계에서 시작된 트랜잭션. 이 트랜잭션이 2단계 커미트 프로토콜의 두 번째 단계에 진입했지만 하나 이상의 페더레이티드 데이터 소스가 두 번째 단계를 완료할 수 없거나 페더레이티드 서버와 통신할 수 없음을 나타냅니다.

예약 첫 번째 바이트는 인다우트(Indoubt) 트랜잭션 유형을 표시하는 데 사용됩니다. 0은 RM을 나타내고 1은 TM을 표시합니다.

rmn 출력. 트랜잭션 커미트 또는 롤백에 실패한 페더레이티드 데이터 소스 수.

rm_list

출력. 실패한 페더레이티드 데이터 소스 항목 목록으로 각각에는 서버 이름 및 xid가 포함됩니다.

rm_entry 데이터 구조 매개변수

이름 출력. 페더레이티드 데이터 소스 이름

xid 출력. 페더레이티드 트랜잭션을 고유하게 식별하기 위해 페더레이티드 데이터베이스가 페더레이티드 데이터 소스에 지정한 XA ID를 지정합니다.

사용 시 참고사항

SQLXA_MAX_FEDRM은 16으로 정의됩니다. 대부분의 페더레이티드 트랜잭션은 10개 미만의 데이터 소스에 연관됩니다. 16개 이상의 페더레이티드 데이터 소스가 트랜잭션에서 커미트 또는 롤백에 실패하면 이 중 16개만 이 인다우트(Indoubt) 트랜잭션을 위해 db2XaListIndTrans API가 리턴합니다. 페더레이티드되지 않은 인다우트(Indoubt) 트랜잭션의 경우, rmn 매개변수가 0으로 설정되고 이는 인다우트(Indoubt) 트랜잭션이 페더레이티드 데이터 소스와 연관되지 않음을 나타냅니다.

페더레이티드 인다우트(Indoubt) 트랜잭션이 실패한 16개 이상의 페더레이티드 데이터 소스에 연관된 경우 경험적 처리가 호출되면 모든 데이터 소스는(db2XaListIndTrans API로 리턴 여부에 상관 없이) 인다우트(Indoubt) 트랜잭션을 커미트하거나 롤백합니다. 성공적으로 인다우트(Indoubt) 트랜잭션을 커미트 또는 롤백한 모든 페더레이티드 데이터 소스는 페더레이티드 인다우트(Indoubt) 트랜잭션의 실패한 페더레이티드 데이터 소스

목록에서 제거됩니다. db2XaListIndTrans API의 다음 호출에서 인다우트(Indoubt) 트랜잭션의 커미트나 롤백이 여전히 실패한 페더레이티드 데이터 소스만 페더레이티드 인다우트(Indoubt) 트랜잭션 목록에 남게 됩니다.

페더레이티드 인다우트(Indoubt) 트랜잭션에서 데이터 소스 목록을 가져오려면 DB2 버전 9.1 헤더 파일을 사용하여 응용프로그램을 컴파일하고 버전 번호 db2Version900 이상(추후 릴리스용)으로 db2XaListIndTrans API에 전달해야 합니다. 이전 버전으로 전달하면 API는 여전히 인다우트(Indoubt) 트랜잭션 목록을 리턴하지만 페더레이티드 데이터 소스가 실행됩니다. 이와는 상관 없이 응용프로그램에서 사용되는 헤더 파일 버전은 API에 전달된 버전과 동기화되어야 합니다. 그렇지 않을 경우 결과를 예상할 수 없습니다.

일반 응용프로그램은 현재 연결을 데이터베이스나 파티션된 데이터베이스 코디네이터 노드로 설정한 후에 다음 단계를 수행합니다.

1. piIndoubtData를 NULL로 설정하여 db2XaListIndTrans를 호출하십시오. 그러면 값이 oReqBufferLen 및 oNumIndoubtsTotal로 리턴됩니다.
2. oReqBufferLen에 리턴된 값을 사용하여 버퍼를 할당하십시오. oReqBufferLen을 가져오는 이 API의 초기 호출로 인해 추가 인다우트(Indoubt) 트랜잭션이 있는 경우 버퍼 크기가 충분하지 않을 수도 있습니다. 응용프로그램은 oReqBufferLen보다 큰 버퍼를 제공할 수도 있습니다.
3. 모든 인다우트(Indoubt) 트랜잭션 레코드가 확보되었는지 판별하십시오. oNumIndoubtsReturned를 oNumIndoubtsTotal에 비교하여 수행할 수 있습니다. oNumIndoubtsTotal이 oNumIndoubtsReturned보다 큰 경우에는 응용프로그램이 위 단계를 반복할 수 있습니다.

제 149 장 sqlxhfrg - 트랜잭션 상태 무시

자원 관리 프로그램이 경험적으로 완료된 트랜잭션에서 보유하고 있는 자원을 릴리스하도록 허용합니다(즉, 경험적으로 커미트 또는 롤백된 자원). 인다우트(Indoubt) XA 트랜잭션을 경험적으로 커미트 또는 롤백한 후에 이 API를 호출할 수 있습니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sqlxa.h
```

API 및 데이터 구조 구문

```
extern int SQL_API_FN sqlxhfrg(
   SQLXA_XID *pTransId,
   struct sqlca *pSqlca
   );
```

sqlxhfrg API 매개변수

pTransId

입력. 데이터베이스 로그에서 제거되거나 경험적으로 무시되는 트랜잭션의 XA ID

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

경험적으로 커미트 또는 롤백 상태인 트랜잭션만 FORGET 조작이 적용될 수 있습니다.

제 150 장 sqlxphcm - 인다우트 트랜잭션 커미트

인다우트 트랜잭션을 커미트합니다(즉, 커미트를 위해 준비된 트랜잭션). 조작이 성공하면 트랜잭션 상태가 경험적으로 커미트가 됩니다.

범위

이 API는 발행된 노드에만 영향을 줍니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sqlxa.h
```

API 및 데이터 구조 구문

```
extern int SQL_API_FN sqlxphcm(
   int exe_type,
   SQLXA_XID *pTransId,
   struct sqlca *pSqlca
   );
```

sqlxphcm API 매개변수

exe_type

입력. EXE_THIS_NODE가 지정되면 이 노드에서만 조작이 실행됩니다.

pTransId

입력. 경험적으로 커미트되는 트랜잭션의 XA ID

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

준비 상태의 트랜잭션만 커미트됩니다. 경험적으로 커미트되면 데이터베이스 관리 프로그램은 sqlxhfrg API가 호출될 때까지 트랜잭션의 상태를 기억합니다.

제 151 장 sqlxphrl - 인다우트(Indoubt) 트랜잭션 롤백

인다우트(Indoubt) 트랜잭션(즉, 준비된 트랜잭션)을 롤백합니다. 조작이 성공하면 트랜잭션의 상태가 경험적으로 롤백됩니다.

범위

이 API는 발행된 노드에만 영향을 줍니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sqlxa.h
```

API 및 데이터 구조 구문

```
extern int SQL_API_FN sqlxphrl(
   int exe_type,
   SQLXA_XID *pTransId,
   struct sqlca *pSqlca
   );
```

sqlxphrl API 매개변수

exe_type

입력. EXE_THIS_NODE가 지정되면 이 노드에서만 조작이 실행됩니다.

pTransId

입력. 경험적으로 롤백되는 트랜잭션의 XA ID

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

준비 또는 유휴 상태의 트랜잭션만 롤백됩니다. 경험적으로 롤백되면 데이터베이스 관리 프로그램은 sqlxhfrg API가 호출될 때까지 트랜잭션의 상태를 기억합니다.

제 8 부 동시 액세스가 있는 스레드 응용프로그램

제 152 장 sqleAttachToCtx - 컨텍스트에 접속

현재 스레드가 지정한 컨텍스트를 사용하도록 합니다. 이 스레드의 모든 후속 데이터베이스 호출에서 이 컨텍스트를 사용합니다. 둘 이상의 스레드가 지정한 컨텍스트에 접속된 경우에는 이 스레드에 대해 액세스가 직렬화되고 커미트 범위를 공유합니다.

범위

이 API의 범위는 즉시 프로세스로 제한됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleAttachToCtx (
        void * pCtx,
        void * reserved,
        struct sqlca * pSqlca);
```

sqleAttachToCtx API 매개변수

pCtx 입력. sqleBeginCtx로 이전에 할당된 유효한 컨텍스트

예약 나중에 사용하기 위해 예약됩니다. NULL로 설정해야 합니다.

pSqlca
출력. sqlca 구조의 포인터

제 153 장 sqleBeginCtx - 응용프로그램 컨텍스트에 접속 및 작성

응용프로그램 컨텍스트를 작성하거나, 응용프로그램 컨텍스트를 작성하고 접속하십시오. 둘 이상의 응용프로그램 컨텍스트를 작성할 수 있습니다. 각 컨텍스트에는 자체 커미트 범위가 있습니다. 다른 스레드를 다른 컨텍스트에 접속할 수 있습니다(sqleAttachToCtx API 참조). 이런 스레드를 통한 모든 데이터베이스 API 호출은 서로 동기화되지 않습니다.

범위

이 API의 범위는 즉시 프로세스로 제한됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleBeginCtx (
        void ** ppCtx,
        sqlint32 lOptions,
        void * reserved,
        struct sqlca * pSqlca);
```

sqleBeginCtx API 매개변수

ppCtx 출력. 컨텍스트 정보의 스토리지에 대해 전용 메모리에서 할당된 데이터 영역.

lOptions

입력. 가능한 값은 다음과 같습니다.

SQL_CTX_CREATE_ONLY

컨텍스트 메모리가 할당되지만 접속되지는 않습니다.

SQL_CTX_BEGIN_ALL

컨텍스트 메모리는 할당되고 sqleAttachToCtx에 대한 호출이 현재 스레드에 대해 작성됩니다. 이 옵션이 사용되는 경우 ppCtx 매개변수는 NULL일 수 있습니다. 스레드가 이미 컨텍스트에 접속된 경우 호출이 실패합니다.

예약 나중에 사용하기 위해 예약됩니다. NULL로 설정해야 합니다.

pSqlca

출력. sqlca 구조의 포인터

제 154 장 sqleDetachFromCtx - 컨텍스트에서 접속 해제

현재 스레드에서 사용 중인 컨텍스트를 접속 해제합니다. 컨텍스트는 해당 컨텍스트에 대한 접속이 이전에 있었던 경우에만 접속 해제됩니다.

범위

이 API의 범위는 즉시 프로세스로 제한됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleDetachFromCtx (
        void * pCtx,
        void * reserved,
        struct sqlca * pSqlca);
```

sqleDetachFromCtx API 매개변수

pCtx 입력. sqleBeginCtx로 이전에 할당된 유효한 컨텍스트

예약 나중에 사용하기 위해 예약됩니다. NULL로 설정해야 합니다.

pSqlca
출력. sqlca 구조의 포인터

제 155 장 sqleEndCtx - 응용프로그램 컨텍스트에서 접속 해제하여 관련된 메모리 사용 가능

지정된 컨텍스트와 연관된 모든 메모리를 사용 가능화합니다.

범위

이 API의 범위는 즉시 프로세스로 제한됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleEndCtx (
        void ** ppCtx,
        sqlint32 lOptions,
        void * reserved,
        struct sqlca * pSqlca);
```

sqleEndCtx API 매개변수

ppCtx 출력. 사용 가능해진 전용 메모리(컨텍스트 정보 스토리지로 사용)의 데이터 영역.

lOptions

입력. 가능한 값은 다음과 같습니다.

SQL_CTX_FREE_ONLY

이전의 접속 해제가 완료된 경우에만 컨텍스트 메모리를 사용할 수 있게 됩니다.

주: pCtx는 sqleBeginCtx로 이전에 할당된 유효한 컨텍스트여야 합니다.

SQL_CTX_END_ALL

필요한 경우 메모리를 사용 가능화하기 전에 sqleDetachFromCtx를 호출합니다.

주: 컨텍스트가 여전히 사용 중인 경우에도 접속 해제가 완료됩니다. 이 옵션을 사용하는 경우 매개변수는 NULL이 될 수 있지만 전달된 경우 sqleBeginCtx로 이전에 할당된 유효한 컨텍스트여야 합니다. sqleGetCurrentCtx가 호출되고 여기에서 현재 컨텍스트가 사용 가능화됩니다.

예약 나중에 사용하기 위해 예약됩니다. NULL로 설정해야 합니다.

pSqlca

출력. sqlca 구조의 포인터

사용 시 참고사항

데이터베이스 연결이 있거나 다른 스레드에 컨텍스트가 접속되어 있는 경우에는 이 호출이 실패합니다.

주: 컨텍스트가 인스턴스 접속을 설정하는 API를 호출한 경우(예: db2CfgGet) sqleEndCtx를 호출하기 전에 sqledtin을 사용하여 인스턴스에서 접속 해제해야 합니다.

제 156 장 sqleGetCurrentCtx - 현재 컨텍스트 가져오기

스레드와 연관된 현재 컨텍스트를 리턴합니다.

범위

이 API의 범위는 즉시 프로세스로 제한됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleGetCurrentCtx (
        void ** ppCtx,
        void * reserved,
        struct sqlca * pSqlca);
```

sqleGetCurrentCtx API 매개변수

ppCtx 출력. 컨텍스트 정보의 스토리지에 대해 전용 메모리에서 할당된 데이터 영역.

예약 나중에 사용하기 위해 예약됩니다. NULL로 설정해야 합니다.

pSqlca

출력. sqlca 구조의 포인터

제 157 장 sqleInterruptCtx - 컨텍스트 인터럽트

지정한 컨텍스트를 인터럽트합니다.

범위

이 API의 범위는 즉시 프로세스로 제한됩니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleInterruptCtx (
        void * pCtx,
        void * reserved,
        struct sqlca * pSqlca);
```

sqleInterruptCtx API 매개변수

pCtx 입력. sqleBeginCtx로 이전에 할당된 유효한 컨텍스트

예약 나중에 사용하기 위해 예약됩니다. NULL로 설정해야 합니다.

pSqlca
출력. sqlca 구조의 포인터

사용 시 참고사항

처리 중에 이 API는 다음을 수행합니다.

- 전달된 컨텍스트로 전환
- 인터럽트 보내기
- 원래 컨텍스트로 전환
- 종료

제 158 장 sqleSetTypeCtx - 응용프로그램 컨텍스트 유형 설정

응용프로그램 컨텍스트 유형을 설정합니다. 이 API는 응용프로그램 내에서 호출되는 첫 번째 데이터베이스 API여야 합니다.

범위

이 API의 범위는 즉시 프로세스로 제한됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sql.h
```

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN
  sqleSetTypeCtx (
        sqlint32 lOptions);
```

sqleSetTypeCtx API 매개변수

lOptions

입력. 가능한 값은 다음과 같습니다.

SQL_CTX_ORIGINAL

모든 스레드는 동일한 컨텍스트를 사용하여 동시 액세스가 차단됩니다. 이 API 중 하나도 호출되지 않은 경우에는 디폴트입니다.

SQL_CTX_MULTI_MANUAL

모든 스레드는 별도의 컨텍스트를 사용하고 각 스레드의 컨텍스트는 응용프로그램이 관리합니다. 다음을 참조하십시오.

- sqleBeginCtx API
- sqleAttachToCtx API
- sqleDetachFromCtx API
- sqleEndCtx API

이 옵션을 사용하는 경우 다음과 같은 제한사항/변경사항이 적용됩니다.

- 종료가 정상적인 경우 프로세스 종료 시에 자동 COMMIT가 사용 불가능합니다. 모든 미해결 트랜잭션은 롤백되고 모든 COMMIT는 명시적으로 수행되어야 합니다.
- sqleintr API는 모든 컨텍스트를 인터럽트합니다. 특정 컨텍스트를 인터럽트하려면 sqleInterruptCtx를 사용하십시오.

사용 시 참고사항

이 API는 다른 모든 데이터베이스 호출보다 먼저 호출되어야 하며 첫 번째 호출만 유효합니다.

제 9 부 데이터베이스 관리 사용자 정의를 위한 DB2 데이터베이스 시스템 플러그인

사용자 및 다른 업체에서 특정 데이터베이스 관리 기능을 사용자 정의하는 데 사용할 수 있는 플러그인 인터페이스가 DB2 데이터베이스 제품과 같이 제공됩니다.

현재 DB2 데이터베이스 시스템에는 세 가지 유형의 플러그인이 있습니다.

- DB2 데이터베이스 시스템 인증 및 그룹 멤버쉽 찾아보기 동작을 사용자 정의하는 보안 플러그인
- DB2 데이터베이스 시스템에서 제공하는 백업 및 리스토어 기능에서 지원하지 않는 디바이스로 데이터를 백업하고 리스토어하기 위한 백업 및 리스토어 플러그인
- 백업 이미지를 압축하고 압축 해제하기 위한 압축 플러그인

위의 세 플러그인에서 제공되는 기능은 DB2 데이터베이스 시스템 제품에서 제공되지만 DB2 데이터베이스 시스템 작동을 사용자 정의하거나 추가하려는 경우 자체 플러그인을 작성하거나 벤더에서 구입할 수 있습니다.

각 플러그인은 API 및 데이터 구조의 동적으로 로드 가능한 라이브러리입니다. API 및 데이터 구조의 프로토타입은 DB2 데이터베이스 시스템에서 제공되며 구현은 벤더에서 제공됩니다. DB2 데이터베이스 시스템은 일부 API 및 데이터 구조에 대한 구현을 제공합니다. DB2 데이터베이스 시스템에서 구현되는 플러그인 API 및 데이터 구조 목록에 대해서는 각 플러그인 주제를 참조하십시오. 구현은 UNIX 시스템의 공유 라이브러리 및 Windows 플랫폼의 DLL 양식으로 되어 있습니다. DB2 데이터베이스 시스템이 특정 플러그인을 검색하는 실제 위치에 대해서는 각 플러그인 주제를 참조하십시오.

플러그인 API는 두 가지 면에서 DB2 API(예: db2Export, db2Backup)와는 다릅니다. 우선, 대부분의 경우 플러그인 API 구현은 벤더에서 제공합니다. 반면에 DB2 API 구현은 DB2에서 제공합니다. 두 번째로 플러그인 API는 DB2가 호출하는 반면 DB2 API는 사용자가 클라이언트 응용프로그램에서 호출합니다. 따라서 플러그인 API 주제에 입력으로 매개변수가 표시되면 이는 DB2가 매개변수 값을 채우는 것이고 매개변수가 출력으로 표시되는 경우에는 벤더의 API 구현은 매개변수 값을 채워야 합니다.

제 159 장 플러그인 사용

그룹 검색 플러그인 전개

DB2 보안 시스템의 그룹 검색 동작을 사용자 정의하기 위해, 해당 그룹 검색 플러그인을 개발하거나 써드파티로부터 구입할 수 있습니다.

데이터베이스 관리 시스템에 적합한 그룹 검색 플러그인을 확보한 후 이를 전개할 수 있습니다.

- 데이터베이스 서버에 그룹 검색 플러그인을 전개하려면 다음 단계를 수행하십시오.
  1. 그룹 검색 플러그인 라이브러리를 서버의 그룹 플러그인 디렉토리에 복사하십시오.
  2. 데이터베이스 관리 프로그램 구성 매개변수 *group_plugin*을 해당 플러그인의 이름으로 갱신하십시오.
- 데이터베이스 클라이언트에 그룹 검색 플러그인을 전개하려면 다음 단계를 수행하십시오.
  1. 그룹 검색 플러그인 라이브러리를 클라이언트의 그룹 플러그인 디렉토리에 복사하십시오.
  2. 데이터베이스 클라이언트에서, 데이터베이스 관리 프로그램 구성 매개변수 *group_plugin*을 해당 플러그인의 이름으로 갱신하십시오.

사용자 ID/암호 플러그인 전개

DB2 보안 시스템의 사용자 ID/암호 인증 동작을 사용자 정의하기 위해, 해당 사용자 ID/암호 인증 플러그인을 개발하거나 써드파티에서 구입할 수 있습니다.

원하는 용도에 따라, 모든 사용자 ID/암호 기반 인증 플러그인을 클라이언트 플러그인 디렉토리 또는 서버 플러그인 디렉토리에 배치해야 합니다. 플러그인을 클라이언트 플러그인 디렉토리에 배치하는 경우, 클라이언트가 서버에 연결을 시도할 때 로컬 인증 검사 및 클라이언트 유효성 확인에 사용됩니다. 플러그인을 서버 플러그인 디렉토리에 배치하는 경우, 서버의 수신 연결을 처리하고 권한 부여 ID가 존재하는지 여부를 검사하는 데 사용됩니다. 또한 USER 또는 GROUP 키워드를 지정하지 않고 GRANT문을 발행할 때마다 유효합니다. 대부분의 경우, 사용자 ID/암호 인증에는 서버 측 플러그인만 필요합니다. 클라이언트 사용자 ID/암호 플러그인만 사용할 수는 있지만 대개 효율성이 떨어집니다. 일반적으로 클라이언트와 서버에서 사용자 ID/암호 플러그인이 일치해야 합니다.

주: *기존* 플러그인의 *새* 버전을 전개하기 전에 플러그인을 사용하는 모든 응용프로그램 또는 DB2 서버를 중지해야 합니다. 프로세스에서 동일한 이름의 새 버전이 복사될 때 프로세스에서 여전히 플러그인을 사용하고 있으면 트랩을 비롯한 정의되지 않은 동작이 발생합니다. 이 제한사항은 처음으로 플러그인을 전개하거나 플러그인을 사용하고 있지 않을 때는 적용되지 않습니다.

데이터베이스 관리 시스템에 적합한 사용자 ID/암호 인증 플러그인을 확보한 후 이를 전개할 수 있습니다.

- 데이터베이스 서버에 사용자 ID/암호 인증 플러그인을 전개하려면 데이터베이스 서버에서 다음 단계를 수행하십시오.
  1. 사용자 ID/암호 인증 플러그인 라이브러리를 서버 플러그인 디렉토리에 복사하십시오.
  2. 데이터베이스 관리 프로그램 구성 매개변수 *srvcon_pw_plugin*을 서버 플러그인의 이름으로 갱신하십시오. 이 플러그인은 CONNECT 및 ATTACH 요청이 처리될 때 서버에 사용됩니다.
  3. 또는 다음과 같습니다.
     - 데이터베이스 관리 프로그램 구성 매개변수 *srvcon_auth*를 CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT 또는 DATA_ENCRYPT_CMP 인증 유형으로 설정하십시오. 또는:
     - 데이터베이스 관리 프로그램 구성 매개변수 *srvcon_auth*를 NOT_SPECIFIED로 설정하고 *authentication*을 CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT 또는 DATA_ENCRYPT_CMP 인증 유형으로 설정하십시오.
- 데이터베이스 클라이언트에 사용자 ID/암호 인증 플러그인을 전개하려면 각 클라이언트에서 다음 단계를 수행하십시오.
  1. 사용자 ID/암호 인증 플러그인 라이브러리를 클라이언트 플러그인 디렉토리에 복사하십시오.
  2. 데이터베이스 관리 프로그램 구성 매개변수 *clnt_pw_plugin*을 클라이언트 플러그인의 이름으로 갱신하십시오. 이 플러그인은 데이터베이스 관리 프로그램 구성 매개변수 *authentication*이 CLIENT로 설정되어 있을 때뿐만 아니라 인증이 수행되는 위치와 관계없이 로드되고 호출됩니다.
- 사용자 ID/암호 인증 플러그인을 사용하여 클라이언트, 서버 또는 게이트웨이에 로컬 인증을 수행하려면 각 클라이언트, 서버 또는 게이트웨이에서 다음 단계를 수행하십시오.
  1. 사용자 ID/암호 인증 플러그인 라이브러리를 클라이언트, 서버 또는 게이트웨이의 클라이언트 플러그인 디렉토리에 복사하십시오.
  2. 데이터베이스 관리 프로그램 구성 매개변수 *clnt_pw_plugin*을 해당 플러그인의 이름으로 갱신하십시오.

3. 데이터베이스 관리 프로그램 구성 매개변수 *authentication*을 CLIENT, SERVER, SERVER_ENCRYPT, DATA_ENCRYPT 또는 DATA_ENCRYPT_CMP로 설정하십시오.

GSS-API 플러그인 전개

DB2 보안 시스템의 인증 동작을 사용자 정의하기 위해, GSS-API를 사용하여 해당 인증 플러그인을 개발하거나 써드파티에서 구입할 수 있습니다.

플러그인 유형이 Kerberos가 아닌 경우, 클라이언트 및 서버의 플러그인 이름이 일치하고 플러그인 유형이 동일해야 합니다. 클라이언트 및 서버의 플러그인을 동일한 벤더가 제공할 필요는 없지만, 호환 가능한 GSS-API 토큰을 생성하고 사용해야 합니다. Kerberos 플러그인은 표준화되어 있으므로 클라이언트 및 서버에 전개된 모든 Kerberos 플러그인을 조합할 수 있습니다. 하지만 표준화되지 않은 다른 GSS-API를 구현하는 경우(예: *x.509* 증명서), DB2 데이터베이스 시스템과 부분적으로만 호환할 수 있습니다. 원하는 용도에 따라, 모든 GSS-API 인증 플러그인을 클라이언트 플러그인 디렉토리 또는 서버 플러그인 디렉토리에 배치해야 합니다. 플러그인을 클라이언트 플러그인 디렉토리에 배치하는 경우, 클라이언트가 서버에 연결을 시도할 때 로컬 인증 검사 및 클라이언트 유효성 확인에 사용됩니다. 플러그인을 서버 플러그인 디렉토리에 배치하는 경우, 서버의 수신 연결을 처리하고 권한 부여 ID가 존재하는지 여부를 검사하는 데 사용됩니다. 또한 USER 또는 GROUP 키워드를 지정하지 않고 GRANT문을 발행할 때마다 유효합니다.

주: *기존* 플러그인의 *새* 버전을 전개하기 전에 플러그인을 사용하는 모든 응용프로그램 또는 DB2 서버를 중지해야 합니다. 프로세스에서 동일한 이름의 새 버전이 복사될 때 프로세스에서 여전히 플러그인을 사용하고 있으면 트랩을 비롯한 정의되지 않은 동작이 발생합니다. 이 제한사항은 처음으로 플러그인을 전개하거나 플러그인을 사용하고 있지 않을 때는 적용되지 않습니다.

데이터베이스 관리 시스템에 적합한 GSS-API 인증 플러그인을 확보한 후 이를 전개할 수 있습니다.

- 데이터베이스 서버에 GSS-API 인증 플러그인을 전개하려면 서버에서 다음 단계를 수행하십시오.
  1. GSS-API 인증 플러그인 라이브러리를 서버 플러그인 디렉토리에 복사하십시오. 이 디렉토리에 다수의 GSS-API 플러그인을 복사할 수 있습니다.
  2. 데이터베이스 관리 프로그램 구성 매개변수 *srvcon_gssplugin_list*를 GSS-API 플러그인 디렉토리에 설치된 플러그인의 순서 지정되어 있고 쉼표로 구분된 이름 목록으로 갱신하십시오.
  3. 또는 다음과 같습니다.

– 데이터베이스 관리 프로그램 구성 매개변수 *srvcon_auth*를 GSSPLUGIN 또는 GSS_SERVER_ENCRYPT로 설정하면 서버가 GSSAPI PLUGIN 인증 방법을 사용할 수 있습니다. 또는:

– 데이터베이스 관리 프로그램 구성 매개변수 *srvcon_auth*를 NOT_SPECIFIED으로 설정하고 *authentication*을 GSSPLUGIN 또는 GSS_SERVER_ENCRYPT로 설정하면 서버가 GSSAPI PLUGIN 인증 방법을 사용할 수 있습니다..

- 데이터베이스 클라이언트에 GSS-API 인증 플러그인을 전개하려면 각 클라이언트에서 다음 단계를 수행하십시오.
  1. GSS-API 인증 플러그인 라이브러리를 클라이언트 플러그인 디렉토리에 복사하십시오. 이 디렉토리에 다수의 GSS-API 플러그인을 복사할 수 있습니다.클라이언트는 클라이언트에서 사용할 수 있는 서버의 플러그인에 포함된 첫 번째 GSS-API 플러그인을 선택하여 CONNECT 또는 ATTACH 조작 중에 인증에 사용할 GSS-API 플러그인을 선택합니다.
  2. 선택사항: 클라이언트가 액세스하는 데이터베이스를 카탈로그화하여 클라이언트가 GSS-API 인증 플러그인만 인증 메커니즘으로 허용함을 나타낼 수 있습니다. 예를 들어, 다음과 같습니다.

     ```
     CATALOG DB testdb AT NODE testnode AUTHENTICATION GSSPLUGIN
     ```

- GSS-API 인증 플러그인을 사용하여 클라이언트, 서버 또는 게이트웨이에 로컬 인증을 수행하려면 다음 단계를 수행하십시오.
  1. GSS-API 인증 플러그인 라이브러리를 클라이언트, 서버 또는 게이트웨이의 클라이언트 플러그인 디렉토리에 복사하십시오.
  2. 데이터베이스 관리 프로그램 구성 매개변수 *local_gssplugin*을 해당 플러그인의 이름으로 갱신하십시오.
  3. 데이터베이스 관리 프로그램 구성 매개변수 *authentication*을 GSSPLUGIN 또는 GSS_SERVER_ENCRYPT로 설정하십시오.

Kerberos 플러그인 전개

DB2 보안 시스템의 Kerberos 인증 동작을 사용자 정의하기 위해, 해당 Kerberos 인증 플러그인을 개발하거나 써드파티에서 구입할 수 있습니다. Kerberos 보안 플러그인은 IPv6를 지원하지 않습니다.

주: *기존* 플러그인의 *새* 버전을 전개하기 전에 플러그인을 사용하는 모든 응용프로그램 또는 DB2 서버를 중지해야 합니다. 프로세스에서 동일한 이름의 새 버전이 복사될 때 프로세스에서 여전히 플러그인을 사용하고 있으면 트랩을 비롯한 정의되지 않은 동작이 발생합니다. 이 제한사항은 처음으로 플러그인을 전개하거나 플러그인을 사용하고 있지 않을 때는 적용되지 않습니다.

데이터베이스 관리 시스템에 적합한 Kerberos 인증 플러그인을 확보한 후 이를 전개할 수 있습니다.

- 데이터베이스 서버에 Kerberos 인증 플러그인을 전개하려면 서버에서 다음 단계를 수행하십시오.
  1. Kerberos 인증 플러그인 라이브러리를 서버 플러그인 디렉토리에 복사하십시오.
  2. 순서 지정되어 있고 쉼표로 구분된 목록으로 제공되는 데이터베이스 관리 프로그램 구성 매개변수 **srvcon_gssplugin_list**를 갱신하여 Kerberos 서버 플러그인 이름을 포함하십시오. 이 목록에 있는 플러그인 하나만 Kerberos 플러그인이 될 수 있습니다. 이 목록이 비어 있고 **authentication**이 KERBEROS 또는 KRB_SVR_ENCRYPT로 설정되어 있으면, 디폴트 DB2 Kerberos 플러그인 IBMkrb5가 사용됩니다.
  3. 필요한 경우, 현재 인증 유형을 겹쳐쓰도록 **srvcon_auth** 데이터베이스 관리 프로그램 구성 매개변수를 설정하십시오. **srvcon_auth** 데이터베이스 관리 프로그램 구성 매개변수가 설정되어 있지 않으면, DB2 데이터베이스 관리 프로그램이 **authentication** 구성 매개변수 값을 사용합니다. **authentication** 구성 매개변수가 현재 다음 인증 유형 중 하나로 설정되어 있으면 Kerberos 플러그인을 전개하여 사용할 수 있습니다.
     - KERBEROS
     - KRB_SERVER_ENCRYPT
     - GSSPLUGIN
     - GSS_SERVER_ENCRYPT

     현재 인증 유형을 겹쳐써야 하는 경우, **srvcon_auth** 구성 매개변수를 다음 인증 유형 중 하나로 설정하십시오.
     - KERBEROS
     - KRB_SERVER_ENCRYPT
     - GSSPLUGIN
     - GSS_SERVER_ENCRYPT
- 데이터베이스 클라이언트에 Kerberos 인증 플러그인을 전개하려면 각 클라이언트에서 다음 단계를 수행하십시오.
  1. Kerberos 인증 플러그인 라이브러리를 클라이언트 플러그인 디렉토리에 복사하십시오.
  2. 데이터베이스 관리 프로그램 구성 매개변수 **clnt_krb_plugin**을 Kerberos 플러그인의 이름으로 갱신하십시오. **clnt_krb_plugin**이 비어 있으면 DB2는 클라이언트가 Kerberos 인증을 사용할 수 없다고 가정합니다. 이 설정은 서버가 플러그인을 지원할 수 없을 때만 적합합니다. 서버 및 클라이언트 둘 다 보안 플러그인을 지원하는 경우, 클라이언트 값 **clnt_krb_plugin**보다 우선하여 디폴트 서버

플러그인 *IBMkrb5*가 사용됩니다. Kerberos 인증 플러그인을 사용하여 클라이언트, 서버 또는 게이트웨이에 로컬 인증을 수행하려면 다음 단계를 수행하십시오.

a. Kerberos 인증 플러그인 라이브러리를 클라이언트, 서버 또는 게이트웨이의 클라이언트 플러그인 디렉토리에 복사하십시오.

b. 데이터베이스 관리 프로그램 구성 매개변수 **clnt_krb_plugin**을 해당 플러그인의 이름으로 갱신하십시오.

c. 데이터베이스 관리 프로그램 구성 매개변수 **authentication**을 KERBEROS 또는 KRB_SERVER_ENCRYPT로 설정하십시오.

3. 선택사항: 클라이언트가 액세스하는 데이터베이스를 카탈로그화하여 클라이언트가 Kerberos 인증 플러그인만 사용함을 나타낼 수 있습니다. 예를 들어, 다음과 같습니다.

```
CATALOG DB testdb AT NODE testnode AUTHENTICATION KERBEROS
        TARGET PRINCIPAL service/host@REALM
```

주: Kerberos를 지원하는 플랫폼의 경우, IBMkrb5 라이브러리는 클라이언트 플러그인 디렉토리에 제공됩니다. Kerberos 플러그인은 GSS-API 플러그인을 사용하여 구현되므로 DB2 데이터베이스 관리 프로그램은 이 라이브러리를 유효한 GSS-API 플러그인으로 인식합니다.

제 160 장 보안 플러그인 쓰기

DB2의 보안 플러그인 로드 방식

DB2 데이터베이스 시스템이 보안 플러그인 함수를 호출하는 데 필요한 정보를 포함하게 하려면 보안 플러그인에 초기화 함수가 올바르게 설정되어 있어야 합니다.

각 플러그인 라이브러리는 플러그인 유형으로 판별되는 특정 이름이 포함된 초기화 함수를 포함해야 합니다.

- 서버 측 인증 플러그인: `db2secServerAuthPluginInit()`
- 클라이언트 측 인증 플러그인: `db2secClientAuthPluginInit()`
- 그룹 플러그인: `db2secGroupPluginInit()`

이 함수를 플러그인 초기화 함수라고 합니다. 플러그인 초기화 함수는 지정된 플러그인을 초기화하며 플러그인 함수를 호출하는 데 필요한 정보를 DB2에 제공합니다. 플러그인 초기화 함수에서 사용할 수 있는 매개변수는 다음과 같습니다.

- 플러그인을 호출하는 DB2 인스턴스가 지원할 수 있는 함수 포인터 구조의 가장 높은 버전 번호
- 구현이 필요한 모든 API에 대한 포인터를 포함하는 구조에 대한 포인터
- db2diag 로그 파일에 로그 메시지를 추가하는 함수에 대한 포인터
- 오류 메시지 문자열에 대한 포인터
- 오류 메시지의 길이

다음은 그룹 검색 플러그인의 초기화 함수에 대한 함수 서명입니다.

```
SQL_API_RC SQL_API_FN db2secGroupPluginInit(
  db2int32 version,
  void *group_fns,
  db2secLogMessage *logMessage_fn,
  char **errormsg,
  db2int32  *errormsglen);
```

주: 플러그인 라이브러리가 C++로 컴파일된 경우 `extern "C"`로 모든 함수를 선언해야 합니다. DB2에서는 기본 운영 체제 플러그인 동적 로더를 사용하여 C++ 사용자 작성 플러그인 라이브러리 내에 사용된 C++ 컨스트럭터와 디스트럭터를 처리합니다.

초기화 함수는 플러그인 라이브러리에서 유일하게 규정된 함수 이름을 사용하는 함수입니다. 다른 플러그인 함수는 초기화 함수에서 리턴된 함수 포인트를 통해 참조됩니다. 서버 플러그인은 DB2 서버가 시작될 때 로드되고, 클라이언트 플러그인은 클라이언트

에서 필요할 때 로드됩니다. DB2에서는 플러그인 라이브러리를 로드하는 즉시 초기화 함수의 위치를 분석하여 이를 호출합니다. 이 함수의 구체적인 태스크는 다음과 같습니다.

- 함수 포인터를 해당 함수 구조에 대한 포인터로 캐스트합니다.
- 라이브러리에 다른 함수에 대한 포인터를 채웁니다.
- 리턴되는 함수 포인터 구조의 버전 번호를 채웁니다.

DB2는 플러그인 초기화 함수를 여러 번 호출할 수도 있습니다. 이러한 상황은 응용프로그램이 DB2 클라이언트 라이브러리를 동적으로 로드, 언로드 및 재로드했는데 재로드하기 이전과 이후에 플러그인의 인증 함수를 수행하는 경우에 발생할 수 있습니다. 이 경우 플러그인 라이브러리가 언로드된 다음 재로드되지 않을 수 있습니다. 그러나 이 동작은 운영 체제에 따라 다릅니다.

플러그인 초기화 함수를 여러 번 호출하는 DB2의 또 다른 경우는 스토어드 프로시저 또는 페더레이티드 시스템 호출 실행 중에 발생합니다. 이때 데이터베이스 서버 자체는 클라이언트로 사용될 수 있습니다. 데이터베이스 서버에서 클라이언트 및 서버 플러그인이 같은 파일에 있는 경우, DB2에서 플러그인 초기화 함수를 두 번 호출할 수 있습니다.

플러그인에서 db2secGroupPluginInit가 여러 번 호출되었음을 발견한 경우, 이 이벤트를 종료한 다음 플러그인 라이브러리를 다시 초기화합니다. 이와 같이 플러그인 초기화 함수는 함수 포인터 세트를 다시 리턴하기 전에 db2secPluginTerm에 대한 호출이 수행하는 전체 정리 태스크를 수행해야 합니다.

UNIX 또는 Linux 기반 운영 체제에서 실행 중인 DB2 서버에서 DB2는 여러 프로세스에서 플러그인 라이브러리를 여러 번 로드하고 초기화할 수도 있습니다.

보안 플러그인 라이브러리 개발의 제한사항

플러그인 라이브러리 개발 방법에 영향을 주는 특정 제한사항이 있습니다.

다음은 플러그인 라이브러리 개발의 제한사항입니다.

C-연계 플러그인 라이브러리는 C-연계와 연결되어야 합니다. 프로토타입을 제공하는 헤더 파일, 플러그인을 구현하는 데 필요한 데이터 구조, 오류 코드 정의는 C/C++에만 제공됩니다. DB2가 로드 시간에 해결하는 함수는 외부 "C"로 선언되어야 합니다(플러그인 라이브러리가 C++로 컴파일되는 경우).

.NET 일반 언어 런타임이 지원되지 않음
플러그인 라이브러리의 소스 코드 컴파일 및 연결에 .NET 일반 언어 런타임(CLR)이 지원되지 않습니다.

신호 핸들러
플러그인 라이브러리가 신호 핸들러를 설치하거나 신호 마스크를 변경하지 않

아야 합니다. DB2의 신호 핸들러를 방해하기 때문입니다. DB2 신호 핸들러를 방해하면 플러그인 코드 자체의 트랩을 포함하여 오류를 보고하고 복구하는 DB2의 기능에 심각한 방해가 될 수 있습니다. DB2의 오류 처리에 방해가 될 수 있으므로 플러그인 라이브러리에서도 C++ 예외가 발생하지 않아야 합니다.

스레드 안전

플러그인 라이브러리는 스레드 안전해야 하며 재진입 가능해야 합니다. 플러그인 초기화 기능은 다시 입력할 필요가 없는 유일한 API입니다. 플러그인 초기화 기능은 다른 프로세스에서 여러 번 호출될 수 있습니다. 이 경우, 플러그인에서 사용된 모든 자원이 정리되고 플러그인 자체가 다시 초기화됩니다.

Exit 핸들러 및 표준 C 라이브러리 및 운영 체제 호출 겹쳐쓰기

플러그인 라이브러리는 표준 C 라이브러리 또는 운영 체제 호출을 겹쳐쓰지 않아야 합니다. 또한 플러그인 라이브러리는 exit 핸들러 또는 `pthread_atfork` 핸들러를 설치하지 않아야 합니다. 프로그램을 종료하기 전에는 exit 핸들러를 언로드할 수 없으므로 이 핸들러를 사용하지 않는 것이 좋습니다.

라이브러리 종속성

Linux 또는 UNIX에서, 플러그인 라이브러리를 로드하는 프로세스는 `setuid` 또는 `setgid` 중 하나입니다. 즉 `$LD_LIBRARY_PATH`, `$SHLIB_PATH` 또는 `$LIBPATH` 환경 변수를 사용하여 종속 라이브러리를 찾을 수 없음을 의미합니다. 따라서 다음과 같은 다른 방법을 통해 종속 라이브러리에 액세스할 수 없는 경우를 제외하고 플러그인 라이브러리는 추가 라이브러리를 사용할 수 없습니다.

- `/lib` 또는 `/usr/lib`에 배치하여
- 지정되는 OS에 상주하는 디렉토리 사용하여 (예: Linux의 `ld.so.conf` 파일)
- 플러그인 라이브러리 자체의 `RPATH`에 지정하여

이 제한사항은 Windows 운영 체제에는 적용되지 않습니다.

기호 충돌

가능하면, 기호 충돌 가능성을 줄일 수 있는 옵션(예: 바인딩되지 않은 외부 기호 참조를 줄이는 옵션)을 사용하여 플러그인 라이브러리를 컴파일하고 연결해야 합니다. 예를 들어, HP, Solaris 및 Linux에서 ″-Bsymbolic″ 링커 옵션을 사용하면 기호 충돌과 관련된 문제점을 방지할 수 있습니다. 하지만 AIX에 작성된 플러그인에는 `"-brtl"` 링커 옵션을 명시적 또는 내재적으로 사용하지 마십시오.

32비트 및 64비트 응용프로그램

32비트 응용프로그램은 32비트 플러그인을 사용해야 하며 64비트 응용프로그램은 64비트 플러그인을 사용해야 합니다. 자세한 정보는 32비트 및 62비트 고려사항에 대한 주제를 참조하십시오.

텍스트 문자열

입력 텍스트 문자열이 널(null) 종료된다고 보장되지 않으며 출력 문자열은 널(null) 종료될 필요가 없습니다. 대신, 모든 입력 문자열에 정수 길이가 지정되고 리턴될 길이에 정수의 포인터가 지정됩니다.

권한 부여 ID 매개변수 전달

DB2가 플러그인에 전달하는 권한 부여 ID(권한 ID) 매개변수(입력 권한 ID 매개변수)에 채워진 공백이 제거된 상태의 대문자 권한 ID가 포함됩니다. 플러그인이 DB2에 리턴하는 권한 ID 매개변수(출력 권한 ID 매개변수)는 특수 처리가 필요하지 않지만, DB2가 내부 DB2 표준에 따라 권한 ID를 대문자로 포함시키고 공백으로 채웁니다.

매개변수 *크기* 한계

플러그인 API에서 사용하는 매개변수의 길이 한계는 다음과 같습니다.

```
#define DB2SEC_MAX_AUTHID_LENGTH 255
#define DB2SEC_MAX_USERID_LENGTH 255
#define DB2SEC_MAX_USERNAMESPACE_LENGTH 255
#define DB2SEC_MAX_PASSWORD_LENGTH 255
#define DB2SEC_MAX_DBNAME_LENGTH 128
```

특정 플러그인을 구현하려면 권한 부여 ID, 사용자 ID 및 암호의 최대 길이가 작아야 하거나 작게 설정해야 합니다. 특히, DB2 데이터베이스 시스템에 제공되는 운영 체제 인증 플러그인은 운영 체제 한계가 위에 설명한 한계보다 작은 경우 운영 체제에서 강제 설정된 최대 사용, 그룹 및 이름 스페이스 길이 한계로 제한됩니다.

AIX의 보안 플러그인 라이브러리 확장자

AIX 시스템에서, 보안 플러그인 라이브러리에 확장자가 *.a* 또는 *.so*인 파일 이름을 사용할 수 있습니다. 플러그인 라이브러리를 로드하는 데 사용되는 메커니즘은 사용되는 확장자에 따라 다릅니다.

- 확장자가 *.a*인 파일 이름이 있는 플러그인 라이브러리는 공유 오브젝트 구성원이 포함된 아카이브로 간주됩니다. 이러한 구성원의 이름은 *shr.o*(32비트) 또는 *shr64.o*(64비트)로 지정해야 합니다. 아카이브 하나에 32비트 및 64비트 구성원 둘 다 포함할 수 있으므로, 두 가지 유형의 플랫폼에 전개할 수 있습니다.

  예를 들어, 32비트 아카이브 스타일 플러그인 라이브러리를 빌드하려면 다음과 같이 구성하십시오.

  ```
  xlc_r -qmkshrobj -o shr.o MyPlugin.c -bE:MyPlugin.exp
  ar rv MyPlugin.a shr.o
  ```

- 확장자가 *.so*인 파일 이름이 있는 플러그인 라이브러리는 동적으로 로드할 수 있는 공유 오브젝트로 간주됩니다. 이러한 오브젝트는 빌드 시 사용된 컴파일러 또는 링커 옵션에 따라 32비트 또는 64비트입니다. 예를 들어, 32비트 플러그인 라이브러리를 빌드하려면 다음과 같이 구성하십시오.

```
xlc_r -qmkshrobj -o MyPlugin.so MyPlugin.c -bE:MyPlugin.exp
```

AIX를 제외한 모든 플랫폼에서, 보안 플러그인 라이브러리는 항상 동적으로 로드할 수 있는 공유 오브젝트로 간주됩니다.

보안 플러그인에 대한 제한사항

보안 플러그인 사용과 관련하여 몇 가지 제한사항이 있습니다.

DB2 데이터베이스 계열 지원 제한사항

GSS-API 플러그인을 사용하여 Linux, UNIX 및 Windows에 설치된 DB2 클라이언트와 다른 DB2 계열 서버(예: z/OS용 DB2) 간의 연결을 인증할 수 없습니다. 또한 클라이언트로 사용되는 다른 DB2 데이터베이스 계열 제품과 Linux, UNIX 또는 Windows에 설치된 DB2 서버 간의 연결도 인증할 수 없습니다.

Linux, UNIX 또는 Windows에서 DB2 클라이언트를 사용하여 다른 DB2 데이터베이스 계열 서버에 연결하는 경우, 클라이언트 측 사용자 ID/암호 플러그인(예: IBM에서 제공하는 운영 체제 인증 플러그인)을 사용하거나 고유한 사용자 ID/암호 플러그인을 작성할 수 있습니다. 내장 Kerberos 플러그인을 사용하거나 고유한 플러그인을 구현할 수도 있습니다.

Linux, UNIX 또는 Windows에서 DB2 클라이언트를 사용할 경우에는 GSSPLUGIN 인증 유형을 사용하여 데이터베이스를 카탈로그해서는 안됩니다.

AUTHID 식별자에 대한 제한사항 DB2 데이터베이스 시스템의 버전 9.5 이상에서는 128바이트의 권한 부여 ID를 사용할 수 있지만, 권한 부여 ID가 운영 체제 사용자 ID 또는 그룹 이름으로 해석되는 경우 운영 체제 이름 지정 제한사항이 적용됩니다(예: 8 또는 30자 사용자 ID 및 30자 그룹 이름 제한). 따라서 128바이트의 권한 부여 ID를 부여할 수는 있지만 해당 권한 부여 ID를 사용하는 사용자로 연결할 수는 없습니다. 사용자 고유의 보안 플러그인을 작성하는 경우에는 확장된 크기의 권한 부여 ID를 사용할 수 있습니다. 예를 들어, 보안 플러그인에 30바이트 사용자 ID를 제공할 수 있으며 이 보안 플러그인은 연결 시 사용할 수 있는 128바이트 권한 부여 ID를 인증 중에 리턴할 수 있습니다.

InfoSphere™ Federation Server 지원 제한사항

DB2 II에서는 GSS_API 플러그인의 위임된 증명서를 사용하여 데이터 소스에 대한 아웃바운드 연결을 설정할 수 없습니다. 데이터 소스에 대한 연결은 계속 CREATE USER MAPPING 명령을 사용해야 합니다.

Database Administration Server 지원 제한사항

DAS(DB2 Administration Server)에서는 보안 플러그인이 지원되지 않습니다. DAS는 운영 체제 인증 메커니즘만 지원합니다.

DB2 클라이언트(Windows용) 보안 플러그인 문제점 및 제한사항

Windows 운영 체제의 DB2 클라이언트에 배치될 보안 플러그인을 개발하는 경우 플러그인 종료 함수에서 보조 라이브러리를 언로드하지 마십시오. 이 제한사항은 그룹, 사용자 ID 및 암호, Kerberos, GSS-API 플러그인 등 모든 유형의 클라이언트 보안 플러그인에 적용됩니다. 이러한 종료 API(예: db2secPluginTerm, db2secClientAuthPluginTerm 및 db2secServerAuthPluginTerm)는 Windows 플랫폼에서 호출되지 않으므로, 적절히 자원을 정리해야 합니다.

이 제한사항은 Windows에서 DLL 언로드와 연관된 정리 문제와 관련이 있습니다.

AIX에서 확장자가 .a 또는 .so인 플러그인 라이브러리 로드

AIX에서 보안 플러그인 라이브러리의 파일 이름 확장자는 .a 또는 .so입니다. 플러그인 라이브러리를 로드하는 데 사용되는 메커니즘은 사용되는 확장자에 따라 다릅니다.

- 파일 이름 확장자가 .a인 플러그인 라이브러리

  파일 이름 확장자가 .a인 플러그인 라이브러리는 공유 오브젝트 구성원을 포함하는 아카이브로 간주됩니다. 이러한 구성원의 이름은 shr.o(32비트) 또는 shr64.o(64비트)여야 합니다. 단일 아카이브는 32비트와 64비트 구성원을 모두 포함할 수 있으므로 두 유형의 플랫폼에 전개할 수 있습니다.

  예를 들어, 32비트 아카이브 스타일 플러그인 라이브러리를 빌드하려면 다음과 같이 구성하십시오.

  ```
  xlc_r -qmkshrobj -o shr.o MyPlugin.c -bE:MyPlugin.exp
  ar rv MyPlugin.a shr.o
  ```

- 파일 이름 확장자가 .so인 플러그인 라이브러리

  파일 이름 확장자가 .so인 플러그인 라이브러리는 동적으로 로드 가능한 공유 오브젝트로 간주됩니다. 이러한 오브젝트는 작성 시 사용된 컴파일러 및 링커 옵션에 따라 32비트 또는 64비트입니다. 예를 들어, 32비트 플러그인 라이브러리를 작성하려면 다음을 실행하십시오.

  ```
  xlc_r -qmkshrobj -o MyPlugin.so MyPlugin.c -bE:MyPlugin.exp
  ```

AIX를 제외한 모든 플랫폼에서는 보안 플러그인 라이브러리가 항상 동적으로 로드 가능한 공유 오브젝트로 간주됩니다.

GSS-API 보안 플러그인은 메시지 암호화 및 서명을 지원하지 않음

GSS-API 보안 플러그인에서는 메시지 암호화 및 서명을 사용할 수 없습니다.

보안 플러그인의 리턴 코드

모든 보안 플러그인 API는 API 실행의 성공 또는 실패를 나타내는 정수 값을 리턴해야 합니다. 리턴 코드 값 0은 API 실행에 성공했음을 나타냅니다. -3, -4 및 -5를 제외한 모든 음수 리턴 코드는 API에 오류가 발생했음을 나타냅니다.

리턴 코드 -3, -4 또는 -5를 제외하고 보안 플러그인 API에서 리턴된 모든 음수 리턴 코드는 SQLCODE -1365, SQLCODE -1366 또는 SQLCODE -30082에 맵핑됩니다. -3, -4 및 -5 값은 권한 부여 ID가 유효한 사용자 또는 그룹을 나타내는지 여부를 표시하는 데 사용됩니다.

모든 보안 플러그인 API 리턴 코드는 SQLLIB/include 디렉토리를 포함하여 DB2에서 볼 수 있는 db2secPlugin.h에 정의됩니다.

다음 표에서 모든 보안 플러그인 리턴 코드에 대한 세부사항을 볼 수 있습니다.

표 10. 보안 플러그인 리턴 코드

리턴 코드	값 정의	의미	적용 가능한 API
0	DB2SEC_PLUGIN_OK	플러그인 API 실행에 성공했습니다.	모두
-1	DB2SEC_PLUGIN_UNKNOWNERROR	플러그인 API에 예기치 않은 오류가 발생했습니다.	모두
-2	DB2SEC_PLUGIN_BADUSER	입력하여 제공된 사용자 ID가 정의되지 않았습니다.	db2secGenerateInitialCred db2secValidatePassword db2secRemapUserid db2secGetGroupsForUser
-3	DB2SEC_PLUGIN_INVALIDUSERORGROUP	해당 사용자 또는 그룹이 없습니다.	db2secDoesAuthIDExist db2secDoesGroupExist
-4	DB2SEC_PLUGIN_USERSTATUSNOTKNOWN	알 수 없는 사용자 상태입니다. 이는 DB2에서 오류로 취급되지 않습니다. GRANT문에 사용되어 권한 ID가 사용자 또는 운영 체제 그룹을 나타내는지 판별하는 데 사용됩니다.	db2secDoesAuthIDExist
-5	DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN	알 수 없는 그룹 상태입니다. 이는 DB2에서 오류로 취급되지 않습니다. GRANT문에 사용되어 권한 ID가 사용자 또는 운영 체제 그룹을 나타내는지 판별하는 데 사용됩니다.	db2secDoesGroupExist

표 10. 보안 플러그인 리턴 코드 (계속)

리턴 코드	값 정의	의미	적용 가능한 API
-6	DB2SEC_PLUGIN_UID_EXPIRED	만기된 사용자 ID입니다.	db2secValidatePassword db2GetGroupsForUser db2secGenerateInitialCred
-7	DB2SEC_PLUGIN_PWD_EXPIRED	만기된 암호입니다.	db2secValidatePassword db2GetGroupsForUser db2secGenerateInitialCred
-8	DB2SEC_PLUGIN_USER_REVOKED	권한 취소된 사용자입니다.	db2secValidatePassword db2GetGroupsForUser
-9	DB2SEC_PLUGIN _USER_SUSPENDED	일시중단된 사용자입니다.	db2secValidatePassword db2GetGroupsForUser
-10	DB2SEC_PLUGIN_BADPWD	잘못된 암호입니다.	db2secValidatePassword db2secRemapUserid db2secGenerateInitialCred
-11	DB2SEC_PLUGIN _BAD_NEWPASSWORD	잘못된 새 암호입니다.	db2secValidatePassword db2secRemapUserid
-12	DB2SEC_PLUGIN _CHANGEPASSWORD _NOTSUPPORTED	암호 변경이 지원되지 않습니다.	db2secValidatePassword db2secRemapUserid db2secGenerateInitialCred
-13	DB2SEC_PLUGIN_NOMEM	메모리가 부족하여 플러그인의 메모리 할당 시도에 실패했습니다.	모두
-14	DB2SEC_PLUGIN_DISKERROR	플러그인에 디스크 오류가 발생했습니다.	모두
-15	DB2SEC_PLUGIN_NOPERM	파일에 잘못된 권한이 있기 때문에 플러그인의 파일 액세스 시도에 실패했습니다.	모두
-16	DB2SEC_PLUGIN_NETWORKERROR	플러그인에 네트워크 오류가 발생했습니다.	모두
-17	DB2SEC_PLUGIN _CANTLOADLIBRARY	플러그인이 필수 라이브러리를 로드할 수 없습니다.	db2secGroupPluginInit db2secClientAuthPluginInit db2secServerAuthPluginInit
-18	DB2SEC_PLUGIN_CANT _OPEN_FILE	파일 누락 또는 파일 권한 부족 이외의 이유로 인해 플러그인이 파일을 열고 읽을 수 없습니다.	모두
-19	DB2SEC_PLUGIN_FILENOTFOUND	파일 시스템에 파일이 누락되어 플러그인이 파일을 열고 읽을 수 없습니다.	모두

표 10. 보안 플러그인 리턴 코드 (계속)

리턴 코드	값 정의	의미	적용 가능한 API
-20	DB2SEC_PLUGIN _CONNECTION_DISALLOWED	연결이 허용되는 데이터베이스 또는 TCP/IP 주소가 특정 데이터베이스에 연결할 수 없는 제한사항으로 인해 플러그인이 연결을 거부합니다.	모든 서버 측 플러그인 API
-21	DB2SEC_PLUGIN_NO_CRED	GSS API 플러그인에만 해당: 초기 클라이언트 증명서가 누락되었습니다.	db2secGetDefaultLoginContext db2secServerAuthPluginInit
-22	DB2SEC_PLUGIN_CRED_EXPIRED	GSS API 플러그인에만 해당: 클라이언트 증명서가 만기되었습니다.	db2secGetDefaultLoginContext db2secServerAuthPluginInit
-23	DB2SEC_PLUGIN _BAD_PRINCIPAL_NAME	GSS API 플러그인에만 해당: 핵심부 이름이 유효하지 않습니다.	db2secProcessServer PrincipalName
-24	DB2SEC_PLUGIN _NO_CON_DETAILS	이 리턴 코드는 db2secGetConDetails 콜백에서 리턴되어 (예: DB2에서 플러그인으로) DB2가 클라이언트의 TCP/IP 주소를 판별할 수 없음을 나타냅니다.	db2secGetConDetails
-25	DB2SEC_PLUGIN _BAD_INPUT_PARAMETERS	플러그인 API가 호출될 때 일부 매개변수가 유효하지 않거나 누락되었습니다.	모두
-26	DB2SEC_PLUGIN _INCOMPATIBLE_VER	플러그인에서 보고한 API 버전을 DB2에 사용할 수 없습니다.	db2secGroupPluginInit db2secClientAuthPluginInit db2secServerAuthPluginInit
-27	DB2SEC_PLUGIN_PROCESS_LIMIT	플러그인에 충분하지 않은 자원을 사용하여 새 프로세스를 작성할 수 있습니다.	모두
-28	DB2SEC_PLUGIN_NO_LICENSES	플러그인에 사용자 허가 문제점이 발생했습니다. 기본 메커니즘 라이센스가 한계에 도달했을 가능성이 있습니다.	모두
-29	DB2SEC_PLUGIN_ROOT_NEEDED	플러그인이 루트 특권이 필요한 응용 프로그램을 실행하려고 시도합니다.	모두
-30	DB2SEC_PLUGIN_UNEXPECTED _SYSTEM_ERROR	플러그인에 예기치 않은 시스템 오류가 발생했습니다. 현재 시스템 구성이 지원되지 않을 가능성이 있습니다.	모두

보안 플러그인의 오류 메시지 조절

보안 플러그인 API에 오류가 발생하면, API가 errormsg 필드의 ASCII 텍스트 문자열을 리턴하여 리턴 코드보다 많은 특정 문제점 설명을 제공할 수 있습니다.

예를 들어, errormsg 문자열에는 "File /home/db2inst1/mypasswd.txt does not exist"가 있을 수 있습니다. DB2가 이 문자열 전체를 DB2 관리 통지 로드에 쓰며 일부 SQL 메시지에 잘린 버전을 토큰으로 포함할 수도 있습니다. SQL 메시지의 토큰

은 제한된 길이로만 포함할 수 있으므로, 이러한 메시지는 짧아야 하며 메시지의 중요 변수 부분이 문자열 앞에 표시되어야 합니다. 디버깅을 위해 오류 메시지에 보안 플러그인 이름을 추가하는 것이 좋습니다.

암호 만기 오류 등과 같은 긴급하지 않은 오류의 경우, errormsg 문자열은 DIAGLEVEL 데이터베이스 관리 프로그램 구성 매개변수가 4로 설정된 경우에만 덤프됩니다.

이러한 오류 메시지의 메모리는 보안 플러그인이 할당해야 합니다. 따라서 플러그인은 API를 제공하여 db2secFreeErrormsg에 대한 메모리를 제거해야 합니다.

API가 0이 아닌 값을 리턴하는 경우에만 DB2에서 errormsg 필드가 선택됩니다. 따라서 오류가 없는 경우에는 플러그인은 리턴된 이 오류 메시지에 메모리를 할당하지 않아야 합니다.

초기화할 때, 메시지 로깅 함수 포인터인 logMessage_fn이 그룹, 클라이언트 및 서버 플러그인에 전달됩니다. 플러그인은 이 함수를 사용하여 db2diag 로그 파일에 디버깅 정보를 로그할 수 있습니다. 예를 들어, 다음과 같습니다.

```
// Log an message indicate init successful
(*(logMessage_fn))(DB2SEC_LOG_CRITICAL,
                   "db2secGroupPluginInit successful",
                    strlen("db2secGroupPluginInit successful"));
```

db2secLogMessage 함수의 각 매개변수에 대한 세부사항은 각 플러그인 유형의 API 초기화를 참조하십시오.

보안 플러그인 API의 호출 시퀀스

DB2 데이터베이스 관리 프로그램이 보안 플러그인 API를 호출하는 시퀀스는 보안 플러그인 API가 호출되는 시나리오에 따라 다릅니다.

다음은 DB2 데이터베이스 관리 프로그램이 보안 플러그인 API를 호출하는 기본 시나리오입니다.

- 데이터베이스 연결을 위해 클라이언트에서(내재적 및 명시적)
  - CLIENT
  - 서버 기반(SERVER, SERVER_ENCRYPT, DATA_ENCRYPT)
  - GSSAPI 및 Kerberos
- 로컬 권한 부여를 위해 클라이언트, 서버 또는 게이트웨이에서
- 데이터베이스 연결을 위해 서버에서
- GRANT문을 위해 서버에서
- 권한 부여 ID가 속하는 그룹의 목록을 가져오기 위해 서버에서

주: DB2 데이터베이스 서버는 로컬 권한 부여가 필요한 데이터베이스 조치(예: db2start, db2stop 및 db2trc)를 클라이언트 응용프로그램처럼 취급합니다.

이러한 각 조작에 대해, DB2 데이터베이스 관리 프로그램이 보안 플러그인 API를 호출하는 시퀀스는 다릅니다. 다음은 각 시나리오에서 DB2 데이터베이스 관리 프로그램이 API를 호출하는 시퀀스입니다.

CLIENT - 내재적

사용자가 구성한 인증 유형이 CLIENT이면, DB2 클라이언트 응용프로그램은 다음과 같은 보안 플러그인 API를 호출합니다.

- `db2secGetDefaultLoginContext();`
- `db2secValidatePassword();`
- `db2secFreetoken();`

내재적 인증의 경우, 특정 사용자 ID 또는 암호를 지정하지 않고 연결할 때 사용자 ID/암호 플러그인을 사용하면 `db2secValidatePassword` API가 호출됩니다. 필요한 경우 플러그인 개발자는 이 API를 사용하여 내재적 인증을 금지할 수 있습니다.

CLIENT - 명시적

명시적 인증 즉, 사용자 ID와 암호 둘 다 지정되어 있는 데이터베이스에 연결하려고 할 때 *authentication* 데이터베이스 관리 프로그램 구성 매개변수가 CLIENT로 설정되어 있으면 DB2 클라이언트 응용프로그램은 구현에 필요할 경우 다음과 같은 보안 플러그인 API를 여러 번 호출합니다.

- `db2secRemapUserid();`
- `db2secValidatePassword();`
- `db2secFreeToken();`

Server 기반(SERVER, SERVER_ENCRYPT, DATA_ENCRYPT) - 내재적

내재적 인증에서, 클라이언트 및 서버에 협상된 사용자 ID/암호 인증이 있는 경우(예: 서버에서 *srvcon_auth* 매개변수가 SERVER; SERVER_ENCRYPT, DATA_ENCRYPT 또는 DATA_ENCRYPT_CMP로 설정되어 있는 경우), 클라이언트 응용프로그램이 다음과 같은 보안 플러그인 API를 호출합니다.

- `db2secGetDefaultLoginContext();`
- `db2secFreeToken();`

Server 기반(SERVER, SERVER_ENCRYPT, DATA_ENCRYPT) - 명시적

명시적 인증에서, 클라이언트 및 서버에 협상된 사용자 ID/암호 인증이 있는 경우(예: 서버에서 srvcon_auth 매개변수가 SERVER; SERVER_ENCRYPT, DATA_ENCRYPT 또는 DATA_ENCRYPT_CMP로 설정되어 있는 경우), 클라이언트 응용프로그램이 1 다음과 같은 보안 플러그인 API를 호출합니다.

- `db2secRemapUserid();`

GSSAPI 및 Kerberos - 내재적

내재적 인증에서, 클라이언트 및 서버에 협상된 GSS-API 또는 Kerberos 인증이 있는 경우(예: 서버에서 *srvcon_auth* 매개변수가 KERBEROS; KRB_SERVER_ENCRYPT, GSSPLUGIN 또는 GSS_SERVER_ENCRYPT로 설정되어 있는 경우), 클라이언트 응용프로그램이 다음과 같은 보안 플러그인 API를 호출합니다. (gss_init_sec_context() 호출에서는 GSS_C_NO_CREDENTIAL을 입력 증명서로 사용합니다.)

- db2secGetDefaultLoginContext();
- db2secProcessServerPrincipalName();
- gss_init_sec_context();
- gss_release_buffer();
- gss_release_name();
- gss_delete_sec_context();
- db2secFreeToken();

멀티플로우 GSS-API 지원 기능을 사용하면, 구현에 필요할 경우 gss_init_sec_context()를 여러 번 호출할 수 있습니다.

GSSAPI 및 Kerberos - 명시적

협상된 인증 유형이 GSS-API 또는 Kerberos일 경우, 클라이언트 응용프로그램은 다음과 같은 순서로 GSS-API 플러그인에 사용할 보안 플러그인 API를 호출합니다. 이러한 API는 별도의 설명이 없는 한 내재적 인증 및 명시적 인증 둘 다에 사용됩니다.

- db2secProcessServerPrincipalName();
- db2secGenerateInitialCred();(명시적 인증에만 해당)
- gss_init_sec_context();
- gss_release_buffer ();
- gss_release_name();
- gss_release_cred();
- db2secFreeInitInfo();
- gss_delete_sec_context();
- db2secFreeToken();

상호 인증 토큰이 서버에서 리턴되고 구현에서 필요할 경우 API gss_init_sec_context()를 여러 번 호출할 수 있습니다.

로컬 권한 부여를 위해 클라이언트, 서버 또는 게이트웨이에서

로컬 권한 부여의 경우, 사용되는 DB2 명령이 다음과 같은 보안 플러그인 API를 호출합니다

- db2secGetDefaultLoginContext();
- db2secGetGroupsForUser();
- db2secFreeToken();
- db2secFreeGroupList();

이러한 API는 사용자 ID/암호 및 GSS-API 인증 메커니즘에 호출됩니다.

데이터베이스 연결을 위해 서버에서

데이터베이스 서버의 데이터베이스 연결의 경우, DB2 에이전트 프로세스 또는 스레드가 사용자 ID/암호 인증 메커니즘에 다음과 같은 보안 플러그인 API를 호출합니다.

- db2secValidatePassword(); *authentication* 데이터베이스 구성 매개변수가 CLIENT가 아닐 경우에만
- db2secGetAuthIDs();
- db2secGetGroupsForUser();
- db2secFreeToken();
- db2secFreeGroupList();

데이터베이스의 CONNECT의 경우, DB2 에이전트 프로세스 또는 스레드가 GSS-API 인증 메커니즘에 다음과 같은 보안 플러그인 API를 호출합니다.

- gss_accept_sec_context();
- gss_release_buffer();
- db2secGetAuthIDs();
- db2secGetGroupsForUser();
- gss_delete_sec_context();
- db2secFreeGroupListMemory();

GRANT문을 위해 서버에서

USER 또는 GROUP 키워드를 지정하지 않는 GRANT문의 경우 (예: "GRANT CONNECT ON DATABASE TO user1"), DB2 에이전트 프로세스 또는 스레드는 user1이 사용자, 그룹 또는 둘 다인지 판별할 수 있어야 합니다. 따라서 DB2 에이전트 프로세스 또는 스레드가 다음과 같은 보안 플러그인 API를 호출합니다.

- db2secDoesGroupExist();
- db2secDoesAuthIDExist();

권한 ID가 속하는 그룹의 목록을 가져오기 위해 서버에서

데이터베이스 서버에서 권한 부여 ID가 속하는 그룹의 목록을 가져와야 하는 경우, DB2 에이전트 프로세스 또는 스레드가 권한 부여 ID만을 입력으로 사용하여 다음과 같은 보안 플러그인 API를 호출합니다.

- db2secGetGroupsForUser();

다른 보안 플러그인의 토큰은 없습니다.

제 161 장 보안 플러그인

DB2 데이터베이스 시스템에 대한 인증은 *보안 플러그인*을 사용하여 수행됩니다. 보안 플러그인은 인증 보안 서비스를 제공하는 동적으로 로드 가능한 라이브러리입니다.

DB2 데이터베이스 시스템에서 제공하는 플러그인의 유형은 다음과 같습니다.

- 그룹 검색 플러그인: 지정된 사용자에 대한 그룹 멤버십 정보를 검색합니다.
- 클라이언트 인증 플러그인: DB2 클라이언트에 대한 인증을 관리합니다.
- 서버 인증 플러그인: DB2 서버에 대한 인증을 관리합니다.

DB2 데이터베이스 관리 프로그램이 플러그인 인증에 대해 지원하는 두 가지 메커니즘은 다음과 같습니다.

사용자 ID/암호 인증

이 인증에는 사용자 ID와 암호를 사용하는 인증이 포함됩니다. 사용자 ID/암호 인증 플러그인을 사용하여 구현되는 인증 유형은 다음과 같습니다.

- CLIENT
- SERVER
- SERVER_ENCRYPT
- DATA_ENCRYPT
- DATA_ENCRYPT_CMP

이러한 인증 유형에 따라 사용자 인증이 수행되는 방식과 위치가 결정됩니다. 사용되는 인증 유형은 *authentication* 데이터베이스 관리 프로그램 구성 매개변수에 지정된 인증 유형에 따라 다릅니다. SRVCON_AUTH 매개변수가 지정된 경우, 연결 또는 접속 조작을 처리할 때 이 매개변수가 AUTHENTICATION보다 우선적으로 처리됩니다.

GSS-API 인증

GSS-API의 공식적인 이름은 *Generic Security Service Application Program Interface, Version 2*(IETF RFC2743) 및 *Generic Security Service API Version 2: C-Bindings*(IETF RFC2744)입니다. GSS-API를 사용하면 Kerberos 인증도 구현됩니다. GSS-API 인증 플러그인을 사용하여 구현되는 인증 유형은 다음과 같습니다.

- KERBEROS
- GSSPLUGIN
- KRB_SERVER_ENCRYPT
- GSS_SERVER_ENCRYPT

KRB_SERVER_ENCRYPT 및 GSS_SERVER_ENCRYPT는 GSS-API 인증과 사용자 ID/암호 인증을 모두 지원하지만, GSS-API 인증이 선호하는 인증 유형입니다.

주: 인증 유형에 따라 사용자 인증이 수행되는 방식과 위치가 결정됩니다. 특정 인증 유형을 사용하려면 인증 데이터베이스 관리 프로그램 구성 매개변수를 갱신하십시오.

각 플러그인을 개별적으로 사용하거나 하나 이상의 다른 플러그인과 함께 사용할 수 있습니다. 예를 들어, 서버 인증만 사용하고 클라이언트 그룹 인증에 대해서는 DB2 디폴트 값을 사용할 수 있습니다. 또는 그룹 또는 클라이언트 인증 플러그인만 사용할 수도 있습니다. GSS-API 인증 플러그인의 경우에만 클라이언트 및 서버 플러그인이 둘 다 필요합니다.

디폴트 동작은 인증에 대해 운영 체제 레벨 메커니즘을 구현하는 사용자 ID/암호 플러그인을 사용하는 것입니다. 이전 릴리스에서는 플러그인 구현 없이 운영 체제 레벨 인증을 직접 사용했습니다. 클라이언트 측 Kerberos는 AIX, Windows 및 Linux 운영 체제에서 지원됩니다. Windows 플랫폼의 경우 Kerberos가 디폴트로 지원됩니다.

DB2 데이터베이스 시스템에는 그룹 검색, 사용자 ID/암호 인증 및 Kerberos 인증에 사용되는 플러그인 세트가 있습니다. 보안 플러그인 인증을 사용할 경우, 사용자 고유의 플러그인을 개발하거나 써드 파티 플러그인을 구매하는 방식으로 DB2 클라이언트 및 서버 인증을 사용자 정의할 수 있습니다.

DB2 클라이언트에서 보안 플러그인 전개

DB2 클라이언트는 하나의 그룹 플러그인과 하나의 사용자 ID/암호 인증 플러그인을 지원할 수 있으며 특정 GSS-API 플러그인에 대해서는 DB2 서버와 협상합니다. 이 협상은 클라이언트에 구현된 인증 플러그인과 처음으로 일치하는 인증 플러그인 이름의 구현된 DB2 서버의 GSS-API 플러그인 목록을 클라이언트에서 스캔하는 과정으로 이루어집니다. 서버의 플러그인 목록은 *srvcon_gssplugin_list* 데이터베이스 관리 프로그램 구성 매개변수 값에 지정되어 있으며 이 값은 서버에 구현된 모든 플러그인의 이름이 포함되어 있습니다. 다음 그림은 DB2 클라이언트의 보안 플러그인 인프라스트럭처를 나타냅니다.

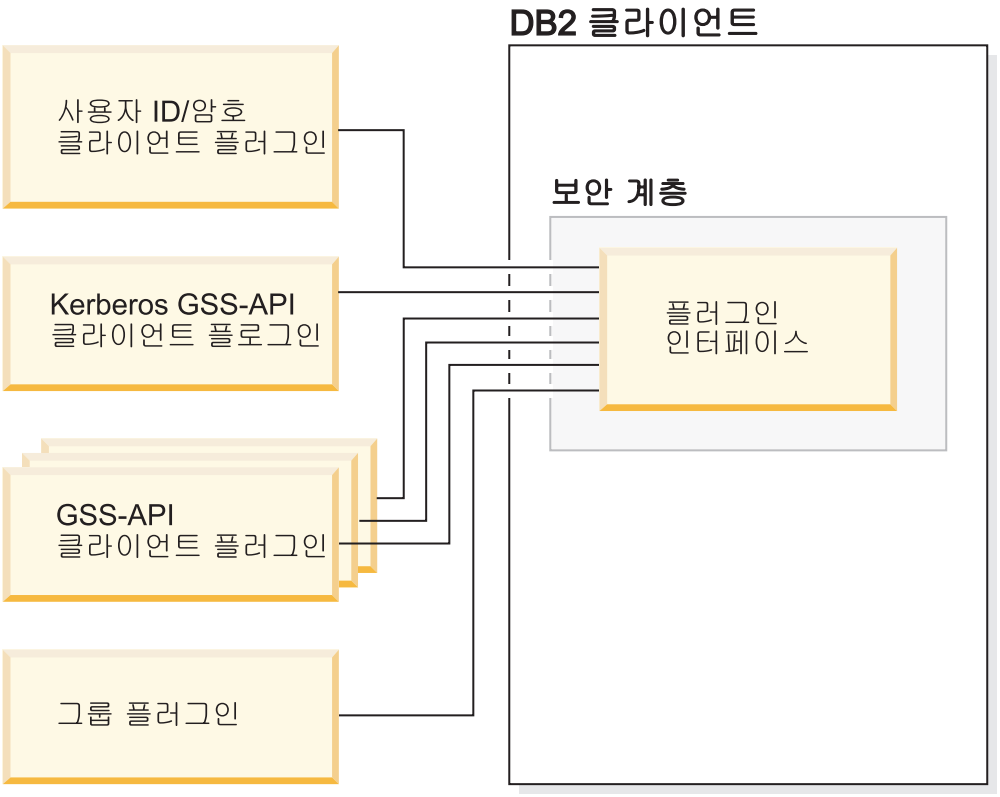


그림 1. DB2 클라이언트에서 보안 플러그인 전개

DB2 서버에서 보안 플러그인 전개

DB2 서버는 하나의 그룹 플러그인, 하나의 사용자 ID/암호 인증 플러그인 및 여러 개의 GSS-API 플러그인을 지원합니다. 여러 개의 GSS-API 플러그인은 *srvcon_gssplugin_list* 관리 프로그램 구성 매개변수 값에 목록으로 지정되어 있으며 목록에 있는 한 개의 GSS-API 플러그인만 Kerberos 플러그인이 될 수 있습니다.

서버 측 보안 이외에도 클라이언트 권한 부여 플러그인을 데이터베이스 서버에 전개해야 할 수도 있습니다. db2start 및 db2trc와 같은 인스턴스 레벨 조작을 실행하는 경우, DB2 데이터베이스 관리 프로그램은 클라이언트 인증 플러그인을 사용하여 이러한 조작에 대한 권한 부여 검사를 수행합니다. 따라서 *authentication* 데이터베이스 관리 프로그램 구성 매개변수에 지정된 서버 플러그인에 해당하는 클라이언트 인증 플러그인을 설치해야 합니다. *authentication*과 *srvcon_auth* 간에는 큰 차이가 있습니다. 특히 이들 매개변수는 데이터베이스 연결 인증과 로컬 권한 부여에 각기 다른 메커니즘이 사용될 수 있도록 서로 다른 값으로 설정할 수 있습니다. 가장 일반적인 경우는 *srvcon_auth*는 GSSPLUGIN으로 설정하고 *authentication*는 SERVER로 설정하는 입니다. 데이터베이스 서버에서 클라이언트 인증 플러그인을 사용하지 않을 경우, 인스턴스 레벨 조작(예: db2start)이 실패하게 됩니다. 예를 들어, 인증 유형이 SERVER이고 사용자가 제공하는 클라이언트 플러그인을 사용하지 않는 경우, DB2 데이터베이스 시스템은 IBM에서 제공하는 디폴트 클라이언트 운영 체제 플러그인을 사용합니다. 다음

그림은 DB2 서버의 보안 플러그인 인프라스트럭처를 나타냅니다.

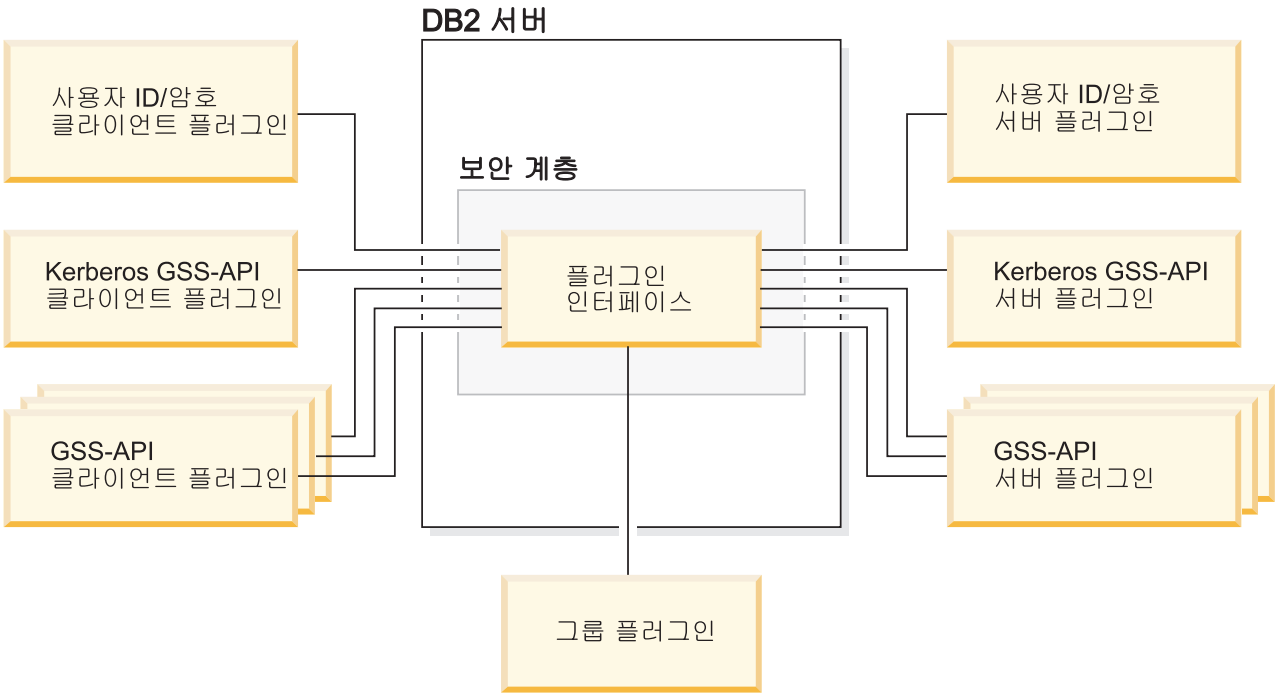


그림 2. DB2 서버에서 보안 플러그인 전개

주: 보안 플러그인의 전개를 적절하게 코딩, 검토 및 테스트하지 않은 경우 DB2 데이터베이스 시스템 설치의 무결성이 손상될 수 있습니다. DB2 데이터베이스 시스템은 일반적으로 발생하는 여러 가지 오류 유형에 대한 예방 조치를 취하고는 있지만, 이러한 예방 조치가 사용자가 작성한 보안 플러그인 배치 시의 무결성을 완전하게 보장하지는 못합니다.

보안 플러그인 사용

시스템 관리자는 특정 플러그인 관련 데이터베이스 관리 프로그램 구성 매개변수를 갱신하여 각 인증 메커니즘에 사용할 플러그인의 이름을 지정할 수 있습니다. 이러한 매개변수가 널(NULL)일 경우, 그룹 검색, 사용자 ID/암호 관리 또는 Kerberos(서버에서 인증이 Kerberos로 설정된 경우)에 대해 DB2 제공 플러그인이 디플트로 제공됩니다. DB2에서는 디폴트 GSS-API 플러그인을 제공하지 않습니다. 따라서 시스템 관리자가 *authentication* 매개변수에 인증 유형을 GSSPLUGIN으로 지정한 경우, *srvcon_gssplugin_list*에 GSS-API 인증 플러그인도 지정해야 합니다.

DB2의 보안 플러그인 로드 방식

데이터베이스 관리 프로그램이 시작되면 데이터베이스 관리 프로그램 구성 매개변수에 지정된 지원되는 플러그인이 모두 로드됩니다.

DB2 클라이언트는 연결 또는 접속 조작 중 서버와 협상한 보안 메커니즘에 적합한 플러그인을 로드합니다. 클라이언트 응용프로그램은 여러 개의 보안 플러그인을 동시에 로드하여 사용할 수 있습니다. 예를 들어, 서로 다른 인스턴스의 서로 다른 데이터베이스에 동시에 연결되어 있는 스레드 프로그램이 이와 같은 경우에 해당합니다.

연결 또는 접속 이외의 조치를 수행하려면 권한 부여(예: 데이터베이스 관리 프로그램 구성 갱신, 데이터베이스 관리 프로그램 시작 및 중지, DB2 추적 설정/해제)도 필요합니다. 이러한 조치의 경우 DB2 클라이언트 프로그램은 다른 데이터베이스 관리 프로그램 구성 매개변수에 지정된 플러그인을 로드합니다. *authentication*이 GSSPLUGIN으로 설정된 경우, DB2 데이터베이스 관리 프로그램은 *local_gssplugin*에 지정된 플러그인을 사용합니다. *authentication*이 KERBEROS로 설정된 경우, DB2 데이터베이스 관리 프로그램은 *clnt_krb_plugin*에 지정된 플러그인을 사용합니다. 또는 DB2 데이터베이스 관리 프로그램은 *clnt_pw_plugin*에 지정된 플러그인을 사용합니다.

보안 플러그인 API는 IPv4 플랫폼 또는 IPv6 플랫폼에서 호출할 수 있습니다. IPv4 주소는 a.b.c.d로 구성된 읽기 가능한 32비트 주소입니다. 여기서 a-d는 0-255의 10진수를 나타냅니다. IPv6 주소는 a:b:c:d:e:f:g:h로 구성된 128비트 주소입니다. 여기서 a-h 각각은 4개의 16진수를 나타냅니다.

보안 플러그인 개발

보안 플러그인을 개발 중인 경우, DB2 데이터베이스 관리 프로그램에서 사용할 표준 인증 함수를 구현해야 합니다. 사용자 정의된 보안 플러그인을 사용 중인 경우, CLP를 통해 발행된 연결 명령문 또는 동적 SQL문에 최대 255자의 사용자 ID를 사용할 수 있습니다. 사용 가능한 유형의 플러그인에 대해 구현해야 할 기능은 다음과 같습니다.

그룹 검색

사용자가 속해 있는 그룹의 목록을 가져옵니다.

사용자 ID/암호 인증

- 디폴트 보안 컨텍스트를 식별합니다(클라이언트만 해당).
- 암호의 유효성을 확인하고 선택적으로 암호를 변경합니다.
- 제공된 문자열이 유효한 사용자인지 판별합니다(서버만 해당).
- 서버로 보내기 전에 클라이언트에 제공된 사용자 ID 또는 암호를 수정합니다(클라이언트만 해당).
- 지정된 사용자와 연관된 DB2 권한 부여 ID를 리턴합니다.

GSS-API 인증

- 필수 GSS-API 함수를 구현합니다.
- 디폴트 보안 컨텍스트를 식별합니다(클라이언트만 해당).
- 사용자 ID 및 암호를 기준으로 초기 증명서를 생성하고 선택적으로 암호를 변경합니다(클라이언트만 해당).

- 보안 티켓을 작성 및 승인합니다.
- 지정된 GSS-API 보안 컨텍스트와 연관된 DB2 권한 부여 ID를 리턴합니다.

보안 플러그인 라이브러리 위치

보안 플러그인을 자체 개발하거나 써드 파티로부터 구매하는 방법으로 보안 플러그인을 얻은 후에는 이를 데이터베이스 서버의 특정 위치로 복사하십시오.

DB2 클라이언트가 클라이언트 측 사용자 인증 플러그인을 찾는 디렉토리는 다음과 같습니다.

- UNIX 32비트: `$DB2PATH/security32/plugin/client`
- UNIX 64비트: `$DB2PATH/security64/plugin/client`
- WINDOWS 32비트 및 64비트: `$DB2PATH¥security¥plugin¥`*instance name*`¥client`

주: Windows 기반 플랫폼의 경우, *instance name* 및 *client* 서브디렉토리는 자동으로 작성되지 않습니다. 인스턴스 소유자가 수동으로 작성해야 합니다.

DB2 데이터베이스 관리 프로그램이 서버 측 사용자 인증 플러그인을 찾는 디렉토리는 다음과 같습니다.

- UNIX 32비트: `$DB2PATH/security32/plugin/server`
- UNIX 64비트: `$DB2PATH/security64/plugin/server`
- WINDOWS 32비트 및 64비트: `$DB2PATH¥security¥plugin¥`*instance name*`¥server`

주: Windows 기반 플랫폼의 경우, *instance name* 및 *server* 서브디렉토리는 자동으로 작성되지 않습니다. 인스턴스 소유자가 수동으로 작성해야 합니다.

DB2 데이터베이스 관리 프로그램이 그룹 플러그인을 찾는 디렉토리는 다음과 같습니다.

- UNIX 32비트: `$DB2PATH/security32/plugin/group`
- UNIX 64비트: `$DB2PATH/security64/plugin/group`
- WINDOWS 32비트 및 64비트: `$DB2PATH¥security¥plugin¥`*instance name*`¥group`

주: Windows 기반 플랫폼의 경우, *instance name* 및 *group* 서브디렉토리는 자동으로 작성되지 않습니다. 인스턴스 소유자가 수동으로 작성해야 합니다.

보안 플러그인 이름 지정 규칙

보안 플러그인 라이브러리의 파일 이름 확장자는 플랫폼별로 달라야 합니다. C 또는 C++에서 작성된 보안 플러그인 라이브러리의 파일 이름 확장자도 플랫폼별로 달라야 합니다.

- Windows: `.dll`
- AIX: `.a` 또는 `.so`, 두 확장자가 모두 있는 경우 `.a` 확장자가 사용됩니다.
- Linux, HP IPF 및 Solaris: `.so`
- HPUX/PA-RISC: `.sl` 또는 `.so`, 두 확장자가 모두 있는 경우 `.sl` 확장자가 사용됩니다.

주: 사용자가 DB2 Universal JDBC 드라이버를 사용하는 보안 플러그인을 개발할 수도 있습니다.
예를 들어, `MyPlugin`이라는 보안 플러그인 라이브러리가 있다고 가정할 경우, 지원되는 운영 체제 각각에 적합한 라이브러리 파일 이름은 다음과 같습니다.

- Windows 32비트: MyPlugin.dll
- Windows 64비트: MyPlugin64.dll
- AIX 32 또는 64비트: MyPlugin.a 또는 MyPlugin.so
- SUN 32 또는 64비트, Linux 32 또는 64비트, HP 32 또는 64비트/IPF: MyPlugin.so
- HP-UX 32 또는 64비트/PA-RISC: MyPlugin.sl 또는 MyPlugin.so

주: 접미부 ″64″는 64비트 Windows 보안 플러그인에 대한 라이브러리 이름에만 필요합니다.

데이터베이스 관리 프로그램 구성을 보안 플러그인의 이름으로 갱신하는 경우, 접미부 ″64″없이 전체 라이브러리 이름을 사용하고 이름의 완전한 경로 부분과 파일 확장자는 생략하십시오. 보안 플러그인 라이브러리 `MyPlugin`은 운영 체제에 관계없이 다음과 같이 등록됩니다.

```
UPDATE DBM CFG USING CLNT_PW_PLUGIN MyPlugin
```

보안 플러그인 이름은 대소문자를 구분하므로 라이브러리 이름과 정확하게 일치해야 합니다. DB2 데이터베이스 시스템은 관련 데이터베이스 관리 프로그램 구성 매개변수의 값을 사용하여 라이브러리 경로를 어셈블한 다음 해당 라이브러리 경로를 사용하여 보안 플러그인 라이브러리를 로드합니다.

보안 플러그인의 이름이 충돌하는 것을 방지하려면 사용된 인증 메소드 및 플러그인을 작성한 회사의 식별 기호를 사용하여 플러그인 이름을 지정해야 합니다. 예를 들어, `Foo, Inc.`라는 회사가 인증 메소드 `FOOsomemethod`를 구현하는 플러그인을 작성했다면 플러그인 이름은 `FOOsomemethod.dll`입니다.

플러그인 이름의 최대 길이(파일 확장자 및 접미부 ″64″ 제외)는 32바이트로 제한됩니다. 데이터베이스 서버에서 지원하는 최대 플러그인 수는 없지만, 데이터베이스 관리 프로그램 구성에 있는 플러그인의 쉼표로 구분된 목록의 최대 길이는 255바이트입니다. 내장 파일 sqlenv.h에 있는 두 개의 define으로 이러한 두 제한을 식별할 수 있습니다.

```
#define SQL_PLUGIN_NAME_SZ     32     /* plug-in name */
#define SQL_SRVCON_GSSPLUGIN_LIST_SZ 255 /* GSS API plug-in list */
```

보안 플러그인 라이브러리 파일은 다음과 같은 파일 권한을 가지고 있어야 합니다.

- 인스턴스 소유자에 의해 소유됨
- 시스템의 모든 사용자가 읽을 수 있음
- 시스템의 모든 사용자가 실행할 수 있음

두 파트 사용자 ID에 대한 보안 플러그인 지원

Windows에 설치된 DB2 데이터베이스 관리 프로그램에서는 두 파트 사용자 ID를 사용할 수 있으며 두 파트 사용자 ID를 두 파트 권한 부여 ID에 맵핑할 수 있습니다.

예를 들어, 도메인과 사용자 ID로 구성된 Windows 운영 체제의 두 파트 사용자 ID(예: MEDWAY₩pieter)가 있다고 가정합니다. 이 예에서 MEDWAY는 도메인이고 pieter는 사용자 이름입니다. DB2 데이터베이스 시스템에서는 이 두 파트 사용자 ID를 한 파트 권한 부여 ID 또는 두 파트 권한 부여 ID에 맵핑할지 여부를 지정할 수 있습니다.

두 파트 사용자 ID를 두 파트 권한 부여 ID에 맵핑할 수는 있지만, 디폴트 동작은 아닙니다. 디폴트로 한 파트 사용자 ID와 두 파트 사용자 ID는 한 파트 권한 부여 ID에 맵핑됩니다. 두 파트 사용자 ID를 두 파트 권한 부여 ID에 맵핑할 수는 있지만, 디폴트 동작은 아닙니다.

두 파트 사용자 ID를 한 파트 사용자 ID에 맵핑하는 디폴트 동작을 사용하는 경우 사용자가 다음을 사용하여 데이터베이스에 연결할 수 있습니다.

```
db2 connect to db user MEDWAY₩pieter using pw
```

이때 디폴트 동작을 사용할 경우, 사용자 ID MEDWAY₩pieter는 권한 부여 ID PIETER로 분석됩니다. 두 파트 사용자 ID를 두 파트 권한 부여 ID에 맵핑할 수 있는 경우, 권한 부여 ID는 MEDWAY₩PIETER입니다.

DB2에서는 두 파트 사용자 ID를 두 파트 권한 부여 ID에 맵핑할 수 있도록 하기 위해 두 가지 인증 플러그인 세트를 제공합니다.

- 한 세트는 배타적으로 한 파트 사용자 ID를 한 파트 권한 부여 ID에 맵핑하고 두 파트 사용자 ID를 한 파트 권한 부여 ID에 맵핑합니다.
- 다른 세트는 한 파트 사용자 ID와 두 파트 사용자 ID를 모두 두 파트 권한 부여 ID에 맵핑합니다.

작업 환경의 사용자 이름을 다른 위치에 정의된 여러 어카운트(예: 로컬 어카운트, 도메인 어카운트, 트러스트된 도메인 어카운트)에 맵핑할 수 있는 경우, 두 파트 권한 부여 ID 맵핑이 가능한 플러그인을 지정할 수 있습니다.

도메인과 사용자 ID로 구성되는 두 파트 권한 부여 ID(예: MEDWAY¥pieter)와 한 파트 권한 부여 ID(예: PIETER)는 기능적으로 구별되는 권한 부여 ID라는 점을 유념하십시오. 이러한 권한 부여 ID 중 하나와 연관된 특권 세트는 다른 권한 부여 ID와 연관된 특권 세트와 완전히 다를 수 있습니다. 따라서 한 파트 권한 부여 ID와 두 파트 권한 부여 ID 관련 작업을 수행할 때는 주의해야 합니다.

다음 표는 DB2 데이터베이스 시스템에서 제공하는 플러그인의 종류 및 특정 인증 구현에 대한 플러그인 이름을 보여줍니다.

표 11. DB2 보안 플러그인

인증 유형	한 파트 사용자 ID 플러그인의 이름	두 파트 사용자 ID 플러그인의 이름
사용자 ID/암호(클라이언트)	IBMOSauthclient	IBMOSauthclientTwoPart
사용자 ID/암호(서버)	IBMOSauthserver	IBMOSauthserverTwoPart
Kerberos	IBMkrb5	IBMkrb5TwoPart

주: Windows 64비트 플랫폼의 경우 여기에 나열된 플러그인 이름에 ″64″가 추가됩니다.

사용자 ID/암호 또는 Kerberos 플러그인이 필요한 인증 유형을 지정한 경우, 위의 표에 있는 ″한 파트 사용자 ID 플러그인의 이름″ 컬럼에 나열된 플러그인이 디폴트로 사용됩니다.

두 파트 사용자 ID를 두 파트 권한 부여 ID에 맵핑하려면 두 파트 플러그인(디폴트 플러그인이 아님)을 사용하도록 지정해야 합니다. 보안 관련 데이터베이스 관리 프로그램 구성 매개변수를 다음과 같이 설정하면 보안 플러그인이 인스턴스 레벨에서 지정됩니다.

두 파트 사용자 ID를 두 파트 권한 부여 ID에 맵핑하는 서버 인증의 경우, 다음과 같이 설정해야 합니다.

- srvcon_pw_plugin을 IBMOSauthserverTwoPart로 설정
- clnt_pw_plugin을 IBMOSauthclientTwoPart로 설정

두 파트 사용자 ID를 두 파트 권한 부여 ID에 맵핑하는 클라이언트 인증의 경우, 다음과 같이 설정해야 합니다.

- srvcon_pw_plugin을 IBMOSauthserverTwoPart로 설정
- clnt_pw_plugin을 IBMOSauthclientTwoPart로 설정

두 파트 사용자 ID를 두 파트 권한 부여 ID에 맵핑하는 Kerberos 인증의 경우, 다음과 같이 설정해야 합니다.

- srvcon_gssplugin_list를 IBMOSkrb5TwoPart로 설정
- clnt_krb_plugin을 IBMkrb5TwoPart로 설정

보안 플러그인 라이브러리는 Microsoft® Windows Security Account Manager 호환 형식으로 지정된 두 파트 사용자 ID를 허용합니다(예: *domain₩user ID* 형식). 도메인 및 사용자 ID 정보는 연결 시 DB2 인증 및 권한 부여 프로세스에 사용됩니다.

기존 데이터베이스의 한 파트 권한 부여 ID와 충돌하지 않도록 새 데이터베이스를 작성하는 경우에는 두 파트 플러그인을 구현해야 합니다. 두 파트 권한 부여 ID를 사용하는 새 데이터베이스는 한 파트 권한 부여 ID를 사용하는 데이터베이스와는 다른 인스턴스에 작성해야 합니다.

보안 플러그인 API 버전화

DB2 데이터베이스 시스템에서는 보안 플러그인 API의 버전 번호를 지원합니다. DB2 UDB 버전 8.2의 경우 이 버전 번호는 1로 시작하는 정수입니다.

DB2가 보안 플러그인 API에 전달하는 버전 번호는 DB2에서 지원할 수 있는 가장 높은 버전 번호로서, 구조 버전 번호에 해당합니다. 더 높은 API 버전을 지원하는 플러그인의 경우, DB2에서 요청한 버전에 대한 함수 포인터를 리턴해야 합니다. 더 낮은 API 버전을 지원하는 플러그인의 경우, 낮은 버전에 대한 함수 포인터를 채워야 합니다. 어떤 경우든지, 보안 플러그인 API는 함수 구조의 버전 필드에서 지원되는 API의 버전 번호를 리턴해야 합니다.

DB2에서는 API의 매개변수가 변경된 경우와 같이 필요한 경우에만 보안 플러그인의 버전 번호가 변경됩니다. 버전 번호는 DB2 릴리스 번호와 함께 자동으로 변경되지 않습니다.

보안 플러그인에 대한 32비트 및 64비트 고려사항

일반적으로 32비트 DB2 인스턴스는 32비트 보안 플러그인을 사용하고, 64비트 DB2 인스턴스는 64비트 보안 플러그인을 사용합니다. 그러나 64비트 인스턴스에서 DB2는 32비트 플러그인 라이브러리가 필요한 32비트 응용프로그램을 지원합니다.

32비트 응용프로그램과 64비트 응용프로그램을 모두 실행할 수 있는 데이터베이스 인스턴스를 하이브리드 인스턴스라고 합니다. 하이브리드 인스턴스에서 32비트 응용프로그램을 실행하려는 경우, 32비트 플러그인 디렉토리에서 32비트 보안 플러그인을 사용할 수 있는지 확인하십시오. Linux 및 UNIX 운영 체제(Linux IPF 제외)에 있는 DB2 인스턴스의 경우, security32 및 security64 디렉토리가 표시됩니다. Windows x64F 또는 IPF에 있는 64비트 DB2 인스턴스의 경우, 32비트 및 64비트 보안 플러그인이 같은 디렉토리에 있지만 64비트 플러그인 이름의 끝에는 ″64″가 붙습니다.

32비트 인스턴스에서 64비트 인스턴스로 업그레이드하려는 경우에는 64비트용으로 재컴파일할 보안 플러그인의 버전을 얻어야 합니다.

64비트 플러그인 라이브러리를 제공하지 않는 벤더로부터 보안 플러그인을 얻은 경우에는 32비트 응용프로그램을 실행하는 64비트 스텁을 구현할 수 있습니다. 이 경우 보안 플러그인은 라이브러리가 아닌 외부 프로그램입니다.

보안 플러그인 문제점 판별

보안 플러그인 관련 문제점은 SQL 오류 및 관리 통지 로그를 통해 보고됩니다.

보안 플러그인과 관련된 SQLCODE 값은 다음과 같습니다.

- db2start 또는 db2stop 중 플러그인 오류가 발생하면 SQLCODE -1365가 리턴됩니다.
- 로컬 권한 부여 문제점이 발생할 때마다 SQLCODE -1366이 리턴됩니다.
- 모든 연결 관련 플러그인 오류에 대해서는 SQLCODE -30082가 리턴됩니다.

관리 통지 로그를 통해 보안 플러그인을 디버깅하고 관리할 수 있습니다. UNIX에서 관리 통지 로그 파일을 확인하려면 sqllib/db2dump/*instance name*.*N*.nfy를 확인하십시오. Windows 운영 체제에서 관리 통지 로그를 확인하려면 이벤트 표시기 도구를 사용하십시오. 이벤트 표시기는 Windows 운영 체제의 "시작" 단추를 누른 다음 설정 -> 제어판 -> 관리 도구 -> 이벤트 표시기로 이동하면 찾을 수 있습니다. 보안 플러그인과 관련된 관리 통지 로그 값은 다음과 같습니다.

- 13000은 GSS-API 보안 플러그인 API에 대한 호출이 오류로 인해 실패했으며 선택적으로 오류 메시지가 리턴되었음을 나타냅니다.

  ```
  SQLT_ADMIN_GSS_API_ERROR (13000)
  Plug-in "plug-in name" received error code "error code" from
  GSS API  "gss api name" with the error message "error message"
  ```

- 13001은 DB2 보안 플러그인 API에 대한 호출이 오류로 인해 실패했으며 선택적으로 오류 메시지가 리턴되었음을 나타냅니다.

  ```
  SQLT_ADMIN_PLUGIN_API_ERROR(13001)
  Plug-in "plug-in name" received error code "error code" from DB2
  security plug-in API "gss api name" with the error message
  "error message"
  ```

- 13002는 DB2에서 플러그인이 언로드되지 않았음을 나타냅니다.

  ```
  SQLT_ADMIN_PLUGIN_UNLOAD_ERROR (13002)
  Unable to unload plug-in "plug-in name". No further action required.
  ```

- 13003은 핵심부 이름이 잘못되었음을 나타냅니다.

  ```
  SQLT_ADMIN_INVALID_PRIN_NAME (13003)
  The principal name "principal name" used for "plug-in name"
  is invalid. Fix the principal name.
  ```

- 13004는 플러그인 이름이 유효하지 않음을 나타냅니다. 플러그인 이름에는 경로 구분자(UNIX의 경우 "/", Windows의 경우 "₩")를 사용할 수 없습니다.

  ```
  SQLT_ADMIN_INVALID_PLGN_NAME (13004)
  The plug-in name "plug-in name" is invalid. Fix the plug-in name.
  ```

- 13005는 보안 플러그인이 로드되지 않았음을 나타냅니다. 플러그인이 올바른 디렉토리에 있으며 적합한 데이터베이스 관리 프로그램 구성 매개변수가 갱신되었는지 확인하십시오.

  ```
  SQLT_ADMIN_PLUGIN_LOAD_ERROR (13005)
  Unable to load plug-in "plug-in name". Verify the plug-in existence and
  directory where it is located is correct.
  ```

- 13006은 보안 플러그인에서 예기치 않은 오류가 발생했음을 나타냅니다. 모든 db2support 정보를 수집하고, 가능한 경우 db2trc를 캡처한 다음 IBM Support에 추가 지원을 요청하십시오.

  ```
  SQLT_ADMIN_PLUGIN_UNEXP_ERROR (13006)
  Plug-in encountered unexpected error. Contact IBM Support for further assistance.
  ```

주: Windows 64비트 데이터베이스 서버에서 보안 플러그인을 사용하는데 보안 플러그인에 대한 로드 오류가 표시되는 경우, 32비트 및 64비트 고려사항 및 보안 플러그인 이름 지정 규칙에 대한 주제를 참조하십시오. 64비트 플러그인 라이브러리의 경우 라이브러리 이름에 "64"라는 접미부가 표시되어 하지만, 보안 플러그인 데이터베이스 관리 프로그램 구성 매개변수의 항목에는 이 접미부가 표시되어서는 안됩니다.

제 162 장 보안 플러그인 API

DB2 데이터베이스 시스템 인증 및 그룹 멤버십 찾아보기 동작을 사용자 정의할 수 있도록 하기 위해, DB2 데이터베이스 시스템에서는 기존 플러그인 모듈을 수정하거나 보안 플러그인 모듈을 새로 작성하는 데 사용할 수 있는 API를 제공합니다.

보안 플러그인 모듈을 개발 중인 경우, DB2 데이터베이스 관리 프로그램에서 호출할 표준 인증 또는 그룹 멤버십 찾아보기 함수를 구현해야 합니다. 사용 가능한 세 가지 유형의 플러그인 모듈에 대해 구현해야 할 기능은 다음과 같습니다.

그룹 검색
: 제공된 사용자에 대한 그룹 멤버십 정보를 검색하고 제공된 문자열이 유효한 사용자인지 판별합니다.

사용자 ID/암호 인증
: 디폴트 보안 컨텍스트를 식별하고(클라이언트만 해당), 암호를 검증하고 선택적으로 변경하며 제공된 문자열이 유효한 사용자인지 판별하고(서버만 해당), 클라이언트에 제공된 사용자 ID나 암호를 서버로 보내기 전에 수정하며(클라이언트만 해당), 제공된 사용자와 연관된 DB2 권한 부여 ID를 리턴하는 인증입니다.

GSS-API 인증
: 필수 GSS-API 함수를 구현하고, 디폴트 보안 컨텍스트를 식별하며(클라이언트만 해당), 사용자 ID와 암호를 기준으로 초기 증명서를 생성하고, 선택적으로 암호를 변경하며(클라이언트만 해당), 보안 티켓을 작성 및 허용하고, 제공된 GSS-API 보안 컨텍스트와 연관된 DB2 권한 부여 ID를 리턴하는 인증입니다.

다음은 플러그인 API 설명에 사용된 용어에 대한 정의입니다.

플러그인
: 사용자가 작성한 인증 또는 그룹 멤버십 찾아보기 함수에 액세스하기 위해 DB2에서 로드하는 동적으로 로드 가능한 라이브러리입니다.

내재된 인증
: 사용자 ID나 암호를 지정하지 않고 데이터베이스에 연결하는 것입니다.

명시적 인증
: 사용자 ID와 암호를 지정하여 데이터베이스에 연결하는 것입니다.

권한 ID
: 데이터베이스 내에 있는 권한과 특권이 부여되는 개인 또는 그룹을 나타내는 내부 ID입니다. 내부적으로 DB2 권한 ID는 대문자로 변환되며 최소 8자(8자까

지 공백으로 채워짐)입니다. 현재 DB2에서는 권한 ID, 사용자 ID, 암호, 그룹 이름, 이름 스페이스 및 도메인 이름을 7비트 ASCII로 나타낼 수 있어야 합니다.

로컬 권한 부여

사용자가 데이터베이스 관리 프로그램 시작 및 중지, DB2 추적 설정 및 해제 또는 데이터베이스 관리 프로그램 구성 갱신 등의 조치(데이터베이스에 연결하는 작업 아님)를 수행할 수 있는 권한이 있는지 확인하는 서버 또는 클라이언트로 국한되는 권한 부여입니다.

이름 스페이스

고유한 개별 사용자 ID를 포함해야 하는 사용자 콜렉션 또는 그룹입니다. 일반적인 예로는 Windows 도메인 및 Kerberos 범주가 있습니다. 예를 들어, Windows 도메인n ″usa.company.com″의 모든 사용자 이름은 고유해야 합니다(예: ″user1@usa.company.com″). 그러나 ″user1@canada.company.com″의 경우와 같이 다른 도메인에 있는 동일한 사용자 ID는 다른 사람을 나타냅니다. 완전한 사용자 ID에는 사용자 ID와 이름 스페이스 쌍(예: ″user@domain.name″ 또는 ″domain₩user″)이 포함됩니다.

입력 DB2가 보안 플러그인 API 매개변수에 대한 값을 채우는 것을 나타냅니다.

출력 보안 플러그인 API가 API 매개변수의 값을 채우는 것을 나타냅니다.

그룹 검색 플러그인용 API

그룹 검색 플러그인 모듈의 경우, 다음과 같은 API를 구현해야 합니다.

- db2secGroupPluginInit

  주: db2secGroupPluginInit API는 다음과 같은 프로토타입을 사용하여 API에 포인터 *logMessage_fn을 입력으로 사용합니다.

```
SQL_API_RC (SQL_API_FN db2secLogMessage)
(
db2int32 level,
void    *data,
db2int32 length
);
```

  플러그인은 db2secLogMessage API를 사용하여 디버깅 또는 정보용으로 db2diag 로그 파일에 메시지를 로그할 수 있습니다. 이 API는 DB2 데이터베이스 시스템이 제공하므로 구현할 필요는 없습니다.

- db2secPluginTerm
- db2secGetGroupsForUser
- db2secDoesGroupExist
- db2secFreeGroupListMemory

- db2secFreeErrormsg
- 외부적으로 해결해야 하는 유일한 API는 db2secGroupPluginInit입니다. 이 API는 void * 매개변수를 사용하므로 해당 유형으로 캐스트해야 합니다.

```
typedef struct db2secGroupFunctions_1
{
db2int32 version;
db2int32 plugintype;
SQL_API_RC (SQL_API_FN * db2secGetGroupsForUser)
(
const char *authid,
db2int32    authidlen,
const char *userid,
db2int32    useridlen,
const char *usernamespace,
db2int32    usernamespacelen,
db2int32    usernamespacetype,
const char *dbname,
db2int32    dbnamelen,
const void *token,
db2int32    tokentype,
db2int32    location,
const char *authpluginname,
db2int32    authpluginnamelen,
void      **grouplist,
db2int32   *numgroups,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secDoesGroupExist)
(
const char *groupname,
db2int32    groupnamelen,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeGroupListMemory)
(
void     *ptr,
char    **errormsg,
db2int32 *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)
(
char *msgtobefree
);

SQL_API_RC (SQL_API_FN * db2secPluginTerm)
(
char    **errormsg,
db2int32 *errormsglen
);

} db2secGroupFunctions_1;
```

db2secGroupPluginInit API는 외부적으로 사용할 수 있는 나머지 함수에 주소를 지정합니다.

주: _1은 API 버전 1에 해당하는 구조임을 나타냅니다. 이후 인터페이스 버전에서의 확장명은 _2, _3 등이 사용됩니다.

---

db2secDoesGroupExist API - 그룹이 존재하는지 여부 확인

authid가 그룹을 나타내는지 여부를 판별합니다.

그룹 이름이 존재하면, API는 DB2SEC_PLUGIN_OK 값을 리턴하여 성공을 나타내야 합니다. 또한 그룹 이름이 유효하지 않으면
API가 DB2SEC_PLUGIN_INVALIDUSERORGROUP 값을 리턴할 수 있어야 합니다. 입력이 유효한 그룹인지 판별할 수 없는 경우 API가
DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN 값을 리턴할 수 있습니다. 유효하지 않은 그룹(DB2SEC_PLUGIN_INVALIDUSERORGROUP) 또는 알 수 없는 그룹(DB2SEC_PLUGIN_GROUPSTATUSNOTKNOWN) 값이 리턴되면, DB2는 USER 및 GROUP 키워드를 사용하지 않고 GRANT문을 발급할 때 authid가 그룹인지, 사용자인지 판별하지 못할 수 있습니다. 이로 인해 SQLCODE -569, SQLSTATE 56092 오류가 사용자에게 리턴됩니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secDoesGroupExist)
                       ( const char *groupname,
                         db2int32 groupnamelen,
                         char      **errormsg,
                         db2int32 *errormsglen );
```

db2secDoesGroupExist API 매개변수

groupname

입력. 뒤 공백이 없는 대문자 authid.

groupnamelen

입력. groupname 매개변수 값의 길이(바이트)

errormsg

출력. db2secDoesGroupExist API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

db2secFreeErrormsg API - 오류 메시지 메모리 제거

이전 API 호출의 오류 메시지를 보유하는 데 사용된 메모리를 제거합니다. 오류 메시지를 리턴하지 않는 유일한 API입니다. 이 API가 오류를 리턴하면 DB2가 이 오류를 로그하고 계속 진행합니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secFreeErrormsg)
                        ( char *errormsg );
```

db2secFreeErrormsg API parameters

msgtofree

입력. 이전 API 호출에서 할당된 메모리의 포인터

db2secFreeGroupListMemory API - 그룹 목록 메모리 제거

이전 db2secGetGroupsForUser API 호출의 그룹 목록을 보유하는 데 사용된 메모리를 제거합니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secFreeGroupListMemory)
                        ( void *ptr,
                          char      **errormsg,
                          db2int32 *errormsglen );
```

db2secFreeGroupListMemory API 매개변수

ptr 입력. 사용 가능해지는 메모리의 포인터

errormsg

출력. db2secFreeGroupListMemory API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

db2secGetGroupsForUser API - 사용자의 그룹 목록 가져오기

사용자가 속해 있는 그룹의 목록을 리턴합니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secGetGroupsForUser)
                        ( const char *authid,
                          db2int32 authidlen,
```

```
const char *userid,
db2int32 useridlen,
const char *usernamespace,
db2int32 usernamespacelen,
db2int32 usernamespacetype,
const char *dbname,
db2int32 dbnamelen,
void *token,
db2int32 tokentype,
db2int32 location,
const char *authpluginname,
db2int32 authpluginnamelen,
void      **grouplist,
db2int32 *numgroups,
char      **errormsg,
db2int32 *errormsglen );
```

db2secGetGroupsForUser API 매개변수

authid

입력. 이 매개변수 값은 SQL 권한 ID입니다. 즉, DB2가 이 값을 뒤에 공백이 없는 대문자 문자열로 변환합니다. DB2는 authid 매개변수에 항상 널(NULL)이 아닌 값을 제공합니다. API는 다른 입력 매개변수와 관계없이 권한 ID가 속하는 그룹 목록을 리턴해야 합니다. 판별할 수 없는 경우 간단한 목록 또는 빈 목록을 리턴할 수 있습니다.

사용자가 존재하지 않는 경우 API는 리턴 코드 DB2SEC_PLUGIN_BADUSER을 리턴해야 합니다. 권한 ID에 연관된 그룹이 없을 수 있으므로 DB2에서는 사용자가 존재하지 않는 상황을 오류로 취급하지 않습니다. 예를 들어, db2secGetAuthids API는 운영 체제에 존재하지 않는 권한 ID를 리턴할 수 있습니다. 권한 ID가 그룹에 연관되어 있지 않지만 직접 특권을 부여할 수 있기 때문입니다.

API가 권한 ID만 사용하여 그룹의 전체 목록을 리턴할 수 없으면, 그룹 지원과 관련된 특정 SQL 함수에 몇 가지 제한사항이 있습니다. 가능한 문제점 시나리오 목록을 보려면 이 주제에 있는 사용법 참고 사항 절을 참조하십시오.

authidlen

입력. authid 매개변수 값의 길이(바이트). DB2 데이터베이스 관리 프로그램은 authidlen 매개변수에 항상 0이 아닌 값을 제공합니다.

userid 입력. 권한 ID에 해당하는 사용자 ID. 이 API가 비연결 시나리오의 서버에서 호출되면, DB2가 이 매개변수를 입력하지 않습니다.

useridlen

입력. userid 매개변수 값의 길이(바이트)

usernamespace

입력. 사용자 ID를 가져온 이름 스페이스. 사용자 ID를 사용할 수 없으면, DB2 데이터베이스 관리 프로그램이 이 매개변수를 입력하지 않습니다.

usernamespacelen

입력. usernamespace 매개변수 값의 길이(바이트)

usernamespacetype

입력. 이름 스페이스 유형. db2secPlugin.h에 정의된 usernamespacetype 매개변수에 유효한 값은 다음과 같습니다.

- DB2SEC_NAMESPACE_SAM_COMPATIBLE: 사용자 이름 스타일에 해당 (예: domain₩myname)
- DB2SEC_NAMESPACE_USER_PRINCIPAL: 사용자 이름 스타일에 해당 (예: myname@domain.ibm.com)

현재 DB2 데이터베이스 시스템은
DB2SEC_NAMESPACE_SAM_COMPATIBLE 값만 지원합니다. 사용자 ID를 사용할 수 없는 경우, usernamespacetype 매개변수가 db2secPlugin.h에 정의된 DB2SEC_USER_NAMESPACE_UNDEFINED 값으로 설정됩니다.

dbname

입력. 연결할 데이터베이스의 이름. 비연결 시나리오에서 이 매개변수는 NULL이 될 수 있습니다.

dbnamelen

입력. dbname 매개변수 값의 길이(바이트). 비연결 시나리오에서 dbname 매개변수가 NULL이면 이 매개변수가 0으로 설정됩니다.

token 입력. 인증 플러그인이 제공하는 데이터의 포인터. DB2에는 사용되지 않습니다. 플러그인 기록기에 사용자 및 그룹 정보를 조정할 수 있는 기능을 제공합니다. 이 매개변수가 제공되지 않는 경우도 있습니다(예: 비연결 시나리오). 이 경우 값은 NULL이 됩니다. 사용되는 인증 플러그인이 GSS-API를 기반으로 하는 경우, 토큰이 GSS-API 컨텍스트 핸들(gss_ctx_id_t)로 설정됩니다.

tokentype

입력. 인증 플러그인이 제공하는 데이터의 유형을 나타냅니다. 사용되는 인증 플러그인이 GSS-API를 기반으로 하는 경우, 토큰이 GSS-API 컨텍스트 핸들(gss_ctx_id_t)로 설정됩니다. 사용되는 인증 플러그인이 사용자 ID/암호를 기반으로 하는 경우, 일반 유형이 됩니다. db2secPlugin.h에 정의된 tokentype 매개변수에 유효한 값은 다음과 같습니다.

- `DB2SEC_GENERIC`: 토큰이 사용자 ID/암호 기반 플러그인에서 제공됨을 나타냅니다.
- `DB2SEC_GSSAPI_CTX_HANDLE`: 토큰이 GSS-API(Kerberos 포함) 기반 플러그인에서 제공됨을 나타냅니다.

location

입력. DB2가 이 API를 클라이언트 측 또는 서버 측에서 호출하는지를 나타냅니다. db2secPlugin.h에 정의된 location 매개변수에 유효한 값은 다음과 같습니다.

- DB2SEC_SERVER_SIDE: API가 데이터베이스 서버에 호출됩니다.
- DB2SEC_CLIENT_SIDE: API가 클라이언트에서 호출됩니다.

authpluginname

입력. 토큰의 데이터를 제공한 인증 플러그인의 이름. db2secGetGroupsForUser API이 이 정보를 사용하여 올바른 그룹 멤버쉽을 판별해야 합니다. 권한 ID가 인증되지 않으면(예: 권한 ID가 현재 연결된 사용자와 일치하지 않는 경우) DB2가 이 매개변수를 입력하지 않습니다.

authpluginnamelen

입력. authpluginname 매개변수 값의 길이(바이트)

grouplist

출력. 사용자가 속해 있는 그룹의 목록. 그룹 목록은 병합된 varchar가 포함된 플러그인에서 할당한 메모리 섹션에 포인터로 리턴되어야 합니다(varchar는 문자 배열로서 첫 번째 바이트가 뒤에 오는 바이트 수를 나타냄). 길이는 서명되지 않은 char(1바이트)이며 그룹 이름을 최대 255자로 제한합니다(예: ″₩006GROUP1₩007MYGROUP₩008MYGROUP3″). 각 그룹 이름에는 유효한 DB2 권한 ID가 있어야 합니다. 이 배열에 사용되는 메모리는 플러그인이 할당해야 합니다. 따라서 플러그인은 API(예: DB2가 메모리를 제거하기 위해 호출하는 db2secFreeGroupListMemory API)를 제공해야 합니다.

numgroups

출력. grouplist 매개변수에 포함된 그룹의 수.

errormsg

출력. db2secGetGroupsForUser API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

사용 시 참고사항

다음은 이 API가 불완전한 그룹 목록을 DB2에 전달할 때 문제점이 발생하는 시나리오 목록입니다.

- 대체 권한 부여가 CREATE SCHEMA문에 제공됨. CREATE SCHEMA문에 중첩된 CREATE문이 있으면 AUTHORIZATION NAME 매개변수에 대해 그룹 찾아보기가 수행됩니다.
- MPP 환경에서 jar 파일 처리. MPP 환경에서, 세션 권한 ID가 있는 코디네이터 노드에서 jar 처리 요청이 전달됩니다. 카탈로그 노드는 이 요청을 받아 세션 권한 ID의 특권을 기반으로 jar 파일을 처리합니다(jar 처리 요청을 실행하는 사용자).
  - jar 파일 설치. 세션 권한 ID에는 DBADM 또는 CREATEIN 권한(jar 스키마에서 내재적 또는 명시적으로 부여) 중 하나가 있어야 합니다. 위의 권한이 세션 권한 ID가 있는 그룹에 부여되지만 세션 권한 ID에 명시적으로 부여되지 않으면 조작에 실패합니다.
  - jar 파일 제거. 세션 권한 ID에는 DBADM 또는 DROPIN 권한(jar 스키마에서 내재적 또는 명시적으로 부여) 중 하나가 있거나 jar 파일의 정의자여야 합니다. 위의 권한이 세션 권한 ID가 있는 그룹에 부여되지만 세션 권한 ID에 명시적으로 부여되지 않고, 세션 권한 ID가 jar 파일의 정의자가 아니면 조작에 실패합니다.
  - jar 파일 교체. jar 파일 제거와 동일하며 교체 후 jar 파일을 설치합니다. 위의 두 사항이 모두 적용됩니다.
- SET SESSION_USER문이 발행될 때. 이 명령문에 의해 지정된 권한 ID의 컨텍스트에 따라 후속 DB2 조작이 실행됩니다. SESSION_USER의 그룹 중 하나가 소유한 필수 특권이 SESSION_USER 권한 ID에 명시적으로 부여되지 않으면 이러한 조작에 실패합니다.

---

db2secGroupPluginInit API - 그룹 플러그인 초기화

플러그인 로딩 직후 DB2 관리 프로그램이 호출하는 그룹 검색 플러그인용 초기화 API

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN db2secGroupPluginInit
                              ( db2int32 version,
                                void *group_fns,
                                db2secLogMessage *logMessage_fn,
                                char     **errormsg,
                                db2int32 *errormsglen );
```

db2secGroupPluginInit API 매개변수

version

입력. 플러그인을 로딩하는 인스턴스에서 지원되는 가장 높은 버전의 API. db2secPlugin.h에 정의된 DB2SEC_API_VERSION 값에는 DB2 데이터베이스 관리 프로그램이 현재 지원하는 최신 버전의 API가 있습니다.

group_fns

출력. db2secGroupFunctions_<version_number>

(group_functions_<version_number>이라고도 함) 구조의 포인터. db2secGroupFunctions_<version_number> 구조에는 그룹 검색 플러그인에 구현된 API의 포인터가 있습니다. 향후에는 다른 버전의 API(예: db2secGroupFunctions_<version_number>)가 포함될 수 있으므로, group_fns 매개변수가 플러그인이 구현한 버전에 해당하는 db2secGroupFunctions_<version_number> 구조의 포인터로 캐스트됩니다. group_functions_<version_number> 구조의 첫 번째 매개변수는 플러그인이 구현한 API 버전을 DB2에 알려줍니다. 참고: 캐스팅은 DB2 버전이 플러그인이 구현한 API 버전과 같거나 높을 경우에만 수행됩니다. 버전 번호는 플러그인이 구현한 API의 버전을 나타내며 pluginType이 DB2SEC_PLUGIN_TYPE_GROUP으로 설정되어야 합니다.

logMessage_fn

입력. DB2 데이터베이스 시스템이 구현하는 db2secLogMessage API의 포인터. db2secGroupPluginInit API는 db2secLogMessage API를 호출하여 디버깅 또는 정보용으로 db2diag 로그 파일에 메시지를 로그할 수 있습니다. db2secLogMessage API의 첫 번째 매개변수(level)는 db2diag 로그 파일에 기록되는 진단 오류의 유형을 지정하고, 마지막 두 매개변수는 각각 메시지 문자열과 길이입니다. db2secPlugin.h에 정의된 dbesecLogMessage API의 첫 번째 매개변수의 올바른 값은 다음과 같습니다.

- DB2SEC_LOG_NONE: (0) 로깅 없음
- DB2SEC_LOG_CRITICAL: (1) 서버 오류 발견
- DB2SEC_LOG_ERROR: (2) 오류 발견
- DB2SEC_LOG_WARNING: (3) 경고
- DB2SEC_LOG_INFO: (4) 정보용

db2secLogMessage API의 'level' 매개변수 값이 diaglevel 데이터베이스 관리 프로그램 구성 매개변수 이하일 경우에만 diag.log 로그 파일에 메시지 텍스트가 표시됩니다. 예를 들어, DB2SEC_LOG_INFO 값을 사용하는 경우 diaglevel 데이터베이스 관리 프로그램 구성 매개변수가 4로 설정된 경우에만 db2diag 로그 파일에 메시지 텍스트가 표시됩니다.

errormsg

출력. db2secGroupPluginInit API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

db2secPluginTerm - 그룹 플러그인 자원 정리

그룹 검색 플러그인에 사용된 자원을 제거합니다.

이 API는 DB2 데이터베이스 관리 프로그램이 그룹 검색 플러그인을 언로드하기 직전에 호출합니다. 이 API는 플러그인 라이브러리가 보유하는 자원을 적절하게 정리하는 방법으로 구현해야 합니다. 예를 들어, 플러그인에서 할당한 메모리를 제거하고, 열려 있는 파일을 닫고, 네트워크 연결을 닫습니다. 플러그인은 자원 제거를 위해 자원을 추적해야 합니다. 이 API는 Windows 플랫폼에서는 호출되지 않습니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secPluginTerm)
                        ( char      **errormsg,
                          db2int32 *errormsglen );
```

db2secPluginTerm API 매개변수

errormsg

출력. db2secPluginTerm API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

제 163 장 사용자 ID/암호 인증 플러그인용 API

사용자 ID/암호 인증 플러그인 모듈에서는 다음과 같은 클라이언트 측 API를 구현해야 합니다.

- db2secClientAuthPluginInit

  주: db2secClientAuthPluginInit API는 다음과 같은 프로토타입을 사용하여 API에 포인터 *logMessage_fn을 입력으로 사용합니다.

  ```
  SQL_API_RC (SQL_API_FN db2secLogMessage)
  (
  db2int32 level,
  void    *data,
  db2int32 length
  );
  ```

  플러그인은 db2secLogMessage API를 사용하여 디버깅 또는 정보용으로 db2diag 로그 파일에 메시지를 로그할 수 있습니다. 이 API는 DB2 데이터베이스 시스템이 제공하므로 구현할 필요는 없습니다.

- db2secClientAuthPluginTerm
- db2secGenerateInitialCred(gssapi에만 사용됨)
- db2secRemapUserid(선택적)
- db2secGetDefaultLoginContext
- db2secValidatePassword
- db2secProcessServerPrincipalName(GSS-API에만 사용됨)
- db2secFreeToken(DLL에서 보유한 메모리를 제거하는 함수)
- db2secFreeErrormsg
- db2secFreeInitInfo
- 외부적으로 해결해야 하는 유일한 API는 db2secClientAuthPluginInit입니다. 이 API는 void * 매개변수를 사용하므로 다음 중 하나로 캐스트해야 합니다.

  ```
  typedef struct db2secUseridPasswordClientAuthFunctions_1
  {
  db2int32 version;
  db2int32 plugintype;

  SQL_API_RC (SQL_API_FN * db2secGetDefaultLoginContext)
  (
  char        authid[DB2SEC_MAX_AUTHID_LENGTH],
  db2int32   *authidlen,
  char        userid[DB2SEC_MAX_USERID_LENGTH],
  db2int32   *useridlen,
  db2int32    useridtype,
  char        usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
  ```

```
db2int32   *usernamespacelen,
db2int32   *usernamespacetype,
const char *dbname,
db2int32    dbnamelen,
void      **token,
char      **errormsg,
db2int32   *errormsglen
);
/* Optional */
SQL_API_RC (SQL_API_FN * db2secRemapUserid)
(
char        userid[DB2SEC_MAX_USERID_LENGTH],
db2int32   *useridlen,
char        usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32   *usernamespacelen,
db2int32   *usernamespacetype,
char        password[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32   *passwordlen,
char        newpassword[DB2SEC_MAX_PASSWORD_LENGTH],
db2int32   *newpasswordlen,
const char *dbname,
db2int32    dbnamelen,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secValidatePassword)
(
const char *userid,
db2int32    useridlen,
const char *usernamespace,
db2int32    usernamespacelen,
db2int32    usernamespacetype,
const char *password,
db2int32    passwordlen,
const char *newpassword,
db2int32    newpasswordlen,
const char *dbname,
db2int32    dbnamelen,
db2Uint32   connection_details,
void      **token,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeToken)
(
void     **token,
char     **errormsg,
db2int32  *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)
(
char *errormsg
);

SQL_API_RC (SQL_API_FN * db2secClientAuthPluginTerm)
```

```
(
char     **errormsg,
db2int32  *errormsglen
);
}
```

또는

```
typedef struct db2secGssapiClientAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;

SQL_API_RC (SQL_API_FN * db2secGetDefaultLoginContext)
(
char        authid[DB2SEC_MAX_AUTHID_LENGTH],
db2int32   *authidlen,
char        userid[DB2SEC_MAX_USERID_LENGTH],
db2int32   *useridlen,
db2int32    useridtype,
char        usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
db2int32   *usernamespacelen,
db2int32   *usernamespacetype,
const char *dbname,
db2int32    dbnamelen,
void      **token,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secProcessServerPrincipalName)
(
const void *data,
gss_name_t *gssName,
char      **errormsg,
db2int32   *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secGenerateInitialCred)
(
const char    *userid,
db2int32       useridlen,
const char    *usernamespace,
db2int32       usernamespacelen,
db2int32       usernamespacetype,
const char    *password,
db2int32       passwordlen,
const char    *newpassword,
db2int32       newpasswordlen,
const char    *dbname,
db2int32       dbnamelen,
gss_cred_id_t *pGSSCredHandle,
void         **initInfo,
char         **errormsg,
db2int32      *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeToken)
```

```
(
void      *token,
char     **errormsg,
db2int32  *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)
(
char *errormsg
);


SQL_API_RC (SQL_API_FN * db2secFreeInitInfo)
(
void      *initInfo,
char     **errormsg,
db2int32  *errormsglen
);

SQL_API_RC (SQL_API_FN * db2secClientAuthPluginTerm)
(
char     **errormsg,
db2int32  *errormsglen
);

/* GSS-API specific functions -- refer to db2secPlugin.h
   for parameter list*/

  OM_uint32 (SQL_API_FN * gss_init_sec_context )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_delete_sec_context )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_display_status )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_release_buffer )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_release_cred )(<parameter list>);
  OM_uint32 (SQL_API_FN * gss_release_name )(<parameter list>);
  }
```

사용자 ID/암호 플러그인을 작성하는 경우에는 `db2secUseridPasswordClientAuthFunctions_1` 구조를 사용해야 합니다. GSS-API(Kerberos 포함) 플러그인을 작성하는 경우에는 `db2secGssapiClientAuthFunctions_1` 구조를 사용해야 합니다.

사용자 ID/암호 플러그인 라이브러리의 경우, 다음과 같은 서버 측 API를 구현해야 합니다.

- `db2secServerAuthPluginInit`

  db2secServerAuthPluginInit API는 다음과 같은 프로토타입을 사용하여 db2secLogMessage API에 포인터 `*logMessage_fn`을, db2secGetConDetails API에 포인터 `*getConDetails_fn`을 입력으로 사용합니다.

```
SQL_API_RC (SQL_API_FN db2secLogMessage)
(
db2int32 level,
void    *data,
db2int32 length
```

```
);

SQL_API_RC (SQL_API_FN db2secGetConDetails)
(
db2int32    conDetailsVersion,
const void *pConDetails
);
```

플러그인은 db2secLogMessage API를 사용하여 디버깅 또는 정보용으로 db2diag 로그 파일에 메시지를 로그할 수 있습니다. 플러그인은 db2secGetConDetails API를 사용하여 데이터베이스 연결을 시도하는 클라이언트에 대한 세부사항을 가져올 수 있습니다. db2secLogMessage API 및 db2secGetConDetails API 둘 다 DB2 데이터베이스 시스템이 제공하므로, 이러한 API를 구현할 필요는 없습니다. 그런 다음 db2secGetConDetails API는 두 번째 매개변수로 다음과 같은 구조 중 하나의 포인터인 pConDetails를 사용합니다.

db2sec_con_details_1:

```
typedef struct db2sec_con_details_1
{
  db2int32  clientProtocol;
  db2Uint32 clientIPAddress;
  db2Uint32 connect_info_bitmap;
  db2int32  dbnameLen;
  char      dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
} db2sec_con_details_1;
```

db2sec_con_details_2:

```
typedef struct db2sec_con_details_2
{
  db2int32  clientProtocol;     /* See SQL_PROTOCOL_ in sqlenv.h */
  db2Uint32 clientIPAddress;    /* Set if protocol is TCPIP4     */
  db2Uint32 connect_info_bitmap;
  db2int32  dbnameLen;
  char dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
  db2Uint32 clientIP6Address[4];/* Set if protocol is TCPIP6     */
} db2sec_con_details_2;
```

db2sec_con_details_3:

```
typedef struct db2sec_con_details_3
{
  db2int32 clientProtocol; /* See SQL_PROTOCOL_ in sqlenv.h */
  db2Uint32 clientIPAddress; /* Set if protocol is TCPIP4 */
  db2Uint32 connect_info_bitmap;
  db2int32 dbnameLen;
  char dbname[DB2SEC_MAX_DBNAME_LENGTH + 1];
  db2Uint32 clientIP6Address[4];/* Set if protocol is TCPIP6 */
  db2Uint32 clientPlatform; /* SQLM_PLATFORM_* from sqlmon.h */
  db2Uint32 _reserved[16];
} db2sec_con_details_3;
```

conDetailsVersion에 사용 가능한 값은 DB2SEC_CON_DETAILS_VERSION_1, DB2SEC_CON_DETAILS_VERSION_2 및 DB2SEC_CON_DETAILS_VERSION_3로서 API의 버전을 나타냅니다.

주: db2sec_con_details_1, db2sec_con_details_2 또는 db2sec_con_details_3를 사용할 때 다음 사항을 고려하십시오.

- db2sec_con_details_1 구조와 DB2SEC_CON_DETAILS_VERSION_1 값을 사용하는 기존 플러그인은 버전 8.2에서 작동되었으므로 db2GetConDetails API를 호출할 때 계속 작동됩니다. 이 API가 IPv4 플랫폼에서 호출되면, 클라이언트 IP 주소가 해당 구조의 clientIPAddress 필드에 리턴됩니다. 이 API가 IPv6 플랫폼에서 호출되면, clientIPAddress 필드에 0 값이 리턴됩니다. IPv6 플랫폼에서 클라이언트 IP 주소를 검색하려면, db2sec_con_details_2 구조와 DB2SEC_CON_DETAILS_VERSION_2 값 또는 db2sec_con_details_3 구조와 DB2SEC_CON_DETAILS_VERSION_3 값을 사용하도록 보안 플러그인 코드를 변경해야 합니다 .
- 새 플러그인은 db2sec_con_details_3 구조와 DB2SEC_CON_DETAILS_VERSION_3 값을 사용해야 합니다. db2secGetConDetails API가 IPv4 플랫폼에서 호출되면 클라이언트 IP 주소가 db2sec_con_details_3 구조의 clientIPAddress 필드에 리턴되고, API가 IPv6 플랫폼에서 호출되면 클라이언트 IP 주소가 db2sec_con_details_3 구조의 clientIP6Address 필드에 리턴됩니다. 연결 세부사항 구조의 *clientProtocol* 필드가 SQL_PROTOCOL_TCPIP(IPv4, v1의 구조 사용), SQL_PROTOCOL_TCPIP4(IPv4, v2의 구조 사용) 또는 SQL_PROTOCOL_TCPIP6(IPv6, v2 또는 v3의 구조 사용) 중 하나로 설정됩니다.
- db2sec_con_details_3 구조는 sqlmon.h에 정의된 플랫폼 유형 상수(예: SQLM_PLATFORM_AIX)를 사용하여 클라이언트 플랫폼 유형(통신 계층이 보고함)을 식별하는 추가 필드(*clientPlatform*)가 포함된 것을 제외하고 db2sec_con_details_2 구조와 동일합니다.

- db2secServerAuthPluginTerm
- db2secValidatePassword
- db2secGetAuthIDs
- db2secDoesAuthIDExist
- db2secFreeToken
- db2secFreeErrormsg
- 외부적으로 해결해야 하는 유일한 API는 db2secServerAuthPluginInit입니다. 이 API는 void * 매개변수를 사용하므로 다음 중 하나로 캐스트해야 합니다.

```
typedef struct db2secUseridPasswordServerAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;

  /* parameter lists left blank for readability
    see above for parameters */
SQL_API_RC (SQL_API_FN * db2secValidatePassword)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secGetAuthIDs)(<parameter list);
SQL_API_RC (SQL_API_FN * db2secDoesAuthIDExist)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secFreeToken)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secServerAuthPluginTerm)();
} userid_password_server_auth_functions;
```

또는

```
typedef struct db2secGssapiServerAuthFunctions_1
{
db2int32 version;
db2int32 plugintype;
gss_buffer_desc serverPrincipalName;
gss_cred_id_t ServerCredHandle;
SQL_API_RC (SQL_API_FN * db2secGetAuthIDs)(<parameter list);
SQL_API_RC (SQL_API_FN * db2secDoesAuthIDExist)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secFreeErrormsg)(<parameter list>);
SQL_API_RC (SQL_API_FN * db2secServerAuthPluginTerm)();

/* GSS-API specific functions
refer to db2secPlugin.h for parameter list*/
OM_uint32 (SQL_API_FN * gss_accept_sec_context )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_display_name )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_delete_sec_context )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_display_status )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_release_buffer )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_release_cred )(<parameter list>);
OM_uint32 (SQL_API_FN * gss_release_name )(<parameter list>);

} gssapi_server_auth_functions;
```

사용자 ID/암호 플러그인을 작성하는 경우에는 `db2secUseridPasswordServerAuthFunctions_1` 구조를 사용해야 합니다. GSS-API(Kerberos 포함) 플러그인을 작성하는 경우에는 `db2secGssapiServerAuthFunctions_1` 구조를 사용해야 합니다.

db2secClientAuthPluginInit API - 클라이언트 인증 플러그인 초기화

플러그인 로딩 직후 DB2 데이터베이스 관리 프로그램이 호출하는 클라이언트 인증 플러그인용 초기화 API

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN db2secClientAuthPluginInit
                              ( db2int32 version,
                                void *client_fns,
```

```
db2secLogMessage *logMessage_fn,
char    **errormsg,
db2int32 *errormsglen );
```

db2secClientAuthPluginInit API 매개변수

version

입력. 현재 DB2 데이터베이스 관리 프로그램을 지원하는 가장 높은 버전의 API. db2secPlugin.h에 정의된 DB2SEC_API_VERSION 값에는 현재 DB2를 지원하는 최신 버전의 API가 있습니다.

client_fns

출력. GSS-API 인증이 사용되는 경우 db2secGssapiClientAuthFunctions_<version_number> 구조(gssapi_client_auth_functions_<version_number>라고도 함), 사용자 ID/암호 인증이 사용되는 경우 db2secUseridPasswordClientAuthFunctions_<version_number> 구조(userid_password_client_auth_functions_<version_number>라고도 함)용으로 DB2 데이터베이스 관리 프로그램에서 제공하는 메모리의 포인터. db2secGssapiClientAuthFunctions_<version_number> 구조 및 db2secUseridPasswordClientAuthFunctions_<version_number> 구조에는 각각 GSS-API 인증 플러그인 및 사용자 ID/암호 인증 플러그인에 구현된 API의 포인터가 있습니다. 향후 DB2 버전에는 다른 버전의 API가 포함되므로, client_fns 매개변수는 플러그인이 구현한 버전에 해당하는 gssapi_client_auth_functions_<version_number> 구조의 포인터로 캐스트됩니다.

gssapi_client_auth_functions_<version_number> 구조 또는 userid_password_client_auth_functions_<version_number> 구조의 첫 번째 매개변수는 플러그인이 구현한 API 버전을 DB2 데이터베이스 관리 프로그램에 알려줍니다.

주: 캐스팅은 DB2 버전이 플러그인이 구현한 API 버전과 같거나 높을 경우에만 수행됩니다.

gssapi_server_auth_functions_<version_number> 또는 userid_password_server_auth_functions_<version_number> 구조 내에서 plugintype 매개변수는 DB2SEC_PLUGIN_TYPE_USERID_PASSWORD, DB2SEC_PLUGIN_TYPE_GSSAPI 또는 DB2SEC_PLUGIN_TYPE_KERBEROS로 설정되어야 합니다. 향후 버전의 API에서는 다른 값이 정의될 수 있습니다.

logMessage_fn

입력. DB2 데이터베이스 관리 프로그램이 구현하는 db2secLogMessage API의 포인터. db2secClientAuthPluginInit API는 db2secLogMessage API를 호

출하여 디버깅 또는 정보용으로 db2diag 로그 파일에 메시지를 로그할 수 있습니다. db2secLogMessage API의 첫 번째 매개변수(level)는 db2diag 로그 파일에 기록되는 진단 오류의 유형을 지정하고, 마지막 두 매개변수는 각각 메시지 문자열과 길이입니다. db2secPlugin.h에 정의된 dbesecLogMessage API의 첫 번째 매개변수의 올바른 값은 다음과 같습니다.

- DB2SEC_LOG_NONE (0) 로깅 없음
- DB2SEC_LOG_CRITICAL (1) 서버 오류 발견
- DB2SEC_LOG_ERROR (2) 오류 발견
- DB2SEC_LOG_WARNING (3) 경고
- DB2SEC_LOG_INFO (4) 정보용

db2secLogMessage API의 'level' 매개변수 값이 diaglevel 데이터베이스 관리 프로그램 구성 매개변수 이하일 경우에만 db2diag 로그 파일에 메시지 텍스트가 표시됩니다. 예를 들어, DB2SEC_LOG_INFO 값을 사용하는 경우 diaglevel 데이터베이스 관리 프로그램 구성 매개변수가 4로 설정된 경우에만 db2diag 로그 파일에 메시지 텍스트가 표시됩니다.

errormsg

출력. db2secClientAuthPluginInit API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

---

db2secClientAuthPluginTerm API - 클라이언트 인증 플러그인 자원 정리

클라이언트 인증 플러그인에 사용된 자원을 제거합니다.

이 API는 DB2 데이터베이스 관리 프로그램이 클라이언트 인증 플러그인을 언로드하기 직전에 호출합니다. 이 API는 플러그인 라이브러리가 보유하는 자원을 적절하게 정리하는 방법으로 구현해야 합니다. 예를 들어, 플러그인에서 할당한 메모리를 제거하고, 열려 있는 파일을 닫고, 네트워크 연결을 닫습니다. 플러그인은 자원 제거를 위해 자원을 추적해야 합니다. 이 API는 Windows 플랫폼에서는 호출되지 않습니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secClientAuthPluginTerm)
                        ( char     **errormsg,
                          db2int32 *errormsglen);
```

db2secClientAuthPluginTerm API 매개변수

errormsg

출력. db2secClientAuthPluginTerm API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

---

db2secDoesAuthIDExist - 인증 ID가 존재하는지 여부 확인

권한 ID가 개별 사용자를 나타내는지 여부를 판별합니다(예: API가 권한 ID를 외부 사용자 ID에 맵핑할 수 있는지 여부)

API는 권한 ID가 유효한 경우(성공) DB2SEC_PLUGIN_OK, 유효하지 않은 경우 DB2SEC_PLUGIN_INVALID_USERORGROUP, 권한 ID 존재를 판별할 수 없는 경우 DB2SEC_PLUGIN_USERSTATUSNOTKNOWN을 리턴합니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secDoesAuthIDExist)
                       ( const char *authid,
                         db2int32 authidlen,
                         char       **errormsg,
                         db2int32 *errormsglen );
```

db2secDoesAuthIDExist API 매개변수

authid

입력. 유효성을 확인할 권한 ID. 대문자이며 뒤 공백이 없습니다.

authidlen

입력. authid 매개변수 값의 길이(바이트)

errormsg

출력. db2secDoesAuthIDExist API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이를 나타내는 정수의 포인터

db2secFreeInitInfo API - db2secGenerateInitialCred에서 보유한 자원 정리

db2secGenerateInitialCred API에서 할당한 자원을 제거합니다. 예를 들어, 여기에는 GSS-API 증명서 캐시용으로 작성된 증명서 캐시 또는 기본 메커니즘 컨텍스트의 핸들이 포함됩니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secFreeInitInfo)
                        ( void *initinfo,
                          char     **errormsg,
                          db2int32 *errormsglen);
```

db2secFreeInitInfo API 매개변수

initinfo

입력. DB2 데이터베이스 관리 프로그램에 알려지지 않은 데이터의 포인터. 플러그인은 이 메모리를 사용하여 증명서 핸들 작성 프로세스에 할당되는 자원 목록을 유지할 수 있습니다. 이러한 자원은 이 API를 호출하면 제거됩니다.

errormsg

출력. db2secFreeInitInfo API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

db2secFreeToken API - 토큰에서 보유한 메모리 제거

토큰에서 보유한 메모리를 제거합니다. 이 API는 DB2 데이터베이스 관리 프로그램이 토큰 매개변수가 보유한 메모리가 더 이상 필요하지 않을 때 호출합니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secFreeToken)
                        ( void *token,
                          char       **errormsg,
                          db2int32 *errormsglen );
```

db2secFreeToken API 매개변수

token 입력. 사용 가능해지는 메모리의 포인터

errormsg

출력. db2secFreeToken API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

db2secGenerateInitialCred API - 초기 증명서 생성

db2secGenerateInitialCred API는 GSS-API 전달되는 사용자 ID 및 암호를 기준으로 초기 증명서를 가져옵니다.

Kerberos의 경우, TGT(Ticket-Granting Ticket)입니다. pGSSCredHandle 매개변수에 리턴되는 증명서 핸들은 gss_init_sec_context API에 사용되는 핸들로서 INITIATE 또는 BOTH 증명서 중 하나여야 합니다. db2secGenerateInitialCred API는 사용자 ID 및 암호가 제공될 때만 호출됩니다. 그렇지 않으면, DB2 데이터베이스 관리 프로그램은 gss_init_sec_context API를 호출할 때 GSS_C_NO_CREDENTIAL 값을 지정하여 현재 로그인 컨텍스트에서 가져온 디폴트 증명서가 사용됨을 나타냅니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secGenerateInitialCred)
                          ( const char *userid,
                            db2int32 useridlen,
                            const char *usernamespace,
                            db2int32 usernamespacelen,
                            db2int32 usernamespacetype,
                            const char *password,
                            db2int32 passwordlen,
                            const char *newpassword,
                            db2int32 newpasswordlen,
                            const char *dbname,
                            db2int32 dbnamelen,
                            gss_cred_id_t *pGSSCredHandle,
                            void          **InitInfo,
                            char          **errormsg,
                            db2int32 *errormsglen );
```

db2secGenerateInitialCred API 매개변수

userid 입력. 데이터베이스 서버에서 암호의 유효성을 확인할 사용자 ID

useridlen

입력. userid 매개변수 값의 길이(바이트)

usernamespace

입력. 사용자 ID를 가져온 이름 스페이스

usernamespacelen

입력. usernamespace 매개변수 값의 길이(바이트)

usernamespacetype

입력. 이름 스페이스 유형

password

입력. 유효성을 확인할 암호.

passwordlen

입력. password 매개변수 값의 길이(바이트)

newpassword

입력. 새 암호(암호가 변경되는 경우). 암호 변경 요청이 없는 경우 newpassword 매개변수는 NULL로 설정됩니다. 이 매개변수가 NULL로 설정되지 않으면, 암호를 새 값으로 설정하기 전에 API가 이전 암호의 유효성을 확인합니다. API가 암호 변경 요청을 이행할 필요는 없지만, 이행하지 않을 경우 이전 암호의 유효성을 확인하지 않고 리턴 값 DB2SEC_PLUGIN_CHANGEPASSWORD_NOTSUPPORTED를 즉시 리턴해야 합니다.

newpasswordlen

입력. newpassword 매개변수 값의 길이(바이트)

dbname

입력. 연결할 데이터베이스의 이름. API가 이 매개변수를 무시하거나, 유효한 암호를 가진 사용자의 특정 데이터베이스에 대한 액세스를 제한하는 규정이 있는 경우 리턴 값 DB2SEC_PLUGIN_CONNECTION_DISALLOWED를 리턴할 수 있습니다.

dbnamelen

입력. dbname 매개변수 값의 길이(바이트)

pGSSCredHandle

출력. GSS-API 증명서 핸들의 포인터

InitInfo

출력. DB2에 알려지지 않은 데이터의 포인터. 플러그인은 이 메모리를 사용하여 증명서 핸들 작성 프로세스에 할당되는 자원 목록을 유지할 수 있습니다. DB2 데이터베이스 관리 프로그램은 이러한 자원을 사용할 수 있는 인증 프로세스 마지막 부분에서 db2secFreeInitInfo API를 호출합니다. db2secGenerateInitialCred API가 이 목록을 유지할 필요가 없으면 NULL을 리턴합니다.

errormsg

출력. db2secGenerateInitialCred API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터.

주: 이 API의 경우, 리턴 값이 잘못된 사용자 ID 또는 암호를 나타내는 경우에는 오류 메시지가 작성되지 않습니다. API가 올바르게 완료되지 못하도록 방해하는 내부 오류가 있을 경우에만 오류 메시지가 리턴됩니다.

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

---

db2secGetAuthIDs API - 인증 ID 가져오기

인증된 사용자에게 SQL 권한 ID를 리턴합니다. 이 API는 데이터베이스에 연결되어 있는 동안 사용자 ID/암호 및 GSS-API 인증 방법에 호출됩니다.

API 및 데이터 구조 구문

```
  SQL_API_RC ( SQL_API_FN *db2secGetAuthIDs)
( const char *userid,
  db2int32 useridlen,
  const char *usernamespace,
  db2int32 usernamespacelen,
  db2int32 usernamespacetype,
  const char *dbname,
  db2int32 dbnamelen,
  void      **token,
  char SystemAuthID[DB2SEC_MAX_AUTHID_LENGTH],
  db2int32 *SystemAuthIDlen,
  char InitialSessionAuthID[DB2SEC_MAX_AUTHID_LENGTH],
  db2int32 *InitialSessionAuthIDlen,
  char username[DB2SEC_MAX_USERID_LENGTH],
  db2int32 *usernamelen,
  db2int32 *initsessionidtype,
  char      **errormsg,
  db2int32 *errormsglen );
```

db2secGetAuthIDs API 매개변수

userid 입력. 인증된 사용자. 인증 없이 사용자 조작 전환이 허용되도록 트러스트된 컨텍스트가 정의되지 않은 경우에는 일반적으로 GSS-API 인증에는 사용되지 않습니다. 이 경우, 사용자 전환 요청에 제공된 사용자 이름이 이 매개변수에 전달됩니다.

useridlen

입력. userid 매개변수 값의 길이(바이트)

usernamespace

입력. 사용자 ID를 가져온 이름 스페이스

usernamespacelen

입력. usernamespace 매개변수 값의 길이(바이트)

usernamespacetype

입력. Namespacetype 값. 현재 지원되는 유일한 이름 스페이스 유형 값은 DB2SEC_NAMESPACE_SAM_COMPATIBLE입니다 (사용자 이름 스타일에 해당, 예: domain₩myname)

dbname

입력. 연결할 데이터베이스의 이름. API는 이 값을 무시하거나, 동일한 사용자가 다른 데이터베이스에 연결할 때 다른 권한 ID를 리턴할 수 있습니다. 이 매개변수는 NULL이 될 수 있습니다.

dbnamelen

입력. dbname 매개변수 값의 길이(바이트). dbname 매개변수가 NULL이면 이 매개변수가 0으로 설정됩니다.

token 입력 또는 출력. 플러그인이 db2secGetGroupsForUser API에 전달하는 데이터. GSS-API의 경우, 컨텍스트 핸들(gss_ctx_id_t)입니다. 일반적으로 token은 입력 전용 매개변수이며 이 값은 db2secValidatePassword API에서 비롯됩니다. 또한 클라이언트에서 인증이 수행되면 출력 매개변수도 될 수 있으며 이 경우 db2secValidatePassword API가 호출되지 않습니다. 트러스트된 컨텍스트가 정의된 환경(인증 없이 사용자 전환 조작이 허용됨)에서, db2secGetAuthIDs API는 이 token 매개변수의 NULL 값을 수신할 수 있어야 하며 위의 userid 및 useridlen 매개변수를 기반으로 시스템 권한 부여 ID를 파생할 수 있어야 합니다.

SystemAuthID

출력. 인증된 사용자의 ID에 해당하는 시스템 권한 부여 ID. 크기는 255바이트이지만 DB2 데이터베이스 관리 프로그램은 현재 최대 30바이트까지 지원합니다.

SystemAuthIDlen

출력. SystemAuthID 매개변수 값의 길이(바이트)

InitialSessionAuthID

출력. 이 연결 세션에 사용된 권한 ID. 이 매개변수는 일반적으로 SystemAuthID 매개변수와 동일하지만, 다른 경우도 있습니다(예: SET SESSION AUTHORIZATION문 발행). 크기는 255바이트이지만 DB2 데이터베이스 관리 프로그램은 현재 최대 30바이트까지 지원합니다.

InitialSessionAuthIDlen

출력. InitialSessionAuthID 매개변수 값의 길이(바이트)

username

출력. 인증된 사용자 및 권한 ID에 해당하는 사용자 이름. 이 매개변수는 감사에만 사용되며 CONNECT문의 감사 레코드에 있는 ″사용자 ID″ 필드에 로그됩니다. 이 API가 username 매개변수에 채워지지 않으면 DB2 데이터베이스 관리 프로그램이 userid에서 이 매개변수를 복사합니다.

usernamelen

출력. username 매개변수 값의 길이(바이트)

initsessionidtype

출력. InitialSessionAuthid 매개변수가 역할 또는 권한 ID인지 여부를 나타내는 세션 권한 ID 유형. API는 db2secPlugin.h에 정의된 다음 값 중 하나를 리턴합니다.

- DB2SEC_ID_TYPE_AUTHID (0)
- DB2SEC_ID_TYPE_ROLE (1)

errormsg

출력. db2secGetAuthIDs API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

---

db2secGetDefaultLoginContext API - 디폴트 로그인 컨텍스트 가져오기

디폴트 로그인 컨텍스트와 연관된 사용자를 판별합니다. 즉, 사용자 ID를 명시적으로 지정하지 않고(데이터베이스에 대한 내재적 인증 또는 로컬 인증) DB2 명령을 호출하는 사용자의 DB2 권한 ID를 판별합니다. 이 API는 권한 ID와 사용자 ID 둘 다 리턴해야 합니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secGetDefaultLoginContext)
                      ( char authid[DB2SEC_MAX_AUTHID_LENGTH],
                        db2int32 *authidlen,
                        char userid[DB2SEC_MAX_USERID_LENGTH],
                        db2int32 *useridlen,
                        db2int32 useridtype,
                        char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
                        db2int32 *usernamespacelen,
                        db2int32 *usernamespacetype,
                        const char *dbname,
                        db2int32 dbnamelen,
                        void      **token,
                        char      **errormsg,
                        db2int32 *errormsglen );
```

db2secGetDefaultLoginContext API 매개변수

authid

출력. 권한 ID를 리턴하는 매개변수. 리턴된 값은 DB2 권한 ID 이름 지정 규칙을 따라야 합니다. 그렇지 않으면 사용자는 요청된 조치를 수행할 권한을 부여 받지 못합니다.

authidlen

출력. authid 매개변수 값의 길이(바이트)

userid 출력. 디폴트 로그인 컨텍스트와 연관된 사용자 ID가 리턴되는 매개변수

useridlen

출력. userid 매개변수 값의 길이(바이트)

useridtype

입력. 프로세스의 실 사용자 ID 또는 유효 사용자 ID가 지정되는지를 나타냅니다. Windows에서는 실 사용자 ID만 존재합니다. UNIX 및 Linux에서, 응용프로그램의 `uid` 사용자 ID가 프로세스를 실행하는 사용자의 ID와 다를 경우 실 사용자 ID와 유효 ID가 다를 수 있습니다. db2secPlugin.h에 정의된 userid 매개변수에 유효한 값은 다음과 같습니다.

DB2SEC_PLUGIN_REAL_USER_NAME

실 사용자 ID가 지정됨을 나타냅니다.

DB2SEC_PLUGIN_EFFECTIVE_USER_NAME

유효 사용자 ID가 지정됨을 나타냅니다.

주: 일부 플러그인 구현에서는 실 사용자 ID와 유효 사용자 ID를 구분하지 않을 수 있습니다. 특히, 사용자의 UNIX 또는 Linux ID를 사용하여 DB2 권한 부여 ID를 설정하지 않는 플러그인에서는 이 두 사용자 ID의 차이가 무시됩니다.

usernamespace

출력. 사용자 ID의 이름 스페이스

usernamespacelen

출력. usernamespace 매개변수 값의 길이(바이트). usernamespacetype 매개변수를 db2secPlugin.h에 정의된 DB2SEC_NAMESPACE_SAM_COMPATIBLE 값으로 설정해야 한다는 제한사항에 따라 현재 지원되는 최대 길이는 15바이트입니다.

usernamespacetype

출력. Namespacetype 값. 현재 지원되는 유일한 이름 스페이스 유형은 DB2SEC_NAMESPACE_SAM_COMPATIBLE입니다(사용자 이름 스타일에 해당, 예: domain₩myname)

dbname

입력. 이 호출이 데이터베이스 연결의 컨텍스트에 사용되는 경우, 연결할 데이터베이스의 이름이 포함됩니다. 로컬 인증 조치 또는 인스턴스 첨부의 경우, 이 매개변수는 NULL로 설정됩니다.

dbnamelen

입력. dbname 매개변수 값의 길이(바이트)

token 출력. 플러그인이 할당하는 데이터의 포인터로서 플러그인에 있는 후속 인증 호출 또는 그룹 검색 플러그인에 전달할 수 있습니다. 이 데이터의 구조는 플러그인 기록기에 의해 판별됩니다.

errormsg

출력. db2secGetDefaultLoginContext API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

---

db2secProcessServerPrincipalName API - 서버에서 리턴된 서비스 핵심부 이름 처리

db2secProcessServerPrincipalName API는 서버에서 리턴된 서비스 핵심부 이름을 처리하며 gss_init_sec_context API에 사용되는 gss_name_t 내부 형식으로 핵심부 이름을 리턴합니다.

또한 db2secProcessServerPrincipalName API는 Kerberos 인증이 사용될 때 데이터베이스 디렉토리에 카탈로그된 서비스 핵심부 이름을 처리합니다. 일반적으로 이 변환에서는 gss_import_name API를 사용합니다. 컨텍스트가 설정된 후 gss_name_t 오브젝트는 gss_release_name API가 호출되면 제거됩니다. gssName 매개변수가 유효한 GSS 이름을 나타내면 db2secProcessServerPrincipalName API는 DB2SEC_PLUGIN_OK 값을 리턴하고 핵심부 이름이 유효하지 않으면 DB2SEC_PLUGIN_BAD_PRINCIPAL_NAME 오류 코드를 리턴합니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secProcessServerPrincipalName)
                      ( const char *name,
                        db2int32 namelen,
                        gss_name_t *gssName,
                        char       **errormsg,
                        db2int32 *errormsglen );
```

db2secProcessServerPrincipalName API 매개변수

name 입력. GSS_C_NT_USER_NAME 형식의 서비스 핵심부 이름(예: service/host@REALM)

namelen

입력. name 매개변수 값의 길이(바이트)

gssName

출력. GSS-API 내부 형식의 출력 서비스 핵심부 이름의 포인터

errormsg

출력. db2secProcessServerPrincipalName API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

---

db2secRemapUserid API - 사용자 ID 및 암호 다시 맵핑

이 API는 DB2 데이터베이스 관리 프로그램이 클라이언트 측에서 호출하여 지정된 사용자 ID 및 암호(새 암호 및 사용자 이름 스페이스 포함)를 연결 시에 지정된 값과 다른 값에 다시 맵핑합니다.

DB2 데이터베이스 관리 프로그램은 연결 시에 사용자 ID 및 암호가 제공된 경우에만 이 API를 호출합니다. 이렇게 하면 플러그인에서 사용자 ID가 자동으로 사용자 ID/암호 쌍에 다시 맵핑되지 않습니다. 이 API는 선택적이며 보안 플러그인에서 제공하거나 구현하지 않으면 호출되지 않습니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secRemapUserid)
                    ( char userid[DB2SEC_MAX_USERID_LENGTH],
                      db2int32 *useridlen,
                      char usernamespace[DB2SEC_MAX_USERNAMESPACE_LENGTH],
                      db2int32 *usernamespacelen,
                      db2int32 *usernamespacetype,
                      char password[DB2SEC_MAX_PASSWORD_LENGTH],
                      db2int32 *passwordlen,
                      char newpasswd[DB2SEC_MAX_PASSWORD_LENGTH],
                      db2int32 *newpasswdlen,
                      const char *dbname,
                      db2int32 dbnamelen,
                      char      **errormsg,
                      db2int32 *errormsglen);
```

db2secRemapUserid API 매개변수

userid 입력 또는 출력. 다시 맵핑되는 사용자 ID. 입력 사용자 ID 값이 없으면, API가 입력 사용자 ID 값과 동일하거나 다른 출력 사용자 ID 값을 제공해야 합니다. 입력 사용자 ID 값이 없으면, API가 출력 사용자 ID 값을 리턴하지 않습니다.

useridlen

입력 또는 출력. userid 매개변수 값의 길이(바이트)

usernamespace

입력 또는 출력. 사용자 ID의 이름 스페이스. 이 값은 선택적으로 다시 맵핑할 수 있습니다. 입력 매개변수 값이 지정되지 않았지만 출력 값이 리턴되는 경우,

DB2 데이터베이스 관리 프로그램이 CLIENT 유형 인증에 usernamespace만 사용하며 다른 인증 유형에서는 무시됩니다.

usernamespacelen

입력 또는 출력. usernamespace 매개변수 값의 길이(바이트). usernamespacetype 매개변수를 db2secPlugin.h에 정의된 DB2SEC_NAMESPACE_SAM_COMPATIBLE 값으로 설정해야 한다는 제한사항에 따라 현재 지원되는 최대 길이는 15바이트입니다.

usernamespacetype

입력 또는 출력. 이전 및 새 namespacetype 값. 현재 지원되는 유일한 이름 스페이스 유형 값은 DB2SEC_NAMESPACE_SAM_COMPATIBLE입니다 (사용자 이름 스타일에 해당, 예: domain₩myname)

password

입력 또는 출력. 입력으로서는 다시 맵핑되는 암호이며 출력으로서는 다시 맵핑된 암호입니다. 이 매개변수에 출력 값이 지정되면, API는 입력 값과 다른 출력 값을 리턴해야 합니다. 출력 값이 지정되지 않으면, API는 출력 암호 값을 리턴하지 않아야 합니다.

passwordlen

입력 또는 출력. password 매개변수 값의 길이(바이트)

newpasswd

입력 또는 출력. 입력으로서는 설정되는 새 암호이며 출력으로서는 확인된 새 암호입니다.

주: DB2 데이터베이스 관리 프로그램이 클라이언트 또는 서버에서(인증 데이터베이스 관리 프로그램 구성 매개변수의 값에 따라) db2secValidatePassword API의 newpassword 매개변수에 전달하는 새 암호입니다. 새 암호가 입력으로 전달되면, API가 출력 값을 리턴해야 하며 다른 새 암호가 될 수 있습니다. 입력으로 전달된 새 암호가 없으면, API가 출력 새 암호를 리턴하지 않아야 합니다.

newpasswdlen

입력 또는 출력. newpasswd 매개변수 값의 길이(바이트)

dbname

입력. 클라이언트가 연결되는 데이터베이스의 이름

dbnamelen

입력. dbname 매개변수 값의 길이(바이트)

errormsg

출력. db2secRemapUserid API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

db2secServerAuthPluginInit - 서버 인증 플러그인 초기화

db2secServerAuthPluginInit API는 플러그인 로딩 직후 DB2 데이터베이스 관리 프로그램이 호출하는 서버 인증 플러그인용 초기화 API입니다.

GSS-API의 경우, 플러그인은 초기화할 때 gssapi_server_auth_functions 구조 내에서 serverPrincipalName 매개변수에 서버의 핵심부 이름을 입력하고, the gssapi_server_auth_functions 구조 내에서 serverCredHandle 매개변수에 서버의 증명서 핸들을 제공해야 합니다. 핵심부 이름과 증명서 핸들을 보유하기 위해 할당된 메모리 제거 작업은 gss_release_name 및 gss_release_cred API를 호출하여 db2secServerAuthPluginTerm API가 수행해야 합니다.

API 및 데이터 구조 구문

```
SQL_API_RC SQL_API_FN db2secServerAuthPluginInit
                              ( db2int32 version,
                                void *server_fns,
                                db2secGetConDetails *getConDetails_fn,
                                db2secLogMessage *logMessage_fn,
                                char               **errormsg,
                                db2int32 *errormsglen );
```

db2secServerAuthPluginInit API 매개변수

version

입력. 현재 DB2 데이터베이스 관리 프로그램을 지원하는 가장 높은 버전의 API. db2secPlugin.h에 정의된 DB2SEC_API_VERSION 값에는 DB2 데이터베이스 관리 프로그램이 현재 지원하는 최신 버전의 API가 있습니다.

server_fns

출력. GSS-API 인증이 사용되는 경우 db2secGssapiServerAuthFunctions_<version_number> 구조(gssapi_server_auth_functions_<version_number>라고도 함), 사용자 ID/암호 인증이 사용되는 경우 db2secUseridPasswordServerAuthFunctions_<version_number> 구조(userid_password_server_auth_functions_<version_number>라고도 함)용으로 DB2 데이터베이스 관리 프로그램에서 제공하는 메모리의 포인터. db2secGssapiServerAuthFunctions_<version_number> 구조 및 db2secUseridPasswordServerAuthFunctions_<version_number> 구조에는 각각 GSS-API 인증 플러그인 및 사용자 ID/암호 인증 플러그인에 구현된 API의 포인터가 있습니다.

server_fns 매개변수는 플러그인이 구현한 버전에 해당하는 gssapi_server_auth_functions_<version_number> 구조의 포인터로 캐스트됩니다. gssapi_server_auth_functions_<version_number> 구조 또는 userid_password_server_auth_functions_<version_number> 구조의 첫 번째 매개변수는 플러그인이 구현한 API 버전을 DB2 데이터베이스 관리 프로그램에 알려줍니다.

주: 캐스팅은 DB2 버전이 플러그인이 구현한 API 버전과 같거나 높을 경우에만 수행됩니다.

gssapi_server_auth_functions_<version_number> 또는 userid_password_server_auth_functions_<version_number> 구조 내에서 plugintype 매개변수는 DB2SEC_PLUGIN_TYPE_USERID_PASSWORD, DB2SEC_PLUGIN_TYPE_GSSAPI 또는 DB2SEC_PLUGIN_TYPE_KERBEROS로 설정되어야 합니다. 향후 버전의 API에서는 다른 값이 정의될 수 있습니다.

getConDetails_fn

입력. DB2 가 구현하는 db2secGetConDetails API의 포인터. db2secServerAuthPluginInit API는 다른 인증 API 중 하나에 db2secGetConDetails API 호출하여 데이터베이스 연결과 관련된 세부사항을 가져올 수 있습니다. 이러한 세부사항에는 연결과 관련된 통신 메커니즘(예: TCP/IP의 경우, IP 주소)에 대한 정보가 포함됩니다. 이 정보는 플러그인 기록기가 인증을 결정할 때 참조해야 하는 정보일 수 있습니다. 예를 들어, 플러그인은 사용자가 특정 IP 주소에서 연결하지 않으면 이 사용자의 연결을 허용하지 않을 수 있습니다. db2secGetConDetails API는 선택적으로 사용할 수 있습니다.

db2secGetConDetails API가 데이터베이스 연결과 관련되지 않은 상황에서 호출되면 DB2SEC_PLUGIN_NO_CON_DETAILS 값을 리턴하고, 그렇지 않으면 성공을 나타내는 0을 리턴합니다.

db2secGetConDetails API는 두 개의 입력 매개변수 즉, db2sec_con_details_<version_number> 구조의 포인터인 pConDetails와 db2sec_con_details 구조가 사용할 버전 번호를 나타내는 conDetailsVersion을 사용합니다. db2sec_con_details1이 사용되면 DB2SEC_CON_DETAILS_VERSION_1, db2sec_con_details2가 사용되면 DB2SEC_CON_DETAILS_VERSION_2 값을 사용할 수 있습니다. 버전 번호 DB2SEC_CON_DETAILS_VERSION_2를 사용하는 것이 좋습니다.

리턴에 성공하면, db2sec_con_details 구조(db2sec_con_details1 또는 db2sec_con_details2)에 다음 정보가 포함됩니다.

- 서버에 연결하는 데 사용된 프로토콜. 프로토콜 정의 목록은 sqlenv.h 파일 (포함 디렉토리에 있음)에 있습니다(SQL_PROTOCOL_*). 이 정보는 clientProtocol 매개변수에 채워집니다.
- clientProtocol이 SQL_PROTOCOL_TCPIP or SQL_PROTOCOL_TCPIP4 일 경우 서버와 인바운드 연결의 TCP/IP 주소. 이 정보는 clientIPAddress 매개변수에 채워집니다.
- 클라이언트가 연결을 시도하는 데이터베이스 이름. 이 정보는 인스턴스 첨부에 설정되지 않습니다. 이 정보는 dbname 및 dbnameLen 매개변수에 채워집니다.
- db2secValidatePassword API의 connection_details 매개변수에 설명된 것과 동일한 세부사항이 있는 연결 정보 비트맵. 이 정보는 connect_info_bitmap 매개변수에 채워집니다.
- clientProtocol이 SQL_PROTOCOL_TCPIP6일 경우 서버와 인바운드 연결의 TCP/IP 주소. 이 정보는 clientIP6Address 매개변수에 채워지며 DB2SEC_CON_DETAILS_VERSION_2가 db2secGetConDetails API 호출에 사용되는 경우에만 사용할 수 있습니다.

logMessage_fn

입력. DB2 데이터베이스 관리 프로그램이 구현하는 db2secLogMessage API의 포인터. db2secClientAuthPluginInit API는 db2secLogMessage API를 호출하여 디버깅 또는 정보용으로 db2diag 로그 파일에 메시지를 로그할 수 있습니다. db2secLogMessage API의 첫 번째 매개변수(level)는 db2diag 로그 파일에 기록되는 진단 오류의 유형을 지정하고, 마지막 두 매개변수는 각각 메시지 문자열과 길이입니다. `db2secPlugin.h`에 정의된 dbesecLogMessage API의 첫 번째 매개변수의 올바른 값은 다음과 같습니다.

DB2SEC_LOG_NONE (0)
로깅 없음

DB2SEC_LOG_CRITICAL (1)
서버 오류 발견

DB2SEC_LOG_ERROR (2)
오류 발견

DB2SEC_LOG_WARNING (3)
경고

DB2SEC_LOG_INFO (4)
정보용

db2secLogMessage API의 'level' 매개변수 값이 diaglevel 데이터베이스 관리 프로그램 구성 매개변수 이하일 경우에만 db2diag 로그 파일에 메시지 텍스트가 표시됩니다.

예를 들어, DB2SEC_LOG_INFO 값을 사용하는 경우 diaglevel 데이터베이스 관리 프로그램 구성 매개변수가 4로 설정된 경우에만 db2diag 로그 파일에 메시지 텍스트가 표시됩니다.

errormsg

출력. db2secServerAuthPluginInit API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

---

db2secServerAuthPluginTerm API - 서버 인증 플러그인 자원 정리

db2secServerAuthPluginTerm API는 서버 인증 플러그인에 사용된 자원을 제거합니다.

이 API는 DB2 데이터베이스 관리 프로그램이 서버 인증 플러그인을 언로드하기 직전에 호출합니다. 이 API는 플러그인 라이브러리가 보유하는 자원을 적절하게 정리하는 방법으로 구현해야 합니다. 예를 들어, 플러그인에서 할당한 메모리를 제거하고, 열려 있는 파일을 닫고, 네트워크 연결을 닫습니다. 플러그인은 자원 제거를 위해 자원을 추적해야 합니다. 이 API는 Windows 플랫폼에서는 호출되지 않습니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secServerAuthPluginTerm)
                        ( char     **errormsg,
                          db2int32 *errormsglen );
```

db2secServerAuthPluginTerm API 매개변수

errormsg

출력. db2secServerAuthPluginTerm API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

---

db2secValidatePassword API - 암호 유효성 확인

데이터베이스 연결을 조작하는 중에 사용자 ID 및 암호 스타일 인증을 수행할 수 있는 방법을 제공합니다.

주: 클라이언트 측에서 API가 실행될 때, API 코드는 CONNECT문을 실행하는 사용자의 특권으로 실행됩니다. 이 API는 인증 구성 매개변수가 CLIENT로 설정된 경우에만 클라이언트 측에서 호출됩니다.

서버 측에서 API가 실행될 때, API 코드는 인스턴스 소유자의 특권으로 실행됩니다.

인증에 특수 특권(예: UNIX에서 루트 수준 시스템 액세스 권한)이 필요할 경우 플러그인 기록기는 이러한 사항을 고려해야 합니다.

이 API는 암호가 유효하면 DB2SEC_PLUGIN_OK(성공)를, 암호가 유효하지 않으면 오류 코드(예: DB2SEC_PLUGIN_BADPWD)를 리턴해야 합니다.

API 및 데이터 구조 구문

```
SQL_API_RC ( SQL_API_FN *db2secValidatePassword)
                       ( const char *userid,
                         db2int32 useridlen,
                         const char *usernamespace,
                         db2int32 usernamespacelen,
                         db2int32 usernamespacetype,
                         const char *password,
                         db2int32 passwordlen,
                         const char *newpasswd,
                         db2int32 newpasswdlen,
                         const char *dbname,
                         db2int32 dbnamelen,
                         db2Uint32 connection_details,
                         void      **token,
                         char      **errormsg,
                         db2int32 *errormsglen );
```

db2secValidatePassword API 매개변수

userid 입력. 암호의 유효성을 확인할 사용자 ID

useridlen

입력. userid 매개변수 값의 길이(바이트)

usernamespace

입력. 사용자 ID를 가져온 이름 스페이스

usernamespacelen

입력. usernamespace 매개변수 값의 길이(바이트)

usernamespacetype

입력. 이름 스페이스 유형. db2secPlugin.h에 정의된 usernamespacetype 매개변수에 유효한 값은 다음과 같습니다.

- DB2SEC_NAMESPACE_SAM_COMPATIBLE 사용자 이름 스타일에 해당(예: domain₩myname)
- DB2SEC_NAMESPACE_USER_PRINCIPAL 사용자 이름 스타일에 해당(예: myname@domain.ibm.com)

현재 DB2 데이터베이스 시스템은
DB2SEC_NAMESPACE_SAM_COMPATIBLE 값만 지원합니다. 사용자 ID를 사용할 수 없는 경우, usernamespacetype 매개변수가 db2secPlugin.h에 정의된 DB2SEC_USER_NAMESPACE_UNDEFINED로 설정됩니다.

password

입력. 유효성을 확인할 암호.

passwordlen

입력. password 매개변수 값의 길이(바이트)

newpasswd

입력. 새 암호(암호가 변경되는 경우). 암호 변경 요청이 없는 경우 이 매개변수는 NULL로 설정됩니다. 이 매개변수가 NULL로 설정되지 않으면, 새 암호로 변경하기 전에 API가 이전 암호의 유효성을 확인합니다. API가 암호 변경 요청을 이행할 필요는 없지만, 이행하지 않을 경우 이전 암호의 유효성을 확인하지 않고 리턴 값
DB2SEC_PLUGIN_CHANGEPASSWORD_NOTSUPPORTED를 즉시 리턴해야 합니다.

newpasswdlen

입력. newpasswd 매개변수 값의 길이(바이트)

dbname

입력. 연결할 데이터베이스의 이름. API가 dbname 매개변수를 무시하거나, 유효한 암호를 가진 사용자의 특정 데이터베이스에 대한 액세스를 제한하는 규정이 있는 경우 리턴 값 DB2SEC_PLUGIN_CONNECTIONREFUSED를 리턴할 수 있습니다. 이 매개변수는 NULL이 될 수 있습니다.

dbnamelen

입력. dbname 매개변수 값의 길이(바이트). dbname 매개변수가 NULL이면 이 매개변수가 0으로 설정됩니다.

connection_details

입력. 다음 정보를 저장하는 데 현재 3비트가 사용되는 32비트 매개변수

- 맨 오른쪽 비트는 사용자 ID 소스가 db2secGetDefaultLoginContext API의 디폴트인지, 연결 중에 명시적으로 제공된 것인지를 나타냅니다.
- 오른쪽에서 두 번째 비트는 연결이 로컬(파티션된 데이터베이스 환경에서 db2nodes.cfg의 노드 중 하나로부터의 연결 또는 IPC(Inter Process Communication) 사용) 또는 리모트(네트워크 또는 루프백을 통해)인지를 나타냅니다. 이를 통해 API는 동일한 시스템에 있는 클라이언트가 암호를 사용하지 않고 DB2 서버에 연결할 수 있는지 결정할 수 있습니다. 디폴트 운

영 체제 기반 사용자 ID/암호 플러그인으로 인해, 동일한 시스템에 있는 클라이언트가 제공하는 암호 없이 로컬 연결이 허용됩니다(사용자가 연결 특권을 가지고 있다고 가정)

- 오른쪽에서 세 번째 비트는 DB2 데이터베이스 관리 프로그램이 API를 서버 측 또는 클라이언트 측에서 호출하는지를 나타냅니다.

비트 값은 db2secPlugin.h에 정의됩니다.

- DB2SEC_USERID_FROM_OS(0x00000001): OS로부터 사용자 ID를 가져오며 연결 명령문에 명시적으로 제공되지 않음을 나타냅니다.
- DB2SEC_CONNECTION_ISLOCAL(0x00000002): 로컬 연결을 나타냅니다.
- DB2SEC_VALIDATING_ON_SERVER_SIDE(0x0000004): DB2 데이터베이스 관리 프로그램이 서버 측 또는 클라이언트 측에서 호출하여 암호의 유효성을 확인하는지를 나타냅니다. 이 비트 값이 설정되면 DB2 데이터베이스 관리 프로그램이 서버 측에서 호출합니다. 그렇지 않으면 클라이언트 측에서 호출합니다.

내재적 인증에 대한 DB2 데이터베이스 시스템 디폴트 동작은 암호의 유효성을 확인하지 않고 연결을 허용하는 것입니다. 하지만 플러그인 개발자가 DB2SEC_PLUGIN_BADPASSWORD 오류를 실행하여 내재적 인증을 허용하지 않을 수 있습니다.

token 입력. 현재 연결 중에 연속 API 호출에 매개변수로 전달할 수 있는 데이터의 포인터. 호출할 수 있는 API로는 db2secGetAuthIDs API 및 db2secGetGroupsForUser API가 있습니다.

errormsg

출력. db2secValidatePassword API 실행이 성공적이지 않을 경우 이 매개변수에 리턴될 수 있는 플러그인이 할당한 ASCII 오류 메시지 문자열 주소의 포인터

errormsglen

출력. errormsg 매개변수에 있는 오류 메시지 문자열의 길이(바이트 단위)를 나타내는 정수의 포인터

제 164 장 GSS-API 인증 플러그인의 필수 API 및 정의

다음은 DB2 보안 플러그인 인터페이스에 필요한 GSS-API의 전체 목록입니다.

지원되는 API는 *Generic Security Service Application Program Interface, 버전 2*(IETF RFC2743) 및 *Generic Security Service API 버전 2: C-바인딩*(IETF RFC2744)과 같은 스펙을 준수합니다. GSS-API 기반 플러그인을 구현하기 전에, 이러한 스펙을 완전히 이해해야 합니다.

표 12. GSS-API 인증 플러그인의 필수 API 및 정의

이름		설명
클라이언트 측 API	gss_init_sec_context	피어 응용프로그램을 사용하여 보안 컨텍스트를 시작합니다.
서버 측 API	gss_accept_sec_context	피어 응용프로그램을 사용하여 시작된 보안 컨텍스트를 승인합니다.
서버 측 API	gss_display_name	내부 형식 이름을 텍스트로 변환합니다.
일반 API	gss_delete_sec_context	설정된 보안 컨텍스트를 삭제합니다.
일반 API	gss_display_status	GSS-API 상태 코드와 연관된 텍스트 오류 메시지를 가져옵니다.
일반 API	gss_release_buffer	버퍼를 삭제합니다.
일반 API	gss_release_cred	GSS-API 증명서와 연관된 로컬 데이터 구조를 릴리스합니다.
일반 API	gss_release_name	내부 형식 이름을 삭제합니다.
필수 정의	GSS_C_DELEG_FLAG	삭제를 요청합니다.
필수 정의	GSS_C_EMPTY_BUFFER	gss_buffer_desc에 데이터가 포함되어 있지 않음을 나타냅니다.
필수 정의	GSS_C_GSS_CODE	GSS 주 상태 코드를 나타냅니다.
필수 정의	GSS_C_INDEFINITE	메커니즘이 컨텍스트 만기를 지원하지 않음을 나타냅니다.
필수 정의	GSS_C_MECH_CODE	GSS 부 상태 코드를 나타냅니다.
필수 정의	GSS_C_MUTUAL_FLAG	상호 인증이 요청되었습니다.
필수 정의	GSS_C_NO_BUFFER	gss_buffer_t 변수가 유효한 gss_buffer_desc 구조를 가리키지 않음을 나타냅니다.
필수 정의	GSS_C_NO_CHANNEL_BINDINGS	통신 채널 바인딩이 없습니다.
필수 정의	GSS_C_NO_CONTEXT	gss_ctx_id_t 변수가 유효한 컨텍스트를 가리키지 않음을 나타냅니다.
필수 정의	GSS_C_NO_CREDENTIAL	gss_cred_id_t 변수가 유효한 증명서 핸들을 가리키지 않음을 나타냅니다.
필수 정의	GSS_C_NO_NAME	gss_name_t 변수가 유효한 내부 이름을 가리키지 않음을 나타냅니다.
필수 정의	GSS_C_NO_OID	디폴트 인증 메커니즘을 사용합니다.
필수 정의	GSS_C_NULL_OID_SET	디폴트 메커니즘을 사용합니다.
필수 정의	GSS_S_COMPLETE	API가 완료되었습니다.
필수 정의	GSS_S_CONTINUE_NEEDED	처리가 완료되지 않았으며 피어에서 수신한 회신 토큰을 사용하여 API를 다시 호출해야 합니다.

GSS-API 인증 플러그인의 제한사항

다음은 GSS-API 인증 플러그인의 제한사항 목록입니다.

- 디폴트 보안 메커니즘이 있는 것으로 항상 간주되므로 OID 고려사항은 없습니다.
- gss_init_sec_context()에 요청되는 유일한 GSS 서비스는 상호 인증 및 위임입니다. DB2 데이터베이스 관리 프로그램은 항상 위임을 위한 티켓을 요청하지만 이 티켓을 사용하여 새 티켓을 생성하지는 않습니다.
- 디폴트 컨텍스트 시간만 요청됩니다.
- gss_delete_sec_context()의 컨텍스트 토큰은 클라이언트에서 서버로, 또는 서버에서 클라이언트로 전송되지 않습니다.
- 익명이 지원되지 않습니다.
- 채널 바인딩이 지원되지 않습니다.
- 처음 증명서가 만기되면 DB2 데이터베이스 관리 프로그램은 이 증명서를 자동으로 갱신하지 않습니다.
- GSS-API 스펙은 gss_init_sec_context() 또는 gss_accept_sec_context()가 실패해도 두 함수 중 하나가 토큰을 리턴하여 피어에 전송하도록 합니다. 하지만 DRDA 제한사항으로 인해, DB2 데이터베이스 관리 프로그램은 gss_init_sec_context()가 실패하는 경우에만 토큰을 전송하고 첫 번째 호출에서 토큰을 생성합니다.

제 165 장 보안 플러그인 샘플

UNIX 및 Linux 디렉토리: 'C' 샘플은 sqllib/samples/security/plugins에 있으며 JCC GSS-API 플러그인 샘플(.java)은 sqllib/samples/java/jdbc에 있습니다.

Windows 디렉토리: 'C' 샘플은 sqllib₩samples₩security₩plugins에 있고 JCC GSS-API 플러그인 샘플(.java)은 sqllib₩samples₩java₩jdbc에 있습니다.

표 13. 보안 플러그인 샘플 프로그램 파일

샘플 프로그램 이름	프로그램 설명
combined.c	사용자 ID 및 암호 인증과 그룹 찾아보기 샘플이 결합된 샘플
group_file.c	단순 파일 기반 그룹 관리 플러그인 샘플
gssapi_simple.c	기본 GSS-API 인증 플러그인 샘플(클라이언트 및 서버 모두)
IBMLDAPauthclient.c	LDAP 사용자 레지스트리와 상호작용하는 클라이언트 측 DB2 보안 플러그인 구현
IBMLDAPauthserver.c	LDAP 사용자 레지스트리와 상호작용하는 서버 측 DB2 보안 플러그인 구현
IBMLDAPconfig.c	DB2 LDAP 보안 플러그인에 적합한 구성 파일 찾기 및 구문 분석과 관련된 기능 포함
IBMLDAPgroups.c	LDAP 기반 그룹 찾아보기에 필요한 DB2 보안 플러그인 구현
IBMLDAPutils.c	DB2 LDAP 보안 플러그인에서 사용되는 유틸리티 기능 포함
IBMLDAPutils.h	LDAP 보안 플러그인 헤더 파일
JCCKerberosPlugin.java	IBM DB2 Universal Driver를 사용하여 Kerberos 인증을 수행하는 GSS-API 플러그인 구현
JCCKerberosPluginTest.java	JCCKerberosPlugin을 사용하여 IBM DB2 Universal Driver를 통해 DB2에 연결
JCCSimpleGSSPlugin.java	IBM DB2 Universal Driver를 사용하여 사용자 ID 및 암호 검사를 수행하는 GSS-API 플러그인 구현
JCCSimpleGSSContext.java	JCCSimpleGSSPlugin에서 사용할 GSSContext 구현
JCCSimpleGSSCredential.java	JCCSimpleGSSPlugin에서 사용할 GSSCredential 구현
JCCSimpleGSSException.java	JCCSimpleGSSPlugin에서 사용할 GSSException 구현
JCCSimpleGSSName.java	JCCSimpleGSSPlugin에서 사용할 GSSName 구현
JCCSimpleGSSPluginTest.java	JCCSimpleGSSPlugin을 사용하여 IBM DB2 Universal Driver를 통해 DB2에 연결

제 166 장 스토리지 관리자로 백업 및 리스토어하는 DB2 API

DB2는 써드 파티 미디어 관리 제품에서 백업 및 리스토어 조작 및 로그 파일의 데이터와 로그 파일을 저장하고 검색하는 데 사용할 수 있는 인터페이스를 제공합니다. 이 인터페이스는 디스켓, 디스크, 테이프 및 DB2의 표준 파트로 지원되는 Tivoli Storage Manager의 백업, 리스토어 및 로그 아카이브 데이터 대상을 기능 보강하도록 설계되었습니다.

이 써드 파티 미디어 관리 제품은 이 섹션의 나머지 부분에서 벤더 제품이라고 합니다.

DB2는 많은 벤더에서 사용할 수 있는 백업, 리스토어 및 로그 아카이브를 위한 일반적인 용도의 데이터 인터페이스를 제공하는 API 프로토타입 세트를 정의합니다. 벤더는 이 API를 UNIX 기반 시스템의 공유 라이브러리 또는 Windows 운영 체제의 DLL로 제공합니다. DB2가 API를 호출하면 백업, 리스토어 또는 로그 아카이브 루틴을 호출하여 지정한 공유 라이브러리나 DLL이 로드되고 벤더가 제공하는 API가 호출되어 필요한 태스크를 수행합니다.

DB2 벤더 기능을 설명하는 샘플 파일은 UNIX 플랫폼의 `sqllib/samples/BARVendor` 디렉토리와 Windows의 `sqllib¥samples¥BARVendor` 디렉토리에 있습니다.

다음은 백업 및 리스토어 벤더 스토리지 플러그인 API 설명에 사용되는 용어 정의입니다.

백업 및 리스토어 벤더 스토리지 플러그인
DB2가 로드하여 벤더 제품용으로 사용자가 작성한 백업 및 리스토어 API에 액세스하는 동적으로 로드할 수 있는 라이브러리.

입력 DB2가 백업 및 리스토어 벤더 스토리지 플러그인 API 매개변수에 값을 채우는 것을 나타냅니다.

출력 백업 및 리스토어 벤더 스토리지 플러그인 API가 API 매개변수의 값을 채우는 것을 나타냅니다.

db2VendorGetNextObj - 디바이스의 다음 오브젝트 가져오기

이 API는 쿼리가 검색 기준에 일치하는 다음 오브젝트(이미지나 아카이브된 로그 파일)를 가져오도록 설정된(sqluvint API 사용) 후에 호출됩니다. 이미지나 로그 파일에 대한 한 번에 한 개의 검색만 설정할 수 있습니다.

권한 부여

없음

필수 연결

데이터베이스.

API 내장 파일

db2VendorApi.h

API 및 데이터 구조 구문

```
int db2VendorGetNextObj ( void                          * vendorCB,
                          struct db2VendorQueryInfo     * queryInfo,
                          struct Return_code            * returnCode);

typedef struct db2VendorQueryInfo
{
   db2Uint64 sizeEstimate;
   db2Uint32 type;
   SQL_PDB_NODE_TYPE dbPartitionNum;
   db2Uint16 sequenceNum;
   char db2Instance[SQL_INSTNAME_SZ + 1];
   char dbname[SQL_DBNAME_SZ + 1];
   char dbalias[SQL_ALIAS_SZ + 1];
   char timestamp[SQLU_TIME_STAMP_LEN + 1];
   char filename[DB2VENDOR_MAX_FILENAME_SZ + 1];
   char owner[DB2VENDOR_MAX_OWNER_SZ + 1];
   char mgmtClass[DB2VENDOR_MAX_MGMTCLASS_SZ + 1];
   char oldestLogfile[DB2_LOGFILE_NAME_LEN + 1];
} db2VendorQueryInfo;
```

db2VendorGetNextObj API 매개변수

vendorCB

입력. 벤더 라이브러리로 할당된 스페이스의 포인터

queryInfo

출력. 벤더 라이브러리로 채워지는 db2VendorQueryInfo 구조의 포인터

returnCode

출력. API 호출의 리턴 코드.

db2VendorQueryInfo 데이터 구조 매개변수

sizeEstimate

오브젝트의 예상 크기를 지정합니다.

type 오브젝트가 백업 이미지인 경우 이미지 유형을 지정합니다.

dbPartitionNum

오브젝트가 속한 데이터베이스 파티션 번호를 지정합니다.

sequenceNum

백업 이미지의 파일 확장자를 지정합니다. 오브젝트가 백업인 경우에만 유효합니다.

db2Instance

오브젝트가 속한 데이터베이스 인스턴스 이름을 지정합니다.

dbname

오브젝트가 속한 데이터베이스 이름을 지정합니다.

dbalias

오브젝트가 속한 데이터베이스 별명을 지정합니다.

timestamp

백업 이미지를 식별하는 데 사용되는 시간소인을 지정합니다. 오브젝트가 백업 이미지인 경우에만 유효합니다.

filename

오브젝트가 로드 사본 이미지 또는 아카이브된 로그 파일인 경우에 오브젝트 이름을 지정합니다.

owner 오브젝트 소유자를 지정합니다.

mgmtClass

오브젝트가 저장된 관리 클래스를 지정합니다(TSM에서 사용).

oldestLogfile

백업 이미지에 저장된 가장 오래된 로그 파일을 지정합니다.

사용 시 참고사항

각 오브젝트나 벤더에 모든 매개변수가 관련되지는 않습니다. 채워야 하는 필수 매개변수는 db2Instance, dbname, dbalias, timestamp(이미지용), filename(로그 및 로드 사본 이미지용), owner, sequenceNum(이미지용) 및 dbPartitionNum입니다. 나머지 매개변수는 정의할 특정 벤더용으로 남아 있습니다. 매개변수가 관련되지 않으면 문자열의 경우 ""로 초기화되고 숫자 유형의 경우 0으로 초기화됩니다.

리턴 코드

다음 표에는 이 API에 사용할 수 있는 모든 리턴 코드가 나열됩니다.

표 14. db2VendorGetNextObj API 리턴 코드

번호	리턴 코드	설명
2	SQLUV_COMM_ERROR	디바이스와의 통신 오류 - 실패.
4	SQLUV_INV_ACTION	입력 매개변수에 대한 유효하지 않은 조치 요청이나 조합으로 인해 조작이 불가능해 집니다. - 실패.
5	SQLUV_NO_DEV_AVAIL	해당 시점에 사용 가능한 디바이스가 없습니다. - 실패.
6	SQLUV_OBJ_NOT_FOUND	삭제할 오브젝트가 없습니다. - 실패.

표 14. db2VendorGetNextObj API 리턴 코드 (계속)

번호	리턴 코드	설명
12	SQLUV_INV_DEV_HANDLE	유효하지 않은 디바이스 핸들 - 실패.
14	SQLUV_END_OF_DATA	더 이상 리턴할 쿼리 오브젝트가 없습니다. - 성공.
18	SQLUV_DEV_ERROR	디바이스 오류 - 실패.
19	SQLUV_WARNING	경고, 리턴 코드 참조 - 성공.
21	SQLUV_MORE_DATA	리턴할 쿼리 오브젝트가 있습니다. - 성공.
25	SQLUV_IO_ERROR	입출력 오류 - 실패.
30	SQLUV_UNEXPECTED_ERROR	심각한 오류 발생 - 실패.

db2VendorQueryApiVersion - 벤더 스토리지 API의 지원되는 레벨 가져오기

백업 및 리스토어 벤더 스토리지 플러그인에서 지원되는 벤더 스토리지 API 레벨을 판별합니다. 지정된 벤더 스토리지 플러그인이 DB2와 호환되지 않으면 사용되지 않습니다.

벤더 스토리지 플러그인에 로그에 대해 구현된 이 API가 없는 경우 사용할 수 없으며 DB2가 오류를 보고합니다. 현재 기존 벤더 라이브러리에서 작업 중인 이미지에는 영향을 주지 않습니다.

권한 부여

없음

필수 연결

데이터베이스.

API 내장 파일

```
db2VendorApi.h
```

API 및 데이터 구조 구문

```
void db2VendorQueryApiVersion ( db2Uint32  * supportedVersion );
```

db2VendorQueryApiVersion API 매개변수

supportedVersion
출력. 벤더 라이브러리가 지원하는 벤더 스토리지 API 버전을 리턴합니다.

사용 시 참고사항

이 API는 다른 벤더 스토리지 API가 호출되기 전에 호출됩니다.

sqluvdel - 커미트된 세션 삭제

벤더 디바이스에서 커미트된 세션을 삭제합니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sqluvend.h
```

API 및 데이터 구조 구문

```
int sqluvdel ( struct Init_input *in,
               struct Init_output *vendorDevData,
               struct Return_code *return_code);
```

sqluvdel API 매개변수

in 입력. Init_input 및 Return_code에 할당된 스페이스

vendorDevData
출력. 벤더 디바이스에서 리턴된 출력이 포함된 구조

return_code
출력. API 호출의 리턴 코드. init_input 구조로 지시된 오브젝트는 삭제됩니다.

사용 시 참고사항

여러 세션이 열려 있으며 일부 세션은 커미트되었지만 이 중 하나라도 실패한 경우 이 API가 커미트된 세션을 삭제하도록 호출됩니다. 시퀀스 번호를 지정하지 않았습니다. sqluvdel이 특정 백업 조작 중에 작성된 모든 오브젝트를 찾아서 삭제합니다. Init_input 구조의 정보를 사용하여 삭제할 출력 데이터를 식별합니다. sqluvdel 호출을 사용하여 벤더 디바이스에서 오브젝트를 삭제하는 데 필요한 연결이나 세션을 작성합니다. 이 호출의 리턴 코드가 SQLUV_DELETE_FAILED인 경우 DB2가 첫 번째 치명적 오류를 리턴하고 후속 오류는 무시하기 때문에 DB2는 호출자에게 통지하지 않습니다. 이 경우 DB2가 sqluvdel API를 호출하도록 하려면 초기의 치명적 오류가 발생해야 합니다.

리턴 코드

표 15. sqluvdel의 유효한 리턴 코드 및 결과 데이터베이스 서버 조치

헤더 파일의 리터럴	설명	가능한 다음 호출
SQLUV_OK	조작 성공	추가 호출 없음
SQLUV_DELETE_FAILED	삭제 요청 실패	추가 호출 없음

sqluvend - 벤더 디바이스 링크 해제 및 해당 자원 릴리스

벤더 디바이스를 링크 해제하고 관련된 모든 자원을 사용 가능하게 합니다. 사용되지 않는 모든 자원(예: 할당된 스페이스 및 파일 핸들)은 sqluvend API 호출이 DB2로 리턴되기 전에 릴리스되어야 합니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sqluvend.h
```

API 및 데이터 구조 구문

```
int sqluvend ( sqlint32             action,
               void *hdle,
               struct Init_output *in_out,
               struct Return_code *return_code);
```

sqluvend API 매개변수

action 입력. 세션을 커미트하거나 중단하는 데 사용됩니다.

- SQLUV_COMMIT(커미트의 경우 0)
- SQLUV_ABORT(중단의 경우 1)

hdle 입력. Init_output 구조의 포인터

in_out 출력. Init_output 할당 해제를 위한 스페이스. 조치가 커미트인 경우 데이터는 백업을 위한 안전한 스토리지로 커미트되었습니다. 조치가 중단인 경우 데이터는 백업을 위해 제거됩니다.

return_code
출력. API 호출의 리턴 코드.

사용 시 참고사항

이 API는 열린 각 세션에 대해 호출됩니다. 두 가지 코드가 가능합니다.

Commit
이 세션의 데이터 출력 또는 세션에서 데이터 읽기가 완료됩니다.

쓰기(백업) 세션의 경우, 벤더가 리턴 코드 SQLUV_OK와 함께 DB2로 리턴하면 DB2에서는 벤더 제품에 의해 출력 데이터가 제대로 저장되어 이후의 sqluvint 호출에서 참조되는 경우 액세스할 수 있다고 간주합니다.

읽기(리스토어) 세션의 경우, 벤더가 DB2에 리턴 코드 SQLUV_OK를 리턴하면 데이터가 다시 필요할 수도 있기 때문에 데이터를 삭제하면 안됩니다. 벤더가 SQLUV_COMMIT_FAILED를 리턴하면 DB2는 전체 백업이나 리스토어 조작에 문제가 발생한 것으로 간주합니다. 모든 사용 중인 세션은 action = SQLUV_ABORT가 포함된 sqluvend 호출로 종료됩니다. 백업 조작의 경우 커미트된 세션은 sqluvint, sqluvdel, sqluvend 시퀀스의 호출을 수신합니다.

Abort DB2에 문제점이 발생했으며 읽거나 쓸 세션의 데이터가 더 이상 없습니다.

쓰기(백업) 세션의 경우 벤더는 부분적인 출력 데이터 세트를 삭제하고 부분 출력이 삭제되는 경우 SQLUV_OK 리턴 코드를 사용해야 합니다. DB2는 전체 백업에 문제가 발생한 것으로 간주합니다. 사용 중인 모든 세션은 action = SQLUV_ABORT가 포함된 sqluvend 호출로 종료되고 커미트된 세션은 sqluvint, sqluvdel, sqluvend 시퀀스의 호출을 수신합니다.

읽기(리스토어) 세션의 경우, 벤더는 데이터가 다시 사용될 수도 있기 때문에 이를 삭제하면 안되지만 이를 제거(clean up)하고 SQLUV_OK 리턴 코드와 함께 DB2로 리턴해야 합니다. DB2는 action = SQLUV_ABORT가 포함된 sqluvend 호출로 모든 리스토어 세션을 종료합니다. 벤더가 DB2에 SQLUV_ABORT_FAILED를 리턴하면 호출자는 DB2가 첫 번째 치명적 오류를 리턴하고 후속 오류는 무시하기 때문에 이 오류에 대한 통지는 수신하지 않습니다. 이 경우 DB2가 action = SQLUV_ABORT가 포함된 sqluvend를 호출하도록 하려면 초기의 치명적 오류가 발생해야 합니다.

리턴 코드

표 16. sqluvend의 유효한 리턴 코드 및 결과 DB2 조치

헤더 파일의 리터럴	설명	가능한 다음 호출	기타 주석
SQLUV_OK	조작 성공	추가 호출 없음	이 세션에 할당된 모든 메모리를 사용 가능화하고 종료합니다.
SQLUV_COMMIT_FAILED	커미트 요청이 실패했습니다.	추가 호출 없음	이 세션에 할당된 모든 메모리를 사용 가능화하고 종료합니다.
SQLUV_ABORT_FAILED	중단 요청이 실패했습니다.	추가 호출 없음	

sqluvget - 벤더 디바이스에서 데이터 읽기

sqluvint API로 벤더 디바이스를 초기화한 후에 DB2가 이 API를 호출하여 리스토어 조작 중에 디바이스에서 데이터를 읽습니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sqluvend.h
```

API 및 데이터 구조 구문

```
int sqluvget ( void * hdle,
               struct Data *data,
               struct Return_code *return_code);
```

sqluvget API 매개변수

hdle 입력. 데이터 구조(데이터 버퍼 포함) 및 Return_code에 할당된 구조의 포인터

데이터 입력 또는 출력. 데이터 구조의 포인터

return_code
출력. API 호출의 리턴 코드.

사용 시 참고사항

이 API는 리스토어 유틸리티에서 사용됩니다.

리턴 코드

표 17. sqluvget의 유효한 리턴 코드 및 결과 DB2 조치

헤더 파일의 리터럴	설명	가능한 다음 호출	기타 주석
SQLUV_OK	조작이 성공했습니다.	sqluvget	DB2에서 데이터를 처리합니다.
SQLUV_COMM_ERROR	디바이스 통신가 발생했습니다.	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_INV_ACTION	유효하지 않은 조치가 요청되었습니다.	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.

표 17. sqluvget의 유효한 리턴 코드 및 결과 DB2 조치 (계속)

헤더 파일의 리터럴	설명	가능한 다음 호출	기타 주석
SQLUV_INV_DEV_HANDLE	유효하지 않은 디바이스 핸들입니다.	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_INV_BUFF_SIZE	유효하지 않은 버퍼 크기가 지정되었습니다.	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_DEV_ERROR	디바이스 오류가 발생했습니다.	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_WARNING	경고. DB2 미디어 끝을 표시하는 데 사용되면 안됩니다. 이런 용도에는 SQLUV_ENDOFMEDIA 또는 SQLUV_ENDOFMEDIA_NO _DATA를 사용하십시오. 그러나 이 리턴 코드로 디바이스가 준비되지 않음 조건을 나타낼 수는 있습니다.	sqluvget 또는 sqluvend, action= SQLU_ABORT	
SQLUV_LINK_NOT_EXIST	현재 링크가 없습니다.	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_MORE_DATA	조작이 성공했습니다. 추가 데이터가 제공되었습니다.	sqluvget	
SQLUV_ENDOFMEDIA_NO_ DATA	미디어 끝 및 0 바이트 읽음(예: 테이프 끝)	sqluvend	
SQLUV_ENDOFMEDIA	미디어 끝 및 > 0 바이트 읽음(예: 테이프 끝)	sqluvend	DB2가 데이터를 처리하고 end-of-media 조건을 처리합니다.
SQLUV_IO_ERROR	입출력 오류가 발생했습니다.	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.

주: 다음 호출: 다음 호출이 sqluvend, action = SQLU_ABORT인 경우 이 세션 및 기타 모든 사용 중인 세션이 종료됩니다.

sqluvint - 벤더 디바이스 초기화 및 링크

벤더 디바이스를 초기화하고 DB2와 벤더 디바이스 사이의 논리 링크 설정에 대한 정보를 제공합니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sqluvend.h
```

API 및 데이터 구조 구문

```
int sqluvint ( struct Init_input *in,
               struct Init_output *out,
               struct Return_code *return_code);
```

sqluvint API 매개변수

in 입력. 벤더 디바이스와의 논리 링크를 작성하기 위해 DB2에서 제공하는 정보를 포함하는 구조.

out 출력. 벤더 디바이스에서 리턴하는 출력을 포함하는 구조.

return_code
출력. DB2에 전달되는 리턴 코드 및 간단한 텍스트 설명을 포함하는 구조.

사용 시 참고사항

각 미디어 입출력 세션에 대해 DB2는 이 API를 호출하여 디바이스 핸들을 확보합니다. 초기화 중에 어떤 이유로든 벤더 스토리지 API에 오류가 발생하면 리턴 코드로 표시됩니다. 리턴 코드로 오류가 표시되면 DB2는 sqluvend API를 호출하여 조작을 종료합니다. 가능한 리턴 코드에 대한 자세한 내용 및 각각에 대한 DB2의 조치는 리턴 코드 테이블(아래 테이블 참조)에서 제공됩니다.

Init_input 구조에는 백업이나 리스토어 진행 가능 여부를 판별하기 위해 벤더 제품에서 사용할 수 있는 요소가 포함됩니다.

size_HI_order 및 size_LOW_order
이는 백업의 예상 *크기*입니다. 이를 사용하여 벤더 디바이스로 백업 이미지 크기를 처리할 수 있는지 판별할 수 있습니다. 이를 사용하여 백업을 보유하는 데 필요한 분리성 매체 수량을 계산할 수 있습니다. 문제가 예상되는 경우에는 첫 번째 sqluvint 호출에서 실패하는 것이 좋을 수도 있습니다.

req_sessions
사용자가 요청한 세션의 수는 예상 *크기* 및 프롬프트 레벨과 같이 사용하여 백업이나 리스토어 조작이 가능한지 판별할 수 있습니다.

prompt_lvl
프롬프트 레벨은 벤더에서 분리성 매체 변경(예: 테이프 드라이브에 다른 테이프 삽입)과 같은 조치 시에 프롬프트를 표시하는 것이 가능한지 표시합니다. 사용자에게 프롬프트를 표시할 방법이 없기 때문에 조작을 계속할 수 없음을 나

타내기도 합니다. 완료 레벨이 WITHOUT PROMPTING이고 분리성 매체 수량이 요청된 세션 수보다 많은 경우 DB2는 조작을 제대로 성공하지 못합니다.

DB2가 기록 중인 백업 또는 DB2_info 구조의 필드를 통해 읽을 수 있는 리스토어 이름을 지정합니다. action = SQLUV_READ인 경우 벤더 제품은 이름이 지정된 오브젝트 존재 여부를 점검합니다. 찾을 수 없는 경우 리턴 코드는 DB2가 적절한 조치를 수행할 수 있도록 SQLUV_OBJ_NOT_FOUND로 설정되어야 합니다.

초기화가 제대로 완료되면 DB2는 다른 전송 API 호출을 계속하지만 sqluvend 호출로 언제든지 세션을 종료할 수도 있습니다.

리턴 코드

표 18. sqluvint의 유효한 리턴 코드 및 결과 DB2 조치

헤더 파일의 리터럴	설명	가능한 다음 호출	기타 주석
SQLUV_OK	조작이 성공했습니다.	sqluvput, sqluvget(주석 참조)	action = SQLUV_WRITE인 경우 다음 호출은 데이터를 백업하는 sqluvput API입니다. action = SQLUV_READ인 경우 SQLUV_OK를 리턴하기 전에 이름이 지정된 오브젝트 존재 여부를 점검하십시오. 다음 호출은 데이터를 리스토어하는 sqluvget API입니다.
SQLUV_LINK_EXIST	세션이 이전에 활성화되었습니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_COMM_ERROR	디바이스 통신가 발생했습니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_INV_VERSION	DB2와 벤더 제품이 호환되지 않습니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_INV_ACTION	유효하지 않은 조치가 요청되었습니다. 이를 사용하여 매개변수 조합으로 불가능한 조작이 초래되었음을 나타낼 수도 있습니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.

표 18. sqluvint의 유효한 리턴 코드 및 결과 DB2 조치 (계속)

헤더 파일의 리터럴	설명	가능한 다음 호출	기타 주석
SQLUV_NO_DEV_AVAIL	해당 시점에 사용 가능한 디바이스가 없습니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_OBJ_NOT_FOUND	지정한 오브젝트를 찾을 수 없습니다. sqluvint 호출의 조치가 ″R″(읽기)이고 DB2_info 구조에 지정한 기준을 사용하여 요청한 오브젝트를 찾을 수가 없는 경우 사용해야 합니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_OBJS_FOUND	지정한 기준에 일치하는 오브젝트가 둘 이상 있습니다. sqluvint 호출의 조치가 ″R″(읽기)이고 둘 이상의 오브젝트가 DB2_info 구조의 기준에 일치하는 경우에 발생합니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_INV_USERID	유효하지 않은 사용자 ID를 지정했습니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_INV_PASSWORD	유효하지 않은 암호를 입력했습니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_INV_OPTIONS	벤더 옵션 필드에 유효하지 않은 옵션이 있습니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_INIT_FAILED	초기화가 실패하고 세션이 종료됩니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_DEV_ERROR	디바이스 오류가 발생했습니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.

표 18. sqluvint의 유효한 리턴 코드 및 결과 DB2 조치 (계속)

헤더 파일의 리터럴	설명	가능한 다음 호출	기타 주석
SQLUV_MAX_LINK_GRANT	최대 수의 링크가 작성되었습니다.	sqluvput, sqluvget(주석 참조)	DB2에서 경고로 처리합니다. DB2는 지원할 수 있는 최대 수의 세션에 도달했기 때문에 벤더 제품에 추가 세션을 열지 않는다는 경고입니다(참고: 이는 디바이스 사용 가능성으로 인해 발생할 수도 있음). action = SQLUV_WRITE (BACKUP)인 경우 다음 호출은 sqluvput API입니다. action = SQLUV_READ인 경우 SQLUV_MAX_LINK_ GRANT를 리턴하기 전에 이름이 지정된 오브젝트의 존재 여부를 점검하십시오. 다음 호출은 데이터를 리스토어하는 sqluvget API입니다.
SQLUV_IO_ERROR	입출력 오류.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.
SQLUV_NOT_ENOUGH_ SPACE	전체 백업 이미지를 저장할 스페이스가 부족합니다. 예상 크기는 64비트 값(바이트 단위)으로 제공됩니다.	추가 호출이 없습니다.	세션 초기화가 실패했습니다. 이 세션에 할당된 메모리를 사용 가능화하고 종료합니다. 세션이 작성되지 않았기 때문에 sqluvend API 호출이 수신되지 않습니다.

sqluvput - 벤더 디바이스에 데이터 쓰기

sqluvint API로 벤더 디바이스를 초기화한 후에 DB2가 이 API를 호출하여 백업 조작 중에 디바이스에 데이터를 기록합니다.

권한 부여

없음

필수 연결

데이터베이스

API 내장 파일

```
sqluvend.h
```

API 및 데이터 구조 구문

```
int sqluvput ( void *hdle,
               struct Data *data,
               struct Return_code *return_code);
```

sqluvput API 매개변수

hdle 입력. DATA 구조(데이터 버퍼 포함) 및 Return_code에 할당된 구조의 포인터

데이터 출력. 기록할 데이터로 채운 데이터 버퍼.

return_code
출력. API 호출의 리턴 코드.

사용 시 참고사항

이 API는 백업 유틸리티에서 사용됩니다.

리턴 코드

표 19. sqluvput의 유효한 리턴 코드 및 결과 DB2 조치

헤더 파일의 리터럴	설명	가능한 다음 호출	기타 주석
SQLUV_OK	조작 성공	완료된 경우 sqluvput 또는 sqluvend(예: DB2에 더 이상 데이터가 없음)	다른 프로세스에 조작 성공을 통보합니다.
SQLUV_COMM_ERROR	디바이스 통신 오류	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_INV_ACTION	유효하지 않은 조치 요청	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_INV_DEV_HANDLE	유효하지 않은 디바이스 핸들	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_INV_BUFF_SIZE	유효하지 않은 버퍼 크기 지정	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_ENDOFMEDIA	미디어 끝에 도달. 예: 테이프 끝	sqluvend	
SQLUV_DATA_RESEND	디바이스에서 버퍼 재전송 요청	sqluvput	DB2가 최종 버퍼를 재전송합니다. 이는 한 번만 수행됩니다.
SQLUV_DEV_ERROR	디바이스 오류	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.

표 19. sqluvput의 유효한 리턴 코드 및 결과 DB2 조치 (계속)

헤더 파일의 리터럴	설명	가능한 다음 호출	기타 주석
SQLUV_WARNING	경고. DB2의 미디어 끝을 표시하는 데 사용되면 안됩니다. 이를 위해서는 SQLUV_ENDOFMEDIA를 사용하십시오. 그러나 이 리턴 코드를 사용하여 디바이스가 준비되지 않음 조건을 표시할 수는 있습니다.	sqluvput	
SQLUV_LINK_NOT_EXIST	현재 링크가 없습니다.	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.
SQLUV_IO_ERROR	입출력 오류	sqluvend, action = SQLU_ABORT(아래 메모 참조)	세션이 종료됩니다.

주: 다음 호출: 다음 호출이 sqluvend, action = SQLU_ABORT인 경우 이 세션 및 기타 모든 사용 중인 세션이 종료됩니다. 커미트된 세션은 호출의 sqluvint, sqluvdel, sqluvend 시퀀스로 삭제됩니다.

DB2_info

DB2 제품 및 백업이나 리스토어 중인 데이터베이스에 대한 정보가 포함됩니다. 이 구조를 사용하여 DB2를 벤더 디바이스에 대해 식별하고 DB2와 벤더 디바이스의 특정 세션에 대해 설명합니다. Init_input 데이터 구조의 파트로 백업 및 리스토어 벤더 스토리지 플러그인에 전달됩니다.

표 20. DB2_info 구조의 필드

필드 이름	데이터 유형	설명
DB2_id	char	DB2 제품의 ID. 지시하는 문자열의 최대 길이는 8자입니다.
version	char	DB2 제품의 현재 버전. 지시하는 문자열의 최대 길이는 8자입니다.
release	char	DB2 제품의 현재 릴리스. 중요하지 않은 경우에는 NULL로 설정하십시오. 지시하는 문자열의 최대 길이는 8자입니다.
level	char	DB2 제품의 현재 레벨. 중요하지 않은 경우에는 NULL로 설정하십시오. 지시하는 문자열의 최대 길이는 8자입니다.
action	char	수행할 조치를 지정합니다. 지시하는 문자열의 최대 길이는 1자입니다.

표 20. DB2_info 구조의 필드 (계속)

필드 이름	데이터 유형	설명
filename	char	백업 이미지를 식별하는 데 사용되는 파일 이름. NULL인 경우 server_id, db2instance, dbname 및 timestamp는 고유하게 백업 이미지를 식별합니다. 지시하는 문자열의 최대 길이는 255자입니다.
server_id	char	데이터베이스가 있는 서버를 식별하는 고유 이름. 지시하는 문자열의 최대 길이는 8자입니다.
db2instance	char	DB2 인스턴스 ID. 명령을 호출하는 사용자 ID입니다. 지시하는 문자열의 최대 길이는 8자입니다.
type	char	수행 중인 백업 유형 또는 수행 중인 리스토어 유형을 지정합니다. 다음은 가능한 값입니다. 조치가 SQLUV_WRITE인 경우: 0 - 전체 데이터베이스 백업 3 - 테이블 스페이스 레벨 백업 조치가 SQLUV_READ인 경우: 0 - 전체 리스토어 3 - 온라인 테이블 스페이스 리스토어 4 - 테이블 스페이스 리스토어 5 - 실행 기록 파일 리스토어
dbname	char	백업 또는 리스토어되는 데이터베이스 이름지시하는 문자열의 최대 길이는 8자입니다.
별명	char	백업 또는 리스토어되는 데이터베이스 별명. 지시하는 문자열의 최대 길이는 8자입니다.
시간소인	char	백업 이미지 식별에 사용되는 시간소인. 지시하는 문자열의 최대 길이는 26자입니다.
시퀀스	char	백업 이미지의 파일 확장자를 지정합니다. 쓰기 조작에서 첫 번째 세션의 값은 1이며 sqluvint 호출로 다른 세션이 시작될 때마다 값이 1씩 증가됩니다. 읽기 조작의 경우 값은 항상 영(0)입니다. 지시하는 문자열의 최대 길이는 3자입니다.
obj_list	struct sqlu_gen_list	나중에 사용하기 위해 예약됨
max_bytes_per_txn	sqlint32	사용자가 지정한 전송 버퍼 크기를 바이트 단위로 벤더에 지정합니다.
image_filename	char	나중에 사용하기 위해 예약됨

표 20. DB2_info 구조의 필드 (계속)

필드 이름	데이터 유형	설명
reserve	void	나중에 사용하기 위해 예약됨
nodename	char	백업이 생성된 노드 이름
password	char	백업이 생성된 노드의 암호
owner	char	백업 시작자 ID
mcNameP	char	관리 클래스.
nodeNum	SQL_PDB_NODE_TYPE	노드 번호. 255 이상의 수는 벤더 인터페이스에서 지원됩니다.

주: 모든 char 데이터 유형 필드는 널(null)로 끝나는 문자열입니다.

filename 또는 server_id, db2instance, type, dbname 및 timestamp는 고유하게 백업 이미지를 식별합니다. 시퀀스로 지정한 시퀀스 번호는 파일 확장자를 식별합니다. 백업 이미지를 리스토어하는 경우 동일한 값을 백업 이미지 검색에 지정해야 합니다. 벤더 제품에 따라 filename을 사용하는 경우 다른 매개변수를 NULL로 설정할 수 있으며 다른 매개변수도 마찬가지로 설정할 수 있습니다.

API 및 데이터 구조 구문

```
typedef struct DB2_info
{
  char *DB2_id;
  char *version;
  char *release;
  char *level;
  char *action;
  char *filename;
  char *server_id;
  char *db2instance;
  char *type;
  char *dbname;
  char *alias;
  char *timestamp;
  char *sequence;
  struct sqlu_gen_list
                 *obj_list;
  sqlint32 max_bytes_per_txn;
  char *image_filename;
  void *reserve;
  char *nodename;
  char *password;
  char *owner;
  char *mcNameP;
  SQL_PDB_NODE_TYPE nodeNum;
} DB2_info ;
```

Vendor_info

벤더 및 벤더 디바이스 버전을 식별하는 Init_output 구조의 파트로 DB2에 리턴되는 정보가 포함됩니다.

표 21. Vendor_info 구조의 필드

필드 이름	데이터 유형	설명
vendor_id	char	벤더 ID. 지시하는 문자열의 최대 길이는 64자입니다.
version	char	벤더 제품의 현재 버전. 지시하는 문자열의 최대 길이는 8자입니다.
release	char	벤더 제품의 현재 릴리스. 중요하지 않은 경우에는 NULL로 설정하십시오. 지시하는 문자열의 최대 길이는 8자입니다.
level	char	벤더 제품의 현재 레벨. 중요하지 않은 경우에는 NULL로 설정하십시오. 지시하는 문자열의 최대 길이는 8자입니다.
server_id	char	데이터베이스가 있는 서버를 식별하는 고유 이름. 지시하는 문자열의 최대 길이는 8자입니다.
max_bytes_per_txn	sqlint32	지원되는 전송 버퍼의 최대 크기. 벤더에서 바이트 단위로 지정합니다. 벤더의 초기화 API 리턴 코드가 유효하지 않는 버퍼 크기 지정을 나타내는 SQLUV_BUFF_SIZE인 경우에만 사용됩니다.
num_objects_in_backup	sqlint32	완전한 백업을 수행하는 데 사용된 세션 수. 리스토어 조작 중에 모든 백업 이미지가 처리된 시기를 판별하는 데 사용됩니다.
reserve	void	나중에 사용하기 위해 예약됨

주: 모든 char 데이터 유형 필드는 널(null)로 끝나는 문자열입니다.

API 및 데이터 구조 구문

```
typedef struct Vendor_info
{
  char *vendor_id;
  char *version;
  char *release;
  char *level;
  char *server_id;
  sqlint32 max_bytes_per_txn;
  sqlint32 num_objects_in_backup;
  void *reserve;
} Vendor_info;
```

Init_input

DB2에서 벤더 디바이스에 대한 논리 링크를 설정하고 작성하기 위해 제공하는 정보가 포함됩니다. DB2가 이 데이터 구조를 사용하여 sqluvint 및 sqluvdel API를 통해 정보를 백업 및 리스토어 벤더 스토리지 플러그인에 보냅니다.

표 22. Init_input 구조의 필드

필드 이름	데이터 유형	설명
DB2_session	struct DB2_info	DB2 측면에서 세션에 대한 설명
size_options	unsigned short	옵션 필드 길이. DB2 백업이나 리스토어 함수를 사용하는 경우 이 필드의 데이터는 VendorOptionsSize 매개변수에서 직접 전달됩니다.
size_HI_order	sqluint32	DB 크기의 상위 32비트 계산(바이트), 총 크기는 64비트.
size_LOW_order	sqluint32	DB 크기의 하위 32비트 계산(바이트), 총 크기는 64비트.
옵션	void	이 정보는 백업이나 리스토어 함수가 호출될 때 응용프로그램에서 전달됩니다. 이 데이터 구조는 플랫해야 합니다. 즉, 간접 레벨이 지원되지 않습니다. 바이트 리버설이 수행되지 않으며 이 데이터의 코드 페이지는 점검되지 않습니다. DB2 백업이나 리스토어 함수를 사용하는 경우 이 필드의 데이터는 pVendorOptions 매개변수에서 직접 전달됩니다.
reserve	void	나중에 사용하기 위해 예약됨
prompt_lvl	char	백업이나 리스토어 조작이 호출될 때 사용자가 요청하는 프롬프트 레벨. 지시하는 문자열의 최대 길이는 1자입니다. 이 필드는 널(null)로 끝나는 문자열입니다.
num_sessions	unsigned short	백업이나 리스토어 조작이 호출될 때 사용자가 요청하는 세션 수.

API 및 데이터 구조 구문

```
typedef struct Init_input
{
   struct DB2_info *DB2_session;
   unsigned short   size_options;
   sqluint32 size_HI_order;
   sqluint32 size_LOW_order;
   void *options;
```

```
   void *reserve;
   char *prompt_lvl;
   unsigned short   num_sessions;
} Init_input;
```

Init_output

백업 및 리스토어 벤더 스토리지 플러그인에서 DB2로 리턴된 세션 및 정보에 대한 제어 블록이 포함됩니다. sqluvint 및 sqluvdel API가 이 데이터 구조를 사용합니다.

표 23. Init_output 구조의 필드

필드 이름	데이터 유형	설명
vendor_session	struct Vendor_info	DB2의 벤더를 식별하는 정보가 포함됩니다.
pVendorCB	void	벤더 제어 블록.
reserve	void	나중에 사용하기 위해 예약됨

API 및 데이터 구조 구문

```
typedef struct Init_output
{
   struct Vendor_info * vendor_session;
   void               * pVendorCB;
   void               * reserve;
} Init_output ;
```

Data

DB2와 벤더 디바이스 사이에 전송된 데이터가 포함됩니다. 이 구조는 데이터가 벤더 디바이스에 기록되는 경우에 sqluvput API에서 사용되고 벤더 디바이스에서 데이터를 읽는 경우에는 sqluvget API에서 사용됩니다.

표 24. Data 구조의 필드

필드 이름	데이터 유형	설명
obj_num	sqlint32	백업 조작 중에 DB2에서 지정된 시퀀스 번호
buff_size	sqlint32	버퍼 크기
actual_buff_size	sqlint32	송신 또는 수신된 실제 바이트 수. buff_size를 초과하면 안됩니다.
dataptr	void	데이터 버퍼의 포인터. DB2가 버퍼에 대해 스페이스를 할당합니다.
reserve	void	나중에 사용하기 위해 예약됨

API 및 데이터 구조 구문

```
typedef struct Data
{
   sqlint32 obj_num;
   sqlint32 buff_size;
   sqlint32 actual_buff_size;
   void *dataptr;
   void *reserve;
} Data;
```

Return_code

백업 및 리스토어 벤더 스토리지 플러그인으로 DB2에 리턴 중인 오류에 대한 간략한 설명의 리턴 코드를 포함합니다. 이 데이터 구조는 모든 벤더 스토리지 플러그인 API에서 사용됩니다.

표 25. Return_code 구조의 필드

필드 이름	데이터 유형	설명
return_code(아래 메모 참조)	sqlint32	벤더 API에서 리턴 코드를 리턴합니다.
설명	char	리턴 코드에 대한 간단한 설명
reserve	void	나중에 사용하기 위해 예약됨

주: 이는 다양한 DB2 API가 리턴하는 값과 동일하지 않은 벤더 고유의 리턴 코드입니다. 벤더 제품에서 승인되는 리턴 코드에 대한 개별적인 API 설명을 참조하십시오.

API 및 데이터 구조 구문

```
typedef struct Return_code
{
        sqlint32 return_code;
        char description[SQLUV_COMMENT_LEN];
        void *reserve;
} Return_code;
```

제 167 장 백업 및 리스토어 조작에서 압축을 사용하기 위한 DB2 API

DB2는 백업 이미지를 압축하고 압축을 해제하는 써드 파티의 압축 제품에 사용할 수 있는 API를 제공합니다. 이 인터페이스는 DB2의 표준 파트로 지원되는 압축 라이브러리를 보강하거나 교체하도록 설계되었습니다. 압축 플러그인 인터페이스는 DB2 API의 백업 및 리스토어에 사용하거나 벤더 스토리지 디바이스의 백업 및 리스토어 플러그인에 사용할 수 있습니다.

DB2는 많은 벤더에서 사용할 수 있는 일반적인 용도의 압축 및 압축 해제를 제공하는 API 프로토타입 세트를 정의합니다. 벤더는 이 API를 Linux 및 UNIX 시스템의 공유 라이브러리 또는 Windows 운영 체제의 DLL로 제공합니다. DB2가 API를 호출하면 백업이나 리스토어 루틴을 호출하여 지정한 공유 라이브러리 또는 DLL이 로드되고 벤더에서 제공하는 API가 호출되어 요청된 태스크를 수행합니다.

운영 개요

DB2 및 벤더 제품과의 인터페이스를 위해 8개의 API가 정의되어 있습니다.

- InitCompression - 압축 라이브러리 초기화
- GetSavedBlock - 백업 이미지를 위한 벤더 블록 가져오기
- Compress - 데이터 블록 압축
- GetMaxCompressedSize - 가능한 최대 버퍼 크기 계산
- TermCompression - 압축 라이브러리 종료
- InitDecompression - 압축 해제 라이브러리 초기화
- Decompress - 데이터 블록 압축 해제
- TermDecompression - 압축 해제 라이브러리 종료

DB2는 COMPR_DB2INFO 구조에 대한 정의를 제공합니다. 벤더는 백업 및 리스토어에 압축을 사용하는 각 기타 구조 및 API에 대한 정의를 제공합니다. 구조, 프로토타입 및 상수는 DB2에서 제공되는 `sqlucompr.h`에 정의됩니다.

DB2가 이 API를 호출하고 벤더는 이 API를 Linux 및 UNIX 시스템의 공유 라이브러리 또는 Windows 운영 체제의 DLL로 제공합니다.

주: 공유 라이브러리 또는 DLL 코드는 데이터베이스 엔진 코드 일부로 실행됩니다. 따라서 다시 입력하고 제대로 디버깅해야 합니다. 오류가 있는 함수는 데이터베이스의 데이터 무결성을 손상시킬 수도 있습니다.

샘플 호출 시퀀스

백업을 위해 다음 호출 시퀀스가 각 세션에 대해 DB2에서 실행됩니다.

```
InitCompression
```

뒤에는 0 - 1이 옵니다.

```
GetMaxCompressedSize
Compress
```

뒤에는 1이 옵니다.

```
TermCompress
```

리스토어의 경우 각 세션에 대한 호출 시퀀스는 다음과 같습니다.

```
InitDecompression
```

뒤에는 1 - n이 옵니다.

```
Decompress
```

뒤에는 1이 옵니다.

```
TermCompression
```

압축 플러그인 인터페이스 리턴 코드

다음은 API가 리턴하는 리턴 코드입니다. 지정된 위치를 제외하고 DB2는 0이 아닌 리턴 코드가 리턴되면 백업이나 리스토어를 종료합니다.

```
SQLUV_OK
```

0

조작 성공

```
SQLUV_BUFFER_TOO_SMALL
```

100

목표 버퍼가 너무 작습니다. 백업에서 표시되면 tgtAct 필드가 오브젝트 압축에 필요한 계산된 *크기*를 표시합니다. DB2는 지정된 *크기* 이상의 버퍼로 조작을 다시 시도합니다. 리스토어에 표시되면 조작이 실패합니다.

```
SQLUV_PARTIAL_BUFFER
```

101

버퍼가 부분적으로 압축됩니다. 백업에 표시되면 srcAct 필드가 실제로 압축된 실제 데이터 양을 표시하고 tgtAct 필드는 압축된 데이터의 실제 *크기*를 나타냅니다. 리스토어에 표시되면 조작이 실패합니다.

SQLUV_NO_MEMORY

102

메모리 부족

SQLUV_EXCEPTION

103

코드에 신호 또는 예외가 발생했습니다.

SQLUV_INTERNAL_ERROR

104

내부 오류가 발견되었습니다.

SQLUV_BUFFER_TOO_SMALL과 SQLUV_PARTIAL_BUFFER의 차이점은 SQLUV_PARTIAL_BUFFER가 리턴될 때 DB2가 출력 버퍼의 데이터를 유효하게 고려하는 점입니다.

COMPR_CB

이 구조는 제어 블록으로 플러그인 라이브러리에서 내부적으로 사용합니다. 여기에는 compression 및 decompression API에서 내부적으로 사용되는 데이터가 포함됩니다. DB2는 플러그인 라이브러리에 대한 각 호출에 이 구조를 전달하지만 구조 관리 정의 및 구조의 메모리 관리를 포함한 구조의 모든 사항은 여전히 라이브러리에 남아 있습니다.

API 및 데이터 구조 구문

```
struct COMPR_CB;
```

COMPR_DB2INFO

DB2 환경에 대해 설명합니다. DB2는 이 구조를 할당하고 정의하여 매개변수로 InitCompression 및 InitDecompression API에 전달합니다. 이 구조는 백업 또는 리스토어 중인 데이터베이스에 대해 설명하고 조작이 발생한 DB2 환경에 대한 세부사항을 제공합니다. dbalias, instance, node, catnode 및 timestamp 매개변수를 사용하여 백업 이미지 이름을 지정합니다.

API 및 데이터 구조 구문

```
struct COMPR_DB2INFO {
   char tag[16];
   db2Uint32 version;
   db2Uint32 size;
```

```
   char dbalias[SQLU_ALIAS_SZ+1];
   char instance[SQL_INSTNAME_SZ+1];
   SQL_PDB_NODE_TYPE node;
   SQL_PDB_NODE_TYPE catnode;
   char timestamp[SQLU_TIME_STAMP_LEN+1];
   db2Uint32 bufferSize;
   db2Uint32 options;
   db2Uint32 bkOptions;
   db2Uint32 db2Version;
   db2Uint32 platform;
   db2int32 comprOptionsByteOrder;
   db2Uint32 comprOptionsSize;
   void *comprOptions;
   db2Uint32 savedBlockSize;
   void *savedBlock;
};
```

COMPR_DB2INFO 데이터 구조 매개변수

tag 구조가 부각되도록 사용됩니다. 이는 항상 "COMPR_DB2INFO ₩0" 문자열로 설정됩니다.

version

API가 추가 필드 표시를 나타낼 수 있도록 사용 중인 구조 버전을 표시합니다. 현재는 버전은 1입니다. 이후에는 이 구조에 더 많은 매개변수가 추가될 수 있습니다.

size COMPR_DB2INFO 구조의 크기를 바이트 단위로 지정합니다.

dbalias

데이터베이스 별명. 리스토어 조작의 경우 dbalias는 소스 데이터베이스의 별명입니다.

인스턴스

인스턴스 이름

node 노드 번호

catnode

카탈로그 노드 번호

시간소인

백업 또는 리스토어 중인 이미지의 시간소인

bufferSize

전송 버퍼 크기를 지정합니다(4K 페이지).

옵션 db2Backup API 또는 db2Restore API에 지정된 iOptions 매개변수

bkOptions

리스토어 조작의 경우 백업이 작성될 때 db2Backup API에서 사용된 iOptions 매개변수를 지정합니다. 백업 조작에서는 영(0)으로 설정됩니다.

db2Version

DB2 엔진의 버전을 지정합니다.

platform

DB2 엔진이 실행 중인 플랫폼을 지정합니다. 값은 sqlmon.h(include 디렉토리에 있음)에 나열된 값 중 하나입니다.

comprOptionsByteOrder

API가 실행되는 클라이언트에서 사용되는 바이트 순서를 지정합니다. DB2는 comprOptions를 통해 전달되는 데이터를 인터럽트하거나 변환하지 않기 때문에 이 필드를 사용하여 데이터를 사용하기 전에 바이트 전환해야 하는지를 판별하십시오. 변환은 플러그인 라이브러리 자체로 수행해야 합니다.

comprOptionsSize

db2Backup 및 db2Restore API에서 piComprOptionsSize 매개변수 값을 지정합니다.

comprOptions

db2Backup 및 db2Restore API에서 piComprOptions 매개변수 값을 지정합니다.

savedBlockSize

savedBlock의 크기(바이트)

savedBlock

DB2에서는 플러그인 라이브러리가 백업 이미지로 임의 데이터 블록을 저장할 수 있습니다. 이런 데이터 블록이 특정 백업으로 저장되는 경우 리스토어 조작에서 이 필드에 리턴됩니다. 백업 조작의 경우 이 필드는 영(0)으로 설정됩니다.

COMPR_PIINFO

플러그인 라이브러리에서 이 구조를 사용하여 자체 DB2를 설명합니다. DB2가 이 구조를 할당하고 초기화하며 키 필드는 InitCompression API 호출 시에 플러그인 라이브러리로 채워집니다.

API 및 데이터 구조 구문

```
struct COMPR_PIINFO {
   char tag[16];
   db2Uint32 version;
   db2Uint32 size;
   db2Uint32 useCRC;
   db2Uint32 useGran;
   db2Uint32 useAllBlocks;
   db2Uint32 savedBlockSize;
};
```

COMPR_PIINFO 데이터 구조 매개변수

tag 구조가 부각되도록 사용됩니다. (DB2가 설정합니다.) 이는 항상 "COMPR_PIINFO ₩0" 문자열로 설정됩니다.

version

API가 추가 필드 표시를 나타낼 수 있도록 사용 중인 구조 버전을 표시합니다. 현재는 버전이 1입니다.

(DB2가 설정합니다.) 이후에는 이 구조에 더 많은 필드가 추가될 수 있습니다.

size COMPR_PIINFO 구조의 *크기*를 나타냅니다(바이트). (DB2가 설정합니다.)

useCRC

DB2는 압축 플러그인에서 32비트 CRC 또는 체크섬 값을 사용하여 압축 또는 압축 해제 중인 데이터 무결성의 유효성을 확인하도록 합니다.

라이브러리에서 이런 점검을 사용하는 경우 이 필드를 1로 설정하고 그 이외의 경우에는 필드를 0으로 설정합니다.

useGran

압축 루틴이 임의 *크기* 증분 방식으로 데이터를 압축할 수 있는 경우 라이브러리는 이 필드를 1로 설정합니다. 압축 루틴이 데이터를 바이트 *크기* 단위의 증분 방식으로만 데이터를 압축하는 경우 라이브러리는 이 필드를 0으로 설정합니다. 이 표시기 설정에 대한 자세한 내용은 압축 API의 srcGran 매개변수 설명을 참조하십시오.

리스토어 조작에서는 이 매개변수가 무시됩니다.

useAllBlocks

DB2가 원래 압축되지 않은 블록보다 *크기*가 큰 압축된 데이터 블록을 백업해야 하는지를 지정합니다. 디폴트로 DB2는 압축된 버전이 더 큰 경우 압축되지 않은 데이터를 저장하지만 플러그인 라이브러리가 이유에 상관 없이 압축된 데이터를 백업하는 경우도 있습니다. DB2가 모든 블록에 대해 압축된 버전의 데이터를 저장하는 경우 라이브러리는 이 값을 1로 설정합니다. DB2가 압축된 데이터 버전이 원래 데이터보다 작은 경우에만 이를 저장하는 경우 라이브러리는 이 값을 0으로 설정합니다. 리스토어 조작에서는 이 필드가 무시됩니다.

savedBlockSize

DB2에서는 플러그인 라이브러리가 백업 이미지로 임의 데이터 블록을 저장할 수 있습니다. 이런 데이터 블록이 특정 백업으로 저장되는 경우 라이브러리는 이 매개변수를 이 데이터에 할당되는 블록 *크기*로 설정합니다. (실제 데이터가 후속 API 호출에서 DB2에 전달됩니다.) 데이터를 저장하지 않는 경우 플러그인 라이브러리는 이 매개변수를 영(0)으로 설정합니다. 리스토어 조작에서는 이 매개변수가 무시됩니다.

Compress - 데이터 블록 압축

데이터 블록을 압축합니다. src 매개변수는 *크기*가 srcLen 바이트인 데이터 블록을 지시합니다. tgt 매개변수는 *크기*가 tgtSize 바이트인 버퍼를 지시합니다. 플러그인 라이브러리는 scr 주소의 데이터를 압축하고 압축된 데이터를 tgt 주소의 버퍼에 기록합니다. 압축되었던 데이터의 압축되지 않은 실제 *크기*가 scrAct에 저장됩니다. 압축된 데이터의 실제 *크기*는 tgtAct로 리턴됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlucompr.h
```

API 및 데이터 구조 구문

```
int Compress(
      struct COMPR_CB *pCB,
      const char *src,
      db2int32 srcSize,
      db2Uint32 srcGran,
      char *tgt,
      db2int32 tgtSize,
      db2int32 *srcAct,
      db2int32 *tgtAct,
      db2Uint32 *tgtCRC);
```

Compress API 매개변수

pCB 입력. InitCompression API 호출이 리턴하는 제어 블록입니다.

src 입력. 압축된 데이터 블록의 포인터

srcLen

입력. 압축되는 데이터의 블록 *크기*(바이트)

srcGran

입력. piInfo->useGran에 대해 라이브러리가 리턴한 값이 1인 경우 srcGran은 데이터의 페이지 *크기*를 log2로 지정합니다. (예를 들어, 데이터의 페이지 *크기*가 4096바이트인 경우 srcGran은 12입니다.) 라이브러리는 실제로 압축된 데이터 *크기*(srcAct)가 이 페이지 *크기*의 정확한 배수인지 확인합니다. 라이브러리가 useGran 플래그를 설정한 경우 DB2는 압축된 데이터를 백업 이미지에 적합하도록 좀 더 효율적인 알고리즘을 사용할 수 있습니다. 즉, 두 플러그인의

성능이 더 좋아지고 압축된 백업 이미지는 더 작아집니다. piInfo->srcGran에 대해 라이브러리가 리턴한 값이 0인 경우 세분화도는 1바이트입니다.

tgt 입력 및 출력. 압축된 데이터의 목표 버퍼. DB2가 이 목표 버퍼를 제공하고 플러그인은 src의 데이터를 압축하여 압축된 데이터를 여기에 작성합니다.

tgtSize

입력. 목표 버퍼의 *크기*(바이트)

srcAct

출력. 압축되었던 src의 압축되지 않은 데이터의 실제 *크기*(바이트)

tgtAct 출력. tgt에 저장된 압축된 데이터의 실제 *크기*(바이트)

tgtCRC

출력. piInfo->useCRC에 대해 라이브러리가 리턴한 값이 1인 경우 압축되지 않은 블록의 CRC 값은 tgtCRC로 리턴됩니다. piInfo->useCRC에 대해 라이브러리가 리턴한 값이 0인 경우 tgtCRC는 NULL 포인터입니다.

Decompress - 데이터 블록 압축 해제

데이터 블록을 압축 해제합니다. src 매개변수는 *크기*가 srcLen 바이트인 데이터 블록을 지시합니다. tgt 매개변수는 *크기*가 tgtSize 바이트인 버퍼를 지시합니다. 플러그인 라이브러리는 src 주소의 데이터를 압축 해제하고 압축되지 않은 데이터를 tgt 주소의 버퍼에 기록합니다. 압축되지 않은 데이터의 실제 *크기*는 tgtLen으로 리턴됩니다. piInfo->useCRC에 대해 라이브러리가 리턴한 값이 1인 경우 압축되지 않은 블록의 CRC는 tgtCRC로 리턴됩니다. piInfo->useCRC에 대해 라이브러리가 리턴한 값이 0인 경우 tgtLen은 NULL 포인터입니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlucompr.h
```

API 및 데이터 구조 구문

```
int Decompress(
      struct COMPR_CB *pCB,
      const char *src,
      db2int32 srcSize,
```

```
char *tgt,
db2int32 tgtSize,
db2int32 *tgtLen,
db2Uint32 *tgtCRC);
```

Decompress API 매개변수

pCB 입력. InitDecompression API 호출이 리턴하는 제어 블록입니다.

src 입력. 압축 해제되는 데이터 블록의 포인터

srcLen
입력. 압축 해제되는 데이터의 블록 *크기*(바이트)

tgt 입력 및 출력. 압축 해제된 데이터의 목표 버퍼. DB2가 이 목표 버퍼를 제공하고 플러그인은 src의 데이터를 압축 해제하여 압축 해제된 데이터를 여기에 작성합니다.

tgtSize
입력. 목표 버퍼의 *크기*(바이트)

tgtLen
출력. tgt에 저장된 압축 해제된 데이터의 실제 *크기*(바이트)

tgtCRC
출력. piInfo->useCRC에 대해 라이브러리가 리턴한 값이 1인 경우 압축되지 않은 블록의 CRC 값은 tgtCRC로 리턴됩니다. piInfo->useCRC에 대해 라이브러리가 리턴한 값이 0인 경우 tgtCRC는 NULL 포인터입니다.

GetMaxCompressedSize - 가능한 최대 버퍼 크기 계산

데이터 블록을 압축하는 데 필요한 가능한 최대 퍼버 *크기*를 계산합니다. srcLen은 압축할 데이터 블록 *크기*를 나타냅니다. 라이브러리는 tgtLen으로 압축한 후에 버퍼의 이론적인 최대 *크기*를 리턴합니다.

DB2는 tgtLen으로 리턴된 값을 사용하여 내부적으로 메모리 사용을 최적화합니다. 값을 계산하지 않거나 잘못된 값으로 계산하면 DB2가 단일 데이터 블록에 대해 Compress API를 두 번 이상 호출하거나 유틸리티 힙의 메모리를 낭비하는 결과를 초래합니다. DB2는 리턴된 값에 상관 없이 여전히 제대로 백업을 작성합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlucompr.h
```

API 및 데이터 구조 구문

```
int GetMaxCompressedSize(
      struct COMPR_CB *pCB,
      db2Uint32 srcLen);
```

GetMaxCompressedSize API 매개변수

pCB 입력. InitCompression API 호출이 리턴하는 제어 블록입니다.

srcLen
입력. 압축하려는 데이터 블록 *크기*(바이트)

GetSavedBlock - 백업 이미지의 블록 데이터 벤더 가져오기

백업 이미지로 저장되는 데이터의 벤더 특정 데이터 블록을 가져옵니다. 라이브러리가 piInfo->savedBlockSize에 대해 0이 아닌 값을 리턴하면 DB2는 blockSize 값을 사용하여 GetSavedBlock을 호출합니다. 플러그인 라이브러리는 지정된 *크기*의 데이터를 데이터에서 참조하는 메모리에 기록합니다. 이 API가 백업 전용의 db2bm의 *초기* 데이터 처리 중에 호출됩니다. db2Backup API에 병렬 처리 > 1이 지정된 경우에도 이 API는 백업별로 1개만 호출됩니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlucompr.h
```

API 및 데이터 구조 구문

```
int GetSavedBlock(
      struct COMPR_CB *pCB,
      db2Uint32 blockSize,
      void *data);
```

GetSavedBlock API 매개변수

pCB 입력. InitCompression API 호출이 리턴하는 제어 블록입니다.

blocksize

입력. InitCompression API 호출로 piInfo->savedBlockSize에 리턴되는 블록 크기입니다.

데이터 출력. 백업 이미지로 저장되는 데이터의 벤더 특정 데이터 블록입니다.

InitCompression - 압축 라이브러리 초기화

압축 라이브러리를 초기화합니다. DB2가 db2Info 및 piInfo 구조를 전달합니다. 라이브러리는 piInfo의 해당하는 매개변수를 채우고 pCB를 할당하며 할당된 메모리의 포인터를 리턴합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlucompr.h
```

API 및 데이터 구조 구문

```
int InitCompression(
      const struct COMPR_DB2INFO
                        *db2Info,
      struct COMPR_PIINFO
                        *piInfo,
      struct COMPR_CB  **pCB);
```

InitCompression API 매개변수

db2Info

입력. 백업 중인 데이터베이스에 대해 설명하고 조작이 수행 중인 DB2 환경에 대한 세부사항을 제공합니다.

piInfo 출력. 이 구조는 플러그인 라이브러리가 DB2에 자체를 설명하는 데 사용됩니다. DB2가 할당하고 초기화하며 키 매개변수는 플러그인 라이브러리로 채워집니다.

pCB 출력. 압축 라이브러리에서 사용되는 제어 블록입니다. 플러그인 라이브러리가 구조의 메모리를 관리합니다.

InitDecompression - 압축 해제 라이브러리 초기화

압축 해제 라이브러리를 초기화합니다. DB2는 db2Info 구조를 전달합니다. 라이브러리가 pCB를 할당하고 할당된 메모리의 포인터를 리턴합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlucompr.h
```

API 및 데이터 구조 구문

```
int InitDecompression(
      const struct COMPR_DB2INFO
                      *db2Info,
      struct COMPR_CB  **pCB);
```

InitDecompression API 매개변수

db2Info

입력. 백업 중인 데이터베이스에 대해 설명하고 조작이 수행 중인 DB2 환경에 대한 세부사항을 제공합니다.

pCB 출력. 압축 해제 라이브러리에서 사용되는 제어 블록입니다. 플러그인 라이브러리가 구조의 메모리를 관리합니다.

TermCompression - 압축 라이브러리 중지

압축 라이브러리를 종료합니다. 라이브러리는 pCB에 사용되는 메모리를 사용 가능화합니다.

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlucompr.h
```

API 및 데이터 구조 구문

```
int TermCompression(
      struct COMPR_CB *pCB);
```

TermCompression API 매개변수

pCB 입력. InitCompression API 호출이 리턴하는 제어 블록입니다.

TermDecompression - 압축 해제 라이브러리 중지

압축 해제 라이브러리를 종료합니다. 라이브러리는 pCB에 사용되는 메모리를 사용 가능화합니다. compression API에서 내부적으로 사용되는 모든 메모리는 라이브러리에서 관리됩니다. 플러그인 라이브러리는 COMPR_CB 구조에서 사용되는 메모리도 관리합니다. DB2는 데이터 버퍼에서 사용되는 메모리를 관리합니다(compression API에서 src 및 tgt 매개변수).

권한 부여

없음

필수 연결

없음

API 내장 파일

```
sqlucompr.h
```

API 및 데이터 구조 구문

```
int TermDecompression(
      struct COMPR_CB *pCB);
```

TermDecompression API 매개변수

pCB 입력. InitDecompression API 호출이 리턴하는 제어 블록입니다.

제 10 부 API에서 사용된 데이터 구조

제 168 장 db2DistMapStruct

이 구조는 db2GetDistMap API에서 사용됩니다.

표 26. db2DistMapStruct 구조의 필드

필드 이름	데이터 유형	설명
tname	unsigned char	완전한 테이블 이름
distmaplen	unsigned short	분산 맵 길이
distmap[SQL_PDB_MAP_SIZE}	SQL_PDB_NODE_TYPE	분산 맵.
sqld	unsigned short	사용된 SQLDPARTKEY 요소 수
sqlpartkey[SQL_MAX_NUM_PART_KEYS]	sqlpartkey	테이블 데이터 분산에 사용되는 분산 키

API 및 데이터 구조 구문

```
SQL_STRUCTURE db2DistMapStruct
{
   unsigned char     *tname;                           /* fully qualified table name          */
   unsigned short    distmaplen;                       /* Length of distribution map          */
   SQL_PDB_NODE_TYPE distmap[SQL_PDB_MAP_SIZE];        /* Distribution map                    */
   unsigned short    sqld;                             /* # of used SQLPARTKEY elements       */
   struct sqlpartkey sqlpartkey[SQL_MAX_NUM_PART_KEYS]; /* Distribution Keys                   */
};
```

제 169 장 db2HistoryData

이 구조를 사용하여 db2HistoryGetEntry API 호출 후에 정보를 리턴합니다.

표 27. db2HistoryData 구조의 필드

필드 이름	데이터 유형	설명
ioHistDataID	char(8)	8바이트 구조 ID 및 스토리지 덤프의 ″아이캐쳐″. 유일한 유효 값은 SQLUHINF입니다. 이 문자열에 대한 기호 정의는 존재하지 않습니다.
oObjectPart	db2Char	처음 14자는 *yyyymmddhhmmss* 형식의 시간소인이며 조작이 시작된 시간을 표시합니다. 다음 세 자는 시퀀스 번호입니다. 백업 이미지가 여러 파일이나 여러 테이프에 저장되는 경우 각 백업 조작은 이 파일에서 여러 개의 항목을 생성할 수 있습니다. 시퀀스 번호를 사용하여 여러 개의 위치를 지정할 수 있습니다. 리스토어 및 로드 조작은 이 파일에서 해당 백업의 시퀀스 번호 ″001″에 해당되는 하나의 항목만 가지고 있습니다. 시퀀스 번호와 결합된 시간소인은 고유해야 합니다.
oEndTime	db2Char	조작이 완료되는 시기를 표시하는 *yyyymmddhhmmss* 형식의 시간소인
oFirstLog	db2Char	가장 빠른 로그 파일 ID(범위 S0000000 - S9999999): • 온라인 백업에 대한 롤 포워드 복구를 적용하는 데 필요합니다. • 오프라인 백업에 대한 롤 포워드 복구를 적용하는 데 필요합니다. • 로드가 시작될 때 현재 사용 중이었던 전체 데이터베이스 또는 테이블 스페이스 레벨 백업을 리스토어한 후 적용됩니다.
oLastLog	db2Char	가장 늦은 로그 파일 ID(범위 S0000000 - S9999999): • 온라인 백업에 대한 롤 포워드 복구를 적용하는 데 필요합니다. • 오프라인 백업에 대한 현재 특정 시점에 롤 포워드 복구를 적용하는 데 필요합니다. • 로드 조작이 완료될 때 현재 사용 중이었던 전체 데이터베이스 또는 테이블 스페이스 레벨 백업을 리스토어한 후 적용됩니다(롤 포워드 복구가 적용되지 않는 경우 **oFirstLog**와 같게 됨).
oID	db2Char	고유한 백업 또는 테이블 ID
oTableQualifier	db2Char	테이블 규정자.
oTableName	db2Char	테이블 이름

표 27. db2HistoryData 구조의 필드 (계속)

필드 이름	데이터 유형	설명
oLocation	db2Char	백업 및 로드 사본의 경우 이 필드는 데이터가 저장된 위치를 표시합니다. 파일의 여러 항목이 필요한 조작의 경우, **oObjectPart** 매개변수가 정의한 시퀀스 번호는 지정된 위치에서 발견되는 백업 파트를 식별합니다. 리스토어 및 로드 조작의 경우, 위치는 항상 리스토어 또는 로드된 데이터의 첫 번째 파트(여러 파트 백업의 경우 시퀀스 ″001″에 해당하는)가 저장되었습니다. **oLocation**의 데이터는 **oDeviceType** 매개변수에 따라 다르게 해석됩니다. • 디스크 또는 디스켓(D 또는 K)의 경우 완전한 파일 이름 • 테이프(T)의 경우 볼륨 레이블 • TSM(A 및 F)의 경우, 백업을 수행한 벤더 라이브러리 이름/경로 • User Exit 또는 기타(U 또는 O)의 경우 무형식 텍스트
oComment	db2Char	무형식 텍스트 주석
oCommandText	db2Char	명령 텍스트 또는 DDL
oLastLSN	db2LSN	마지막 로그 시퀀스 번호
oEID	구조	고유한 항목 ID
poEventSQLCA	구조	기록된 이벤트의 결과 sqlca
poTablespace	db2Char	테이블 스페이스 이름 목록
iNumTablespaces	db2Uint32	**poTablespace** 목록에서 db2HistoryGetEntry API에 사용할 수 있는 항목 수
oNumTablespaces	db2Uint32	**poTablespace** 목록에서 db2HistoryGetEntry API에 사용된 항목 수. 각 테이블 스페이스 백업에는 하나 이상의 테이블 스페이스가 포함됩니다. 각 테이블 스페이스 리스토어 조작은 하나 이상의 테이블 스페이스를 교체합니다. 이 필드가 0(테이블 스페이스 레벨 백업 또는 리스토어 표시)이 아니면, 이 파일의 다음 라인은 백업 또는 리스토어된 테이블 스페이스의 이름을 포함하며 17자 문자열로 표시됩니다. 하나의 테이블 스페이스 이름이 라인마다 나타납니다.
oOperation	char	751 페이지의 표 28의 내용을 참조하십시오.
oObject	char	조작의 세분화도: 전체 데이터베이스의 경우 D, 테이블 스페이스의 경우 P 및 테이블의 경우 T.
oOptype	char	751 페이지의 표 29의 내용을 참조하십시오.
oStatus	char	항목 상태: 활성의 경우 A, 비활성의 경우 I, 만기된 경우 E, 삭제되는 경우 D, 삭제하지 않는 경우 X
oDeviceType	char	디바이스 유형. 이 필드는 **oLocation** 필드가 인터럽트되는 방법을 판별합니다. TSM의 경우 A, 클라이언트의 경우 C, 디스크의 경우 D, 스냅샷 백업의 경우 F, 디스켓의 경우 K, 로컬의 경우 L, 기타의 경우(다른 벤더 디바이스 지원의 경우) O, 파이프의 경우 P, 커서의 경우 Q, 서버의 경우 S, 테이프의 경우 T, User Exit의 경우 U입니다.

*표 28. db2HistoryData 구조에서 유효한 **oOperation** 값*

값	설명	C 정의	COBOL/FORTRAN 정의
A	테이블 스페이스 추가	DB2HISTORY_OP_ADD_TABLESPACE	DB2HIST_OP_ADD_TABLESPACE
B	백업	DB2HISTORY_OP_BACKUP	DB2HIST_OP_BACKUP
C	로드 사본	DB2HISTORY_OP_LOAD_COPY	DB2HIST_OP_LOAD_COPY
D	삭제된 테이블	DB2HISTORY_OP_DROPPED_TABLE	DB2HIST_OP_DROPPED_TABLE
F	롤 포워드	DB2HISTORY_OP_ROLLFWD	DB2HIST_OP_ROLLFWD
G	테이블 재구성	DB2HISTORY_OP_REORG	DB2HIST_OP_REORG
L	load	DB2HISTORY_OP_LOAD	DB2HIST_OP_LOAD
N	테이블 스페이스 이름 바꾸기	DB2HISTORY_OP_REN_TABLESPACE	DB2HIST_OP_REN_TABLESPACE
O	테이블 스페이스 삭제	DB2HISTORY_OP_DROP_TABLESPACE	DB2HIST_OP_DROP_TABLESPACE
Q	Quiesce	DB2HISTORY_OP_QUIESCE	DB2HIST_OP_QUIESCE
R	리스토어	DB2HISTORY_OP_RESTORE	DB2HIST_OP_RESTORE
T	테이블 스페이스 변경	DB2HISTORY_OP_ALT_TABLESPACE	DB2HIST_OP_ALT_TBS
U	언로드	DB2HISTORY_OP_UNLOAD	DB2HIST_OP_UNLOAD
X	로그 아카이브	DB2HISTORY_OP_ARCHIVE_LOG	DB2HIST_OP_ARCHIVE_LOG

*표 29. db2HistData 구조에서 유효한 **oOptype** 값*

oOperation	oOptype	설명	C/COBOL/FORTRAN 정의
B	F N I O D E	오프라인, 온라인, 증분 오프라인, 증분 온라인, 델타 오프라인, 델타 온라인	DB2HISTORY_OPTYPE_OFFLINE, DB2HISTORY_OPTYPE_ONLINE, DB2HISTORY_OPTYPE_INCR_OFFLINE, DB2HISTORY_OPTYPE_INCR_ONLINE, DB2HISTORY_OPTYPE_DELTA_OFFLINE, DB2HISTORY_OPTYPE_DELTA_ONLINE
F	E P	로그의 끝, 특정 시점	DB2HISTORY_OPTYPE_EOL, DB2HISTORY_OPTYPE_PIT
G	F N	오프라인, 온라인	DB2HISTORY_OPTYPE_OFFLINE, DB2HISTORY_OPTYPE_ONLINE
L	I R	삽입, 교체	DB2HISTORY_OPTYPE_INSERT, DB2HISTORY_OPTYPE_REPLACE
Q	S U X Z	Quiesce 공유, Quiesce 갱신, Quiesce 독점, Quiesce 재설정	DB2HISTORY_OPTYPE_SHARE, DB2HISTORY_OPTYPE_UPDATE, DB2HISTORY_OPTYPE_EXCL, DB2HISTORY_OPTYPE_RESET
R	F N I O R	오프라인, 온라인, 증분 오프라인, 증분 온라인, 재빌드	DB2HISTORY_OPTYPE_OFFLINE, DB2HISTORY_OPTYPE_ONLINE, DB2HISTORY_OPTYPE_INCR_OFFLINE, DB2HISTORY_OPTYPE_INCR_ONLINE, DB2HISTORY_OPTYPE_REBUILD

표 29. *db2HistData 구조에서 유효한* **oOptype** *값 (계속)*

oOperation	oOptype	설명	C/COBOL/FORTRAN 정의
T	C R	컨테이너 추가, 재조정	DB2HISTORY_OPTYPE_ADD_CONT, DB2HISTORY_OPTYPE_REB
X	N P M F 1 2	아카이브 로그 명령, 1차 로그 경로, 미러 로그 경로, 아카이브 실패 경로, 로그 아카이브 메소드 1, 로그 아카이브 메소드 2	DB2HISTORY_OPTYPE_ARCHIVE_CMD, DB2HISTORY_OPTYPE_PRIMARY, DB2HISTORY_OPTYPE_MIRROR, DB2HISTORY_OPTYPE_ARCHFAIL, DB2HISTORY_OPTYPE_ARCH1, DB2HISTORY_OPTYPE_ARCH2

표 30. *db2HistoryEID 구조의 필드*

필드 이름	데이터 유형	설명
ioNode ioHID	SQL_PDB_NODE_TYPE db2Uint32	노드 번호. 로컬 실행기록 파일 항목 ID

API 및 데이터 구조 구문

```
typedef SQL_STRUCTURE db2HistoryData
{
   char ioHistDataID[8];
   db2Char oObjectPart;
   db2Char oEndTime;
   db2Char oFirstLog;
   db2Char oLastLog;
   db2Char oID;
   db2Char oTableQualifier;
   db2Char oTableName;
   db2Char oLocation;
   db2Char oComment;
   db2Char oCommandText;
   db2LSN oLastLSN;
   db2HistoryEID oEID;
   struct sqlca *poEventSQLCA;
   struct db2Char *poTablespace;
   db2Uint32 iNumTablespaces;
   db2Uint32 oNumTablespaces;
   char oOperation;
   char oObject;
   char oOptype;
   char oStatus;
   char oDeviceType;
} db2HistoryData;

typedef SQL_STRUCTURE db2Char
{
   char *pioData;
   db2Uint32 iLength;
   db2Uint32 oLength;
} db2Char;
typedef SQL_STRUCTURE db2HistoryEID
```

```
{
   SQL_PDB_NODE_TYPE ioNode;
   db2Uint32 ioHID;
} db2HistoryEID;
```

db2Char 데이터 구조 매개변수

pioData

문자 데이터 버퍼의 포인터. NULL인 경우 데이터가 리턴되지 않습니다.

iLength

입력. **pioData** 버퍼 크기

oLength

출력. **pioData** 버퍼에 있는 데이터의 유효한 문자 수

db2HistoryEID 데이터 구조 매개변수

ioNode

이 매개변수는 입력이나 출력 매개변수로 사용할 수 있습니다. 노드 번호를 표시합니다.

ioHID 이 매개변수는 입력이나 출력 매개변수로 사용할 수 있습니다. 로컬 실행기록 파일 항목 ID를 표시합니다.

제 170 장 db2LSN 데이터 구조

db2ReadLog 및 db2ReadLogNoConn API에서 사용되는 이 통합에는 로그 시퀀스 번호의 정의가 포함됩니다. 로그 시퀀스 번호(LSN)는 데이터베이스 로그 내에서 상대 바이트 주소를 나타냅니다. 모든 로그 레코드는 이 번호로 식별됩니다. LSN은 데이터베이스 로그 시작으로부터의 로그 레코드 바이트 오프셋을 나타냅니다.

표 31. db2LSN 구조의 필드

필드 이름	데이터 유형	설명
lsnU64	db2Uint64	로그 시퀀스 번호를 지정합니다.

API 및 데이터 구조 구문

```
typedef SQL_STRUCTURE db2LSN
{
    db2Uint64   lsnU64;    /* Log sequence number           */
} db2LSN;
```

제 171 장 sql_dir_entry

이 구조는 DCS 디렉토리 API에서 사용됩니다.

표 32. SQL-DIR-ENTRY 구조의 필드

필드 이름	데이터 유형	설명
STRUCT_ID RELEASE CODEPAGE COMMENT LDB TDB AR PARM	SMALLINT SMALLINT SMALLINT CHAR(30) CHAR(8) CHAR(18) CHAR(32) CHAR(512)	구조 ID SQL_DCS_STR_ID(sqlenv에 정의)로 설정. 릴리스 버전(API가 지정). 주석의 코드 페이지. 데이터베이스의 선택적 설명. 데이터베이스의 로컬 이름. 시스템 데이터베이스 디렉토리의 데이터베이스 별명과 일치해야 함. 데이터베이스의 실제 이름. 응용프로그램 클라이언트의 이름. 트랜잭션 프로그램 접두부, 트랜잭션 프로그램 이름, SQLCODE 맵핑 파일 이름 및 연결 끊기 및 보안 옵션 포함.

주: 이 구조에서 전달되는 문자 필드는 널(NULL)로 종료되거나 필드 길이까지 채워진 공백이어야 합니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sql_dir_entry
{
        unsigned short         struct_id;
        unsigned short         release;
        unsigned short         codepage;
        _SQLOLDCHAR comment[SQL_CMT_SZ + 1];
        _SQLOLDCHAR ldb[SQL_DBNAME_SZ + 1];
        _SQLOLDCHAR tdb[SQL_LONG_NAME_SZ + 1];
        _SQLOLDCHAR ar[SQL_AR_SZ + 1];
        _SQLOLDCHAR parm[SQL_PARAMETER_SZ + 1];
};
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQL-DIR-ENTRY.
    05 STRUCT-ID             PIC 9(4) COMP-5.
    05 RELEASE-LVL           PIC 9(4) COMP-5.
    05 CODEPAGE              PIC 9(4) COMP-5.
    05 COMMENT               PIC X(30).
    05 FILLER                PIC X.
    05 LDB                   PIC X(8).
    05 FILLER                PIC X.
    05 TDB                   PIC X(18).
```

```
           05 FILLER                PIC X.
           05 AR                    PIC X(32).
           05 FILLER                PIC X.
           05 PARM                  PIC X(512).
           05 FILLER                PIC X.
           05 FILLER                PIC X(1).
       *
```

제 172 장 SQLB_TBS_STATS

이 구조를 사용하여 응용프로그램에 추가 테이블 스페이스 통계를 리턴합니다.

표 33. SQLB-TBS-STATS 구조의 필드

필드 이름	데이터 유형	설명
TOTALPAGES	INTEGER	테이블 스페이스에서 차지하는 총 운영 체제 스페이스(4KB 페이지 단위). DMS의 경우 이는 컨테이너 크기 합계입니다(오버헤드 포함). SMS의 테이블의 경우 이 테이블 스페이스에 저장되는 테이블에 사용되는 모든 파일 스페이스 합계입니다. 이는 SMS 테이블 스페이스에 리턴되는 일부 정보일 뿐이며 다른 필드는 이 값이나 영(0)으로 설정됩니다.
USEABLEPAGES	INTEGER	DMS의 경우 TOTALPAGES 빼기 (오버헤드 더하기 부분 Extent)와 동등. SMS의 경우 TOTALPAGES와 동등.
USEDPAGES	INTEGER	DMS의 경우 사용 중인 총 페이지 수. SMS의 경우 TOTALPAGES와 동등.
FREEPAGES	INTEGER	DMS의 경우 USEABLEPAGES 빼기 USEDPAGES와 동등. SMS의 경우 적용되지 않습니다.
HIGHWATERMARK	INTEGER	DMS의 경우 상위 워터 마크(water mark)는 현재 테이블 스페이스 어드레스 스페이스의 "끝"입니다. 즉, 테이블 스페이스의 최종 할당 Extent가 뒤에 오는 첫 번째 사용 가능한 테이블 스페이스의 페이지 번호

주: "상위 워터 마크(water mark)"는 아니지만 값을 줄일 수 있기 때문에 "현재 워터 마크(water mark)"입니다. SMS의 경우 적용되지 않습니다.

테이블 스페이스 재조정 중에 사용 가능한 페이지 수에는 새로 추가된 컨테이너 페이지가 포함되지만 이 새 페이지는 재조정이 완료될 때까지 사용 가능한 페이지 수에는 반영되지 않습니다. 테이블 스페이스 재조정이 수행되지 않으면 사용된 페이지 수에 사용 가능한 페이지 수를 더한 수가 사용할 수 있는 페이지 수와 동일하게 됩니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE SQLB_TBS_STATS
{
   sqluint32 totalPages;
   sqluint32 useablePages;
   sqluint32 usedPages;
   sqluint32 freePages;
   sqluint32 highWaterMark;
};
```

COBOL 구조

```
* File: sqlutil.cbl
01 SQLB-TBS-STATS.
    05 SQL-TOTAL-PAGES        PIC 9(9) COMP-5.
    05 SQL-USEABLE-PAGES      PIC 9(9) COMP-5.
    05 SQL-USED-PAGES         PIC 9(9) COMP-5.
    05 SQL-FREE-PAGES         PIC 9(9) COMP-5.
    05 SQL-HIGH-WATER-MARK    PIC 9(9) COMP-5.
*
```

제 173 장 SQLB_TBSCONTQRY_DATA

이 구조를 사용하여 응용프로그램에 컨테이너 데이터를 리턴합니다.

표 34. SQLB-TBSCONTQRY-DATA 구조의 필드

필드 이름	데이터 유형	설명
ID	INTEGER	컨테이너 ID
NTBS	INTEGER	항상 1입니다.
TBSID	INTEGER	테이블 스페이스 ID
NAMELEN	INTEGER	컨테이너 이름 길이(C 이외의 언어용).
NAME	CHAR(256)	컨테이너 이름
UNDERDBDIR	INTEGER	1(DB 디렉토리의 컨테이너) 또는 0(DB 디렉토리에 없는 컨테이너)
CONTTYPE	INTEGER	컨테이너 유형
TOTALPAGES	INTEGER	테이블 스페이스 컨테이너가 차지하는 총 페이지 수.
USEABLEPAGES	INTEGER	DMS의 경우 TOTALPAGES 빼기 오버헤드. SMS의 경우 TOTALPAGES와 동등.
OK	INTEGER	1(컨테이너에 액세스 가능) 또는 0(컨테이너에 액세스할 수 없음). 영(0)은 일반적으로 데이터베이스 관리자가 주의해야 하는 비정상적인 상황을 나타냅니다.

CONTTYPE의 가능한 값(sqlutil에 정의)은 다음과 같습니다.

SQLB_CONT_PATH
디렉토리 경로를 지정합니다(SMS 전용).

SQLB_CONT_DISK
원시 디바이스를 지정합니다(DMS 전용).

SQLB_CONT_FILE
파일을 지정합니다(DMS 전용).

API 및 데이터 구조 구문

```
SQL_STRUCTURE SQLB_TBSCONTQRY_DATA
{
   sqluint32 id;
   sqluint32 nTbs;
   sqluint32 tbsID;
   sqluint32 nameLen;
   char name[SQLB_MAX_CONTAIN_NAME_SZ];
   sqluint32 underDBDir;
   sqluint32 contType;
```

```
   sqluint32 totalPages;
   sqluint32 useablePages;
   sqluint32 ok;
};
```

COBOL 구조

```
* File: sqlutbcq.cbl
01 SQLB-TBSCONTQRY-DATA.
    05 SQL-ID                PIC 9(9) COMP-5.
    05 SQL-N-TBS             PIC 9(9) COMP-5.
    05 SQL-TBS-ID            PIC 9(9) COMP-5.
    05 SQL-NAME-LEN          PIC 9(9) COMP-5.
    05 SQL-NAME              PIC X(256).
    05 SQL-UNDER-DBDIR       PIC 9(9) COMP-5.
    05 SQL-CONT-TYPE         PIC 9(9) COMP-5.
    05 SQL-TOTAL-PAGES       PIC 9(9) COMP-5.
    05 SQL-USEABLE-PAGES     PIC 9(9) COMP-5.
    05 SQL-OK                PIC 9(9) COMP-5.
*
```

제 174 장 SQLB_TBSPQRY_DATA

이 구조를 사용하여 응용프로그램에 테이블 스페이스 데이터를 리턴합니다.

표 35. SQLB-TBSPQRY-DATA 구조의 필드

필드 이름	데이터 유형	설명
TBSPQVER	CHAR(8)	구조 버전 ID
ID	INTEGER	테이블 스페이스의 내부 ID
NAMELEN	INTEGER	테이블 스페이스 이름의 길이.
NAME	CHAR(128)	테이블 스페이스의 널(Null) 종료 이름
TOTALPAGES	INTEGER	CREATE TABLESPACE에 지정된 페이지 수(DMS만).
USEABLEPAGES	INTEGER	TOTALPAGES - 오버헤드 (DMS만). 이 값은 다음 4KB 배수로 잘라 버립니다.
FLAGS	INTEGER	테이블 스페이스의 비트 속성.
PAGESIZE	INTEGER	테이블 스페이스의 페이지 크기 (바이트). 현재 4KB로 고정되어 있습니다.
EXTSIZE	INTEGER	테이블 스페이스의 Extent 크기 (페이지 수).
PREFETCHSIZE	INTEGER	프리페치 크기.
NCONTAINERS	INTEGER	테이블 스페이스에 있는 컨테이너 수.
TBSSTATE	INTEGER	테이블 스페이스 상태.
LIFELSN	INTEGER(64-BIT)	테이블 스페이스의 원점을 식별하는 시간소인
FLAGS2	INTEGER	테이블 스페이스의 비트 속성.
MINIMUMRECTIME	CHAR(27)	시간에서 특정 시점 테이블 스페이스 롤 포워드에서 지정될 수 있는 가장 빠른 시점.
STATECHNGOBJ	INTEGER	TBSSTATE가 SQLB_LOAD_PENDING 또는 SQLB_DELETE_PENDING인 경우, 테이블 스페이스 상태가 설정되도록 한 테이블 스페이스 STATECHANGEID의 오브젝트 ID. 그렇지 않으면 0입니다.

표 35. SQLB-TBSPQRY-DATA 구조의 필드 (계속)

필드 이름	데이터 유형	설명
STATECHNGID	INTEGER	TBSSTATE가 SQLB_LOAD_PENDING 또는 SQLB_DELETE_PENDING인 경우, 테이블 스페이스 상태가 설정되도록 한 오브젝트 STATECHANGEOBJ의 테이블 스페이스 ID. 그렇지 않으면 0입니다.
NQUIESCERS	INTEGER	TBSSTATE가 SQLB_QUIESCED_SHARE, UPDATE 또는 EXCLUSIVE인 경우, 테이블 스페이스의 Quiescer 수 및 QUIESCER의 항목 수
QUIESCER	SQLB_QUIESCER_ DATA 구조 배열	각각의 배열 항목은 Quiesce 상태 오브젝트의 Quiesce 데이터로 구성됩니다.
FSCACHING	UNSIGNED CHAR	직접 입출력을 지원하기 위한 파일 시스템 캐싱 규정. 이는 31비트 필드입니다.
RESERVED	CHAR(31)	나중에 사용하기 위해 예약됨

FLAGS에 대해 가능한 값(sqlutil에 정의된)은 다음과 같습니다.

SQLB_TBS_SMS
시스템 관리 스페이스

SQLB_TBS_DMS
데이터베이스 관리 스페이스

SQLB_TBS_ANY
모든 유형의 영구 데이터. 일반 테이블 스페이스

SQLB_TBS_LONG
모든 유형의 영구 데이터. 대형 테이블 스페이스

SQLB_TBS_SYSTMP
시스템 임시 데이터.

SQLB_TBS_USRTMP
사용자 임시 데이터.

TBSSTATE의 가능한 값(sqlutil에 정의)은 다음과 같습니다.

SQLB_NORMAL
일반

SQLB_QUIESCED_SHARE
Quiesced: SHARE

SQLB_QUIESCED_UPDATE
Quiesced: UPDATE

SQLB_QUIESCED_EXCLUSIVE
Quiesced: EXCLUSIVE

SQLB_LOAD_PENDING
로드 보류

SQLB_DELETE_PENDING
삭제 보류

SQLB_BACKUP_PENDING
백업 보류

SQLB_ROLLFORWARD_IN_PROGRESS
롤 포워드 진행 중

SQLB_ROLLFORWARD_PENDING
롤 포워드 보류

SQLB_RESTORE_PENDING
리스토어 보류

SQLB_DISABLE_PENDING
작동 불가능화 보류

SQLB_REORG_IN_PROGRESS
재구성 진행 중

SQLB_BACKUP_IN_PROGRESS
백업 진행 중

SQLB_STORDEF_PENDING
스토리지를 정의해야 함

SQLB_RESTORE_IN_PROGRESS
리스토어 진행 중

SQLB_STORDEF_ALLOWED
스토리지를 정의할 수 있음

SQLB_STORDEF_FINAL_VERSION
스토리지 정의가 '최종' 상태에 있음

SQLB_STORDEF_CHANGED
롤 포워드 이전에 스토리지 정의가 변경되었음

SQLB_REBAL_IN_PROGRESS
DMS 재조정 프로그램이 활성 상태임

SQLB_PSTAT_DELETION
테이블 스페이스 삭제 진행 중

SQLB_PSTAT_CREATION
테이블 스페이스 작성 진행 중

FLAGS2에 대해 가능한 값(sqlutil에 정의된)은 다음과 같습니다.

SQLB_STATE_SET
서비스 전용.

API 및 데이터 구조 구문

```
SQL_STRUCTURE SQLB_TBSPQRY_DATA
{
   char tbspqver[SQLB_SVERSION_SIZE];
   sqluint32 id;
   sqluint32 nameLen;
   char name[SQLB_MAX_TBS_NAME_SZ];
   sqluint32 totalPages;
   sqluint32 useablePages;
   sqluint32 flags;
   sqluint32 pageSize;
   sqluint32 extSize;
   sqluint32 prefetchSize;
   sqluint32 nContainers;
   sqluint32 tbsState;
   sqluint64 lifeLSN;
   sqluint32 flags2;
   char minimumRecTime[SQL_STAMP_STRLEN+1];
   char pad1[1];
   sqluint32 StateChngObj;
   sqluint32 StateChngID;
   sqluint32 nQuiescers;
   struct SQLB_QUIESCER_DATA quiescer[SQLB_MAX_QUIESCERS];
   unsigned char   fsCaching;
   char reserved[31];
};

SQL_STRUCTURE SQLB_QUIESCER_DATA
{
   sqluint32 quiesceId;
   sqluint32 quiesceObject;
};
```

SQLB_QUIESCER_DATA 데이터 구조 매개변수

pad 예약됨. 구조 맞추기에 사용되며 사용자 데이터로 채우면 안됩니다.

pad1 예약됨. 구조 맞추기에 사용되며 사용자 데이터로 채우면 안됩니다.

quiesceId
입력. Quiesce 상태의 오브젝트가 작성된 테이블 스페이스의 ID

quiesceObject

입력. Quiesce 상태 오브젝트의 오브젝트 ID

COBOL 구조

```
* File: sqlutbsp.cbl
01 SQLB-TBSPQRY-DATA.
    05 SQL-TBSPQVER           PIC X(8).
    05 SQL-ID                 PIC 9(9) COMP-5.
    05 SQL-NAME-LEN           PIC 9(9) COMP-5.
    05 SQL-NAME               PIC X(128).
    05 SQL-TOTAL-PAGES        PIC 9(9) COMP-5.
    05 SQL-USEABLE-PAGES      PIC 9(9) COMP-5.
    05 SQL-FLAGS              PIC 9(9) COMP-5.
    05 SQL-PAGE-SIZE          PIC 9(9) COMP-5.
    05 SQL-EXT-SIZE           PIC 9(9) COMP-5.
    05 SQL-PREFETCH-SIZE      PIC 9(9) COMP-5.
    05 SQL-N-CONTAINERS       PIC 9(9) COMP-5.
    05 SQL-TBS-STATE          PIC 9(9) COMP-5.
    05 SQL-LIFE-LSN           PIC 9(18) COMP-5.
    05 SQL-FLAGS2             PIC 9(9) COMP-5.
    05 SQL-MINIMUM-REC-TIME   PIC X(26).
    05 FILLER                 PIC X.
    05 SQL-PAD1               PIC X(1).
    05 SQL-STATE-CHNG-OBJ     PIC 9(9) COMP-5.
    05 SQL-STATE-CHNG-ID      PIC 9(9) COMP-5.
    05 SQL-N-QUIESCERS        PIC 9(9) COMP-5.
    05 SQL-QUIESCER OCCURS 5 TIMES.
        10 SQL-QUIESCE-ID     PIC 9(9) COMP-5.
        10 SQL-QUIESCE-OBJECT PIC 9(9) COMP-5.
    05 SQL-FSCACHING          PIC X(1).
    05 SQL-RESERVED           PIC X(31).
*
```

제 175 장 SQLCA

SQL 통신 영역(SQLCA) 구조는 데이터베이스 관리 프로그램에서 응용프로그램의 오류 정보를 리턴하는 데 사용됩니다. 이 구조는 모든 API 호출 및 SQL문이 발행된 후에 갱신됩니다.

언어 구문

C 구조

```
/* File: sqlca.h */
/* Structure: SQLCA */
/* ... */
SQL_STRUCTURE  sqlca
{
  _SQLOLDCHAR    sqlcaid[8];
  sqlint32       sqlcabc;
  #ifdef DB2_SQL92E
  sqlint32       sqlcade;
  #else
  sqlint32       sqlcode;
  #endif
  short          sqlerrml;
  _SQLOLDCHAR    sqlerrmc[70];
  _SQLOLDCHAR    sqlerrp[8];
  sqlint32       sqlerrd[6];
  _SQLOLDCHAR    sqlwarn[11];
  #ifdef DB2_SQL92E
  _SQLOLDCHAR    sqlstat[5];
  #else
  _SQLOLDCHAR    sqlstate[5];
  #endif
};
/* ... */
```

COBOL 구조

```
* File: sqlca.cbl
01 SQLCA SYNC.
    05 SQLCAID PIC X(8) VALUE "SQLCA   ".
    05 SQLCABC PIC S9(9) COMP-5 VALUE 136.
    05 SQLCODE PIC S9(9) COMP-5.
    05 SQLERRM.
    05 SQLERRP PIC X(8).
    05 SQLERRD OCCURS 6 TIMES PIC S9(9) COMP-5.
    05 SQLWARN.
        10 SQLWARN0 PIC X.
        10 SQLWARN1 PIC X.
        10 SQLWARN2 PIC X.
        10 SQLWARN3 PIC X.
        10 SQLWARN4 PIC X.
        10 SQLWARN5 PIC X.
        10 SQLWARN6 PIC X.
```

```
           10 SQLWARN7 PIC X.
           10 SQLWARN8 PIC X.
           10 SQLWARN9 PIC X.
           10 SQLWARNA PIC X.
        05 SQLSTATE PIC X(5).
*
```

제 176 장 sqlchar

이 구조를 사용하여 가변 길이 데이터를 데이터베이스 관리 프로그램에 전달합니다.

표 36. SQLCHAR 구조의 필드

필드 이름	데이터 유형	설명
LENGTH	SMALLINT	DATA로 지시되는 문자열 길이
DATA	CHAR(n)	길이 LENGTH의 문자 배열.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqlchar
{
        short length;
        _SQLOLDCHAR data[1];
};
```

COBOL 구조

어떤 헤더 파일에도 정의되지 않습니다. 다음은 COBOL에 구조를 정의하는 방법을 보여주는 예입니다.

```
* Replace maxlen with the appropriate value:
01 SQLCHAR.
49 SQLCHAR-LEN  PIC S9(4) COMP-5.
49 SQLCHAR-DATA PIC X(maxlen).
```

제 177 장 SQLDA

SQL 디스크립터 영역(SQLDA) 구조는 SQL DESCRIBE 문 실행에 필요한 변수의 콜렉션입니다. SQLDA 변수는 PREPARE, OPEN, FETCH, EXECUTE, CALL 문과 같이 사용할 수 있는 옵션입니다.

SQLDA는 동적 SQL과 통신하여, DESCRIBE문에 사용되고, 호스트 변수 주소로 수정된 후 FETCH 문에서 재사용될 수 있습니다.

SQLDA는 모든 언어를 지원하지만 사전 정의된 선언은 C, REXX, FORTRAN 및 COBOL에 대해서만 제공됩니다.

SQLDA에 있는 정보 의미는 사용 방식에 따라 다릅니다. PREPARE 및 DESCRIBE에서 SQLDA는 준비된 명령문에 대한 정보를 응용프로그램에 제공합니다. OPEN, EXECUTE, FETCH, CALL에서 SQLDA는 호스트 변수를 설명합니다.

언어 구문

C 구조

```
/* File: sqlda.h */
/* Structure: SQLDA */
/* ... */
SQL_STRUCTURE  sqlda
{
  _SQLOLDCHAR    sqldaid[8];
  long           sqldabc;
  short          sqln;
  short          sqld;
  struct sqlvar  sqlvar[1];
};
/* ... */
```

```
/* File: sqlda.h */
/* Structure: SQLVAR */
/* ... */
SQL_STRUCTURE  sqlvar
{
  short          sqltype;
  short          sqllen;
  _SQLOLDCHAR   *SQL_POINTER sqldata;
  short         *SQL_POINTER sqlind;
  struct sqlname sqlname;
};
/* ... */
```

```
/* File: sqlda.h */
/* Structure: SQLNAME */
/* ... */
SQL_STRUCTURE  sqlname
{
```

```
  short          length;
  _SQLOLDCHAR    data[30];
};
/* ... */

/* File: sqlda.h */
/* Structure: SQLVAR2 */
/* ... */
SQL_STRUCTURE  sqlvar2
{
  union sql8bytelen len;
  char *SQL_POINTER sqldatalen;
  struct sqldistinct_type sqldatatype_name;
};
/* ... */

/* File: sqlda.h */
/* Structure: SQL8BYTELEN */
/* ... */
union sql8bytelen
{
  long        reserve1[2];
  long        sqllonglen;
};
/* ... */

/* File: sqlda.h */
/* Structure: SQLDISTINCT-TYPE */
/* ... */
SQL_STRUCTURE  sqldistinct_type
{
  short          length;
  char           data[27];
  char           reserved1[3];
};
/* ... */
```

COBOL 구조

```
* File: sqlda.cbl
01 SQLDA SYNC.
    05 SQLDAID PIC X(8) VALUE "SQLDA  ".
    05 SQLDABC PIC S9(9) COMP-5.
    05 SQLN PIC S9(4) COMP-5.
    05 SQLD PIC S9(4) COMP-5.
    05 SQLVAR-ENTRIES OCCURS 0 TO 1489 TIMES
        10 SQLVAR.
        10 SQLVAR2 REDEFINES SQLVAR.
*
```

제 178 장 sqldcol

이 구조를 사용하여 변수 컬럼 정보를 db2Export, db2Import, db2Load API에 전달합니다.

표 37. SQLDCOL 구조의 필드

필드 이름	데이터 유형	설명
DCOLMETH	SMALLINT	데이터 파일에서 컬럼 선택에 사용되는 메소드를 나타내는 문자.
DCOLNUM	SMALLINT	DCOLNAME 배열에 지정된 컬럼 수.
DCOLNAME	Array	DCOLNUM sqldcoln 구조 배열.

DCOLMETH의 유효한 값(sqlutil에 정의)은 다음과 같습니다.

SQL_METH_N

이름. 임포트 또는 로드 시에 이 구조를 통해 제공된 컬럼 이름을 사용하여 외부 파일에서 임포트하거나 로드할 데이터를 식별하십시오. 이 컬럼 이름의 대소문자는 시스템 카탈로그의 해당 이름의 대소문자와 일치해야 합니다. 익스포트 시에 이 구조를 통해 제공된 컬럼 이름을 출력 파일의 컬럼 이름으로 사용하십시오.

dcolname 배열의 각 요소에 대한 dcolnptr 포인터는 문자 배열 및 임포트 또는 로드하는 컬럼 이름을 구성하는 길이 dcolnlen 바이트를 지시합니다. dcolnum 필드는 양수여야 하며 dcolname 배열의 요소 수를 나타냅니다.

이 메소드는 외부 파일에 컬럼 이름이 없는 경우에는 유효하지 않습니다(예: DEL 또는 ASC 형식 파일).

SQL_METH_P

위치. 임포트 또는 로드 시에 이 구조를 통해 제공된 시작 컬럼 위치를 사용하여 외부 파일에서 임포트하거나 로드할 데이터를 식별하십시오. 데이터 익스포트 시에는 이 메소드가 유효하지 않습니다.

dcolname 배열의 각 요소에 대한 dcolnptr 포인터는 무시되지만 dcolnlen 필드에는 외부 파일의 컬럼 위치가 포함됩니다. dcolnum 필드는 양수여야 하며 dcolname 배열의 요소 수를 나타냅니다.

가장 낮은 유효한 컬럼 위치 값은 1(첫 번째 컬럼을 표시)이며 가장 높은 유효한 값은 외부 파일 유형에 따라 다릅니다. 위치 선택은 ASC 파일 임포트 시에는 유효하지 않습니다.

SQL_METH_L

위치. 임포트 또는 로드 시에 이 구조를 통해 제공된 시작 및 종료 컬럼 위치를 사용하여 외부 파일에서 임포트하거나 로드할 데이터를 식별하십시오. 데이터 익스포트 시에는 이 메소드가 유효하지 않습니다.

dcolname 배열의 첫 번째 요소에 대한 dcolnptr 필드는 sqlloctab 구조를 지시하며 이는 sqllocpair 구조 배열로 구성됩니다. 이 배열의 요소 수는 반드시 양수여야 하는 sqldcol 구조의 dcolnum 필드로 판별합니다. 배열의 각 요소는 컬럼의 시작 및 종료 위치를 나타내는 2바이트 정수 쌍입니다. 각 위치 쌍의 첫 번째 요소는 컬럼이 시작되는 파일 내의 바이트이며 두 번째 요소는 컬럼이 종료되는 위치의 바이트입니다. 파일의 행에서 첫 번째 바이트 위치가 바이트 위치 1이 됩니다. 컬럼은 오버랩할 수 있습니다.

SQL_METH_D

디폴트. DEL 및 IXF 파일 임포트나 로드 시에는 파일의 첫 번째 컬럼이 테이블의 첫 번째 컬럼으로 로드되거나 임포트되며 이렇게 순서대로 진행됩니다. 익스포트 시에 디폴트 이름은 외부 파일의 컬럼에 사용됩니다.

sqldcol 구조의 dcolnum 및 dcolname 필드는 모두 무시되며 외부 파일의 컬럼은 기본 순서대로 사용됩니다.

외부 파일의 컬럼은 두 번 이상 배열에서 사용할 수 있습니다. 외부 파일의 모든 컬럼을 사용하지 않아도 됩니다.

표 38. SQLDCOLN 구조의 필드

필드 이름	데이터 유형	설명
DCOLNLEN	SMALLINT	DCOLNPTR로 지시된 데이터 길이
DCOLNPTR	Pointer	DCOLMETH로 판별되는 데이터 요소의 포인터

주: DCOLNLEN 및 DCOLNPTR 필드는 지정된 각 컬럼에 대해 반복됩니다.

표 39. SQLLOCTAB 구조의 필드

필드 이름	데이터 유형	설명
LOCPAIR	Array	sqllocpair 구조의 배열.

표 40. SQLLOCPAIR 구조의 필드

필드 이름	데이터 유형	설명
BEGIN_LOC	SMALLINT	외부 파일의 컬럼 데이터 시작 위치.
END_LOC	SMALLINT	외부 파일의 컬럼 데이터 종료 위치.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqldcol
{
   short dcolmeth;
   short dcolnum;
   struct sqldcoln dcolname[1];
};

SQL_STRUCTURE sqldcoln
{
   short dcolnlen;
   char *dcolnptr;
};

SQL_STRUCTURE sqlloctab
{
   struct sqllocpair locpair[1];
};

SQL_STRUCTURE sqllocpair
{
   short begin_loc;
   short end_loc;
};
```

COBOL 구조

```
* File: sqlutil.cbl
01 SQL-DCOLDATA.
    05 SQL-DCOLMETH           PIC S9(4) COMP-5.
    05 SQL-DCOLNUM            PIC S9(4) COMP-5.
    05 SQLDCOLN OCCURS 0 TO 255 TIMES DEPENDING ON SQL-DCOLNUM.
        10 SQL-DCOLNLEN       PIC S9(4) COMP-5.
        10 FILLER             PIC X(2).
        10 SQL-DCOLN-PTR      USAGE IS POINTER.
*


* File: sqlutil.cbl
01 SQL-LOCTAB.
    05 SQL-LOC-PAIR OCCURS 1 TIMES.
        10 SQL-BEGIN-LOC      PIC S9(4) COMP-5.
        10 SQL-END-LOC        PIC S9(4) COMP-5.
*
```

제 179 장 sqle_addn_options

이 구조는 sqleaddn API에 정보를 전달하는 데 사용됩니다.

표 41. SQLE-ADDN-OPTIONS 구조의 필드

필드 이름	데이터 유형	설명
SQLADDID	CHAR	SQLE_ADDOPTID_V51로 설정되어야 하는 "아이캐처" 값.
TBLSPACE_TYPE	sqluint32	추가 중인 노드에 사용되는 시스템 임시 테이블 스페이스 정의 유형을 지정합니다. 값은 아래를 참조하십시오. 참고: 자동 스토리지를 사용하도록 정의된 시스템 임시 테이블 스페이스에 대해서는 이 옵션이 무시됩니다(즉, CREATE TABLESPACE 문의 MANAGED BY AUTOMATIC STORAGE절로 작성되었거나 MANAGED BY절이 전혀 지정되지 않은 시스템 임시 테이블 스페이스). 이 테이블 스페이스의 경우 다른 파티션에 정의된 것처럼 컨테이너 작성을 지연하거나 컨테이너 세트를 작성하도록 선택할 수 없습니다. 컨테이너는 데이터베이스에 연결된 스토리지 경로를 기반으로 하는 데이터베이스 관리 프로그램에 의해 자동으로 할당됩니다.
TBLSPACE_NODE	SQL_PDB_NODE_TYPE	시스템 임시 테이블 스페이스 정의가 확보해야 하는 노드 번호를 지정합니다. 노드 번호는 db2nodes.cfg 파일에 있어야 하며 tblspace_type 필드를 SQLE_TABLESPACES_LIKE_NODE로 설정한 경우에만 사용됩니다.

TBLSPACE_TYPE의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

SQLE_TABLESPACES_NONE
시스템 임시 테이블 스페이스를 작성하지 않습니다.

SQLE_TABLESPACES_LIKE_NODE
시스템 임시 테이블 스페이스 컨테이너는 지정한 노드의 컨테이너와 동일해야 합니다.

SQLE_TABLESPACES_LIKE_CATALOG
시스템 임시 테이블 스페이스 컨테이너는 각 데이터베이스의 카탈로그 노드의 컨테이너와 동일해야 합니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqle_addn_options
{
        char sqladdid[8];
        sqluint32 tblspace_type;
        SQL_PDB_NODE_TYPE tblspace_node;
};
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQLE-ADDN-OPTIONS.
    05 SQLADDID              PIC X(8).
    05 SQL-TBLSPACE-TYPE     PIC 9(9) COMP-5.
    05 SQL-TBLSPACE-NODE     PIC S9(4) COMP-5.
    05 FILLER                PIC X(2).
*
```

제 180 장 sqle_client_info

이 구조는 sqleseti 및 sqleqryi API에 정보를 전달하는 데 사용됩니다. 이 구조는 다음을 지정합니다.

- 설정 또는 쿼리 중인 정보 유형
- 설정 또는 쿼리 중인 데이터 길이
- 다음의 포인터
  - 설정 중인 데이터가 포함될 영역
  - 쿼리 중인 데이터가 포함된 충분한 길이의 영역

응용프로그램은 다음 유형의 정보를 지정할 수 있습니다.

- 설정 또는 쿼리 중인 클라이언트 사용자 ID. 최대 255자를 설정할 수 있으며 서버는 이를 일부 플랫폼 고유의 값으로 절단하기도 합니다.

  주: 이 사용자 ID는 식별을 위해서만 사용되며 권한 부여에는 전혀 사용되지 않습니다.
- 설정 또는 쿼리 중인 클라이언트 워크스테이션 이름. 최대 255자를 설정할 수 있으며 서버는 이를 일부 플랫폼 고유의 값으로 절단하기도 합니다.
- 설정 또는 쿼리 중인 클라이언트 응용프로그램 이름. 최대 255자를 설정할 수 있으며 서버는 이를 일부 플랫폼 고유의 값으로 절단하기도 합니다.
- 설정 또는 쿼리 중인 클라이언트 현재 패키지. 최대 255자를 설정할 수 있으며 서버는 이를 일부 플랫폼 고유의 값으로 절단하기도 합니다.
- 설정 또는 쿼리 중인 클라이언트 프로그램 ID. 최대 80자를 설정할 수 있으며 서버는 이를 일부 플랫폼 고유의 값으로 절단하기도 합니다.
- 설정 또는 쿼리 중인 클라이언트 어카운팅 문자열. 최대 200자를 설정할 수 있으며 서버는 이를 일부 플랫폼 고유의 값으로 절단하기도 합니다.

  주: 정보는 sqlesact API를 사용하여 설정할 수 있습니다. 그러나 sqlesact는 이미 연결되어 있으면 어카운팅 문자열의 변경을 허용하지 않는 반면 sqleseti는 이미 연결되어 있는 경우 뿐만 아니라 이후에도 어카운팅 정보를 변경할 수 있도록 허용합니다.

표 42. SQLE-CLIENT-INFO 구조의 필드

필드 이름	데이터 유형	설명
TYPE	sqlint32	설정 유형

표 42. SQLE-CLIENT-INFO 구조의 필드 (계속)

필드 이름	데이터 유형	설명
LENGTH	sqlint32	값 길이. sqleseti 호출에서 길이는 0에서 유형에 대해 정의된 최대 길이 사이로 지정할 수 있습니다. 길이가 0인 경우에는 널(NULL) 값을 나타냅니다. sqleqryi 호출에서 길이가 리턴되지만 pValue로 지시된 영역은 유형의 최대 길이를 포함할 수 있을 만큼 커야 합니다. 길이가 0인 경우에는 널(NULL) 값을 나타냅니다.
PVALUE	Pointer	지정한 값을 포함하는 응용프로그램에서 할당한 버퍼의 포인터. 이 값의 데이터 유형은 유형 필드에 따라 다릅니다.

SQLE-CLIENT-INFO TYPE 요소에 대한 유효한 항목 및 각 항목에 대한 관련 설명이 아래 나열됩니다.

표 43. 연결 설정

유형	데이터 유형	설명
SQLE_CLIENT_INFO_USERID	CHAR(255)	클라이언트의 사용자 ID. 일부 서버는 이 값을 절단하기도 합니다. 예를 들어, z/OS용 DB2 서버는 최대 16자를 지원합니다. 이 사용자 ID는 식별을 위해서만 사용되며 권한 부여에는 전혀 사용되지 않습니다.
SQLE_CLIENT_INFO_WRKSTNNAME	CHAR(255)	클라이언트의 워크스테이션 이름. 일부 서버는 이 값을 절단하기도 합니다. 예를 들어, z/OS용 DB2 서버는 최대 18자를 지원합니다.
SQLE_CLIENT_INFO_APPLNAME	CHAR(255)	클라이언트의 응용프로그램 이름. 일부 서버는 이 값을 절단하기도 합니다. 예를 들어, z/OS용 DB2 서버는 최대 32자를 지원합니다.
SQLE_CLIENT_INFO_PROGRAMID	CHAR(80)	클라이언트의 프로그램 ID. 이 요소를 설정하면 z/OS용 DB2 Universal Database 버전 8은 이 ID를 동적 SQL문 캐시에 삽입된 임의의 명령문과 연관시킵니다. 이 요소는 z/OS용 DB2 UDB 버전 8에 액세스한 응용프로그램에 대해서만 지원됩니다.
SQLE_CLIENT_INFO_ACCTSTR	CHAR(200)	클라이언트의 어카운팅 문자열. 일부 서버는 이 값을 절단하기도 합니다. 예를 들어, z/OS용 DB2 서버는 최대 200자를 지원합니다.
SQLE_CLIENT_INFO_AUTOCOMMIT	CHAR(1)	클라이언트의 자동 커미트 설정. SQLE_CLIENT_AUTOCOMMIT_ON 또는 SQLE_CLIENT_AUTOCOMMIT_OFF로 설정할 수 있습니다.

주: 이 필드 이름은 C 프로그래밍 언어용으로 정의됩니다. 동일한 시맨틱을 사용하는 FORTRAN 및 COBOL에도 유사한 이름이 있습니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqle_client_info
{
        unsigned short         type;
        unsigned short         length;
        char *pValue;
};
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQLE-CLIENT-INFO.
    05 SQLE-CLIENT-INFO-ITEM OCCURS 4 TIMES.
        10 SQLE-CLIENT-INFO-TYPE   PIC S9(4) COMP-5.
        10 SQLE-CLIENT-INFO-LENGTH PIC S9(4) COMP-5.
        10 SQLE-CLIENT-INFO-VALUE  USAGE IS POINTER.
*
```

제 181 장 sqle_conn_setting

이 구조를 사용하여 sqleqryc 및 sqlesetc API의 연결 설정 유형 및 값을 지정합니다.

표 44. SQLE-CONN-SETTING 구조의 필드

필드 이름	데이터 유형	설명
TYPE VALUE	SMALLINT SMALLINT	설정 유형. 설정 값.

SQLE-CONN-SETTING TYPE 요소에 대한 유효한 항목 및 각 항목에 대한 관련 설명이 아래 나열됩니다(sqlenv 및 sql에 정의)

표 45. 연결 설정

유형	값	설명
SQL_CONNECT_TYPE	SQL_CONNECT_1 SQL_CONNECT_2	유형 1 CONNECT에서는 리모트 작업 단위(RUOW)의 규칙이라고도 하는 이전 릴리스에 대한 작업 단위(UOW) 시맨틱당 한 개의 데이터베이스를 강제로 사용합니다. 유형 2 CONNECT는 DUOW의 작업 단위(UOW) 시맨틱당 여러 개의 데이터베이스를 지원합니다.
SQL_RULES	SQL_RULES_DB2 SQL_RULES_STD	SQL CONNECT문으로 현재 연결을 설정된(유휴) 연결로 전환할 수 있습니다. SQL CONNECT 문을 사용하는 새 연결 작성만 허용합니다. SQL SET CONNECTION 문을 사용하여 현재 연결을 유휴 연결로 전환해야 합니다.
SQL_DISCONNECT	SQL_DISCONNECT_EXPL SQL_DISCONNECT_COND SQL_DISCONNECT_AUTO	커미트 시에 SQL RELEASE 문을 사용하여 명시적으로 릴리스용으로 표시한 해당 연결을 제거합니다. 커미트 시에 열려 있는 WITH HOLD 커서가 없는 연결 및 SQL RELEASE 문으로 릴리스용으로 표시된 연결만 중단합니다. 커미트 시에 모든 연결을 중단합니다.
SQL_SYNCPOINT	SQL_SYNC_TWOPHASE SQL_SYNC_ONEPHASE SQL_SYNC_NONE	트랜잭션 관리 프로그램(TM)이 이 프로토콜을 지원하는 데이터베이스 사이의 2단계 커미트를 조정하도록 합니다. 1단계 커미트를 사용하여 여러 데이터베이스 트랜잭션에서 각 데이터베이스로 수행하는 작업을 커미트합니다. 단일 갱신자와 다중 읽기 동작을 실행합니다. 1단계 커미트를 사용하여 완료된 작업을 커미트하지만 단일 갱신자와 다중 읽기 동작을 강제로 수행하지는 않습니다.

표 45. 연결 설정 (계속)

유형	값	설명
SQL_DEFERRED_PREPARE	SQL_DEFERRED_PREPARE_NO SQL_DEFERRED_PREPARE_YES SQL_DEFERRED_PREPARE_ALL	PREPARE문은 발행된 시점에서 실행됩니다. PREPARE문 실행은 해당 OPEN, DESCRIBE 또는 EXECUTE 문이 발행될 때까지 지연됩니다. SQLDA가 즉시 리턴되어야 하는 INTO 절이 사용된 경우에는 PREPARE문은 지연되지 않습니다. 그러나 매개변수 표시문자를 사용하지 않는 커서에 대해 PREPARE INTO문이 발행된 경우에는 PREPARE를 실행할 때 커서를 pre-OPENing하여 처리가 최적화됩니다. 매개변수 표시문자가 포함된 PREPARE INTO가 지연되는 점만 제외하면 YES와 동일합니다. PREPARE INTO 문에 매개변수 표시문자가 포함되지 않은 경우에는 커서의 pre-OPENing이 계속 수행됩니다. PREPARE문이 INTO 절을 사용하여 SQLDA를 리턴하는 경우에는 OPEN, DESCRIBE 또는 EXECUTE 문이 발행되고 리턴될 때까지 응용프로그램은 이 SQLDA 컨텐츠를 참조하면 안됩니다.
SQL_CONNECT_NODE	0 - 999 또는 키워드 SQL_CONN_CATALOG_NODE.	연결이 작성되는 노드를 지정합니다. 환경변수 DB2NODE 값을 겹쳐씁니다. 예를 들어 노드 1, 2, 3을 정의한 경우 클라이언트는 이 노드 중 하나에 액세스할 수 있으면 됩니다. 데이터베이스를 포함하는 노드 1만 카탈로그되고 이 매개변수가 3으로 설정된 경우 노드 1의 초기 연결 후에 다음 연결 시도는 노드 3으로 연결됩니다.
SQL_ATTACH_NODE	0 - 999 사이.	접속하려는 노드를 지정합니다. 환경 변수 DB2NODE 값을 겹쳐씁니다. 예를 들어 노드 1, 2, 3을 정의한 경우 클라이언트는 이 노드 중 하나에 액세스할 수 있으면 됩니다. 데이터베이스를 포함하는 노드 1만 카탈로그되고 이 매개변수가 3으로 설정된 경우 노드 1의 초기 접속 후에 다음 접속 시도는 노드 3으로 접속됩니다.

주: 이 필드 이름은 C 프로그래밍 언어용으로 정의됩니다. 동일한 시맨틱을 사용하는 FORTRAN 및 COBOL에도 유사한 이름이 있습니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqle_conn_setting
{
        unsigned short          type;
        unsigned short          value;
};
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQLE-CONN-SETTING.
    05 SQLE-CONN-SETTING-ITEM OCCURS 7 TIMES.
        10 SQLE-CONN-TYPE  PIC S9(4) COMP-5.
        10 SQLE-CONN-VALUE PIC S9(4) COMP-5.
*
```

제 182 장 sqle_node_local

이 구조를 사용하여 sqlectnd API의 로컬 노드를 카탈로그합니다.

표 46. SQLE-NODE-LOCAL 구조의 필드

필드 이름	데이터 유형	설명
INSTANCE_NAME	CHAR(8)	인스턴스 이름

주: 이 구조에서 전달되는 문자 필드는 널(NULL)로 종료되거나 필드 길이까지 채워진 공백이어야 합니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqle_node_local
{
        char instance_name[SQL_INSTNAME_SZ+1];
};
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQL-NODE-LOCAL.
    05 SQL-INSTANCE-NAME      PIC X(8).
    05 FILLER                 PIC X.
*
```

제 183 장 sqle_node_npipe

이 구조는 sqlectnd API의 named pipe 노드 카탈로그에 사용됩니다.

표 47. SQLE-NODE-NPIPE 구조의 필드

필드 이름	데이터 유형	설명
COMPUTERNAME	CHAR(15)	컴퓨터 이름
INSTANCE_NAME	CHAR(8)	인스턴스 이름

주: 이 구조에서 전달되는 문자 필드는 널(NULL)로 종료되거나 필드 길이까지 채워진 공백이어야 합니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqle_node_npipe
{
        char computername[SQL_COMPUTERNAME_SZ+1];
        char instance_name[SQL_INSTNAME_SZ+1];
};
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQL-NODE-NPIPE.
    05 COMPUTERNAME           PIC X(15).
    05 FILLER                 PIC X.
    05 INSTANCE-NAME          PIC X(8).
    05 FILLER                 PIC X.
*
```

제 184 장 sqle_node_struct

이 구조는 sqlectnd API에 대해 노드를 카탈로그하는 데 사용됩니다.

주: NetBIOS는 더 이상 지원되지 않습니다. 해당 API인 APPC, APPN 및 CPI-C가 포함된 SNA가 더 이상 지원되지 않습니다. 이 프로토콜을 사용하는 경우에는 TCP/IP와 같이 지원되는 프로토콜을 사용하여 노드와 데이터베이스를 재카탈로그해야 합니다. 이 프로토콜 참조는 무시해야 합니다.

표 48. SQLE-NODE-STRUCT 구조의 필드

필드 이름	데이터 유형	설명
STRUCT_ID	SMALLINT	구조 ID
CODEPAGE	SMALLINT	주석의 코드 페이지.
COMMENT	CHAR(30)	노드의 선택적 설명
NODENAME	CHAR(8)	데이터베이스가 위치한 노드의 로컬 이름
PROTOCOL	CHAR(1)	통신 프로토콜 유형

주: 이 구조에서 전달되는 문자 필드는 널(NULL)로 종료되거나 필드 길이까지 채워진 공백이어야 합니다.

PROTOCOL의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

- SQL_PROTOCOL_APPC
- SQL_PROTOCOL_APPN
- SQL_PROTOCOL_CPIC
- SQL_PROTOCOL_LOCAL
- SQL_PROTOCOL_NETB
- SQL_PROTOCOL_NPIPE
- SQL_PROTOCOL_SOCKS
- SQL_PROTOCOL_TCPIP

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqle_node_struct
{
        unsigned short struct_id;
        unsigned short codepage;
        _SQLOLDCHAR comment[SQL_CMT_SZ + 1];
        _SQLOLDCHAR nodename[SQL_NNAME_SZ + 1];
        unsigned char  protocol;
};
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQL-NODE-STRUCT.
    05 STRUCT-ID             PIC 9(4) COMP-5.
    05 CODEPAGE              PIC 9(4) COMP-5.
    05 COMMENT               PIC X(30).
    05 FILLER                PIC X.
    05 NODENAME              PIC X(8).
    05 FILLER                PIC X.
    05 PROTOCOL              PIC X.
    05 FILLER                PIC X(1).
*
```

제 185 장 sqle_node_tcpip

이 구조는 sqlectnd API에 대해 TCP/IP 노드를 카탈로그하는 데 사용됩니다.

주: TCP/IP, TCP/IPv4 또는 TCP/IPv6 노드를 카탈로그하려면 sqlectnd API를 호출하기 전에 노드 디렉토리 구조에서 PROTOCOL 유형을 SQLE-NODE-STRUCT 구조에서 각각 SQL_PROTOCOL_TCPIP, SQL_PROTOCOL_TCPIP4 또는 SQL_PROTOCOL_TCPIP6으로 설정하십시오. TCP/IP 또는 TCP/IPv4 SOCKS 노드를 카탈로그하려면 sqlectnd API를 호출하기 전에 노드 디렉토리 구조의 PROTOCOL 유형을 SQLE-NODE-STRUCT 구조에서 각각 SQL_PROTOCOL_SOCKS 또는 SQL_PROTOCOL_SOCKS4로 설정하십시오. SOCKS는 IPv6에서 지원되지 않습니다. 예를 들어, IPv6 주소의 SQL_PROTOCOL_SOCKS는 지원되지 않습니다.

표 49. SQLE-NODE-TCPIP 구조의 필드

필드 이름	데이터 유형	설명
HOSTNAME	CHAR(255)	DB2 서버가 있는 호스트 이름 또는 IP 주소. 승인된 IP 주소 유형은 선택한 프로토콜에 따라 다릅니다.
SERVICE_NAME	CHAR(14)	DB2 서버 인스턴스의 TCP/IP 서비스 이름 또는 관련 포트 번호

주: 이 구조에서 전달되는 문자 필드는 널(NULL)로 종료되거나 필드 길이까지 채워진 공백이어야 합니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqle_node_tcpip
{
        _SQLOLDCHAR hostname[SQL_HOSTNAME_SZ+1];
        _SQLOLDCHAR service_name[SQL_SERVICE_NAME_SZ+1];
};
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQL-NODE-TCPIP.
    05 HOSTNAME              PIC X(255).
    05 FILLER                PIC X.
    05 SERVICE-NAME          PIC X(14).
    05 FILLER                PIC X.
*
```

제 186 장 sqledbdesc

sqlecrea API 호출 중에 데이터베이스 설명 블록(SQLEDBDESC) 구조를 사용하여 데이터베이스 속성에 대한 영구적인 값을 지정할 수 있습니다. 이 속성에는 데이터베이스 주석 조합 순서 및 테이블 스페이스 정의가 포함됩니다.

표 50. SQLEDBDESC 구조의 필드

필드 이름	데이터 유형	설명
SQLDBDID	CHAR(8)	구조 ID 및 스토리지 덤프의 ″아이캐처″. SQLE_DBDESC_2 값(sqlenv에 정의)으로 초기화해야 하는 8바이트의 문자열입니다. 이 필드의 컨텐츠는 버전 제어를 위해 유효성 확인됩니다.
SQLDBCCP	INTEGER	데이터베이스 주석의 코드 페이지. 이 값은 데이터베이스 관리 프로그램에서 더 이상 사용되지 않습니다.
SQLDBCSS	INTEGER	데이터베이스 조합 시퀀스 소스를 타나내는 값. 값은 아래를 참조하십시오. 참고: 데이터베이스의 조합 시퀀스가 IDENTITY(2진 조합 시퀀스를 구현)가 됨을 지정하도록 SQL_CS_NONE을 지정하십시오. SQL_CS_NONE이 디폴트입니다.
SQLDBUDC	CHAR(256)	SQLDBCSS가 SQL_CS_USER로 설정된 경우 이 필드에 *n*번째 바이트에는 데이터베이스 코드 페이지에서 기본 10진수 표현이 n인 코드 포인트의 정렬 가중치가 포함됩니다. SQLDBCSS가 SQL_CS_UNICODE로 설정된 경우 이 필드에는 언어에서 인식하는 또는 로케일 고유의 UCA 기반 조합 이름(길이가 최대 128 바이트인 NULL로 종료되는 문자열)이 포함됩니다. SQLDBCSS가 SQL_CS_USER 또는 SQL_CS_UNICODE와 동등하지 않은 경우 이 필드는 무시됩니다.
SQLDBCMT	CHAR(30)	데이터베이스의 주석.
SQLDBSGP	INTEGER	예약된 필드. 더 이상 사용되지 않습니다.

표 50. SQLEDBDESC 구조의 필드 (계속)

필드 이름	데이터 유형	설명
SQLDBNSG	SHORT	데이터베이스에 작성되는 파일 세그먼트 수를 나타내는 값. 이 필드의 최소 값은 1이고 최대 값은 256입니다. 값으로 -1을 제공하면 이 필드는 디폴트로 1을 사용합니다. 참고: SQLDBNSG를 영(0)으로 설정하면 버전 1 호환성을 위한 디폴트가 작성됩니다.
SQLTSEXT	INTEGER	데이터베이스의 각 테이블 스페이스에 대한 디폴트 Extent 크기를 나타내는 4KB 페이지 단위의 값. 이 필드의 최소 값은 2이고 최대 값은 256입니다. 값으로 -1을 제공하면 이 필드는 디폴트로 32를 사용합니다.
SQLCATTS	Pointer	테이블 스페이스 설명 SQLETSDESC의 포인터로 카탈로그 테이블 스페이스를 정의합니다. 널(NULL)인 경우 SQLTSEXT 및 SQLDBNSG 값을 기반으로 하는 디폴트 카탈로그 테이블 스페이스가 작성됩니다.
SQLUSRTS	Pointer	테이블 스페이스 설명 SQLETSDESC의 포인터로 사용자 테이블 스페이스를 정의합니다. 널(NULL)인 경우 SQLTSEXT 및 SQLDBNSG 값을 기반으로 하는 디폴트 사용자 테이블 스페이스가 작성됩니다.
SQLTMPTS	Pointer	테이블 스페이스 설명 SQLETSDESC의 포인터로 시스템 임시 테이블 스페이스를 정의합니다. 널(NULL)인 경우 SQLTSEXT 및 SQLDBNSG 값을 기반으로 하는 디폴트 시스템 임시 테이블 스페이스가 작성됩니다.

테이블 스페이스 설명 블록 구조(SQLETSDESC)를 사용하여 세 초기 테이블 스페이스 중 임의의 테이블 스페이스 속성을 지정합니다.

표 51. SQLETSDESC 구조의 필드

필드 이름	데이터 유형	설명
SQLTSDID	CHAR(8)	구조 ID 및 스토리지 덤프의 ″아이캐처″. SQLE_DBTSDESC_1 값(sqlenv에 정의)으로 초기화해야 하는 8바이트의 문자열입니다. 이 필드의 컨텐츠는 버전 제어를 위해 유효성 확인됩니다.
SQLEXTNT	INTEGER	테이블 스페이스 Extent 크기 (4KB 페이지). 값으로 -1을 제공하면 이 필드는 디폴트로 dft_extent_sz 구성 매개변수의 현재 값을 사용합니다.
SQLPRFTC	INTEGER	테이블 스페이스 프리페치 크기 (4KB 페이지). 값으로 -1을 제공하면 이 필드는 디폴트로 dft_prefetch_sz 구성 매개변수의 현재 값을 사용합니다.
SQLFSCACHING	UNSIGNED CHAR	파일 시스템 캐싱. 값으로 1을 제공하면 파일 시스템 캐싱이 현재 테이블 스페이스에 대해 해제됩니다. 값으로 0을 제공하면 파일 시스템 캐싱이 현재 테이블 스페이스에 대해 설정됩니다. 2를 지정하여 디폴트 설정을 표시하십시오. 이 경우 파일 시스템 캐싱은 AIX, Linux, Solaris 및 Windows에서 해제되며 AIX JFS, System z® 기반 Linux, SMS 임시 테이블 스페이스 파일을 위한 Solaris 비VxFS 및 SMS 대형 오브젝트(LOB) 또는 대형 파일은 제외됩니다. 파일 시스템 캐싱은 기타 모든 플랫폼에 대해 설정됩니다.
SQLPOVHD	DOUBLE	테이블 스페이스 입출력 오버헤드 (밀리초). 값으로 -1을 제공하면 이 필드는 추후 릴리스에 변경될 수도 있는 내부 데이터베이스 관리 프로그램 값(현재 24.1ms)을 디폴트로 사용합니다.
SQLTRFRT	DOUBLE	테이블 스페이스 입출력 전송률 (밀리초). 값으로 -1을 제공하면 이 필드는 추후 릴리스에 변경될 수도 있는 내부 데이터베이스 관리 프로그램 값(현재 0.9ms)을 디폴트로 사용합니다.

표 51. SQLETSDESC 구조의 필드 (계속)

필드 이름	데이터 유형	설명
SQLTSTYP	CHAR(1)	테이블 스페이스가 시스템 관리 또는 데이터베이스 관리인지를 나타냅니다. 값은 아래를 참조하십시오.
SQLCCNT	SMALLINT	테이블 스페이스에 지정되는 컨테이너 수. 사용되는 SQLCTYPE/SQLCSIZE/ SQLCLEN/SQLCONTR 값 수량을 나타냅니다.
CONTAINR	Array	sqlccnt SQLETSCDESC 구조 배열.

표 52. SQLETSCDESC 구조의 필드

필드 이름	데이터 유형	설명
SQLCTYPE	CHAR(1)	이 컨테이너 유형을 식별합니다. 값은 아래를 참조하십시오.
SQLCSIZE	INTEGER	SQLCONTR에서 식별된 컨테이너 *크기*(4KB 페이지). SQLTSTYP를 SQL_TBS_TYP_DMS로 설정한 경우에만 유효합니다.
SQLCLEN	SMALLINT	다음 SQLCONTR 값의 길이.
SQLCONTR	CHAR(256)	컨테이너 문자열.

SQLDBCSS의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

SQL_CS_SYSTEM

유니코드가 아닌 데이터베이스의 경우 이 옵션은 데이터베이스 지역을 기반으로 하는 조합 시퀀스가 포함된 디폴트 옵션입니다. 유니코드 데이터베이스의 경우 이 옵션은 IDENTITY 옵션과 동등합니다. NULL 포인터를 전달하는 경우 운영 체제의 조합 시퀀스(현재 로케일 코드 및 코드 페이지를 바탕으로)가 사용됩니다. 이는 SQL_CS_SYSTEM(0)과 동일한 SQLDBCSS를 지정하는 것과 동일합니다.

SQL_CS_USER

조합 시퀀스는 사용자가 입력한 256바이트 가중치 테이블로 지정합니다. 테이블의 각 가중치의 길이는 1바이트입니다.

SQL_CS_NONE

문자열이 바이트별로 비교되는 조합 시퀀스를 식별합니다. 유니코드 데이터베이스의 경우 디폴트입니다.

SQL_CS_COMPATABILITY

이전 버전의 조합 시퀀스를 사용합니다.

SQL_CS_SYSTEM_NLSCHAR

문자 유형에 대한 비교 루틴으로 NLS 버전을 사용하는 시스템의 조합 순서. 이 값은 태국어 TIS620-1 데이터베이스 작성 시에만 지정할 수 있습니다.

SQL_CS_USER_NLSCHAR

조합 시퀀스는 사용자가 입력한 256바이트 가중치 테이블로 지정합니다. 테이블의 각 가중치의 길이는 1바이트입니다. 이 값은 태국어 TIS620-1 데이터베이스 작성 시에만 지정할 수 있습니다.

SQL_CS_IDENTITY_16BIT

유니코드 컨소시엄 웹 사이트(www.unicode.org)에서 제공되는 Unicode Technical Report #26으로 지정하는 CESU-8(UTF-16의 호환성 인코딩 스킴 : 8비트) 조합 시퀀스. 이 값은 유니코드 데이터베이스 작성 시에만 지정할 수 있습니다.

SQL_CS_UCA400_NO

정규화 내재성이 '설정'된 Unicode Standard 버전 4.0.0을 기반으로 하는 UCA(Unicode Collation Algorithm) 조합 시퀀스. UCA에 대한 자세한 내용은 유니코드 컨소시엄 웹 사이트(www.unicode.org)에서 제공되는 Unicode Technical Standard #10에서 확인할 수 있습니다. 이 값은 유니코드 데이터베이스 작성 시에만 지정할 수 있습니다.

SQL_CS_UCA400_LSK

Unicode Standard 버전 4.0.0을 기반으로 하지만 슬로바키아 문자를 해당 순서로 정렬하는 UCA(Unicode Collation Algorithm) 조합 시퀀스. UCA에 대한 자세한 내용은 유니코드 컨소시엄 웹 사이트(www.unicode.org)에서 제공되는 Unicode Technical Standard #10에서 확인할 수 있습니다. 이 값은 유니코드 데이터베이스 작성 시에만 지정할 수 있습니다.

SQL_CS_UCA400_LTH

모든 타이 문자를 Royal 타이 사전 순서로 정렬하는 Unicode Standard 버전 4.0.0을 기반으로 하는 UCA(Unicode Collation Algorithm) 조합 시퀀스. UCA에 대한 자세한 내용은 유니코드 컨소시엄 웹 사이트(www.unicode.org)에서 제공되는 Unicode Technical Standard #10에서 확인할 수 있습니다. 이 값은 유니코드 데이터베이스 작성 시에만 지정할 수 있습니다.

SQL_CS_UNICODE

조합 순서는 유니코드 데이터베이스에 대해 언어를 기반으로 합니다. 특정 조합 이름이 SQLDBUDC 필드에 지정되며 0x00 바이트로 종료되어야 합니다. 조합 이름은 "유니코드 데이터에 대한 언어 인식 조합"에 정의된 모든 언어 인식 조합 또는 "UCA(Unicode Collation Algorithm) 기반 조합"에 식별된 로케일 인식 UCA 기반 조합을 모두 식별할 수 있습니다.

예를 들어, 코드 페이지 819의 미국 영어에 해당하는 조합을 사용하려면 SQLDBCSS를 SQL_CS_UNICODE로 설정하고 SQLDBUDC를 SYSTEM_819_US로 설정하십시오.

주: 9.5 이전 버전에 대해 CREATE DATABASE를 수행하는 경우 이 옵션을 사용할 수 없습니다. 디폴트로 이런 서버에서의 유니코드 데이터베이스는 SYSTEM 조합으로 작성됩니다.

SQLTSTYP의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

SQL_TBS_TYP_SMS
시스템 관리

SQL_TBS_TYP_DMS
데이터베이스 관리

SQLCTYPE의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

SQL_TBSC_TYP_DEV
디바이스. SQLTSTYP = SQL_TBS_TYP_DMS에서만 유효.

SQL_TBSC_TYP_FILE
파일. SQLTSTYP = SQL_TBS_TYP_DMS에서만 유효.

SQL_TBSC_TYP_PATH
경로(디렉토리). SQLTSTYP = SQL_TBS_TYP_SMS에서만 유효.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqledbdesc
{
        _SQLOLDCHAR sqldbdid[8];
        sqlint32 sqldbccp;
        sqlint32 sqldbcss;
        unsigned char  sqldbudc[SQL_CS_SZ];
        _SQLOLDCHAR sqldbcmt[SQL_CMT_SZ+1];
        _SQLOLDCHAR pad[1];
        sqluint32 sqldbsgp;
        short sqldbnsg;
        char pad2[2];
        sqlint32 sqltsext;
        struct SQLETSDESC *sqlcatts;
        struct SQLETSDESC *sqlusrts;
        struct SQLETSDESC *sqltmpts;
};

SQL_STRUCTURE SQLETSDESC
{
        char sqltsdid[8];
        sqlint32 sqlextnt;
        sqlint32 sqlprftc;
        double sqlpovhd;
        double sqltrfrt;
```

```
        char sqltstyp;
        unsigned char  sqlfscaching;
        short sqlccnt;
        struct SQLETSCDESC containr[1];
};

SQL_STRUCTURE SQLETSCDESC
{
        char sqlctype;
        char pad1[3];
        sqlint32 sqlcsize;
        short sqlclen;
        char sqlcontr[SQLB_MAX_CONTAIN_NAME_SZ];
        char pad2[2];
};
```

sqledbdesc 구조 매개변수

pad1 예약됨. 구조 맞추기에 사용되며 사용자 데이터로 채우면 안됩니다.

pad2 예약됨. 구조 맞추기에 사용되며 사용자 데이터로 채우면 안됩니다.

SQLETSCDESC 구조 매개변수

pad1 예약됨. 구조 맞추기에 사용되며 사용자 데이터로 채우면 안됩니다.

pad2 예약됨. 구조 맞추기에 사용되며 사용자 데이터로 채우면 안됩니다.

COBOL 구조

```
* File: sqlenv.cbl
01 SQLEDBDESC.
    05 SQLDBDID              PIC X(8).
    05 SQLDBCCP              PIC S9(9) COMP-5.
    05 SQLDBCSS              PIC S9(9) COMP-5.
    05 SQLDBUDC              PIC X(256).
    05 SQLDBCMT              PIC X(30).
    05 FILLER                PIC X.
    05 SQL-PAD               PIC X(1).
    05 SQLDBSGP              PIC 9(9) COMP-5.
    05 SQLDBNSG              PIC S9(4) COMP-5.
    05 SQL-PAD2              PIC X(2).
    05 SQLTSEXT              PIC S9(9) COMP-5.
    05 SQLCATTS              USAGE IS POINTER.
    05 SQLUSRTS              USAGE IS POINTER.
    05 SQLTMPTS              USAGE IS POINTER.
*


* File: sqletsd.cbl
01 SQLETSDESC.
    05 SQLTSDID              PIC X(8).
    05 SQLEXTNT              PIC S9(9) COMP-5.
    05 SQLPRFTC              PIC S9(9) COMP-5.
    05 SQLPOVHD              USAGE COMP-2.
    05 SQLTRFRT              USAGE COMP-2.
    05 SQLTSTYP              PIC X.
    05 SQL-PAD1              PIC X.
```

```
    05 SQLCCNT                PIC S9(4) COMP-5.
    05 SQL-CONTAINR OCCURS 001 TIMES.
        10 SQLCTYPE           PIC X.
        10 SQL-PAD1           PIC X(3).
        10 SQLCSIZE           PIC S9(9) COMP-5.
        10 SQLCLEN            PIC S9(4) COMP-5.
        10 SQLCONTR           PIC X(256).
        10 SQL-PAD2           PIC X(2).
*


* File: sqlenv.cbl
01 SQLETSCDESC.
    05 SQLCTYPE               PIC X.
    05 SQL-PAD1               PIC X(3).
    05 SQLCSIZE               PIC S9(9) COMP-5.
    05 SQLCLEN                PIC S9(4) COMP-5.
    05 SQLCONTR               PIC X(256).
    05 SQL-PAD2               PIC X(2).
*
```

제 187 장 sqledbdescext

sqlecrea API 호출 중에 확장된 데이터베이스 설명 블록(sqledbdescext) 구조를 사용하여 데이터베이스 속성에 대한 영구적인 값을 지정합니다. 확장된 데이터베이스 설명 블록을 사용하면 데이터베이스에 자동 스토리지를 사용할 수 있고 데이터베이스의 디폴트 페이지 *크기*를 선택하거나 새로 추가된 새 테이블 스페이스 속성에 대한 값을 지정합니다. 이 구조는 데이터베이스 설명 블록(sqledbdesc) 구조 대신이 아니라 추가적으로 사용됩니다.

이 구조를 sqlecrea API에 전달하지 않으면 다음과 같은 동작이 사용됩니다.

- 데이터베이스에 대해 자동 스토리지 사용 가능
- 데이터베이스의 디폴트 페이지 *크기*는 4096바이트(4KB)
- 관련된 경우 DB2 데이터베이스 시스템은 자동으로 확장된 테이블 스페이스 속성 값을 판별합니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqledbdescext
{
        sqluint32 sqlPageSize;
        struct sqleAutoStorageCfg *sqlAutoStorage;
        struct SQLETSDESCEXT *sqlcattsext;
        struct SQLETSDESCEXT *sqlusrtsext;
        struct SQLETSDESCEXT *sqltmptsext;
        void *reserved;
};

SQL_STRUCTURE sqleAutoStorageCfg
{
        char sqlEnableAutoStorage;
        char pad[3];
        sqluint32 sqlNumStoragePaths;
        char **sqlStoragePaths;
};

SQL_STRUCTURE SQLETSDESCEXT
{
        sqlint64 sqlInitSize;
        sqlint64 sqlIncreaseSize;
        sqlint64 sqlMaximumSize;
        char sqlAutoResize;
        char sqlInitSizeUnit;
        char sqlIncreaseSizeUnit;
        char sqlMaximumSizeUnit;
};

SQL_STRUCTURE sqledboptions
{
```

```
        void *piAutoConfigInterface;
        sqlint32 restrictive;
        void *reserved;
};
```

sqledbdescext 데이터 구조 매개변수

표 53. sqledbdescext 구조의 필드

필드 이름	데이터 유형	설명
SQLPAGESIZE	sqluint32	데이터베이스 작성 시의 디폴트 버퍼 풀 *크기* 뿐만 아니라 초기 테이블 스페이스(SYSCATSPACE, TEMPSPACE1, USERSPACE1)의 *크기*도 지정합니다. 지정된 값은 모든 전체 CREATE BUFFERPOOL 및 CREATE TABLESPACE 문의 디폴트 페이지 크기도 나타냅니다. 값에 대해서는 이 테이블 뒤에 오는 정보를 참조하십시오.
SQLAUTOSTORAGE	Pointer	자동 스토리지 구성 구조의 포인터. 이 포인터를 사용하여 데이터베이스의 자동 스토리지를 사용하거나 사용하지 않을 수 있습니다. 포인터를 지정하면 자동 스토리지를 사용하거나 사용하지 않을 수 있습니다. NULL인 경우 자동 스토리지가 사용 가능하고 단일 스토리지 경로는 dbpath로 전달 또는 데이터베이스 관리 프로그램 구성 매개변수인 dftdbpath에 의해 판별된 값으로 간주됩니다.
SQLCATTSEXT	Pointer	SQLETSDESC 속성에 대해 추가 속성을 정의하는 시스템 카탈로그 테이블 스페이스의 확장된 테이블 스페이스 설명 제어 블록(SQLETSDESCEXT)의 포인터. NULL인 경우 데이터베이스 관리 프로그램은 자동으로 이 속성 값을 판별합니다(관련된 경우).
SQLUSRTSEXT	Pointer	SQLETSDESC 속성에 대해 추가 속성을 정의하는 사용자 테이블 스페이스의 확장된 테이블 스페이스 설명 제어 블록(SQLETSDESCEXT)의 포인터. NULL인 경우 데이터베이스 관리 프로그램은 자동으로 이 속성 값을 판별합니다(관련된 경우).
SQLTMPTSEXT	Pointer	SQLETSDESC 속성에 대해 추가 속성을 정의하는 시스템 임시 테이블 스페이스의 확장된 테이블 스페이스 설명 제어 블록(SQLETSDESCEXT)의 포인터. NULL인 경우 데이터베이스 관리 프로그램은 자동으로 이 속성 값을 판별합니다(관련된 경우).
RESERVED	Pointer	데이터베이스 옵션 제어 블록(sqledboptions)의 포인터

SQLPAGESIZE의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

SQL_PAGESIZE_4K

데이터베이스의 디폴트 페이지 *크기*는 4096바이트입니다.

SQL_PAGESIZE_8K

데이터베이스의 디폴트 페이지 *크기*는 8192바이트입니다.

SQL_PAGESIZE_16K

데이터베이스의 디폴트 페이지 크기는 16 384바이트입니다.

SQL_PAGESIZE_32K

데이터베이스의 디폴트 페이지 크기는 32,768바이트입니다.

자동 스토리지 구성(sqleAutoStorageCfg) 데이터 구조 매개변수

자동 스토리지 구성(sqleAutoStorageCfg) 구조는 sqlecrea API 호출 중에 사용할 수 있습니다. sqledbdescext 구조의 요소이며 데이터베이스에 자동 스토리지 사용 여부를 지정합니다.

표 54. sqleAutoStorageCfg 구조의 필드

필드 이름	데이터 유형	설명
SQLENABLEAUTOSTORAGE	CHAR(1)	데이터베이스의 자동 스토리지 사용 가능 여부를 지정합니다. 값에 대해서는 이 테이블 뒤에 오는 정보를 참조하십시오.
SQLNUMSTORAGEPATHS	sqluint32	SQLSTORAGEPATHS 배열로 지시 중인 스토리지 경로 수를 나타내는 값. 값이 0인 경우 SQLSTORAGEPATHS 포인터는 NULL이어야 합니다. 스토리지 경로의 최대 수는 128입니다(SQL_MAX_STORAGE_PATHS).
SQLSTORAGEPATHS	Pointer	스토리지 경로를 지시하는 문자열 포인터의 배열. 배열의 포인터 수는 SQLNUMSTORAGEPATHS로 반영됩니다. 제공된 스토리지 경로가 없는 경우에는 SQLSTORAGEPATHS를 NULL로 설정하십시오(이 경우 SQLNUMSTORAGEPATHS는 0으로 설정해야 함). 각 경로의 최대 길이는 175자입니다.

SQLENABLEAUTOSTORAGE의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

SQL_AUTOMATIC_STORAGE_NO

데이터베이스에 대해 자동 스토리지를 사용할 수 없습니다. 이 값을 사용하는 경우 SQLNUMSTORAGEPATHS를 0으로 설정해야 하며 SQLSTORAGEPATHS는 NULL로 설정해야 합니다.

SQL_AUTOMATIC_STORAGE_YES

데이터베이스에 대해 자동 스토리지를 사용할 수 있습니다. 자동 스토리지에 사용되는 스토리지 경로는 SQLSTORAGEPATHS 포인터를 사용하여 지정합니다. 이 포인터가 NULL인 경우 단일 스토리지 경로는 데이터베이스 관리 프로그램 구성 매개변수로 판별된 값으로 간주됩니다.

SQL_AUTOMATIC_STORAGE_DFT

데이터베이스 관리 프로그램은 자동 스토리지 사용 가능 여부를 판별합니다. 현재 SQLSTORAGEPATHS 포인터에서 선택됩니다. 이 포인터가 NULL인 경

우 자동 스토리지를 사용할 수 없으며 그 이외의 경우에는 사용할 수 있습니다. 디폴트값은 SQL_AUTOMATIC_STORAGE_YES와 동등합니다.

확장된 테이블 스페이스 설명 블록(SQLETSDESCEXT) 구조 매개변수

확장된 테이블 스페이스 설명 블록(SQLETSDESCEXT) 구조를 사용하여 세 개의 초기 테이블 스페이스 속성을 지정합니다. 이 구조는 테이블 스페이스 설명 블록(SQLETSDESC) 구조 대신이 아니라 추가적으로 사용됩니다.

표 55. SQLETSDESCEXT 구조의 필드

필드 이름	데이터 유형	설명
SQLINITSIZE	sqlint64	자동 스토리지를 사용하는 각 테이블 스페이스의 초기 크기를 정의합니다. 이 필드는 일반 또는 대형 자동 스토리지 테이블 스페이스에만 관련됩니다. 다른 테이블 스페이스에 대해 SQL_TBS_AUTOMATIC_INITSIZE 값을 사용하거나 DB2가 자동으로 초기 크기를 판별하도록 하려는 경우에 사용하십시오. 참고: 데이터베이스 관리 프로그램에서 사용되는 실제 값은 지정된 값보다 조금 크거나 작을 수도 있습니다. 이 조치는 테이블 스페이스의 컨테이너에서 크기를 일관성 있게 유지하는 데 사용되며 제공된 값은 해당 일관성을 허용하지 않을 수도 있습니다.
SQLINCREASESIZE	sqlint64	테이블 스페이스가 가득 찰 때 데이터베이스 관리 프로그램이 자동으로 테이블 스페이스의 값을 증가시키는 크기를 정의합니다. 이 필드는 자동 크기 조정이 사용 가능한 테이블 스페이스에만 연관됩니다. 자동 크기 조정이 사용 불가능하거나 데이터베이스 관리 프로그램이 자동으로 크기 증가를 판별하도록 하려면 SQL_TBS_AUTOMATIC_INCSIZE 값을 사용하십시오. 참고: 데이터베이스 관리 프로그램에서 사용되는 실제 값은 지정된 값보다 조금 크거나 작을 수도 있습니다. 이 조치는 테이블 스페이스의 컨테이너에서 크기를 일관성 있게 유지하는 데 사용되며 제공된 값은 해당 일관성을 허용하지 않을 수도 있습니다.

표 55. SQLETSDESCEXT 구조의 필드 (계속)

필드 이름	데이터 유형	설명
SQLMAXIMUMSIZE	sqlint64	데이터베이스 관리 프로그램이 자동으로 테이블 스페이스 크기를 증가시키는 최대 크기를 정의합니다. 또는 SQL_TBS_NO_MAXSIZE 값을 사용하여 최대 크기를 "무제한"으로 지정할 수도 있으며 이 경우 테이블 스페이스는 테이블 스페이스의 구조적인 한계까지 확장되거나 "파일 시스템 가득 참" 조건이 발생될 때까지 확장될 수 있습니다. 이 필드는 자동 크기 조정이 사용 가능한 테이블 스페이스에만 연관됩니다. 자동 크기 조정이 사용 불가능하거나 데이터베이스 관리 프로그램이 자동으로 최대 크기를 판별하도록 하려면 SQL_TBS_AUTOMATIC_MAXSIZE 값을 사용하십시오. 참고: 데이터베이스 관리 프로그램에서 사용되는 실제 값은 지정된 값보다 조금 크거나 작을 수도 있습니다. 이 조치는 테이블 스페이스의 컨테이너에서 크기를 일관성 있게 유지하는 데 사용되며 제공된 값은 해당 일관성을 허용하지 않을 수도 있습니다.
SQLAUTORESIZE	CHAR(1)	테이블 스페이스에 자동 크기 조정의 사용 가능 여부를 지정합니다. 값에 대해서는 이 테이블 뒤에 오는 정보를 참조하십시오.
SQLINITSIZEUNIT	CHAR(1)	관련된 경우 SQLINITSIZE가 바이트, KB, MB 또는 GB 단위로 제공되는지를 나타냅니다. 값에 대해서는 이 테이블 뒤에 오는 정보를 참조하십시오.
SQLINCREASESIZEUNIT	CHAR(1)	관련된 경우 SQLINCREASESIZE가 바이트, KB, MB, GB 또는 백분율로 제공되는지를 나타냅니다. 값에 대해서는 이 테이블 뒤에 오는 정보를 참조하십시오.
SQLMAXIMUMSIZEUNIT	CHAR(1)	관련된 경우 SQLMAXIMUMSIZE가 바이트, KB, MB 또는 GB 단위로 제공되는지를 나타냅니다. 값에 대해서는 이 테이블 뒤에 오는 정보를 참조하십시오.

SQLAUTORESIZE의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

SQL_TBS_AUTORESIZE_NO

테이블 스페이스에 자동 크기 조정을 사용할 수 없습니다. 이 값은 데이터베이스 관리 스페이스(DMS) 테이블 스페이스 또는 자동 스토리지 테이블 스페이스에만 지정할 수 있습니다.

SQL_TBS_AUTORESIZE_YES

테이블 스페이스에 자동 크기 조정을 사용할 수 있습니다. 이 값은 데이터베이스 관리 스페이스(DMS) 테이블 스페이스 또는 자동 스토리지 테이블 스페이스에만 지정할 수 있습니다.

SQL_TBS_AUTORESIZE_DFT

데이터베이스 관리 프로그램은 테이블 스페이스 유형에 따라 자동 크기 조정이 사용 가능한지를 판별할 수 있습니다. 데이터베이스 관리 스페이스(DMS) 테이

블 스페이스 및 자동 스토리지 테이블 스페이스에 대해 자동 *크기* 조정이 해제됩니다. 테이블 스페이스에는 자동 *크기* 조정이 적용되지 않기 때문에 시스템 관리 스페이스(SMS) 테이블 스페이스에 대해 이 값을 사용하십시오.

SQLINITSIZEUNIT, SQLINCREASESIZEUNIT 및 SQLMAXIMUMSIZEUNIT의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

SQL_TBS_STORAGE_UNIT_BYTES
해당 *크기* 필드에 지정된 값은 바이트 단위입니다.

SQL_TBS_STORAGE_UNIT_KILOBYTES
해당 *크기* 필드에 지정된 값은 KB 단위입니다(1KB = 1024바이트)

SQL_TBS_STORAGE_UNIT_MEGABYTES
해당 *크기* 필드에 지정된 값은 MB 단위입니다(1MB = 1,048,576바이트)

SQL_TBS_STORAGE_UNIT_GIGABYTES
해당 *크기* 필드에 지정된 값은 GB 단위입니다(1GB = 1 073 741 824바이트)

SQL_TBS_STORAGE_UNIT_PERCENT
해당 *크기* 필드에 지정된 값은 백분율입니다(유효한 범위는 1 - 100). 이 값은 SQLINCREASESIZEUNIT에만 사용할 수 있습니다.

sqledboptions 데이터 구조 매개변수

piAutoConfigInterface
입력. 구성 어드바이저의 입력으로 사용되는 정보가 포함된 db2AutoConfigInterface 구조의 포인터

restrictive
제한 필드의 설정은 RESTRICT_ACCESS 데이터베이스 구성 매개변수에 저장되고 이후의 모든 이 데이터베이스 업그레이드에 영향을 줍니다. 즉, 데이터베이스가 후속 DB2 릴리스로 업그레이드되면 UPGRADE DATABASE가 RESTRICT_ACCESS 데이터베이스 구성 매개변수 설정을 점검하여 디폴트 조치의 제한 세트를 새 DB2 릴리스에서 소개된 모든 새 오브젝트(예를 들어, 새 시스템 카탈로그 테이블)에 적용해야 하는지를 판별합니다.

이 매개변수에 대해 유효한 값(include 디렉토리에 있는 sqlenv 헤더 파일에 정의)은 다음과 같습니다.

SQL_DB_RESTRICT_ACCESS_NO 또는 SQL_DB_RESTRICT_ACCESS_DFT
데이터베이스가 디폴트 조치의 제한 세트를 사용하지 않고 작성되는 것을 나타냅니다. 이 설정을 사용하여 PUBLIC에 다음 특권이 부여됩니다.

- CREATETAB 특권

- BINDADD 특권
- CONNECT 특권
- IMPLICIT_SCHEMA 특권
- 스키마 SQLJ의 모든 프로시저에 GRANT가 포함된 EXECUTE 특권
- 스키마 SYSPROC의 모든 함수 및 프로시저에 GRANT가 포함된 EXECUTE 특권
- NULLID 스키마에 작성된 모든 패키지의 BIND 특권
- NULLID 스키마에 작성된 모든 패키지의 EXECUTE 특권
- 스키마 SQLJ의 CREATEIN 특권
- 스키마 NULLID의 CREATEIN 특권
- 테이블 스페이스 USERSPACE1의 USE 특권
- SYSIBM 카탈로그 테이블의 SELECT 특권
- SYSCAT 카탈로그 뷰의 SELECT 특권
- SYSSTAT 카탈로그 뷰의 SELECT 특권
- SYSSTAT 카탈로그 뷰의 UPDATE 특권

SQL_DB_RESTRICT_ACCESS_YES
데이터베이스가 디폴트 조치의 제한 세트를 사용하여 작성되는 것을 나타냅니다. 이 경우 위의 SQL_DB_RESTRICT_ACCESS_NO에 나열된 권한 부여 조치가 수행되지 않습니다.

예약 나중에 사용하기 위해 예약됨

제 188 장 sqledbterritoryinfo

이 구조를 사용하여 sqlecrea API에 대한 코드 세트와 지역 옵션을 제공합니다.

표 56. SQLEDBTERRITORYINFO 구조의 필드

필드 이름	데이터 유형	설명
SQLDBCODESET	CHAR(9)	데이터베이스 코드 세트.
SQLDBLOCALE	CHAR(5)	데이터베이스 지역.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqledbcountryinfo
{
        char sqldbcodeset[SQL_CODESET_LEN + 1];
        char sqldblocale[SQL_LOCALE_LEN + 1];
};
typedef SQL_STRUCTURE sqledbcountryinfo SQLEDBTERRITORYINFO;
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQLEDBTERRITORYINFO.
    05 SQLDBCODESET          PIC X(9).
    05 FILLER                PIC X.
    05 SQLDBLOCALE           PIC X(5).
    05 FILLER                PIC X.
*
```

제 189 장 sqleninfo

이 구조는 sqlengne API 호출 후에 정보를 리턴합니다.

주: NetBIOS는 더 이상 지원되지 않습니다. 해당 API인 APPC, APPN 및 CPI-C가 포함된 SNA가 더 이상 지원되지 않습니다. 이 프로토콜을 사용하는 경우에는 TCP/IP와 같이 지원되는 프로토콜을 사용하여 노드와 데이터베이스를 재카탈로그해야 합니다. 이 프로토콜 참조는 무시해야 합니다.

표 57. SQLENINFO 구조의 필드

필드 이름	데이터 유형	설명
NODENAME	CHAR(8)	NetBIOS 프로토콜에 사용. 데이터베이스가 있는 노드의 nname(시스템 디렉토리에서만 유효)
LOCAL_LU	CHAR(8)	APPN 프로토콜에 사용. 로컬 논리 장치.
PARTNER_LU	CHAR(8)	APPN 프로토콜에 사용. 상대 논리 장치.
MODE	CHAR(8)	APPN 프로토콜에 사용. 전송 서비스 모드
COMMENT	CHAR(30)	노드에 연관된 주석
COM_CODEPAGE	SMALLINT	주석의 코드 페이지. 이 필드는 데이터베이스 관리 프로그램에서 더 이상 사용되지 않습니다.
ADAPTER	SMALLINT	NetBIOS 프로토콜에 사용. 로컬 네트워크 어댑터.
NETWORKID	CHAR(8)	APPN 프로토콜에 사용. 네트워크 ID
PROTOCOL	CHAR(1)	통신 프로토콜.
SYM_DEST_NAME	CHAR(8)	APPC 프로토콜에 사용. 기호 목적지 이름
SECURITY_TYPE	SMALLINT	APPC 프로토콜에 사용. 보안 유형. 값은 아래를 참조하십시오.
HOSTNAME	CHAR(255)	TCP/IP 프로토콜에 사용됩니다. DB2 서버 인스턴스가 있는 TCP/IP 호스트 또는 IPv4 또는 IPv6 주소 이름
SERVICE_NAME	CHAR(14)	TCP/IP 프로토콜에 사용됩니다. DB2 서버 인스턴스의 TCP/IP 서비스 이름 또는 관련 포트 번호

표 57. *SQLENINFO 구조의 필드 (계속)*

필드 이름	데이터 유형	설명
FILESERVER	CHAR(48)	IPX/SPX 프로토콜에 사용됩니다. DB2 서버 인스턴스가 등록된 Netware 파일 서버의 이름
OBJECTNAME	CHAR(48)	데이터베이스 관리 프로그램 서버 인스턴스가 NetWare 파일 서버에서 object, objectname으로 표시됩니다. 서버의 IPX/SPX 인터네트워크 주소가 저장되어 이 오브젝트에서 검색됩니다.
INSTANCE_NAME	CHAR(8)	로컬 및 NPIPE 프로토콜에 사용됩니다. 서버 인스턴스 이름
COMPUTERNAME	CHAR(15)	NPIPE 프로토콜에 사용됩니다. 서버 노드의 컴퓨터 이름
SYSTEM_NAME	CHAR(21)	리모트 서버의 DB2 시스템 이름
REMOTE_INSTNAME	CHAR(8)	DB2 서버 인스턴스 이름
CATALOG_NODE_TYPE	CHAR	카탈로그 노드 유형
OS_TYPE	UNSIGNED SHORT	서버의 운영 체제를 식별합니다.

주: 리턴된 각 문자 필드는 필드 길이까지 채워지는 공백입니다.

SECURITY_TYPE의 유효한 값(sqlenv에 정의)은 다음과 같습니다.

- SQL_CPIC_SECURITY_NONE
- SQL_CPIC_SECURITY_SAME
- SQL_CPIC_SECURITY_PROGRAM

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqleninfo
{
        _SQLOLDCHAR nodename[SQL_NNAME_SZ];
        _SQLOLDCHAR local_lu[SQL_LOCLU_SZ];
        _SQLOLDCHAR partner_lu[SQL_RMTLU_SZ];
        _SQLOLDCHAR mode[SQL_MODE_SZ];
        _SQLOLDCHAR comment[SQL_CMT_SZ];
        unsigned short com_codepage;
        unsigned short adapter;
        _SQLOLDCHAR networkid[SQL_NETID_SZ];
        _SQLOLDCHAR protocol;
        _SQLOLDCHAR sym_dest_name[SQL_SYM_DEST_NAME_SZ];
        unsigned short security_type;
        _SQLOLDCHAR hostname[SQL_HOSTNAME_SZ];
        _SQLOLDCHAR service_name[SQL_SERVICE_NAME_SZ];
        char fileserver[SQL_FILESERVER_SZ];
        char objectname[SQL_OBJECTNAME_SZ];
        char instance_name[SQL_INSTNAME_SZ];
        char computername[SQL_COMPUTERNAME_SZ];
        char system_name[SQL_SYSTEM_NAME_SZ];
        char remote_instname[SQL_REMOTE_INSTNAME_SZ];
```

```
        _SQLOLDCHAR catalog_node_type;
        unsigned short os_type;
        _SQLOLDCHAR chgpwd_lu[SQL_RMTLU_SZ];
        _SQLOLDCHAR transpn[SQL_TPNAME_SZ];
        _SQLOLDCHAR lanaddr[SQL_LANADDRESS_SZ];
};
```

COBOL 구조

```
* File: sqlenv.cbl
01 SQLENINFO.
    05 SQL-NODE-NAME          PIC X(8).
    05 SQL-LOCAL-LU           PIC X(8).
    05 SQL-PARTNER-LU         PIC X(8).
    05 SQL-MODE               PIC X(8).
    05 SQL-COMMENT            PIC X(30).
    05 SQL-COM-CODEPAGE       PIC 9(4) COMP-5.
    05 SQL-ADAPTER            PIC 9(4) COMP-5.
    05 SQL-NETWORKID          PIC X(8).
    05 SQL-PROTOCOL           PIC X.
    05 SQL-SYM-DEST-NAME      PIC X(8).
    05 FILLER                 PIC X(1).
    05 SQL-SECURITY-TYPE      PIC 9(4) COMP-5.
    05 SQL-HOSTNAME           PIC X(255).
    05 SQL-SERVICE-NAME       PIC X(14).
    05 SQL-FILESERVER         PIC X(48).
    05 SQL-OBJECTNAME         PIC X(48).
    05 SQL-INSTANCE-NAME      PIC X(8).
    05 SQL-COMPUTERNAME       PIC X(15).
    05 SQL-SYSTEM-NAME        PIC X(21).
    05 SQL-REMOTE-INSTNAME    PIC X(8).
    05 SQL-CATALOG-NODE-TYPE  PIC X.
    05 SQL-OS-TYPE            PIC 9(4) COMP-5.
*
```

제 190 장 sqlfupd

이 구조는 데이터베이스 구성 파일 및 데이터베이스 관리 프로그램 구성 파일에 대한 정보를 전달합니다.

표 58. SQLFUPD 구조의 필드

필드 이름	데이터 유형	설명
TOKEN	UINT16	리턴 또는 갱신할 구성 값을 지정합니다.
PTRVALUE	포인터	TOKEN으로 지정한 데이터를 보유하는 응용프로그램 할당 버퍼의 포인터

토큰 요소에 유효한 데이터 유형은 다음과 같습니다.

Uint16
부호없는 2바이트 정수

Sint16
부호있는 2바이트 정수

Uint32
부호없는 4바이트 정수

Sint32
부호있는 4바이트 정수

Uint64
부호없는 8바이트 정수

float 4바이트 부동 소수점 10진수

char(n)
n 길이의 문자열(널(NULL) 종료는 포함되지 않음).

SQLFUPD 토큰 요소에 유효한 항목은 다음과 같습니다.

표 59. 갱신 가능한 데이터베이스 구성 매개변수

매개변수 이름	토큰	토큰 값	데이터 유형
alt_collate	SQLF_DBTN_ALT_COLLATE	809	Uint32
app_ctl_heap_sz	SQLF_DBTN_APP_CTL_HEAP_SZ	500	Uint16
appgroup_mem_sz	SQLF_DBTN_APPGROUP_MEM_SZ	800	Uint32
applheapsz	SQLF_DBTN_APPLHEAPSZ	51	Uint16
archretrydelay	SQLF_DBTN_ARCHRETRYDELAY	828	Uint16

표 59. 갱신 가능한 데이터베이스 구성 매개변수 (계속)

매개변수 이름	토큰	토큰 값	데이터 유형
• auto_maint • auto_db_backup • auto_tbl_maint • auto_runstats • auto_stats_prof • auto_prof_upd • auto_reorg	• SQLF_DBTN_AUTO_MAINT • SQLF_DBTN_AUTO_DB_BACKUP • SQLF_DBTN_AUTO_TBL_MAINT • SQLF_DBTN_AUTO_RUNSTATS • SQLF_DBTN_AUTO_STATS_PROF • SQLF_DBTN_AUTO_PROF_UPD • SQLF_DBTN_AUTO_REORG	• 831 • 833 • 835 • 837 • 839 • 844 • 841	Uint16
autorestart	SQLF_DBTN_AUTO_RESTART	25	Uint16
avg_appls	SQLF_DBTN_AVG_APPLS	47	Uint16
blk_log_dsk_ful	SQLF_DBTN_BLK_LOG_DSK_FUL	804	Uint16
catalogcache_sz	SQLF_DBTN_CATALOGCACHE_SZ	56	Sint32
chngpgs_thresh	SQLF_DBTN_CHNGPGS_THRESH	38	Uint16
database_memory	SQLF_DBTN_DATABASE_MEMORY	803	Uint64
dbheap	SQLF_DBTN_DB_HEAP	58	Uint64
db_mem_thresh	SQLF_DBTN_DB_MEM_THRESH	849	Uint16
dft_degree	SQLF_DBTN_DFT_DEGREE	301	Sint32
dft_extent_sz	SQLF_DBTN_DFT_EXTENT_SZ	54	Uint32
dft_loadrec_ses	SQLF_DBTN_DFT_LOADREC_SES	42	Sint16
dft_mttb_types	SQLF_DBTN_DFT_MTTB_TYPES	843	Uint32
dft_prefetch_sz	SQLF_DBTN_DFT_PREFETCH_SZ	40	Sint16
dft_queryopt	SQLF_DBTN_DFT_QUERYOPT	57	Sint32
dft_refresh_age	SQLF_DBTN_DFT_REFRESH_AGE	702	char(22)
dft_sqlmathwarn	SQLF_DBTN_DFT_SQLMATHWARN	309	Sint16
discover	SQLF_DBTN_DISCOVER	308	Uint16
dlchktime	SQLF_DBTN_DLCHKTIME	9	Uint32
dyn_query_mgmt	SQLF_DBTN_DYN_QUERY_MGMT	604	Uint16
failarchpath	SQLF_DBTN_FAILARCHPATH	826	char(243)
groupheap_ratio	SQLF_DBTN_GROUPHEAP_RATIO	801	Uint16
hadr_local_host	SQLF_DBTN_HADR_LOCAL_HOST	811	char(256)
hadr_local_svc	SQLF_DBTN_HADR_LOCAL_SVC	812	char(41)
hadr_remote_host	SQLF_DBTN_HADR_REMOTE_HOST	813	char(256)
hadr_remote_inst	SQLF_DBTN_HADR_REMOTE_INST	815	char(9)
hadr_remote_svc	SQLF_DBTN_HADR_REMOTE_SVC	814	char(41)
hadr_syncmode	SQLF_DBTN_HADR_SYNCMODE	817	Uint32
hadr_timeout	SQLF_DBTN_HADR_TIMEOUT	816	Sint32
indexrec	SQLF_DBTN_INDEXREC	30	Uint16
locklist	SQLF_DBTN_LOCK_LIST	704	Uint64
locktimeout	SQLF_DBTN_LOCKTIMEOUT	34	Sint16
logarchmeth1	SQLF_DBTN_LOGARCHMETH1	822	Uint16

표 59. 갱신 가능한 데이터베이스 구성 매개변수 (계속)

매개변수 이름	토큰	토큰 값	데이터 유형
logarchmeth2	SQLF_DBTN_LOGARCHMETH2	823	Uint16
logarchopt1	SQLF_DBTN_LOGARCHOPT1	824	char(243)
logarchopt2	SQLF_DBTN_LOGARCHOPT2	825	char(243)
logbufsz	SQLF_DBTN_LOGBUFSZ	33	Uint16
logfilsiz	SQLF_DBTN_LOGFIL_SIZ	92	Uint32
logindexbuild	SQLF_DBTN_LOGINDEXBUILD	818	Uint32
logprimary	SQLF_DBTN_LOGPRIMARY	16	Uint16
logretain	SQLF_DBTN_LOG_RETAIN	23	Uint16
logsecond	SQLF_DBTN_LOGSECOND	17	Uint16
max_log	SQLF_DBTN_MAX_LOG	807	Uint16
maxappls	SQLF_DBTN_MAXAPPLS	6	Uint16
maxfilop	SQLF_DBTN_MAXFILOP	3	Uint16
maxlocks	SQLF_DBTN_MAXLOCKS	15	Uint16
max_log	SQLF_DBTN_MAX_LOG	807	Uint16
mincommit	SQLF_DBTN_MINCOMMIT	32	Uint16
mirrorlogpath	SQLF_DBTN_MIRRORLOGPATH	806	char(242)
newlogpath	SQLF_DBTN_NEWLOGPATH	20	char(242)
num_db_backups	SQLF_DBTN_NUM_DB_BACKUPS	601	Uint16
num_freqvalues	SQLF_DBTN_NUM_FREQVALUES	36	Uint16
num_iocleaners	SQLF_DBTN_NUM_IOCLEANERS	37	Uint16
num_ioservers	SQLF_DBTN_NUM_IOSERVERS	39	Uint16
num_log_span	SQLF_DBTN_NUM_LOG_SPAN	808	Uint16
num_quantiles	SQLF_DBTN_NUM_QUANTILES	48	Uint16
numarchretry	SQLF_DBTN_NUMARCHRETRY	827	Uint16
overflowlogpath	SQLF_DBTN_OVERFLOWLOGPATH	805	char(242)
pckcachesz	SQLF_DBTN_PCKCACHE_SZ	505	Uint32
rec_his_retentn	SQLF_DBTN_REC_HIS_RETENTN	43	Sint16
self_tuning_mem	SQLF_DBTN_SELF_TUNING_MEM	848	Uint16
seqdetect	SQLF_DBTN_SEQDETECT	41	Uint16
sheapthres_shr	SQLF_DBTN_SHEAPTHRES_SHR	802	Uint32
softmax	SQLF_DBTN_SOFTMAX	5	Uint16
sortheap	SQLF_DBTN_SORT_HEAP	52	Uint32
stat_heap_sz	SQLF_DBTN_STAT_HEAP_SZ	45	Uint32
stmtheap	SQLF_DBTN_STMTHEAP	53	Uint16
trackmod	SQLF_DBTN_TRACKMOD	703	Uint16
tsm_mgmtclass	SQLF_DBTN_TSM_MGMTCLASS	307	char(30)
tsm_nodename	SQLF_DBTN_TSM_NODENAME	306	char(64)
tsm_owner	SQLF_DBTN_TSM_OWNER	305	char(64)
tsm_password	SQLF_DBTN_TSM_PASSWORD	501	char(64)
userexit	SQLF_DBTN_USER_EXIT	24	Uint16

표 59. 갱신 가능한 데이터베이스 구성 매개변수 (계속)

매개변수 이름	토큰	토큰 값	데이터 유형
util_heap_sz	SQLF_DBTN_UTIL_HEAP_SZ	55	Uint32
vendoropt	SQLF_DBTN_VENDOROPT	829	char(242)

SQLF_DBTN_AUTONOMIC_SWITCHES 비트는 다수의 자동 유지보수 구성 매개변수 디폴트 설정입니다. 이 복합 매개변수를 구성하는 각 비트는 다음과 같습니다.

```
Default => Bit 1 on  (xxxx xxxx xxxx xxx1): auto_maint
           Bit 2 off (xxxx xxxx xxxx xx0x): auto_db_backup
           Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint
           Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
           Bit 5 off (xxxx xxxx xxx0 xxxx): auto_stats_prof
           Bit 6 off (xxxx xxxx xx0x xxxx): auto_prof_upd
           Bit 7 off (xxxx xxxx x0xx xxxx): auto_reorg
           Bit 8 off (xxxx xxxx 0xxx xxxx): auto_storage
           Bit 9 off (xxxx xxx0 xxxx xxxx): auto_stmt_stats
                       0    0    0    D

Maximum => Bit 1 on  (xxxx xxxx xxxx xxx1): auto_maint
           Bit 2 off (xxxx xxxx xxxx xx1x): auto_db_backup
           Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint
           Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
           Bit 5 off (xxxx xxxx xxx1 xxxx): auto_stats_prof
           Bit 6 off (xxxx xxxx xx1x xxxx): auto_prof_upd
           Bit 7 off (xxxx xxxx x1xx xxxx): auto_reorg
           Bit 8 off (xxxx xxxx 1xxx xxxx): auto_storage
           Bit 9 off (xxxx xxx1 xxxx xxxx): auto_stmt_stats
                       0    1    F    F
```

indexrec에 유효한 값(sqlutil.h에 정의):

- SQLF_INX_REC_SYSTEM (0)
- SQLF_INX_REC_REFERENCE (1)
- SQLF_INX_REC_RESTART (2)

logretain에 유효한 값(sqlutil.h에 정의):

- SQLF_LOGRETAIN_NO (0)
- SQLF_LOGRETAIN_RECOVERY (1)
- SQLF_LOGRETAIN_CAPTURE (2)

표 60. 갱신 불가능한 데이터베이스 구성 매개변수

매개변수 이름	토큰	토큰 값	데이터 유형
backup_pending	SQLF_DBTN_BACKUP_PENDING	112	Uint16
codepage	SQLF_DBTN_CODEPAGE	101	Uint16
codeset	SQLF_DBTN_CODESET	120	char(9)(아래 메모 1 참조)
collate_info	SQLF_DBTN_COLLATE_INFO	44	char(260)

표 60. 갱신 불가능한 데이터베이스 구성 매개변수 (계속)

매개변수 이름	토큰	토큰 값	데이터 유형
country/region	SQLF_DBTN_COUNTRY	100	Uint16
database_consistent	SQLF_DBTN_CONSISTENT	111	Uint16
database_level	SQLF_DBTN_DATABASE_LEVEL	124	Uint16
log_retain_status	SQLF_DBTN_LOG_RETAIN_STATUS	114	Uint16
loghead	SQLF_DBTN_LOGHEAD	105	char(12)
logpath	SQLF_DBTN_LOGPATH	103	char(242)
multipage_alloc	SQLF_DBTN_MULTIPAGE_ALLOC	506	Uint16
numsegs	SQLF_DBTN_NUMSEGS	122	Uint16
release	SQLF_DBTN_RELEASE	102	Uint16
restore_pending	SQLF_DBTN_RESTORE_PENDING	503	Uint16
rollfwd_pending	SQLF_DBTN_ROLLFWD_PENDING	113	Uint16
territory	SQLF_DBTN_TERRITORY	121	char(5) (see note 2 below)
user_exit_status	SQLF_DBTN_USER_EXIT_STATUS	115	Uint16

주:

1. HP-UX, Solaris 및 Linux 운영 체제의 char(17)
2. HP-UX, Solaris 및 Linux 운영 체제의 char(33)

SQLFUPD 토큰 요소에 유효한 항목은 다음과 같습니다.

표 61. 갱신 가능한 데이터베이스 관리 프로그램 구성 매개변수

매개변수 이름	토큰	토큰 값	데이터 유형
agent_stack_sz	SQLF_KTN_AGENT_STACK_SZ	61	Uint16
agentpri	SQLF_KTN_AGENTPRI	26	Sint16
alternate_auth_enc	SQLF_KTN_ALTERNATE_AUTH_ENC	938	Uint16
aslheapsz	SQLF_KTN_ASLHEAPSZ	15	Uint32
audit_buf_sz	SQLF_KTN_AUDIT_BUF_SZ	312	Sint32
authentication	SQLF_KTN_AUTHENTICATION	78	Uint16
catalog_noauth	SQLF_KTN_CATALOG_NOAUTH	314	Uint16
clnt_krb_plugin	SQLF_KTN_CLNT_KRB_PLUGIN	812	char(33)
clnt_pw_plugin	SQLF_KTN_CLNT_PW_PLUGIN	811	char(33)
comm_bandwidth	SQLF_KTN_COMM_BANDWIDTH	307	float
conn_elapse	SQLF_KTN_CONN_ELAPSE	508	Uint16
cpuspeed	SQLF_KTN_CPUSPEED	42	float
dft_account_str	SQLF_KTN_DFT_ACCOUNT_STR	28	char(25)
dft_monswitches	SQLF_KTN_DFT_MONSWITCHES	29	Uint16
dft_mon_bufpool	SQLF_KTN_DFT_MON_BUFPOOL	33	Uint16
dft_mon_lock	SQLF_KTN_DFT_MON_LOCK	34	Uint16
dft_mon_sort	SQLF_KTN_DFT_MON_SORT	35	Uint16

표 61. 갱신 가능한 데이터베이스 관리 프로그램 구성 매개변수 (계속)

매개변수 이름	토큰	토큰 값	데이터 유형
dft_mon_stmt	SQLF_KTN_DFT_MON_STMT	31	Uint16
dft_mon_table	SQLF_KTN_DFT_MON_TABLE	32	Uint16
dft_mon_timestamp	SQLF_KTN_DFT_MON_ TIMESTAMP	36	Uint16
dft_mon_uow	SQLF_KTN_DFT_MON_UOW	30	Uint16
dftdbpath	SQLF_KTN_DFTDBPATH	27	char(215)
diaglevel	SQLF_KTN_DIAGLEVEL	64	Uint16
diagpath	SQLF_KTN_DIAGPATH	65	char(215)
dir_cache	SQLF_KTN_DIR_CACHE	40	Uint16
discover	SQLF_KTN_DISCOVER	304	Uint16
discover_inst	SQLF_KTN_DISCOVER_INST	308	Uint16
fcm_num_buffers	SQLF_KTN_FCM_NUM_BUFFERS	503	Uint32
fcm_num_channels	SQLF_KTN_FCM_NUM_CHANNELS	902	Uint32
fed_noauth	SQLF_KTN_FED_NOAUTH	806	Uint16
federated	SQLF_KTN_FEDERATED	604	Sint16
federated_async	SQLF_KTN_FEDERATED_ASYNC	849	Sint32
fenced_pool	SQLF_KTN_FENCED_POOL	80	Sint32
group_plugin	SQLF_KTN_GROUP_PLUGIN	810	char(33)
health_mon	SQLF_KTN_HEALTH_MON	804	Uint16
indexrec	SQLF_KTN_INDEXREC	20	Uint16
instance_memory	SQLF_KTN_INSTANCE_MEMORY	803	Uint64
intra_parallel	SQLF_KTN_INTRA_PARALLEL	306	Sint16
java_heap_sz	SQLF_KTN_JAVA_HEAP_SZ	310	Sint32
jdk_path	SQLF_KTN_JDK_PATH	311	char(255)
keepfenced	SQLF_KTN_KEEPFENCED	81	Uint16
local_gssplugin	SQLF_KTN_LOCAL_GSSPLUGIN	816	char(33)
max_connections	SQLF_DBTN_MAX_CONNECTIONS	802	Sint32
max_connretries	SQLF_KTN_MAX_CONNRETRIES	509	Uint16
max_coordagents	SQLF_KTN_MAX_COORDAGENTS	501	Sint32
max_querydegree	SQLF_KTN_MAX_QUERYDEGREE	303	Sint32
max_time_diff	SQLF_KTN_MAX_TIME_DIFF	510	Uint16
mon_heap_sz	SQLF_KTN_MON_HEAP_SZ	79	Uint16
notifylevel	SQLF_KTN_NOTIFYLEVEL	605	Sint16
num_initagents	SQLF_KTN_NUM_INITAGENTS	500	Uint32
num_initfenced	SQLF_KTN_NUM_INITFENCED	601	Sint32
num_poolagents	SQLF_KTN_NUM_POOLAGENTS	502	Sint32
numdb	SQLF_KTN_NUMDB	6	Uint16
query_heap_sz	SQLF_KTN_QUERY_HEAP_SZ	49	Sint32
resync_interval	SQLF_KTN_RESYNC_INTERVAL	68	Uint16
rqrioblk	SQLF_KTN_RQRIOBLK	1	Uint16
sheapthres	SQLF_KTN_SHEAPTHRES	21	Uint32

표 61. 갱신 가능한 데이터베이스 관리 프로그램 구성 매개변수 (계속)

매개변수 이름	토큰	토큰 값	데이터 유형
spm_log_file_sz	SQLF_KTN_SPM_LOG_FILE_SZ	90	Sint32
spm_log_path	SQLF_KTN_SPM_LOG_PATH	313	char(226)
spm_max_resync	SQLF_KTN_SPM_MAX_RESYNC	91	Sint32
spm_name	SQLF_KTN_SPM_NAME	92	char(8)
srvcon_auth	SQLF_KTN_SRVCON_AUTH	815	Uint16
srvcon_gssplugin_list	SQLF_KTN_SRVCON_GSSPLUGIN_ LIST	814	char(256)
srv_plugin_mode	SQLF_KTN_SRV_PLUGIN_MODE	809	Uint16
srvcon_pw_plugin	SQLF_KTN_SRVCON_PW_PLUGIN	813	char(33)
ssl_cipherspecs	SQLF_KTN_SSL_CIPHERSPECS	934	char(255)
ssl_clnt_keydb	SQLF_KTN_SSL_CLNT_KEYDB	936	char(1023)
ssl_clnt_stash	SQLF_KTN_SSL_CLNT_STASH	937	char(1023)
ssl_svcename	SQLF_KTN_SSL_SVCENAME	933	char(14)
ssl_svr_keydb	SQLF_KTN_SSL_SVR_KEYDB	930	char(1023)
ssl_svr_label	SQLF_KTN_SSL_SVR_LABEL	932	char(1023)
ssl_svr_stash	SQLF_KTN_SSL_SVR_STASH	931	char(1023)
ssl_versions	SQLF_KTN_SSL_VERSIONS	935	char(255)
start_stop_time	SQLF_KTN_START_STOP_TIME	511	Uint16
svcename	SQLF_KTN_SVCENAME	24	char(14)
sysadm_group	SQLF_KTN_SYSADM_GROUP	39	char(16)
sysctrl_group	SQLF_KTN_SYSCTRL_GROUP	63	char(16)
sysmaint_group	SQLF_KTN_SYSMAINT_GROUP	62	char(16)
sysmon_group	SQLF_KTN_SYSMON_GROUP	808	char(30)
tm_database	SQLF_KTN_TM_DATABASE	67	char(8)
tp_mon_name	SQLF_KTN_TP_MON_NAME	66	char(19)
trust_allclnts	SQLF_KTN_TRUST_ALLCLNTS	301	Uint16
trust_clntauth	SQLF_KTN_TRUST_CLNTAUTH	302	Uint16
util_impact_lim	SQLF_KTN_UTIL_IMPACT_LIM	807	Uint32

주: 구성 매개변수 `maxagents` 및 `maxcagents`는 더 이상 사용되지 않습니다. 이 구성 매개변수는 추후 릴리스에서 완전히 제거될 수 있습니다.

alternate_auth_enc에 유효한 값(sqlenv.h에 정의):

- SQL_ALTERNATE_AUTH_ENC_AES (0)
- SQL_ALTERNATE_AUTH_ENC_AES_CMP (1)
- SQL_ALTERNATE_AUTH_ENC_NOTSPEC (255)

authentication에 유효한 값(sqlenv.h에 정의):

- SQL_AUTHENTICATION_SERVER (0)
- SQL_AUTHENTICATION_CLIENT (1)

- SQL_AUTHENTICATION_DCS (2)
- SQL_AUTHENTICATION_DCE (3)
- SQL_AUTHENTICATION_SVR_ENCRYPT (4)
- SQL_AUTHENTICATION_DCS_ENCRYPT (5)
- SQL_AUTHENTICATION_DCE_SVR_ENC (6)
- SQL_AUTHENTICATION_KERBEROS (7)
- SQL_AUTHENTICATION_KRB_SVR_ENC (8)
- SQL_AUTHENTICATION_GSSPLUGIN (9)
- SQL_AUTHENTICATION_GSS_SVR_ENC (10)
- SQL_AUTHENTICATION_DATAENC (11)
- SQL_AUTHENTICATION_DATAENC_CMP (12)
- SQL_AUTHENTICATION_NOT_SPEC (255)

SQLF_KTN_DFT_MONSWITCHES는 디폴트 모니터 스위치 설정을 나타내는 비트로 Uint16 매개변수입니다. 이를 사용하면 동시에 몇 개의 매개변수 스펙이 허용됩니다. 이 복합 매개변수를 구성하는 각 비트는 다음과 같습니다.

- 비트 1(xxxx xxx1): dft_mon_uow
- 비트 2(xxxx xx1x): dft_mon_stmt
- 비트 3(xxxx x1xx): dft_mon_table
- 비트 4(xxxx 1xxx): dft_mon_buffpool
- 비트 5(xxx1 xxxx): dft_mon_lock
- 비트 6(xx1x xxxx): dft_mon_sort
- 비트 7(x1xx xxxx): dft_mon_timestamp

discover에 유효한 값(sqlutil.h에 정의):

- SQLF_DSCVR_KNOWN (1)
- SQLF_DSCVR_SEARCH (2)

indexrec에 유효한 값(sqlutil.h에 정의):

- SQLF_INX_REC_SYSTEM (0)
- SQLF_INX_REC_REFERENCE (1)
- SQLF_INX_REC_RESTART (2)

trust_allclnts에 유효한 값(sqlutil.h에 정의):

- SQLF_TRUST_ALLCLNTS_NO (0)
- SQLF_TRUST_ALLCLNTS_YES (1)
- SQLF_TRUST_ALLCLNTS_DRDAONLY (2)

표 62. 갱신 불가능한 데이터베이스 관리 프로그램 구성 매개변수

매개변수 이름	토큰	토큰 값	데이터 유형
nodetype	SQLF_KTN_NODETYPE	100	Uint16
release	SQLF_KTN_RELEASE	101	Uint16

nodetype에 유효한 값(sqlutil.h에 정의):

- SQLF_NT_STANDALONE (0)
- SQLF_NT_SERVER (1)
- SQLF_NT_REQUESTOR (2)
- SQLF_NT_STAND_REQ (3)
- SQLF_NT_MPP (4)
- SQLF_NT_SATELLITE (5)

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqlfupd
{
   unsigned short  token;
   char *ptrvalue;
};
```

COBOL 구조

```
* File: sqlutil.cbl
01 SQL-FUPD.
    05 SQL-TOKEN             PIC 9(4) COMP-5.
    05 FILLER                PIC X(2).
    05 SQL-VALUE-PTR         USAGE IS POINTER.
*
```

제 191 장 sqllob

이 구조를 사용하여 프로그래밍 언어에서 LOB 데이터 유형을 나타냅니다.

표 63. sqllob 구조의 필드

필드 이름	데이터 유형	설명
length	sqluint32	데이터 매개변수의 길이(바이트)
데이터	char(1)	전달 중인 데이터.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqllob
{
        sqluint32 length;
        char data[1];
};
```

제 192 장 sqlma

SQL 모니터 영역(SQLMA) 구조를 사용하여 데이터베이스 모니터 스냅샷 요청을 데이터베이스 관리 프로그램에 보냅니다. 스냅샷 출력의 크기(바이트)를 계산하는 데도 사용됩니다.

표 64. SQLMA 구조의 필드

필드 이름	데이터 유형	설명
OBJ_NUM	INTEGER	모니터하려는 오브젝트 수.
OBJ_VAR	Array	모니터하려는 오브젝트 설명이 포함된 sqlm_obj_struct 구조의 배열. 배열의 길이는 OBJ_NUM로 판별합니다.

표 65. SQLM-OBJ-STRUCT 구조의 필드

필드 이름	데이터 유형	설명
AGENT_ID	INTEGER	모니터하려는 응용프로그램의 응용프로그램 핸들. OBJ_TYPE에 agent_id(응용프로그램 핸들)가 필요한 경우에만 지정합니다. 전체 콜렉션 정보가 포함된 Health 스냅샷을 검색하려면 이 필드에 SQLM_HMON_OPT_COLL_FULL을 지정하십시오. **주:** SQLM_HMON_OPT_COLL_FULL 값이 사용되지 않아서 버전 9.7에 Health Monitor가 사용되지 않으므로 추후 릴리스에서 제거될 수 있습니다.
OBJ_TYPE	INTEGER	모니터하려는 오브젝트 유형
OBJECT	CHAR(128)	모니터하려는 오브젝트 이름. OBJ_TYPE에 appl_id 또는 데이터베이스 별명과 같은 이름이 필요한 경우에만 지정합니다.

OBJ_TYPE에 유효한 값(include 디렉토리에 있는 sqlmon 헤더 파일에 정의)은 다음과 같습니다.

SQLMA_DB2

인스턴스 관련 정보.

SQLMA_DBASE

특정 데이터베이스에 대해 데이터베이스 관련 정보. SQLMA_DBASE 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_APPL

제공된 응용프로그램 ID에 일치하는 응용프로그램의 응용프로그램 정보. SQLMA_APPL 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 응용프로그램 ID를 제공해야 합니다.

SQLMA_AGENT_ID

제공된 에이전트 ID에 일치하는 응용프로그램의 응용프로그램 정보. SQLMA_AGENT_ID 값을 사용하는 경우에는 sqlm_obj_struct 구조의 agent_id 매개변수에 에이전트 ID를 제공해야 합니다.

SQLMA_DBASE_TABLES

특정 데이터베이스의 테이블 정보. SQLMA_DBASE_TABLES 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_DBASE_APPLS

특정 데이터베이스에 연결된 모든 응용프로그램의 응용프로그램 정보. SQLMA_DBASE_APPLS 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_DBASE_APPLINFO

특정 데이터베이스 연결에 대한 요약 응용프로그램 정보.
SQLMA_DBASE_APPLINFO 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_DBASE_LOCKS

특정 데이터베이스의 잠금 보유 목록. SQLMA_DBASE_LOCKS 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_APPL_LOCKS

응용프로그램에서 보유하는 잠금 목록으로 일치하는 응용프로그램 ID 포함. SQLMA_APPL_LOCKS 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 응용프로그램 ID를 제공해야 합니다.

SQLMA_APPL_LOCKS_AGENT_ID

응용프로그램에서 보유하는 잠금 목록으로 일치하는 에이전트 ID 포함. SQLMA_APPL_LOCKS_AGENT_ID 값을 사용하는 경우에는 sqlm_obj_struct 구조의 agent_id 매개변수에 에이전트 ID를 제공해야 합니다.

SQLMA_DBASE_ALL

인스턴스의 모든 사용 중인 데이터베이스에 대한 데이터베이스 정보.

SQLMA_APPL_ALL

인스턴스의 모든 데이터베이스 연결에 대한 응용프로그램 정보.

SQLMA_APPLINFO_ALL

인스턴스의 모든 연결에 대한 요약 응용프로그램 정보.

SQLMA_DCS_APPLINFO_ALL

인스턴스의 데이터베이스 연결 서비스(DCS) 연결 목록

SQLMA_DYNAMIC_SQL

특정 데이터베이스에 대한 동적 SQL 문 정보. SQLMA_DYNAMIC_SQL 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_DCS_DBASE

특정 데이터베이스 연결 서비스(DCS) 데이터베이스에 대한 정보. SQLMA_DCS_DBASE 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_DCS_DBASE_ALL

모든 사용 중인 데이터베이스 연결 서비스(DCS) 데이터베이스에 대한 정보.

SQLMA_DCS_APPL_ALL

모든 연결에 대한 데이터베이스 연결 서비스(DCS) 응용프로그램 정보.

SQLMA_DCS_APPL

제공된 응용프로그램 ID에 일치하는 응용프로그램에 대한 데이터베이스 연결 서비스(DCS) 응용프로그램 정보. SQLMA_DCS_APPL 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 응용프로그램 ID를 제공해야 합니다.

SQLMA_DCS_APPL_HANDLE

제공된 에이전트 ID에 일치하는 응용프로그램에 대한 데이터베이스 연결 서비스(DCS) 응용프로그램 정보. SQLMA_DCS_APPL_HANDLE 값을 사용하는 경우에는 sqlm_obj_struct 구조의 agent_id 매개변수에 에이전트 ID를 제공해야 합니다.

SQLMA_DCS_DBASE_APPLS

특정 데이터베이스에 대한 모든 활성 연결에 대한 데이터베이스 연결 서비스(DCS) 응용프로그램 정보. SQLMA_DCS_DBASE_APPLS 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_DBASE_TABLESPACES

특정 데이터베이스에 대한 테이블 스페이스 정보.
SQLMA_DBASE_TABLESPACES 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_DBASE_BUFFERPOOLS

특정 데이터베이스의 버퍼 풀 정보. SQLMA_DBASE_BUFFERPOOLS 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_BUFFERPOOLS_ALL

모든 버퍼 풀에 대한 정보.

SQLMA_DBASE_REMOTE

특정 페더레이티드 데이터베이스에 대한 리모트 액세스 정보. SQLMA_DBASE_REMOTE 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_DBASE_REMOTE_ALL

모든 페더레이티드 데이터베이스에 대한 리모트 액세스 정보.

SQLMA_DBASE_APPLS_REMOTE

특정 페더레이티드 데이터베이스에 연결된 응용프로그램에 대한 리모트 액세스 정보. SQLMA_DBASE_APPLS_REMOTE 값을 사용하는 경우에는 sqlm_obj_struct 구조의 object 매개변수에 데이터베이스 이름을 제공해야 합니다.

SQLMA_APPL_REMOTE_ALL

모든 응용프로그램에 대한 리모트 액세스 정보.

API 및 데이터 구조 구문

```
typedef struct sqlma
{
   sqluint32 obj_num;
   sqlm_obj_struct obj_var[1];
}sqlma;

typedef struct sqlm_obj_struct
{
   sqluint32 agent_id;
   sqluint32 obj_type;
   _SQLOLDCHAR object[SQLM_OBJECT_SZ];
}sqlm_obj_struct;
```

COBOL 구조

```
* File: sqlmonct.cbl
01 SQLMA.
    05 OBJ-NUM                 PIC 9(9) COMP-5.
    05 OBJ-VAR OCCURS 0 TO 100 TIMES DEPENDING ON OBJ-NUM.
        10 AGENT-ID             PIC 9(9) COMP-5.
        10 OBJ-TYPE             PIC 9(9) COMP-5.
        10 OBJECT               PIC X(128).
*
```

제 193 장 sqlopt

이 구조를 사용하여 바인드 옵션을 sqlabndx API에 전달하고 프리컴파일 옵션을 sqlaprep API에 전달하며 리바인드 옵션을 sqlarbnd API에 전달합니다.

표 66. SQLOPT 구조의 필드

필드 이름	데이터 유형	설명
HEADER	Structure	sqloptheader 구조
OPTION	Array	sqloptions 구조의 배열. 이 배열의 요소 수는 헤더의 할당된 필드 값으로 판별됩니다.

표 67. SQLOPTHEADER 구조의 필드

필드 이름	데이터 유형	설명
ALLOCATED	INTEGER	sqlopt 구조의 옵션 배열에 있는 요소 수
USED	INTEGER	실제로 사용된 sqlopt 구조의 옵션 배열에 있는 요소 수. 이는 입력된 옵션 쌍(TYPE 및 VAL)의 수입니다.

표 68. SQLOPTIONS 구조의 필드

필드 이름	데이터 유형	설명
TYPE VAL	INTEGER INTEGER	바인드/프리컴파일/리바인드 옵션 유형 바인드/프리컴파일/리바인드 옵션 값.

주: TYPE 및 VAL 필드는 지정된 각 바인드, 프리컴파일 또는 리바인드에 대해 반복됩니다.

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqlopt
{
   SQL_STRUCTURE sqloptheader header;
   SQL_STRUCTURE sqloptions   option[1];
};

SQL_STRUCTURE sqloptheader
{
   sqluint32 allocated;
   sqluint32 used;
};

SQL_STRUCTURE sqloptions
{
   sqluint32 type;
   sqluintptr val;
};
```

COBOL 구조

```
* File: sql.cbl
01 SQLOPT.
    05 SQLOPTHEADER.
        10 ALLOCATED   PIC 9(9) COMP-5.
        10 USED        PIC 9(9) COMP-5.
    05 SQLOPTIONS OCCURS 1 TO 50 DEPENDING ON ALLOCATED.
        10 SQLOPT-TYPE        PIC 9(9) COMP-5.
        10 SQLOPT-VAL         PIC 9(9) COMP-5.
        10 SQLOPT-VAL-PTR     REDEFINES SQLOPT-VAL
*
```

제 194 장 SQLU_LSN

이 통합에는 로그 시퀀스 번호의 정의가 포함됩니다. 로그 시퀀스 번호(LSN)는 데이터베이스 로그 내에서 상대 바이트 주소를 나타냅니다. 모든 로그 레코드는 이 번호로 식별됩니다. LSN은 데이터베이스 로그 시작으로부터의 로그 레코드 바이트 오프셋을 나타냅니다.

표 69. SQLU-LSN 통합의 필드

필드 이름	데이터 유형	설명
lsnChar	UNSIGNED CHAR의 배열	6 구성원 문자 배열 로그 시퀀스 번호를 지정합니다.
lsnWord	UNSIGNED SHORT의 배열	3 구성원 short 배열 로그 시퀀스 번호를 지정합니다.

주: SQLU_LSN 구조는 db2LSN 구조로 교체되었습니다.

API 및 데이터 구조 구문

```
typedef union SQLU_LSN
{
   unsigned char   lsnChar[6];
   unsigned short  lsnWord[3];
} SQLU_LSN;
```

제 195 장 sqlu_media_list

이 구조는 db2Load API에 정보를 전달하는 데 사용됩니다.

표 70. SQLU-MEDIA-LIST 구조의 필드

필드 이름	데이터 유형	설명
MEDIA_TYPE	CHAR(1)	미디어 유형을 표시하는 문자.
SESSIONS	INTEGER	이 구조의 대상 필드에서 지시하는 배열의 요소 수를 나타냅니다.
TARGET	단위	이 필드는 구조의 네 유형 중 하나에 대한 포인터입니다. 지시된 구조 유형은 media_type 필드 값으로 판별됩니다. 이 필드에 입력해야 하는 내용에 대한 자세한 정보는 해당 API를 참조하십시오.
FILLER	CHAR(3)	메모리에서 데이터 구조를 적절하게 맞추는 데 사용되는 필러

표 71. SQLU-MEDIA-LIST-TARGETS 구조의 필드

필드 이름	데이터 유형	설명
MEDIA	Pointer	sqlu_media_entry 구조의 포인터
VENDOR	Pointer	sqlu_vendor 구조의 포인터
LOCATION	Pointer	sqlu_location_entry 구조의 포인터
PSTATEMENT	Pointer	sqlu_statement_entry 구조의 포인터
PREMOTEFETCH	Pointer	sqlu_remotefetch_entry 구조의 포인터

표 72. SQLU-MEDIA-ENTRY 구조의 필드

필드 이름	데이터 유형	설명
RESERVE_LEN MEDIA_ENTRY	INTEGER CHAR(215)	media_entry 필드 길이. C 이외의 언어용. 백업 및 리스토어 유틸리티에서 사용되는 백업 이미지의 경로

표 73. SQLU-VENDOR 구조의 필드

필드 이름	데이터 유형	설명
RESERVE_LEN1	INTEGER	shr_lib 필드 길이. C 이외의 언어용.

표 73. SQLU-VENDOR 구조의 필드 (계속)

필드 이름	데이터 유형	설명
SHR_LIB	CHAR(255)	데이터 저장 또는 검색을 위해 벤더에서 제공하는 공유 라이브러리 이름
RESERVE_LEN2	INTEGER	파일 이름 필드 길이. C 이외의 언어용.
FILENAME	CHAR(255)	공유 라이브러리 사용 시에 로드 입력 소스를 식별하는 파일 이름

표 74. SQLU-LOCATION-ENTRY 구조의 필드

필드 이름	데이터 유형	설명
RESERVE_LEN	INTEGER	location_entry 필드 길이. C 이외의 언어용.
LOCATION_ENTRY	CHAR(256)	로드 유틸리티의 입력 데이터 파일 이름

표 75. SQLU-STATEMENT-ENTRY 구조의 필드

필드 이름	데이터 유형	설명
LENGTH	INTEGER	데이터 필드 길이
PDATA	Pointer	SQL 쿼리의 포인터

표 76. SQLU-REMOTEFETCH-ENTRY 구조의 필드

필드 이름	데이터 유형	설명
pDatabaseName	Pointer	소스 데이터베이스 이름
iDatabaseNameLen	INTEGER	소스 데이터베이스 이름 길이.
pUserID	Pointer	사용자 ID의 포인터
iUserIDLen	INTEGER	사용자 ID 길이.
pPassword	Pointer	암호의 포인터
iPasswordLen	INTEGER	암호 길이.
pTableSchema	Pointer	소스 테이블 스키마의 포인터
iTableSchemaLen	INTEGER	스키마 길이.
pTableName	Pointer	소스 테이블 이름의 포인터
iTableNameLen	INTEGER	소스 테이블 이름 길이.
pStatement	Pointer	명령문 이름의 포인터
iStatementLen	INTEGER	명령문 길이.
pIsolationLevel	Pointer	분리 레벨의 포인터(디폴트 CS).

MEDIA_TYPE의 유효한 값(sqlutil에 정의)은 다음과 같습니다.

SQLU_LOCAL_MEDIA

로컬 디바이스(테이프, 디스크 또는 디스켓).

SQLU_SERVER_LOCATION
서버 디바이스(테이프, 디스크 또는 디스켓: 로드 전용). piSourceList 매개변수에 대해서만 지정할 수 있습니다.

SQLU_CLIENT_LOCATION
클라이언트 디바이스(파일 또는 Named Pipes). piLobFileList 매개변수의 piSourceList 매개변수에 대해서만 지정할 수 있습니다.

SQLU_SQL_STMT
SQL 쿼리(로드 전용). piSourceList 매개변수에 대해서만 지정할 수 있습니다.

SQLU_TSM_MEDIA
TSM

SQLU_XBSA_MEDIA
XBSA

SQLU_OTHER_MEDIA
벤더 라이브러리

SQLU_REMOTEFETCH
리모트 페치 미디어(로드 전용). piSourceList 매개변수에 대해서만 지정할 수 있습니다.

SQLU_DISK_MEDIA
디스크(벤더 API 전용)

SQLU_DISKETTE_MEDIA
디스켓(벤더 API 전용)

SQLU_NULL_MEDIA
널(NULL)(DB2 데이터베이스에서 내부적으로 생성)

SQLU_TAPE_MEDIA
테이프(로드 전용).

SQLU_PIPE_MEDIA
Named Pipe(벤더 API 전용)

API 및 데이터 구조 구문

```
typedef SQL_STRUCTURE sqlu_media_list
{
   char media_type;
   char filler[3];
   sqlint32 sessions;
   union sqlu_media_list_targets target;
} sqlu_media_list;

union sqlu_media_list_targets
{
   struct sqlu_media_entry *media;
```

```
   struct sqlu_vendor *vendor;
   struct sqlu_location_entry *location;
   struct sqlu_statement_entry *pStatement;
   struct sqlu_remotefetch_entry *pRemoteFetch;
};

typedef SQL_STRUCTURE sqlu_media_entry
{
   sqluint32 reserve_len;
   char media_entry[SQLU_DB_DIR_LEN+1];
} sqlu_media_entry;

typedef SQL_STRUCTURE sqlu_vendor
{
   sqluint32 reserve_len1;
   char shr_lib[SQLU_SHR_LIB_LEN+1];
   sqluint32 reserve_len2;
   char filename[SQLU_SHR_LIB_LEN+1];
} sqlu_vendor;

typedef SQL_STRUCTURE sqlu_location_entry
{
   sqluint32 reserve_len;
   char location_entry[SQLU_MEDIA_LOCATION_LEN+1];
} sqlu_location_entry;

typedef SQL_STRUCTURE sqlu_statement_entry
{
   sqluint32 length;
   char *pEntry;
} sqlu_statement_entry;

typedef SQL_STRUCTURE sqlu_remotefetch_entry
{
   char *pDatabaseName;
   sqluint32 iDatabaseNameLen;
   char *pUserID;
   sqluint32 iUserIDLen;
   char *pPassword;
   sqluint32 iPasswordLen;
   char *pTableSchema;
   sqluint32 iTableSchemaLen;
   char *pTableName;
   sqluint32 iTableNameLen;
   char *pStatement;
   sqluint32 iStatementLen;
   sqlint32 *pIsolationLevel;
   sqluint32 *piEnableParallelism;
} sqlu_remotefetch_entry;
```

COBOL 구조

```
* File: sqlutil.cbl
01 SQLU-MEDIA-LIST.
    05 SQL-MEDIA-TYPE         PIC X.
    05 SQL-FILLER             PIC X(3).
    05 SQL-SESSIONS           PIC S9(9) COMP-5.
    05 SQL-TARGET.
```

```
        10 SQL-MEDIA          USAGE IS POINTER.
        10 SQL-VENDOR         REDEFINES SQL-MEDIA
        10 SQL-LOCATION       REDEFINES SQL-MEDIA
        10 SQL-STATEMENT      REDEFINES SQL-MEDIA
        10 FILLER             REDEFINES SQL-MEDIA
*


* File: sqlutil.cbl
01 SQLU-MEDIA-ENTRY.
    05 SQL-MEDENT-LEN         PIC 9(9) COMP-5.
    05 SQL-MEDIA-ENTRY        PIC X(215).
    05 FILLER                 PIC X.
*


* File: sqlutil.cbl
01 SQLU-VENDOR.
    05 SQL-SHRLIB-LEN         PIC 9(9) COMP-5.
    05 SQL-SHR-LIB            PIC X(255).
    05 FILLER                 PIC X.
    05 SQL-FILENAME-LEN       PIC 9(9) COMP-5.
    05 SQL-FILENAME           PIC X(255).
    05 FILLER                 PIC X.
*


* File: sqlutil.cbl
01 SQLU-LOCATION-ENTRY.
    05 SQL-LOCATION-LEN       PIC 9(9) COMP-5.
    05 SQL-LOCATION-ENTRY     PIC X(255).
    05 FILLER                 PIC X.
*


* File: sqlutil.cbl
01 SQLU-STATEMENT-ENTRY.
    05 SQL-STATEMENT-LEN       PIC 9(9) COMP-5.
    05 SQL-STATEMENT-ENTRY     USAGE IS POINTER.
*
```

제 196 장 SQLU_RLOG_INFO

이 구조에는 db2ReadLog API 및 데이터베이스 로그의 호출 상태에 대한 정보가 포함됩니다.

표 77. SQLU-RLOG-INFO 구조의 필드

필드 이름	데이터 유형	설명
initialLSN	SQLU_LSN	첫 번째 데이터베이스 CONNECT 문 발행 이후 첫 번째 로그 레코드의 LSN 값을 지정합니다. 자세한 정보는 SQLU-LSN을 참조하십시오.
firstReadLSN	SQLU_LSN	첫 번째 로그 레코드 읽기에 대한 LSN 값을 지정합니다.
lastReadLSN	SQLU_LSN	마지막 로그 레코드 읽기에 대한 LSN 값을 지정합니다.
curActiveLSN	SQLU_LSN	현재(사용 중) 로그의 LSN 값을 지정합니다.
logRecsWritten	sqluint32	버퍼에 기록된 로그 레코드 수를 지정합니다.
logBytesWritten	sqluint32	버퍼에 기록된 바이트 수를 지정합니다.

API 및 데이터 구조 구문

```
typedef SQL_STRUCTURE SQLU_RLOG_INFO
{
   SQLU_LSN initialLSN;
   SQLU_LSN firstReadLSN;
   SQLU_LSN lastReadLSN;
   SQLU_LSN curActiveLSN;
   sqluint32 logRecsWritten;
   sqluint32 logBytesWritten;
} SQLU_RLOG_INFO;
```

제 197 장 sqlupi

이 구조를 사용하여 테이블의 분산 맵 및 분산 키와 같은 파티셔닝 정보를 저장합니다.

표 78. SQLUPI 구조의 필드

필드 이름	데이터 유형	설명
PMAPLEN	INTEGER	분산 맵의 길이(바이트). 단일 노드 테이블의 경우 값은 sizeof(SQL_PDB_NODE_TYPE)입니다. 다중 노드 테이블의 경우 값은 SQL_PDB_MAP_SIZE * sizeof(SQL_PDB_NODE_TYPE)입니다.
PMAP	SQL_PDB_NODE_TYPE	분산 맵.
SQLD	INTEGER	사용된 SQLPARTKEY 요소 수. 즉, 분산 키의 키 파트 수
SQLPARTKEY	Structure	분산 키의 분산 컬럼에 대한 설명. 분산 컬럼의 최대 수는 SQL_MAX_NUM_PART_KEYS입니다.

다음 표에는 SQLUPI 데이터 구조의 SQL 데이터 유형 및 길이가 표시됩니다. SQLTYPE 컬럼은 항목의 데이터 유형을 나타내는 숫자 값을 지정합니다.

표 79. SQLUPI 구조의 SQL 데이터 유형 및 길이

데이터 유형	**SQLTYPE (NULL 값이 허용되지 않음)**	**SQLTYPE (NULL 값 허용)**	**SQLLEN**	**AIX**
날짜	384	385	무시	예
시간	388	389	무시	예
시간소인	392	393	무시	예
가변 길이 문자열	448	449	문자열 길이	예
고정 길이 문자열	452	453	문자열 길이	예
Long 문자열	456	457	무시	없음
널(Null)로 종료되는 문자열	460	461	문자열 길이	예
부동 소수점	480	481	무시	예
10진수	484	485	바이트 1 = 정밀도 바이트 2 = 스케일	예
큰 정수(large integer)	496	497	무시	예

표 79. SQLUPI 구조의 SQL 데이터 유형 및 길이 (계속)

데이터 유형	SQLTYPE (NULL 값이 허용되지 않음)	SQLTYPE (NULL 값 허용)	SQLLEN	AIX
작은 정수(small integer)	500	501	무시	예
가변 길이 그래픽 문자열	464	465	2바이트 문자 길이	예
고정 길이 그래픽 문자열	468	469	2바이트 문자 길이	예
Long 그래픽 문자열	472	473	무시	없음

sqlpartkey 데이터 구조 매개변수 설명

sqltype

입력. 분산 키의 데이터 유형

sqllen 입력. 분산 키의 데이터 길이

API 및 데이터 구조 구문

```
SQL_STRUCTURE sqlupi
{
   unsigned short  pmaplen;
   SQL_PDB_NODE_TYPE pmap[SQL_PDB_MAP_SIZE];
   unsigned short  sqld;
   struct sqlpartkey sqlpartkey[SQL_MAX_NUM_PART_KEYS];
};

SQL_STRUCTURE sqlpartkey
{
   unsigned short  sqltype;
   unsigned short  sqllen;
};
```

제 198 장 SQLXA_XID

트랜잭션 API가 이 구조를 사용하여 XA 트랜잭션을 식별합니다. sqlxhfrg, sqlxphcm, sqlxphrl, sqlcspqy 및 db2XaListIndTrans API가 트랜잭션 API 그룹을 대체합니다. 이 API는 인다우트(Indoubt) 트랜잭션 관리에 사용됩니다.

표 80. SQLXA-XID 구조의 필드

필드 이름	데이터 유형	설명
FORMATID	INTEGER	XA 형식 ID
GTRID_LENGTH	INTEGER	전역 트랜잭션 ID 길이.
BQUAL_LENGTH	INTEGER	분기 ID 길이.
DATA	CHAR[128]	BQUAL 및 뒤 공백이 오는 총 128바이트의 GTRID

주: GTRID 및 BQUAL의 최대 크기는 각각 64바이트입니다.

API 및 데이터 구조 구문

```
struct sqlxa_xid_t {
  sqlint32 formatID;
  sqlint32 gtrid_length;
  sqlint32 bqual_length;
  char data[SQLXA_XIDDATASIZE];
  } ;
typedef struct sqlxa_xid_t SQLXA_XID;
```

제 11 부 부록

부록 A. 프리컴파일러 사용자 정의 API

프리컴파일러 사용자 정의 API

다른 응용프로그램 개발 도구를 해당 제품 내에서 직접 DB2에 대한 프리컴파일러 지원을 구현할 수 있도록 하는 문서화된 API 세트. 예를 들어, AIX의 IBM COBOL이 이 인터페이스를 사용합니다. 프리컴파일러 서비스 API 세트에 대한 정보는 다음 웹 사이트에서 PDF 파일, prepapi.pdf로 사용할 수 있습니다.

http://www.ibm.com/software/data/db2/udb/support/manualsv9.html

부록 B. DB2 로그 레코드

DB2 로그 레코드

이 섹션에서는 iFilterOption 입력 값 DB2READLOG_FILTER_ON이 지정된 경우 db2ReadLog API가 리턴하는 DB2 로그 레코드 구조에 대해 설명합니다. 이 값이 사용되는 경우에는 전파된 로그 레코드만 리턴됩니다. 전파된 로그 레코드만 문서화됩니다. 다른 모든 로그 레코드는 IBM의 내부 사용 전용으로 문서화되지 않습니다.

모든 DB2 로그 레코드는 Long 로그 관리 프로그램 헤더로 시작됩니다. 이 헤더는 총 로그 레코드 크기, 로그 레코드 유형 및 트랜잭션 고유의 정보가 포함됩니다. 어카운팅, 통계, 추적 또는 성능 평가에 대한 정보는 포함하지 않습니다. 자세한 정보는 857 페이지의 『로그 관리 프로그램 헤더』의 내용을 참조하십시오.

로그 레코드는 로그 시퀀스 번호(LSN)로 고유하게 식별됩니다. LSN은 로그 레코드의 첫 번째 바이트에 대해 데이터베이스 로그 내에서 관련 바이트 주소를 나타냅니다. 데이터베이스 로그 시작에서 로그 레코드의 오프셋을 표시합니다.

단일 트랜잭션으로 작성된 로그 레코드는 로그 레코드 헤더의 필드로 고유하게 식별됩니다. 고유 트랜잭션 ID는 새 트랜잭션이 시작될 때마다 하나씩 증가되는 6바이트의 필드입니다. 단일 트랜잭션으로 작성된 모든 로그 레코드에는 동일한 ID가 포함됩니다.

트랜잭션이 DATA CAPTURE CHANGES가 설정된 테이블에 대해 작성 가능한 작업을 수행하거나 로그 작성 유틸리티를 호출하는 경우 트랜잭션이 전파 가능으로 표시됩니다. 전파 가능한 트랜잭션에만 전파 가능으로 표시되는 해당 트랜잭션 관리 프로그램 로그 레코드가 있습니다.

표 81. DB2 로그 레코드

유형	레코드 이름	설명
데이터 관리 프로그램	873 페이지의 『테이블 초기화 로그 레코드』	영구적인 새 테이블이 작성됩니다.
데이터 관리 프로그램	875 페이지의 『임포트 바꾸기(절단) 로그 레코드』	교체 활동을 임포트합니다.
데이터 관리 프로그램	875 페이지의 『처음에 로그되지 않은 것 활성화 로그 레코드』	ACTIVATE NOT LOGGED INITIALLY절이 포함되는 테이블 활동을 변경합니다.
데이터 관리 프로그램	876 페이지의 『삽입 롤백 로그 레코드』	행 삽입을 롤백합니다.
데이터 관리 프로그램	876 페이지의 『테이블 재구성 로그 레코드』	REORG가 커미트됩니다.
데이터 관리 프로그램	877 페이지의 『인덱스 작성, 인덱스 삭제 로그 레코드』	인덱스 활동.

표 81. DB2 로그 레코드 (계속)

유형	레코드 이름	설명
데이터 관리 프로그램	877 페이지의 『테이블 작성, 테이블 삭제, 테이블 작성 롤백, 테이블 삭제 롤백 로그 레코드』	테이블 활동.
데이터 관리 프로그램	878 페이지의 『테이블 속성 변경 로그 레코드』	전파, 점검 보류 및 추가 모드 활동.
데이터 관리 프로그램	878 페이지의 『컬럼 추가 테이블 변경, 컬럼 추가 롤백 로그 레코드』	기존 테이블에 컬럼 추가.
데이터 관리 프로그램	879 페이지의 『컬럼 속성 변경 로그 레코드』	컬럼 활동.
데이터 관리 프로그램	879 페이지의 『컬럼 속성 변경 실행 취소 로그 레코드』	컬럼 활동.
데이터 관리 프로그램	880 페이지의 『레코드 삽입, 레코드 삭제 롤백, 레코드 갱신 롤백 로그 레코드』	테이블 레코드 활동.
데이터 관리 프로그램	884 페이지의 『비어 있는 페이지에 레코드 삽입, 페이지를 비우기 위해 레코드 삭제, 페이지를 비우기 위해 레코드 삭제 롤백, 비어 있는 페이지에 레코드 삽입 롤백 로그 레코드』	다차원적으로 클러스터된(MDC) 테이블 활동
데이터 관리 프로그램	885 페이지의 『레코드 갱신 로그 레코드』	스토리지 위치가 변경된 행 갱신.
데이터 관리 프로그램	886 페이지의 『테이블 또는 스키마의 이름 바꾸기 로그 레코드』	테이블 또는 스키마 이름 활동.
데이터 관리 프로그램	886 페이지의 『테이블 또는 스키마의 이름 바꾸기 실행 취소 로그 레코드』	테이블 또는 스키마 이름 활동.
Long 필드 관리 프로그램	868 페이지의 『Long 필드 레코드 로그 레코드 추가/삭제/갱신 안함』	Long 필드 레코드 활동.
트랜잭션 관리 프로그램	859 페이지의 『일반 커미트 로그 레코드』	트랜잭션 커미트.
트랜잭션 관리 프로그램	860 페이지의 『경험적 커미트 로그 레코드』	인다우트(Indoubt) 트랜잭션 커미트.
트랜잭션 관리 프로그램	860 페이지의 『MPP 코디네이터 커미트 로그 레코드』	트랜잭션 커미트. 하나 이상의 종속 요소 노드에서 갱신을 수행하는 응용프로그램에 대한 코디네이터 노드에 기록됩니다.
트랜잭션 관리 프로그램	861 페이지의 『MPP 종속 커미트 로그 레코드』	트랜잭션 커미트. 종속 요소 노드에 기록됩니다.
트랜잭션 관리 프로그램	861 페이지의 『일반 중단 로그 레코드』	트랜잭션 중단.
트랜잭션 관리 프로그램	862 페이지의 『경험적 중단 로그 레코드』	인다우트(Indoubt) 트랜잭션 중단.
트랜잭션 관리 프로그램	862 페이지의 『로컬 보류 목록 로그 레코드』	보류 목록이 포함된 트랜잭션 커미트가 있음.
트랜잭션 관리 프로그램	863 페이지의 『전역 보류 목록 로그 레코드』	보류 목록이 포함된 트랜잭션 커미트(2단계)가 있음.

표 81. DB2 로그 레코드 (계속)

유형	레코드 이름	설명
트랜잭션 관리 프로그램	863 페이지의 『XA 준비 로그 레코드』	2단계 커미트 환경에서 XA 트랜잭션 커미트.
트랜잭션 관리 프로그램	864 페이지의 『MPP 종속 요소 준비 로그 레코드』	2단계 커미트 환경에서 MPP 트랜잭션 커미트. 이 로그 레코드는 종속 요소 노드에만 있습니다.
트랜잭션 관리 프로그램	865 페이지의 『TM 준비 로그 레코드』	데이터베이스가 TM 데이터베이스로 사용되는 2단계 커미트의 파트로 조정된 트랜잭션 준비.
트랜잭션 관리 프로그램	865 페이지의 『백아웃 제거 로그 레코드』	백아웃 제거 구간 종료를 표시합니다. 백아웃 제거 구간은 트랜잭션이 중단된 경우 보상되지 않는 로그 레코드 세트입니다.
트랜잭션 관리 프로그램	866 페이지의 『응용프로그램 정보 로그 레코드』	트랜잭션을 시작한 응용프로그램에 대한 정보.
트랜잭션 관리 프로그램	866 페이지의 『페더레이티드 compensat 로그 레코드』	트랜잭션에 관련된 페더레이티드 자원 관리자의 정보.
유틸리티 관리 프로그램	869 페이지의 『시스템 카탈로그 이주 시작 로그 레코드』	시스템 카탈로그 이주가 시작됩니다.
유틸리티 관리 프로그램	869 페이지의 『시스템 카탈로그 이주 종료 로그 레코드』	시스템 카탈로그 이주가 완료됩니다.
유틸리티 관리 프로그램	869 페이지의 『로드 시작 로그 레코드』	테이블 로드가 시작됩니다.
유틸리티 관리 프로그램	870 페이지의 『백업 종료 로그 레코드』	백업 활동이 완료됩니다.
유틸리티 관리 프로그램	870 페이지의 『테이블 스페이스 롤 포워드 로그 레코드』	테이블 스페이스 롤 포워드가 완료됩니다.
유틸리티 관리 프로그램	870 페이지의 『특정 시점으로 테이블 스페이스 롤 포워드 시작 로그 레코드』	테이블 스페이스 롤 포워드 시작을 특정 시점으로 표시합니다.
유틸리티 관리 프로그램	871 페이지의 『특정 시점으로 테이블 스페이스 롤 포워드 종료 로그 레코드』	테이블 스페이스 롤 포워드 종료를 특정 시점으로 표시합니다.

로그 관리 프로그램 헤더

모든 DB2 로그 레코드는 Long 로그 관리 프로그램 헤더로 시작됩니다. 이 헤더에는 로그 레코드와 로그 레코드 기록기의 트랜잭션 정보를 자세히 설명하는 정보가 포함됩니다.

주: 유형 'i'의 로그 레코드는 정보용 로그 레코드입니다. 이 로그 레코드는 롤 포워드, 롤백 및 응급 복구 중에 DB2에서 무시됩니다.

표 82. 로그 관리 프로그램 로그 레코드 헤더(LogManagerLogRecordHeader)

설명	유형	오프셋(바이트)
전체 로그 레코드의 길이	int	0(4)
로그 레코드의 유형(859 페이지의 표 83 참조)	short	4(2)
로그 레코드 일반 플래그[1]	short	6(2)

표 82. 로그 관리 프로그램 로그 레코드 헤더(LogManagerLogRecordHeader) (계속)

설명	유형	오프셋(바이트)
해당 트랜잭션에 의해 기록된 이전 로그 레코드의 로그 시퀀스 번호. 트랜잭션별로 로그 레코드를 체인으로 연결하기 위해 사용됩니다. 값이 0000 0000 0000 0000이면, 트랜잭션에서 기록된 첫 번째 로그 레코드입니다.	db2LSN[2]	8(8)
고유한 트랜잭션 ID	SQLU_TID[3]	16(6)
보상 중인 로그 레코드 이전에 해당 트랜잭션에 대한 로그 레코드의 로그 시퀀스 번호 (참고: 보상 및 백아웃 해제 로그 레코드의 경우에만)	db2LSN	22(8)
보상 중인 해당 트랜잭션에 대한 로그 레코드의 로그 시퀀스 번호 (참고: 전파 가능 보상 로그 레코드의 경우에만)	db2LSN	30(8)
로그 관리 프로그램 로그 레코드 헤더의 총 길이: • *보상이 아닌 경우: 22바이트* • *보상: 30바이트* • *전파 가능 보상: 38바이트*		

주:

1. 로그 레코드 일반 플래그 상수

```
Redo Always                                              0x0001
Propagatable                                             0x0002
Temp Table                                               0x0004
Tablespace rollforward undo                              0x0008
Singular transaction (no commit/rollback)                0x0010
Conditionally Recoverable                                0x0080
Tablespace rollforward at check constraint process       0x0100
```

2. 로그 시퀀스 번호(LSN)

```
A unique log record identifier representing the relative byte address
   of the log record within the database log.

   db2LSN: { db2Uint64 lsnU64;
                    }
```

3. 트랜잭션 ID(TID)

```
A unique log record identifier representing the transaction.

   SQLU_TID: union { unsigned char  [6] ;
                     unsigned short [3] ;
                     }
```

4. 레코드 ID(RID)

```
A unique number identifying the position of a record.
   RID: Page number  char [4];
   slot number  char [2];
```

표 83. 로그 관리 프로그램 로그 레코드 헤더 로그 유형 값 및 정의

값	정의
0x0041	일반 중단
0x0042	백아웃 해제
0x0043	보상
0x0049	경험적 중단
0x004A	로드 시작
0x004E	일반 로그 레코드
0x004F	백업 종료
0x0051	전역 보류 목록
0x0052	재실행
0x0055	실행 취소
0x0056	시스템 카탈로그 이주 시작
0x0057	시스템 카탈로그 이주 종료
0x0069	정보용
0x006F	백업 시작
0x0071	특정 시점으로 테이블 스페이스 롤 포워드 종료
0x007B	MPP 준비
0x007C	XA 준비
0x007D	트랜잭션 관리 프로그램(TM) 준비
0x0084	일반 커미트
0x0085	MPP 종속 커미트
0x0086	MPP 코디네이터 커미트
0x0087	경험적 커미트
0x0089	특정 시점으로 테이블 스페이스 롤 포워드 시작
0x008A	로컬 보류 목록
0x008B	응용프로그램 정보

트랜잭션 관리 프로그램 로그 레코드

트랜잭션 관리 프로그램은 트랜잭션 이벤트(예: 커미트 또는 롤백)의 완료를 나타내는 로그 레코드를 생성합니다. 로그 레코드의 시간소인은 세계 표준시(UTC)로 되어 있고 1970 1월 1일 이후의 시간(초)을 표시합니다.

일반 커미트 로그 레코드

이 로그 레코드는 단일 노드 환경이나 다중 노드 환경에서 하나의 트랜잭션에 대해 기록됩니다. 트랜잭션은 하나의 노드에만 영향을 줍니다. 로그 레코드는 다음 이벤트 중 하나가 발생한 후에 트랜잭션이 커미트될 때 기록됩니다.

1. 사용자가 COMMIT를 발행했습니다.
2. CONNECT RESET 중에 내재된 커미트가 발생합니다.

표 84. 일반 커미트 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
커미트된 시간 트랜잭션	sqluint64	22 (8)
권한 부여 ID 길이[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	unsigned short	30(2)
응용프로그램의 권한 부여 ID[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	char[ ]	32(변수[2])
총 길이: 32바이트 + 변수 전파 가능(30바이트 전파 불가능)		

주:

1. 로그 레코드가 전파 가능한 것으로 표시되는 경우
2. 권한 부여 ID 길이를 기초로 하는 변수

경험적 커미트 로그 레코드

이 로그 레코드는 인다우트(Indoubt) 트랜잭션이 커미트될 때 기록됩니다.

표 85. 경험적 커미트 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0(22)
커미트된 시간 트랜잭션	sqluint64	22(8)
권한 부여 ID 길이[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	unsigned short	30(2)
응용프로그램의 권한 부여 ID[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	char[ ]	32(변수[2])
총 길이: 32바이트 + 변수 전파 가능(30바이트 전파 불가능)		

주:

1. 로그 레코드가 전파 가능한 것으로 표시되는 경우
2. 권한 부여 ID 길이를 기초로 하는 변수

MPP 코디네이터 커미트 로그 레코드

하나 이상의 종속 요소 노드에서 갱신을 수행하는 응용프로그램에 대한 코디네이터 노드에 기록됩니다.

표 86. MPP 코디네이터 커미트 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0(22)
커미트된 시간 트랜잭션	sqluint64	22(8)
트랜잭션의 MPP ID	SQLP_GXID	30(20)

표 86. MPP 코디네이터 커미트 로그 레코드 구조 (계속)

설명	유형	오프셋(바이트)
최대 노드 번호	unsigned short	50(2)
TNL	unsigned char[ ]	52(최대 노드 번호/8 + 1)
권한 부여 ID 길이[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	unsigned short	변수(2)
응용프로그램의 권한 부여 ID[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	char[ ]	변수(변수[2])
총 길이: 변수		

주:

1. TNL은 트랜잭션에 포함된 코디네이터 노드를 제외하고 노드를 정의합니다.
2. 권한 부여 ID 길이를 기초로 하는 변수

MPP 종속 커미트 로그 레코드

이 로그 레코드는 MPP의 종속 요소 노드에 기록됩니다.

표 87. MPP 종속 요소 커미트 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0(22)
커미트된 시간 트랜잭션	sqluint64	22(8)
트랜잭션의 MPP ID	SQLP_GXID	30(20)
예약됨	unsigned short	50(2)
권한 부여 ID 길이[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	unsigned short	52(2)
응용프로그램의 권한 부여 ID[2](로그 레코드가 전파 가능한 것으로 표시된 경우)	char[ ]	54(변수[3])
총 길이: 54바이트 + 변수		

주:

1. 이는 트랜잭션이 하나의 데이터베이스 파티션에만 있는 경우 현재 데이터베이스 파티션 번호입니다. 그렇지 않으면 코디네이터 파티션 번호입니다.
2. 로그 레코드가 전파 가능한 것으로 표시되는 경우
3. 권한 부여 ID 길이를 기초로 하는 변수

일반 중단 로그 레코드

이 로그 레코드는 다음 이벤트 중 하나가 발생한 후에 트랜잭션이 중단될 때 기록됩니다.

- 사용자가 ROLLBACK을 발행했습니다.
- 교착 상태가 발생합니다.
- 응급 복구 중에 내재된 롤백이 발생합니다.
- ROLLFORWARD 복구 중에 내재된 롤백이 발생합니다.

표 88. 일반 중단 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0(22)
권한 부여 ID 길이[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	unsigned short	22(2)
응용프로그램의 권한 부여 ID[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	char[ ]	24(변수[2])
총 길이: 24바이트 + 변수 전파 가능(22바이트 전파 불가능)		

주:

1. 로그 레코드가 전파 가능한 것으로 표시되는 경우
2. 권한 부여 ID 길이를 기초로 하는 변수

경험적 중단 로그 레코드

이 로그 레코드는 인다우트(Indoubt) 트랜잭션이 중단될 때 기록됩니다.

표 89. 경험적 중단 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0(22)
권한 부여 ID 길이[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	unsigned short	22(2)
응용프로그램의 권한 부여 ID[1](로그 레코드가 전파 가능한 것으로 표시된 경우)	char[ ]	24(변수[2])
총 길이: 24바이트 + 변수 전파 가능(22바이트 전파 불가능)		

주:

1. 로그 레코드가 전파 가능한 것으로 표시되는 경우
2. 권한 부여 ID 길이를 기초로 하는 변수

로컬 보류 목록 로그 레코드

이 로그 레코드는 트랜잭션이 커미트되고 보류 목록이 존재하는 경우에 기록됩니다. 보류 목록은 사용자/응용프로그램이 COMMIT를 발행할 때만 수행할 수 있는 복구 불가능한 조작(예: 파일 삭제)의 링크 목록입니다. 변수 길이 구조에는 보류 목록 항목이 포함됩니다.

표 90. 로컬 보류 목록 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0(22)
커미트된 시간 트랜잭션	sqluint64	22(8)
권한 부여 ID 길이[1]	unsigned short	30(2)
응용프로그램의 권한 부여 ID[1]	char[ ]	32(변수)[2]
보류 목록 항목	variable	변수(변수)
총 길이: 32바이트 + 변수 전파 가능(30바이트 + 보류 목록 항목 전파 불가능)		

주:

1. 로그 레코드가 전파 가능한 것으로 표시되는 경우
2. 권한 부여 ID 길이를 기초로 하는 변수

전역 보류 목록 로그 레코드

이 로그 레코드는 2단계 커미트에 포함된 트랜잭션이 커미트되는 경우에 기록됩니다. 보류 목록은 사용자/응용프로그램이 COMMIT를 발행할 때만 수행할 수 있는 복구 불가능한 조작(예: 파일 삭제)이 포함됩니다. 변수 길이 구조에는 보류 목록 항목이 포함됩니다.

표 91. 전역 보류 목록 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
권한 부여 ID 길이[1]	unsigned short	22(2)
응용프로그램의 권한 부여 ID[1]	char[ ]	24(변수)[2]
전역 보류 목록 항목	variable	변수(변수)
총 길이: 24바이트 + 변수 전파 가능(22바이트 + 보류 목록 항목 전파 불가능)		

주:

1. 로그 레코드가 전파 가능한 것으로 표시되는 경우
2. 권한 부여 ID 길이를 기초로 하는 변수

XA 준비 로그 레코드

이 로그 레코드는 단일 노드 환경이나 MPP의 코디네이터 노드에서 XA 트랜잭션에 대해 기록됩니다. XA 응용프로그램의 경우에만 사용됩니다. 로그 레코드는 트랜잭션의 준비를 2단계 커미트의 일부로 표시하기 위해 기록됩니다. XA 준비 로그 레코드는 트랜잭션을 시작한 응용프로그램을 설명하며 인다우트(Indoubt) 트랜잭션을 재작성하기 위해 사용됩니다.

표 92. XA 준비 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
준비된 시간 트랜잭션	sqluint64	22 (8)
트랜잭션에서 사용되는 로그 스페이스	sqluint64	30(8)
트랜잭션 노드 목록 크기	sqluint32	38(4)
트랜잭션 노드 목록	unsigned char[ ]	42(변수)
예약	sqluint32	변수(2)
트랜잭션의 XA ID	SQLXA_XID[1]	변수(140)
Synclog 정보	variable	변수(변수)
총 길이: 184바이트 + 변수		

주: 1. SQLXA_XID 로그 레코드 유형에 대한 세부사항은 849 페이지의 제 198 장 『SQLXA_XID』의 설명을 참조하십시오.

MPP 종속 요소 준비 로그 레코드

이 로그 레코드는 종속 요소 노드의 MPP 트랜잭션에 대해 기록됩니다. 로그 레코드는 트랜잭션의 준비를 2단계 커미트의 일부로 표시하기 위해 기록됩니다. MPP 종속 요소 준비 로그 레코드는 트랜잭션을 시작한 응용프로그램을 설명하며 인다우트(Indoubt) 트랜잭션을 재작성하기 위해 사용됩니다.

표 93. MPP 종속 요소 준비 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
준비된 시간 트랜잭션	sqluint64	22 (8)
트랜잭션에서 사용되는 로그 스페이스	sqluint64	30(8)
코디네이터 LSN	db2LSN	38(8)
채우기	char[ ]	46(2)
트랜잭션의 MPP ID	SQLP_GXID[1]	48(20)
총 길이: 68바이트		

주: 1. SQLP-GXID 로그 레코드는 MPP 환경에서 트랜잭션을 식별하기 위해 사용됩니다.

표 94. SQLP-GXID 구조의 필드

필드 이름	데이터 유형	설명
FORMATID	INTEGER	GXID 형식 ID
GXID_LENGTH	INTEGER	GXID 길이
BQAL_LENGTH	INTEGER	분기 ID의 길이

표 94. SQLP-GXID 구조의 필드 (계속)

필드 이름	데이터 유형	설명
DATA	CHAR(8)	처음 2바이트에는 노드 번호가 포함되고, 나머지는 트랜잭션 ID입니다.

TM 준비 로그 레코드

이 로그 레코드는 단일 파티션 데이터베이스 환경이나 MPP의 코디네이터 파티션에서 DB2 조정 데이터베이스에 대해 기록됩니다. 로그 레코드는 트랜잭션의 준비를 2단계 커미트의 일부로 표시하기 위해 기록됩니다.

표 95. TM 준비 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
준비된 시간 트랜잭션	sqluint64	22 (8)
트랜잭션에서 사용되는 로그 스페이스	sqluint64	30(8)
트랜잭션 노드 목록 크기	sqluint32	38(4)
트랜잭션 노드 목록	unsigned char[ ]	42(변수)
예약	sqluint32	변수(2)
트랜잭션의 XA ID	SQLXA_XID	변수(140)
Synclog 정보	variable	변수(변수)
총 길이: 184바이트 + 변수		

백아웃 제거 로그 레코드

이 로그 레코드는 백아웃 제거 구간의 끝을 표시하기 위해 사용됩니다. 백아웃 제거 구간은 트랜잭션이 중단된 경우 보상되지 않는 로그 레코드 세트입니다. 이 로그 레코드에는 8바이트 로그 시퀀스 번호(오프셋 22에서 시작하여 로그 레코드 헤더에 저장되는 *complsn*)가 포함됩니다. 특정 시나리오에서는 백아웃 제거 로그 레코드에 해당되는 데이터 관리 프로그램 로그 레코드에 로그된 데이터와 동일한 로그 데이터(오프셋 30에서 시작)도 포함됩니다. 롤백 시(중단된 트랜잭션 다음에) 이 로그 레코드를 읽을 때, *complsn*은 보상할 다음 로그 레코드를 표시합니다.

표 96. 백아웃 제거 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
Complsn	db2LSN	22 (8)
로그 데이터[1]	variable	variable
총 길이: 30바이트 + 변수		

주: 1. 특정 시나리오에만 적용되며 사용되는 경우 로그 헤더에서 전체 로그 레코드의 길이는 28바이트를 초과합니다.

응용프로그램 정보 로그 레코드

이 로그 레코드에는 해당 트랜잭션을 시작한 응용프로그램에 대한 정보가 포함됩니다.

표 97. 응용프로그램 정보 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
트랜잭션 시작 시간	sqluint32	22 (4)
예약됨	char[ ]	26(16)
코드 페이지	sqluint32	42(4)
응용프로그램 이름 길이	sqluint32	46(4)
응용프로그램 이름	char[ ]	50(변수)
응용프로그램 ID 길이	sqluint32	변수(4)
응용프로그램 ID	char[ ]	변수(변수)
시퀀스 번호 길이	sqluint32	변수(4)
시퀀스 번호	char[ ]	변수(변수)
클라이언트 길이에 사용되는 데이터베이스 별명	sqluint32	변수(4)
클라이언트에 사용되는 데이터베이스 별명	char[ ]	변수(변수)
권한 부여 ID 길이	sqluint32	변수(4)
권한 부여 ID	char[ ]	변수(변수)
총 길이: 66바이트 + 변수		

페더레이티드 compensat 로그 레코드

이 로그 레코드에는 트랜잭션에 포함된 페더레이티드 자원 관리 프로그램에 대한 정보가 포함됩니다.

표 98. 페더레이티드 준비 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
자원 관리 프로그램 수	sqluint32	22 (4)
권한 부여 ID 길이	sqluint16	26(2)
암호화된 암호 길이	sqluint16	28(2)
권한 부여 ID	char[128]	30(128)
암호화된 암호	char[255]	158(255)
자원 관리 프로그램 항목	variable	413(변수)
총 길이: 413바이트 + 변수		

Long 필드 관리 프로그램 로그 레코드

Long 필드 레코드 로그 레코드는 LOG RETAIN 또는 USEREXITS를 사용할 수 있도록 데이터베이스가 구성된 경우에만 기록됩니다. Long 필드 데이터가 삽입, 삭제 또는 갱신될 때마다 기록됩니다.

주: LOB 관리 프로그램 로그 레코드는 전파할 수 없기 때문에 문서화되지 않습니다.

로그 스페이스를 유지하기 위해 테이블에 삽입된 Long 필드 데이터는 데이터베이스가 순환 로깅하도록 구성된 경우 로깅되지 않습니다. 또한, Long 필드 값이 갱신되면 사전 이미지가 쉐도우되어 로깅되지 않습니다.

모든 Long 필드 레코드 로그 레코드는 헤더로 시작됩니다.

모든 Long 필드 관리 프로그램 로그 레코드 오프셋은 로그 관리 프로그램 로그 레코드 헤더 끝에서 옵니다.

LONG VARCHAR OR LONG VARGRAPHIC 컬럼을 캡처하도록 테이블이 수정된 경우(ALTER TABLE 문에서 INCLUDE LONGVAR COLUMNS 지정):

- Long 필드 관리 프로그램이 해당 Long 필드 로그 레코드를 기록합니다.
- Long 필드 데이터가 갱신되면 뒤에 새 값이 삽입되어 갱신사항이 이전 Long 필드 값을 삭제하는 것처럼 처리됩니다. Long 필드 레코드 삭제/추가 여부가 테이블의 갱신 조작과 연관되어 있는지 판별하기 위해 원래 조작 값이 Long 필드 관리 프로그램 로그 레코드에 로그됩니다.
- Long 필드 컬럼이 포함된 테이블이 갱신되지만 Long 필드 컬럼 자체는 갱신되지 않는 경우 Long 필드 레코드 갱신 안함이 기록됩니다.
- Long 필드 레코드 삭제 및 Long 필드 레코드 갱신 안함은 정보 제공을 위한 로그 레코드입니다.

표 99. Long 필드 관리 프로그램 로그 레코드 헤더(LongFieldLogRecordHeader)

설명	유형	오프셋(바이트)
시작자 코드(구성요소 ID = 3)	unsigned char	0 (1)
조작 유형(868 페이지의 표 100 참조.)	unsigned char	1 (1)
테이블 스페이스 ID	unsigned short	2 (2)
오브젝트 ID	unsigned short	4 (2)
상위 테이블 스페이스 ID[1]	unsigned short	6 (2)
상위 오브젝트 ID[2]	unsigned short	8 (2)
총 길이: 10바이트		

주:

1. 데이터 오브젝트의 테이블 스페이스 ID
2. 데이터 오브젝트의 오브젝트 ID

표 100. Long 필드 관리 프로그램 로그 레코드 헤더 조작 유형 값 및 정의

값	정의
113	Long 필드 레코드 추가
114	Long 필드 레코드 삭제
115	Long 필드 레코드 갱신 안함

Long 필드 레코드 로그 레코드 추가/삭제/갱신 안함

이 로그 레코드는 Long 필드 데이터가 삽입, 삭제 또는 갱신될 때마다 기록됩니다. 데이터 길이는 다음 512바이트 경계로 반올림됩니다.

표 101. Long 필드 레코드 로그 레코드 구조 추가/삭제/갱신 안함

설명	유형	오프셋(바이트)
로그 헤더	LongFieldLogRecordHeader	0 (10)
내부	Internal	10 (1)
원래 조작 유형[1]	char	11 (1)
컬럼 ID[2]	unsigned short	12 (2)
Long 필드 길이[3]	unsigned short	14 (2)
파일 오프셋[4]	sqluint32	16 (4)
Long 필드 데이터	char[ ]	20 (변수)

주:

1. 원래 조작 유형

   ```
   1     Insert
   2     Delete
   4     Update
   ```

2. 로그 레코드가 적용되는 컬럼 번호. 컬럼 번호는 0부터 시작됩니다.
3. 512바이트 섹터의 Long 필드 데이터 길이(실제 데이터 길이는 Long 필드 디스크립터(LF 디스크립터)의 처음 4바이트로 기록되고, 이는 다음 삽입/삭제/갱신 로그 레코드에 형식화된 사용자 데이터 레코드의 일부로 로그됨). 이 필드 값은 항상 양수입니다.

   Long 필드 관리 프로그램은 삽입, 삭제 또는 갱신되는 로그 레코드의 길이를 0 길이의 Long 필드 데이터로 기록하지 않습니다.
4. 데이터가 있는 Long 필드 오브젝트에 512바이트 섹터 오프셋.

유틸리티 관리 프로그램 로그 레코드

유틸리티 관리 프로그램은 다음 DB2 유틸리티에 연관된 로그 레코드를 작성합니다.

- 데이터베이스 업그레이드
- 로드
- 백업

- 테이블 스페이스 롤 포워드

로그 레코드는 요청된 활동의 시작이나 끝을 표시합니다. 이러한 유틸리티에 대해 전파 가능한 로그 레코드만 문서화됩니다.

시스템 카탈로그 이주 시작 로그 레코드

데이터베이스 업그레이드 중에 시스템 카탈로그 오브젝트는 새 릴리스 형식으로 변환됩니다. 이 로그 레코드는 시스템 카탈로그 이주 시작을 나타냅니다.

표 102. 시스템 카탈로그 이주 시작 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
시작 시간	char[ ]	22 (10)
이전 릴리스	unsigned short	32 (2)
새 릴리스	unsigned short	34 (2)
총 길이: 36바이트		

시스템 카탈로그 이주 종료 로그 레코드

데이터베이스 업그레이드 중에 시스템 카탈로그 오브젝트는 새 릴리스 형식으로 변환됩니다. 이 로그 레코드는 시스템 카탈로그 이주의 성공적인 완료를 나타냅니다.

표 103. 시스템 카탈로그 이주 종료 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
종료 시간	char[ ]	22 (10)
새 릴리스	unsigned short	32 (2)
총 길이: 34바이트		

로드 시작 로그 레코드

이 로그 레코드는 로드 시작과 연관됩니다.

이 레코드는 전파 가능한 로드 로그 레코드 전용입니다.

로그 레코드 전파가 목적인 경우 로그 시작 로그 레코드를 읽은 후에 특정 테이블의 로그 레코드를 목표 테이블에 전파하지 않는 것이 좋습니다. 로드 시작 로그 레코드 이후에 로드 중인 테이블에 속한 모든 전파 가능한 로그 레코드는 콜드 스타트가 시작되는 시점까지 트랜잭션 경계에 상관 없이 무시할 수 있습니다. 콜드 스타트는 소스와 목표 테이블을 동기화하는 데 필요합니다.

표 104. 로드 시작 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)

표 104. 로드 시작 로그 레코드 구조 (계속)

설명	유형	오프셋(바이트)
로그 레코드 ID	sqluint32	22 (4)
풀 ID	unsigned short	26 (2)
오브젝트 ID	unsigned short	28 (2)
플래그	sqluint32	30 (4)
오브젝트 풀 목록	variable	34 (변수)
총 길이: 34바이트 더하기 변수		

백업 종료 로그 레코드

이 로그 레코드는 성공적인 백업 종료와 연관됩니다.

표 105. 백업 종료 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
백업 종료 시간	sqluint64	22 (8)
총 길이: 30바이트		

테이블 스페이스 롤 포워드 로그 레코드

이 로그 레코드는 테이블 스페이스 ROLLFORWARD 복구와 연관됩니다. 이는 성공적으로 롤 포워드된 각 테이블 스페이스에 대해 기록됩니다.

표 106. 테이블 스페이스 롤 포워드 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
테이블 스페이스 ID	sqluint32	22 (4)
총 길이: 26바이트		

특정 시점으로 테이블 스페이스 롤 포워드 시작 로그 레코드

이 로그 레코드는 테이블 스페이스 ROLLFORWARD 복구와 연관됩니다. 이는 테이블 스페이스 롤 포워드 시작을 특정 시점으로 표시합니다.

표 107. 특정 시점으로 테이블 스페이스 롤 포워드 시작 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
이 로그 레코드의 시간소인	sqluint64	22 (8)
테이블 스페이스가 롤 포워드 중인 시간소인	sqluint32	30 (4)
롤 포워드 중인 풀 수	sqluint32	34 (4)
총 길이: 38바이트		

특정 시점으로 테이블 스페이스 롤 포워드 종료 로그 레코드

이 로그 레코드는 테이블 스페이스 ROLLFORWARD 복구와 연관됩니다. 이는 테이블 스페이스 롤 포워드 종료를 특정 시점으로 표시합니다.

표 108. 특정 시점으로 테이블 스페이스 롤 포워드 종료 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	LogManagerLogRecordHeader	0 (22)
이 로그 레코드의 시간소인	sqluint64	22 (8)
테이블 스페이스가 롤 포워드된 시간소인	sqluint32	30 (4)
롤 포워드가 성공인 경우 값이 TRUE이거나 롤 포워드가 취소된 경우 FALSE인 플래그.	sqluint32	34 (4)
총 길이: 38바이트		

이벤트 로그와 이벤트 타이밍이 구분될 수 있도록 적절한 정밀도를 제공하는 데 두 개의 필드가 필요합니다. 첫 번째 시간소인에서는 8바이트를 사용하여 로그가 작성된 시간을 초 단위의 정밀도로 표시합니다. 이 시간소인의 처음 4바이트는 초 분할 영역을 나타냅니다. 1초 동안 여러 조치가 발생될 수 있기 때문에 이벤트 순서를 이해하려면 더 자세한 정밀도가 필요합니다. 두 번째 시간소인 필드는 나노초를 나타내는 데 사용되는 4바이트를 제공합니다. 두 로그 레코드의 로그 레코드 시간소인이 동일한 경우 추가 4바이트 시간소인 필드를 사용하여 관련 로그 이벤트 순서를 판별할 수 있습니다.

데이터 관리 프로그램 로그 레코드

데이터 관리 프로그램 로그 레코드는 DDL, DML 또는 유틸리티 활동의 결과입니다.

데이터 관리 프로그램 로그 레코드에는 다음 두 가지 유형이 있습니다.

- DMS(Data Management System) 로그는 헤더에 1 구성요소 ID를 가지고 있습니다.
- DOM(Data Object Manager) 로그는 헤더에 4 구성요소 ID를 가지고 있습니다.

표 109. DMS 로그 레코드 헤더 구조(DMSLogRecordHeader)

설명	유형	오프셋(바이트)
구성요소 ID(=1)	unsigned char	0(1)
함수 ID(872 페이지의 표 110 참조)	unsigned char	1(1)
테이블 ID		
테이블 스페이스 ID	unsigned short	2(2)
테이블 ID	unsigned short	4(2)
총 길이: 6바이트		

표 110. DMS 로그 레코드 헤더 구조 함수 ID 값 및 정의

값	정의
102	테이블에 컬럼 추가
104	컬럼 추가 실행 취소
108	파티션 상태 갱신 실행 취소
110	레코드 삽입 실행 취소
111	레코드 삭제 실행 취소
112	레코드 갱신 실행 취소
113	컬럼 변경
115	컬럼 변경 실행 취소
122	스키마 또는 테이블 이름 바꾸기
123	스키마 또는 테이블 이름 바꾸기 실행 취소
124	테이블 속성 변경
128	테이블 초기화
131	비어 있는 페이지에 레코드 삽입 실행 취소
137	파티션 상태 갱신
161	레코드 삭제
162	레코드 삽입
163	레코드 갱신
164	페이지를 비우기 위해 레코드 삭제
165	비어 있는 페이지에 레코드 삽입
166	페이지를 비우기 위해 레코드 삭제 실행 취소
167	여러 레코드 삽입
168	여러 레코드 삽입 실행 취소

표 111. DOM 로그 레코드 헤더 구조(DOMLogRecordHeader)

설명	유형	오프셋(바이트)
구성요소 ID(=4)	unsigned char	0(1)
함수 ID(873 페이지의 표 112 참조)	unsigned char	1(1)
오브젝트 ID		
테이블 스페이스 ID	unsigned short	2(2)
오브젝트 ID	unsigned short	4(2)
테이블 ID		
테이블 스페이스 ID	unsigned short	6(2)
테이블 ID	unsigned short	8(2)
오브젝트 유형	unsigned char	10(1)
플래그	unsigned char	11(1)
총 길이: 12바이트		

표 112. DOM 로그 레코드 헤더 구조 함수 ID 값 및 정의

값	정의
2	인덱스 작성
3	인덱스 삭제
4	테이블 삭제
5	테이블 삭제 실행 취소
11	테이블 절단(임포트 바꾸기)
12	NOT LOGGED INITIALLY 활성화
35	테이블 재구성
101	테이블 작성
130	테이블 작성 실행 취소

주: 모든 데이터 관리 프로그램 로그 레코드 오프셋은 로그 관리 프로그램 레코드 헤더 끝에서 옵니다.

함수 ID 단축 이름이 UNDO로 시작하는 모든 로그 레코드는 문제 조치의 UNDO 또는 ROLLBACK 중에 기록된 로그 레코드입니다.

ROLLBACK은 다음의 결과가 될 수 있습니다.

- ROLLBACK 트랜잭션 명령문을 발행하는 사용자
- 선택된 트랜잭션의 ROLLBACK을 야기하는 교착 상태
- 응급 복구 다음에 언커미트된 트랜잭션의 ROLLBACK
- 로그 RESTORE 및 ROLLFORWARD 다음에 언커미트된 트랜잭션의 ROLLBACK

테이블 초기화 로그 레코드

테이블 초기화 로그 레코드는 새 영구 테이블이 작성될 때 기록되며 테이블 초기화를 표시합니다. 이 레코드는 DATA 및 블록 맵 스토리지 오브젝트를 작성하는 로그 레코드 다음에, LF 및 LOB 스토리지 오브젝트를 작성하는 로그 레코드 이전에 나타납니다. 이는 재실행 로그 레코드입니다. 함수 ID는 128입니다.

표 113. 테이블 초기화 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
파일 작성 LSN	db2LSN	6(8)
내부	내부	14(74)
테이블 설명 길이	sqluint32	88(4)
테이블 설명 레코드	variable	92(변수)
총 길이: 92바이트 + 테이블 설명 레코드 길이		

표 114. 테이블 설명 레코드

설명	유형	오프셋(바이트)
레코드 유형	unsigned char	0(1)
내부	내부	1(1)
컬럼 수	unsigned short	2(2)
컬럼 디스크립터 배열	변수 long	variable
총 길이: 4바이트 + 컬럼 디스크립터 배열 길이		

테이블 설명 레코드: 컬럼 디스크립터 배열

(컬럼 수) * 8. 배열의 각 요소는 다음을 포함합니다.

- 필드 유형(unsigned short, 2바이트)

```
SMALLINT     0x0000
INTEGER      0x0001
DECIMAL      0x0002
DOUBLE       0x0003
REAL         0x0004
BIGINT       0x0005
DECFLOAT64   0x0006
DECFLOAT128  0x0007
CHAR         0x0100
VARCHAR      0x0101
LONG VARCHAR 0x0104
DATE         0x0105
TIME         0x0106
TIMESTAMP    0x0107
BLOB         0x0108
CLOB         0x0109
STRUCT       0x010D
XMLTYPE      0x0112
GRAPHIC      0x0200
VARGRAPH     0x0201
LONG VARG    0x0202
DBCLOB       0x0203
```

- 길이(2바이트)
  - BLOB, CLOB 또는 DBCLOB의 경우 이 필드는 사용되지 않습니다. 이 필드의 최대 길이의 경우, 컬럼 디스크립터 배열 다음에 오는 배열을 참조하십시오.
  - DECIMAL이 아닌 경우, 길이는 필드의 최대 길이입니다(short).
  - PACKED DECIMAL의 경우, 바이트 0, unsigned char, 정밀도(총 길이) 바이트 1, unsigned char, 스케일(소수 자릿수)입니다.
- 널(NULL) 플래그(unsigned short, 2바이트)
  - 상호 배타적: 널(NULL)을 허용하거나 허용하지 않습니다.
  - 유효한 옵션: 디폴트 없음, 유형 디폴트, 사용자 디폴트, 생성됨 또는 압축 유형 디폴트

```
ISNULL                    0x0001
NONULLS                   0x0002
TYPE_DEFAULT              0x0004
USER_DEFAULT              0x0008
GENERATED                 0x0040
COMPRESS_SYSTEM_DEFAULT 0x0080
```

- 필드 오프셋(unsigned short, 2바이트). 필드의 고정 값을 찾을 수 있는 사용자 레코드의 고정 길이 시작 부분으로부터의 오프셋입니다.

테이블 설명 레코드: LOB 컬럼 디스크립터 배열

(LOB, CLOB 및 DBCLOB 필드 수) * 12. 배열의 각 요소는 다음을 포함합니다.

- 길이(MAX LENGTH OF FIELD, sqluint32, 4바이트)
- 인라인 길이(INLINE_LENGTH, sqluint16, 2바이트)
- 로그 플래그(IS COLUMN LOGGED, sqluint16, 2바이트)
- 예약됨(internal, sqluint32, 4바이트)

컬럼 디스크립터 배열에서 발견된 첫 번째 LOB, CLOB 또는 DBCLOB는 LOB 디스크립터 배열에서 첫 번째 요소를 사용합니다. 컬럼 디스크립터 배열에서 발견된 두 번째 LOB, CLOB 또는 DBCLOB는 LOB 디스크립터 배열에서 두 번째 요소를 사용합니다.

임포트 바꾸기(절단) 로그 레코드

임포트 바꾸기(절단) 로그 레코드는 IMPORT REPLACE 조치가 실행될 때 기록됩니다. 이 레코드는 테이블의 다시 초기화를 표시합니다(사용자 레코드가 없음, 새 수명 LSN). 로그 헤더의 테이블 ID는 절단되는 테이블을 식별합니다(IMPORT REPLACE). 이는 일반 로그 레코드입니다. 함수 ID는 11입니다.

표 115. 임포트 바꾸기(절단) 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DOMLogRecordHeader	0(12)
내부	내부	12(변수)
총 길이: 12바이트 + 변수 길이		

처음에 로그되지 않은 것 활성화 로그 레코드

사용자가 ACTIVATE NOT LOGGED INITIALLY절이 포함되는 ALTER TABLE 문을 발행할 때 처음에 로그되지 않은 것 활성화 로그 레코드가 기록됩니다. 이는 일반 로그 레코드입니다. 함수 ID는 12입니다.

표 116. 처음에 로그되지 않은 것 활성화 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DOMLogRecordHeader	0(12)

표 116. 처음에 로그되지 않은 것 활성화 로그 레코드 구조 (계속)

설명	유형	오프셋(바이트)
내부	내부	12(4)
긴 테이블 스페이스 ID*	unsigned short	16(2)
인덱스 테이블 스페이스 ID*	unsigned short	18(2)
인덱스 오브젝트 ID	unsigned short	20(2)
LF 오브젝트 ID	unsigned short	22(2)
LOB 오브젝트 ID	unsigned short	24(2)
XML 오브젝트 ID	unsigned short	26(2)
총 길이: 28바이트		

** DOM 헤더의 테이블 스페이스 ID와 동일하며 데이터베이스에 정의된 각 테이블 스페이스에 대해 고유한 ID입니다.*

삽입 롤백 로그 레코드

삽입 롤백 로그 레코드는 행 삽입 조치(INSERT RECORD)가 롤백될 때 기록됩니다. 이는 보상 로그 레코드입니다. 함수 ID는 110입니다.

표 117. 삽입 롤백 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
내부	내부	6(2)
레코드 길이	unsigned short	8(2)
여유 공간	unsigned short	10(2)
RID	char[]	12(6)
총 길이: 16바이트		

테이블 재구성 로그 레코드

테이블 재구성 로그 레코드는 테이블 재구성을 완료하기 위해 REORG 유틸리티가 커미트될 때 기록됩니다. 이는 일반 로그 레코드입니다. 함수 ID는 35입니다.

표 118. 테이블 Reorg 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DOMLogRecordHeader	0(12)
내부	variable	12(476)
인덱스 토큰[1]	unsigned short	488(2)
임시 테이블 스페이스 ID[2]	unsigned short	490(2)
Long 임시 테이블 스페이스 ID	unsigned short	492(2)
총 길이: 494바이트		

주:

1. 인덱스 토큰의 값이 0이 아니면, 재구성이 클러스터되는 인덱스입니다(클러스터링 인덱스).
2. 임시 테이블 스페이스 ID의 값이 0이 아니면, 재구성된 테이블을 빌드하기 위해 사용된 시스템 임시 테이블 스페이스입니다.

인덱스 작성, 인덱스 삭제 로그 레코드

이 로그 레코드는 인덱스가 작성되거나 삭제될 때 기록됩니다. 로그 레코드의 두 요소는 다음과 같습니다.

- 내부 ID인 인덱스 루트 페이지
- SYSIBM.SYSINDEXES의 IID 컬럼과 같은 인덱스 토큰. 이 요소의 값이 0이면 로그 레코드는 내부 인덱스의 조치를 나타내며 사용자 인덱스에는 관련되지 않습니다.

이는 일반 로그 레코드입니다. 함수 ID는 2(인덱스 작성) 또는 3(인덱스 삭제)입니다.

표 119. 인덱스 작성, 인덱스 삭제(drop) 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DOMLogRecordHeader	0(12)
내부	내부	12(2)
인덱스 토큰	unsigned short	14(2)
인덱스 루트 페이지	sqluint32	16(4)
총 길이: 20바이트		

테이블 작성, 테이블 삭제, 테이블 작성 롤백, 테이블 삭제 롤백 로그 레코드

이 로그 레코드는 영구 테이블에 대한 DATA 오브젝트가 작성되거나 삭제될 때 기록됩니다. MDC 테이블 작성의 경우, 블록 맵 오브젝트 작성에 대해 테이블 작성 로그 레코드가 있습니다. DATA 오브젝트(적용 가능한 경우 블록 맵 오브젝트도)는 CREATE TABLE 조작 중에, 테이블 초기화(Initialize Table) 이전에 작성됩니다. 테이블 작성 및 테이블 삭제는 일반 로그 레코드입니다. 테이블 작성 롤백 및 테이블 삭제 롤백은 보상 로그 레코드입니다. 함수 ID는 101(테이블 작성), 4(테이블 삭제), 130(테이블 작성 롤백) 또는 5(테이블 삭제 롤백)입니다.

표 120. 테이블 작성, 테이블 삭제(drop), 테이블 작성 롤백, 테이블 삭제 롤백 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DOMLogRecordHeader	0(12)
내부	variable	12(72)
총 길이: 84바이트		

테이블 속성 변경 로그 레코드

테이블 속성 변경 로그 레코드는 ALTER TABLE 문을 사용하거나, 제한조건 추가 또는 유효성 확인 결과로 테이블 상태가 변경될 때 기록됩니다. 이는 일반 또는 보상 로그 레코드가 될 수 있습니다. 함수 ID는 124입니다.

표 121. 테이블 속성 변경, 테이블 속성 변경 실행 취소

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
비트(속성) 변경 마스크	sqluint64	6(8)
비트(속성) 변경 값	sqluint64	14(8)
총 길이: 22바이트		

속성 비트

```
0x00000001   Propagation
0x00000002   Check Pending
0x00000010   Value Compression
0x00010000   Append Mode
0x00200000   LF Propagation
```

다른 모든 비트는 내부용입니다.

위의 비트 중 하나가 비트 변경 마스크에 있는 경우, 이 테이블 속성은 변경됩니다. 테이블 속성의 새 값을 판별하려면(0 = OFF 및 1 = ON), 비트 변경 값에서 해당되는 비트를 점검하십시오.

컬럼 추가 테이블 변경, 컬럼 추가 롤백 로그 레코드

컬럼 추가 테이블 변경 로그 레코드는 사용자가 ALTER TABLE 문을 사용하여 기존 테이블에 컬럼을 추가할 때 기록됩니다. 이전 컬럼 및 새 컬럼에 대한 전체 정보가 로그됩니다.

- 컬럼 계수 요소는 이전 컬럼 수와 새로운 총 컬럼 수를 나타냅니다.
- 병렬 배열에는 테이블에 정의된 컬럼에 대한 정보가 포함됩니다. 이전 병렬 배열은 ALTER TABLE 문 이전에 테이블을 정의하지만 새 병렬 배열은 ALTER TABLE 문의 결과로 생성되는 테이블을 정의합니다.
- 각 병렬 배열은 다음으로 구성됩니다.
  - 컬럼마다 하나의 8바이트 요소
  - LOB 컬럼이 있는 경우 LOB 컬럼마다 하나의 12바이트 요소. 이는 8바이트 요소 배열 뒤에 옵니다.

컬럼 추가 테이블 변경은 일반 로그 레코드입니다. 컬럼 추가 롤백은 보상 로그 레코드입니다. 함수 ID는 102(컬럼 추가) 또는 104(컬럼 추가 실행 취소)입니다.

표 122. 컬럼 추가 테이블 변경, 컬럼 추가 롤백 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordheader	0(6)
내부	내부	6(2)
이전 컬럼 계수	sqluint32	8(4)
새 컬럼 계수	sqluint32	12(4)
이전 병렬 배열[1]	variable	16(변수)
새 병렬 배열	variable	변수(변수)
총 길이: 16바이트 + 두 개의 병렬 배열 세트		

배열 요소:

1. 이 배열에 있는 요소의 길이는 다음과 같이 정의됩니다.
   - 요소가 컬럼 디스크립터인 경우 요소 길이는 8바이트입니다.
   - 요소가 LOB 컬럼 디스크립터인 경우 요소 길이는 12바이트입니다.

컬럼 디스크립터 배열 또는 LOB 컬럼 디스크립터 배열에 대한 정보는 874 페이지의 표 114 다음에 있는 설명을 참조하십시오.

컬럼 속성 변경 로그 레코드

함수 ID는 113입니다.

표 123. 컬럼 속성 변경 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordheader	0(6)
컬럼 ID	unsigned short	6(2)
이전 컬럼 정의	컬럼 디스크립터[1]	8(8)
새 컬럼 정의	컬럼 디스크립터[1]	16(8)
총 길이: 24바이트 + 레코드 길이		

[1]컬럼 디스크립터 배열의 설명은 874 페이지의 표 114 다음에 있는 설명을 참조하십시오.

컬럼 속성 변경 실행 취소 로그 레코드

함수 ID는 115입니다.

표 124. 컬럼 속성 변경 실행 취소 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
컬럼 ID	unsigned short	6(2)
이전 컬럼 정의	컬럼 디스크립터[1]	8(8)
새 컬럼 정의	컬럼 디스크립터[1]	16(8)

표 124. 컬럼 속성 변경 실행 취소 로그 레코드 구조 (계속)

설명	유형	오프셋(바이트)
총 길이: 24바이트 + 레코드 길이		

[1]컬럼 디스크립터 배열의 설명은 874 페이지의 표 114 다음에 있는 설명을 참조하십시오.

레코드 삽입, 레코드 삭제 롤백, 레코드 갱신 롤백 로그 레코드

이 로그 레코드는 행이 테이블에 삽입될 때, 또는 삭제나 갱신이 롤백될 때 기록됩니다. 갱신되는 레코드의 위치가 수정된 레코드 데이터를 수용하기 위해 변경되어야 하는 경우 갱신 중에 레코드 삽입 및 레코드 삭제 로그 레코드도 생성될 수 있습니다. 레코드 삽입 로그 레코드는 일반 로그 레코드입니다. 레코드 삭제 롤백 및 레코드 갱신 롤백은 보상 로그 레코드입니다. 함수 ID는 162(삽입), 111(삭제 롤백) 또는 112(갱신 롤백)입니다.

표 125. 레코드 삽입, 레코드 삭제 롤백, 레코드 갱신 롤백 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
내부	내부	6(2)
레코드 길이	unsigned short	8(2)
여유 공간	unsigned short	10(2)
RID	char[]	12(6)
레코드 오프셋	unsigned short	18(2)
레코드 헤더 및 데이터	variable	20(변수)
총 길이: 20바이트 + 레코드 길이		

다음은 레코드 헤더 및 데이터에 대한 세부사항입니다.

레코드 헤더

- 4바이트
- 레코드 유형(unsigned char, 1바이트)
- 예약됨(char, 1바이트)
- 레코드 길이(unsigned short, 2바이트)

레코드

- 변수 길이
- 레코드 유형(unsigned char, 1바이트)
- 예약됨(char, 1바이트)
- 레코드의 나머지는 레코드 유형과 테이블에 대해 정의된 테이블 디스크립터 레코드에 종속됩니다.

- 다음 필드는 1비트 세트를 가지고 있는 레코드 유형의 사용자 데이터 레코드에 적용됩니다.
  - 고정 길이(unsigned short, 2바이트). 이는 데이터 행의 고정 길이 섹션 길이입니다.
  - 형식화된 레코드(뒤에 가변 길이 컬럼이 오는 모든 고정 길이 컬럼)
- 다음 필드는 2비트 세트를 가지고 있는 레코드 유형의 사용자 데이터 레코드에 적용됩니다.
  - 컬럼 수(unsigned short, 2바이트). 데이터 행의 데이터 분할 영역에 있는 컬럼 수입니다. 883 페이지의 『VALUE COMPRESSION 없이 테이블에 대해 형식화된 사용자 데이터 레코드』의 내용을 참조하십시오.

    주: 오프셋 배열은 1 + 컬럼 수를 포함합니다.
  - 형식화된 레코드(뒤에 데이터 컬럼이 있는 오프셋 배열)

사용자 레코드는 다음 특성으로 완전히 지정됩니다.

1. 외부(Outer) 레코드 유형은 0입니다. 또는
2. 외부(Outer) 레코드 유형은 0x10입니다. 또는
3. 외부(Outer) 레코드 유형은 0x04 비트 세트를 가지고 있습니다. 그리고

1. 내부(Inner) 레코드 유형은 0x01 비트 세트를 가지고 있습니다. 또는
2. 내부(Inner) 레코드 유형은 0x02 비트 세트를 가지고 있습니다.

주: 행 압축 및 데이터 캡처는 호환 가능하지 않습니다.

형식화된 사용자 데이터에서 인라인 LOB 데이터 추출

인라인 LOB 데이터는 사용자 데이터 레코드에서 추출할 수 있습니다. LOB 컬럼 데이터의 시작 부분을 찾으면(테이블에서 VALUE COMPRESSION이 사용 가능한지 여부를 기초로) 첫 번째 바이트의 조사로 인라인 LOB 데이터 존재를 확인할 수 있습니다.

첫 번째 바이트가 0x69이면, 4바이트의 인라인 LOB 데이터 헤더를 표시합니다. 인라인 LOB 데이터는 이 4바이트 헤더 이후에 시작됩니다.

첫 번째 바이트가 0x80이면, LOB 데이터는 비어 있는 문자열입니다.

VALUE COMPRESSION 없이 테이블에 대해 형식화된 사용자 데이터 레코드

VALUE COMPRESSION 없이 형식화된 레코드의 경우 모든 필드는 고정 길이 분할 영역을 포함합니다. 또한 변수 길이 파트를 가지고 있는 8개의 필드 유형이 있습니다.

- VARCHAR

- LONG VARCHAR
- BLOB
- CLOB
- VARGRAPHIC
- LONG VARG
- DBCLOB

다른 필드 유형의 고정 분할 영역 길이는 다음과 같이 판별할 수 있습니다.

- DECIMAL

  이 필드는 *nnnnnn...s* 양식의 표준 압축 10진수입니다. 필드 길이는 (정밀도 + 2)/2 입니다. 부호 니블은 양수(+)의 경우 xC이고 음수(-)의 경우 xD 또는 xB입니다.

- SMALLINT INTEGER BIGINT DOUBLE REAL CHAR GRAPHIC

  테이블 디스크립터 레코드에서 이 컬럼에 대해 요소의 길이 필드에 필드의 고정 길이 *크기*가 포함됩니다.

- DATE

  이 필드는 *yyyymmdd* 양식의 4바이트 압축 10진수입니다. 예를 들어, 1996년 4월 3일은 x '19960403'으로 표시됩니다.

- TIME

  이 필드는 *hhmmss* 양식의 3바이트 압축 10진수입니다. 예를 들어, 1:32PM은 x '133200'으로 표시됩니다.

- TIMESTAMP

  이 필드는 *yyyymmddhhmmssuuuuuu*(DATE|TIME|microseconds) 양식의 10바이트 압축 10진수입니다.

- VARCHAR LONG VARCHAR BLOB CLOB VARGRAPHIC LONG VARG DBCLOB

  모든 변수 길이 필드의 고정 분할 영역 길이는 4입니다.

다음 섹션에서는 형식화된 레코드 내에서 각 필드의 고정 분할 영역 위치를 설명합니다.

테이블 디스크립터 레코드는 테이블의 컬럼 형식을 설명합니다. 컬럼 구조 배열을 포함하며 해당 요소는 필드 유형, 필드 길이, 널(NULL) 플래그 및 필드 오프셋을 나타냅니다. 맨 마지막은 필드의 고정 길이 분할 영역이 위치되는, 형식화된 레코드 시작으로부터의 오프셋입니다.

표 126. 테이블 디스크립터 레코드 구조

레코드 유형	컬럼 수	컬럼 구조 • 필드 유형 • 길이 • 널(NULL) 플래그 • 필드 오프셋	LOB 정보

주: 자세한 정보는 다음 873 페이지의 표 113의 설명을 참조하십시오.

널(NULL) 입력 가능한(널(NULL) 플래그로 지정되는) 컬럼의 경우, 필드의 고정 길이 분할 영역 다음에 추가 바이트가 있습니다. 이 바이트에는 두 개의 값 중 하나가 포함됩니다.

- NOT NULL (0x00)
- NULL (0x01)

널(NULL) 입력 가능한 컬럼에 대한 형식화된 레코드 내의 널(NULL) 플래그가 0x00으로 설정된 경우, 레코드의 고정 길이 데이터 분할 영역에 유효한 값이 있습니다. 널(NULL) 플래그 값이 0x01인 경우 데이터 필드 값은 NULL입니다.

형식화된 사용자 데이터 레코드는 사용자가 볼 수 있는 테이블 데이터를 포함합니다. 고정 길이 레코드로 형식화되며 뒤에는 변수 길이 섹션이 있습니다.

표 127. VALUE COMPRESSION을 사용하지 않는 테이블에 대한 형식화된 사용자 데이터 레코드 구조

레코드 유형	고정 길이 섹션	고정 길이 섹션	변수 데이터 섹션

주: 자세한 정보는 다음 880 페이지의 표 125의 설명을 참조하십시오.

모든 변수 필드 유형은 고정 길이 섹션에 4바이트의 고정 데이터 분할 영역을 가지고 있습니다(또한, 컬럼이 널(NULL) 입력 가능한 경우 널(NULL) 플래그도 추가됨). 처음 2바이트(short)는 변수 데이터가 위치되는 고정 길이 섹션 시작으로부터의 오프셋을 나타냅니다. 다음 2바이트(short)는 오프셋 값에 언급된 변수 데이터의 길이를 지정합니다.

VALUE COMPRESSION 없이 테이블에 대해 형식화된 사용자 데이터 레코드

VALUE COMPRESSION으로 형식화된 레코드는 오프셋 배열과 데이터 분할 영역으로 구성됩니다. 배열의 각 항목은 데이터 분할 영역에 있는 해당 컬럼 데이터로의 2바이트 오프셋입니다. 데이터 분할 영역에 있는 컬럼 수 데이터는 레코드 헤더에서 찾을 수 있으며 오프셋 배열의 항목 수는 (1 + 데이터 분할 영역에 있는 컬럼 수 데이터)입니다.

1. 압축된 컬럼 값은 속성 바이트에 사용되는 디스크 스페이스의 1바이트만 소비합니다. 속성 바이트는 컬럼 데이터가 압축되었음을 나타냅니다. 예를 들어, 데이터 값은 알려져 있지만 디스크에 저장되어 있지 않습니다. 오프셋에서 상위 비트(0x8000)는 액세스한 데이터가 속성 바이트임을 표시하기 위해 사용됩니다. (해당 컬럼 데이터의 오프셋을 나타내기 위해 15비트만 사용됩니다.)
2. 일반 컬럼 데이터의 경우 컬럼 데이터는 오프셋 배열 다음에 있습니다. 속성 바이트나 길이 표시기는 존재하지 않습니다.
3. 액세스한 데이터는 속성 바이트인 경우 두 개의 다른 값을 사용할 수 있습니다.
   - NULL 0x01 (값이 NULL임)
   - COMPRESSED SYSTEM DEFAULT 0x80 (값이 시스템 디폴트와 같음)
4. 컬럼 데이터의 길이는 현재 오프셋과 다음 컬럼 오프셋 사이의 차이입니다.

표 128. VALUE COMPRESSION을 사용하는 테이블에 대한 형식화된 사용자 데이터 레코드 구조

레코드 유형	데이터 분할 영역에 있는 컬럼 수	오프셋 배열	데이터 분할 영역

주: 자세한 정보는 다음 880 페이지의 표 125의 설명을 참조하십시오.

비어 있는 페이지에 레코드 삽입, 페이지를 비우기 위해 레코드 삭제, 페이지를 비우기 위해 레코드 삭제 롤백, 비어 있는 페이지에 레코드 삽입 롤백 로그 레코드

이 로그 레코드는 테이블이 다차원적으로 클러스터된(MDC) 테이블인 경우에 기록됩니다. 비어 있는 페이지에 레코드 삽입 로그 레코드는 레코드가 삽입될 때 기록되고 페이지의 첫 번째 레코드로서, 해당 페이지는 블록의 첫 번째 페이지가 아닙니다. 이 로그 레코드는 페이지에 대한 삽입 뿐만 아니라 페이지가 더 이상 비어있지 않음을 나타내는 블록 첫 번째 페이지에서의 비트 갱신도 로그합니다. 페이지를 비우기 위해 레코드 삭제 로그 레코드는 마지막 레코드가 페이지에서 삭제될 때 기록됩니다. 해당 페이지는 블록의 첫 번째 페이지가 아닙니다. 이 로그 레코드는 페이지로부터의 삭제를 로그하고, 블록의 첫 번째 페이지에 대한 비트 갱신을 로그합니다. 이는 페이지가 비어 있음을 표시합니다. 비어 있는 페이지에 레코드 삽입 로그 레코드 및 페이지를 비우기 위해 레코드 삭제 로그 레코드는 일반 로그 레코드입니다. 레코드 삭제 롤백 로그 레코드와 레코드 삽입 롤백 로그 레코드는 보상 로그 레코드입니다. 함수 ID는 165(비어 있는 페이지에 레코드 삽입), 164(페이지를 비우기 위해 레코드 삭제), 166(페이지를 비우기 위해 레코드 삭제 롤백) 또는 131(비어 있는 페이지에 레코드 삽입 롤백)입니다.

표 129. 비어 있는 페이지로 레코드 삽입 롤백

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
내부	내부	6(2)
레코드 길이	unsigned short	8(2)

표 129. 비어 있는 페이지로 레코드 삽입 롤백 (계속)

설명	유형	오프셋(바이트)
여유 공간	unsigned short	10(2)
RID	char[]	12(6)
내부	내부	18(2)
블록의 첫 번째 페이지	sqluint32	20(4)
총 길이: 24바이트		

표 130. 비어 있는 페이지로 레코드 삽입, 비어 있는 페이지로 레코드 삭제 롤백, 비어 있는 페이지로 레코드 삭제

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
내부	내부	6(2)
레코드 길이	unsigned short	8(2)
여유 공간	unsigned short	10(2)
RID	char[]	12(6)
내부	내부	18(2)
블록의 첫 번째 페이지	sqluint32	20(4)
레코드 오프셋	unsigned short	24(2)
레코드 헤더 및 데이터	variable	26(변수)
총 길이: 26바이트 + 레코드 길이		

주: 레코드 헤더 및 데이터 세부사항은 다음 880 페이지의 표 125의 설명을 참조하십시오.

레코드 갱신 로그 레코드

레코드 갱신 로그 레코드는 행이 갱신되고 해당되는 스토리지 위치가 동일하게 유지되는 경우에 기록됩니다. 두 가지의 사용 가능한 레코드 형식이 있습니다. 두 형식은 레코드 삽입(로그 레코드 삭제와도 같음) 로그 레코드와 같습니다(880 페이지의 『레코드 삽입, 레코드 삭제 롤백, 레코드 갱신 롤백 로그 레코드』 참조). 하나의 형식에는 갱신되는 행의 *사전* 갱신 이미지가 포함되고, 다른 형식에는 갱신되는 행의 *사후* 갱신 이미지가 포함됩니다. 이는 일반 로그 레코드입니다. 함수 ID는 163입니다.

표 131. 레코드 갱신 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
내부	내부	6(2)
새 레코드 길이	unsigned short	8(2)
여유 공간	unsigned short	10(2)
RID	char[]	12(6)
레코드 오프셋	unsigned short	18(2)
이전 레코드 헤더 및 데이터	variable	20(변수)

표 131. 레코드 갱신 로그 레코드 구조 (계속)

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	변수(6)
내부	내부	변수(2)
이전 레코드 길이	unsigned short	변수(2)
여유 공간	unsigned short	변수(2)
RID	char[]	변수(6)
레코드 오프셋	unsigned short	변수(2)
새 레코드 헤더 및 데이터	variable	변수(변수)
총 길이: 40바이트 + 2 레코드 길이		

테이블 또는 스키마의 이름 바꾸기 로그 레코드

테이블 스키마의 이름 바꾸기 로그 레코드는 테이블 또는 스키마 이름이 수정될 때 기록됩니다. 함수 ID는 122입니다.

표 132. 테이블 또는 스키마 이름 바꾸기 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
총 길이: 6바이트		

테이블 또는 스키마의 이름 바꾸기 로그 레코드에는 테이블 또는 스키마 오브젝트의 이전 및 새 이름에 관한 정보가 포함되지 않습니다. 테이블 또는 스키마 이름 바꾸기가 발생할 때 시스템 카탈로그 테이블에 대한 조작과 연관되는 별도의 삽입, 갱신 및 삭제 로그 레코드가 생성됩니다.

테이블 또는 스키마의 이름 바꾸기 실행 취소 로그 레코드

테이블 스키마의 이름 바꾸기 실행 취소 로그 레코드는 테이블 또는 스키마 이름 수정 사항이 롤백될 때 기록됩니다. 함수 ID는 123입니다.

표 133. 테이블 또는 스키마 이름 바꾸기 실행 취소 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
총 길이: 6바이트		

테이블 또는 스키마의 이름 바꾸기 로그 레코드에는 테이블 또는 스키마 오브젝트의 이전 및 새 이름에 관한 정보가 포함되지 않습니다. 테이블 또는 스키마 이름 바꾸기가 발생할 때 시스템 카탈로그 테이블에 대한 조작과 연관되는 별도의 삽입, 갱신 및 삭제 로그 레코드가 생성됩니다.

여러 레코드 삽입, 여러 레코드 삽입 실행 취소

이 로그 레코드는 여러 행이 테이블의 동일 페이지에 삽입될 때 기록됩니다. 여러 레코드 삽입 롤백은 보상 로그 레코드입니다. 함수 ID는 167 및 168입니다.

표 134. 다중 레코드 삽입 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
채우기	char[]	6(2)
레코드 수	unsigned short	8(2)
여유 공간	unsigned short	10(2)
레코드 길이의 합	unsigned short	12(2)
변수 파트 길이	unsigned short	14(2)
풀 페이지 번호	sqluint32	16(4)
레코드 설명 또는 롤백 설명 표 135 및 표 136의 설명을 참조하십시오.	variable	20(변수)
총 길이: 20바이트 + 레코드 길이		

표 135. 레코드 설명(레코드마다 하나)

설명	유형	오프셋(바이트)
RID	unsigned char[6]	0(6)
레코드 오프셋	unsigned short	6(2)
레코드 헤더 및 데이터	variable	8(변수)
총 길이: 8바이트 + 레코드 길이		

표 136. 롤백 설명(레코드마다 하나)

설명	유형	오프셋(바이트)
RID	unsigned char[6]	0(6)
레코드 오프셋	unsigned short	6(2)
총 길이: 8바이트		

레코드 헤더 및 데이터 세부사항은 다음 880 페이지의 표 125의 설명을 참조하십시오.

파티션 상태 갱신, 파티션 상태 롤백 로그 레코드

파티션 상태 갱신 로그 레코드는 사용자가 ALTER TABLE문에 ADD PARTITION, ATTACH PARTITION 또는 DETACH PARTITION절을 추가하여 실행한 경우에 기록됩니다. 이는 SET INTEGRITY문이 파티션된 테이블에서 실행되어 이전에 접속된 파티션이 온라인으로 표시되는 경우에도 기록됩니다. 로그 레코드 함수 ID는 137이지만 로그 레코드 실행 취소 또는 롤백 함수 ID는 108입니다.

표 137. 파티션 상태 갱신, 파티션 상태 롤백 로그 레코드 구조

설명	유형	오프셋(바이트)
로그 헤더	DMSLogRecordHeader	0(6)
마스터 테이블 스페이스 ID	unsigned short	6(2)
마스터 테이블 ID	unsigned short	8(2)
내부	내부	10(2)
데이터 파티션 ID	unsigned short	12(2)
내부	내부	14(6)
내부	내부	20(2)
파티션 조치	unsigned short	22(2)
전체 길이: 24바이트		

로그 헤더

871 페이지의 표 109의 내용을 참조하십시오. DMSLogRecordHeader의 테이블 스페이스 및 테이블 ID는 테이블 파티션에 대한 SYSCAT.DATAPARTITIONS 카탈로그 뷰에서 각각 TBSPACEID 및 PARTITIONOBJECTID 컬럼 값과 일치합니다.

마스터 테이블 스페이스 ID

마스터 테이블 스페이스 ID는 파티션된 테이블에 대한 SYSCAT.TABLES 카탈로그 뷰에서 TBSPACEID 컬럼 값과 일치합니다.

마스터 테이블 ID

마스터 테이블 ID는 파티션된 테이블에 대한 SYSCAT.TABLES 카탈로그 뷰에서 TABLEID 컬럼 값과 일치합니다.

데이터 파티션 ID

데이터 파티션 ID는 테이블 파티션에 대한 SYSCAT.DATAPARTITIONS 카탈로그 뷰에서 DATAPARTITIONID 컬럼 값과 일치합니다.

파티션 조치

파티션 조치 값은 다음과 같은 정의를 포함합니다.

```
1   ADD PARTITION
2   ATTACH PARTITION
4   SET INTEGRITY after ATTACH PARTITION
5   DETACH PARTITION (지연됨: 구체화된 쿼리 테이블(MQT)이 유지됨)
6   DETACH PARTITION (지연됨: 파티션되지 않은 인덱스는 정리됨)
7   DETACH PARTITION (즉시: 접속된 파티션은 SET INTEGRITY를 실행할 수 없음)
8   DETACH PARTITION (즉시: MQT 또는 파티션되지 않은 인덱스가 없음)
```

부록 C. DB2 기술 정보 개요

DB2 기술 정보는 다음 도구 및 메소드를 통해 사용할 수 있습니다.

- DB2 정보 센터
  - 주제 항목(태스크, 개념 및 참조 항목)
  - DB2 도구에 대한 도움말
  - 샘플 프로그램
  - 자습서
- DB2 서적
  - PDF 파일(다운로드)
  - PDF 파일(DB2 PDF DVD)
  - 인쇄된 서적
- 명령행 도움말
  - 명령 도움말
  - 메시지 도움말

주: DB2 정보 센터의 주제는 PDF 또는 하드카피 서적보다 더 자주 갱신됩니다. 최신 정보를 보려면 사용 가능한 문서 갱신사항을 설치하거나 ibm.com에서 DB2 정보 센터를 참조하십시오.

ibm.com에서 추가 DB2 기술 정보(예: 기술 노트, 백서 및 IBM Redbooks® 서적)를 온라인으로 액세스할 수 있습니다. 다음은 DB2 정보 관리 라이브러리 소프트웨어 사이트의 주소입니다. http://www.ibm.com/software/data/sw-library/

문서 피드백

DB2 문서에 대한 피드백을 환영합니다. DB2 문서를 향상시키는 방법에 대해서 제안사항이 있는 경우 db2docs@ca.ibm.com으로 전자 우편을 보내십시오. DB2 문서 팀에서는 고객의 모든 피드백을 읽지만 직접 응답할 수는 없습니다. 고객의 문제를 더 잘 이해할 수 있도록 가능한 한 구체적인 예를 제공해주십시오. 특정 주제 또는 도움말 파일에 대한 피드백을 보내실 경우, 제목 및 URL을 알려주십시오.

DB2 고객 지원에 문의할 때는 이 전자 우편 주소를 사용하지 마십시오. 문서에서 해결할 수 없는 DB2 기술 문제점이 있는 경우, 해당 지역의 IBM 서비스 센터에 도움을 요청하십시오.

DB2 기술 라이브러리(하드카피 또는 PDF 형식)

다음 표는 IBM Publications Center(www.ibm.com/shop/publications/order)에서 사용할 수 있는 DB2 라이브러리에 대한 설명입니다. PDF 형식의 영문 DB2 버전 9.7 매뉴얼 및 번역된 버전은 www.ibm.com/support/docview.wss?rs=71&uid=swg2700947에서 다운로드할 수 있습니다.

표에 인쇄할 수 있는 책으로 설명된 경우라도, 사용 국가 또는 지역에 따라 해당 책을 사용할 수 없을 수도 있습니다.

매뉴얼이 갱신될 때마다 문서 번호가 증가합니다. 다음 사항을 참조하여 읽고 있는 매뉴얼이 최신 버전인지 확인하십시오.

주: *DB2 정보 센터*는 PDF 또는 하드카피 서적보다 자주 갱신됩니다.

표 138. DB2 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
관리 API 참조서	SA30-3958-00	예	2009년 8월
관리 루틴 및 뷰	SA30-3955-00	아니오	2009년 8월
Call Level Interface Guide and Reference, Volume 1	SC27-2437-00	예	2009년 8월
Call Level Interface Guide and Reference, Volume 2	SC27-2438-00	예	2009년 8월
명령어 참조서	SA30-3959-00	예	2009년 8월
데이터 이동 유틸리티 안내서 및 참조서	SA30-3969-00	예	2009년 8월
데이터 복구 및 고가용성 안내서 및 참조서	SA30-3970-00	예	2009년 8월
데이터베이스 관리 개념 및 구성 참조서	SA30-3951-00	예	2009년 8월
데이터베이스 모니터링 안내서 및 참조서	SA30-3953-00	예	2009년 8월
데이터베이스 보안 안내서	SA30-3971-00	예	2009년 8월
DB2 Text Search Guide	SC27-2459-00	예	2009년 8월
Developing ADO.NET and OLE DB Applications	SC27-2444-00	예	2009년 8월
Developing Embedded SQL Applications	SC27-2445-00	예	2009년 8월
Developing Java Applications	SC27-2446-00	예	2009년 8월
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-2447-00	아니오	2009년 8월

표 138. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Developing User-defined Routines(SQL and External)	SC27-2448-00	예	2009년 8월
Getting Started with Database Application Development	GI11-9410-00	예	2009년 8월
Linux 및 Windows에서 DB2 설치 및 관리 시작하기	GA30-3960-00	예	2009년 8월
자국어 안내서	SA30-3972-00	예	2009년 8월
DB2 Server 설치	GA30-3962-00	예	2009년 8월
IBM Data Server Client 설치	GA30-3963-00	아니오	2009년 8월
Message Reference Volume 1	SC27-2450-00	아니오	2009년 8월
Message Reference Volume 2	SC27-2451-00	아니오	2009년 8월
Net Search Extender Administration and User's Guide	SC27-2469-00	아니오	2009년 8월
파티셔닝 및 클러스터링 안내서	SA30-3973-00	예	2009년 8월
pureXML Guide	SC27-2465-00	예	2009년 8월
Query Patroller 관리 및 사용자 안내서	SA30-3974-00	아니오	2009년 8월
Spatial Extender and Geodetic Data Management Feature User's Guide and Reference	SC27-2468-00	아니오	2009년 8월
SQL Procedural Languages: Application Enablement and Support	SC27-2470-00	예	2009년 8월
SQL 참조서, 볼륨 1	SA30-3956-00	예	2009년 8월
SQL 참조서, 볼륨 2	SA30-3957-00	예	2009년 8월
문제점 해결 및 데이터베이스 성능 조정	SA30-3952-00	예	2009년 8월
DB2 버전 9.7로 업그레이드	SA30-3961-00	예	2009년 8월
Visual Explain 자습서	SA30-3968-00	아니오	2009년 8월
DB2 버전 9.7의 새로운 내용	SA30-3967-00	예	2009년 8월
Workload Manager Guide and Reference	SC27-2464-00	예	2009년 8월

표 138. DB2 기술 정보 (계속)

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
XQuery Reference	SC27-2466-00	아니오	2009년 8월

표 139. DB2 Connect 특정 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
DB2 Connect Personal Edition 설치 및 구성	SA30-3965-00	예	2009년 8월
DB2 Connect Server 설치 및 구성	SA30-3966-00	예	2009년 8월
DB2 Connect 사용자 안내서	SA30-3964-00	예	2009년 8월

표 140. Information Integration 기술 정보

이름	문서 번호	인쇄 가능	마지막 갱신 날짜
Information Integration: Administration Guide for Federated Systems	SC19-1020-02	예	2009년 8월
Information Integration: ASNCLP Program Reference for Replication and Event Publishing	SC19-1018-04	예	2009년 8월
Information Integration: Configuration Guide for Federated Data Sources	SC19-1034-02	아니오	2009년 8월
Information Integration: SQL Replication Guide and Reference	SC19-1030-02	예	2009년 8월
Information Integration: Introduction to Replication and Event Publishing	GC19-1028-02	예	2009년 8월

인쇄된 DB2 서적 주문

인쇄된 DB2 서적이 필요한 경우, 대부분 온라인으로 구매할 수 있으나 모든 국가 또는 지역에서 가능한 것은 아닙니다. 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. *DB2 PDF 문서* DVD의 일부 소프트카피 서적은 인쇄할 수 없다는 점에 유의하십시오. 예를 들어, *DB2 메시지 참조서*의 볼륨은 인쇄된 서적으로 사용할 수 없습니다.

DB2 PDF 문서 DVD에서 사용할 수 있는 다수의 DB2 서적의 인쇄된 버전은 IBM에서 유료로 주문할 수 있습니다. 주문하는 위치에 따라 IBM Publications Center에서 온라인으로 서적을 주문할 수도 있습니다. 해당 국가 또는 지역에서 온라인 주문이

불가능하면, 언제든지 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문할 수 있습니다. DB2 PDF 문서 DVD의 모든 서적을 인쇄할 수는 없다는 점에 유의하십시오.

주: 가장 최신의 완전한 DB2 문서는 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7의 DB2 정보 센터에서 유지보수됩니다.

인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.

- 해당 국가 또는 지역에서 인쇄된 DB2 서적을 온라인으로 주문할 수 있는지 여부를 확인하려면 http://www.ibm.com/shop/publications/order의 IBM Publications Center를 확인하십시오. 서적 주문 정보를 액세스하려면 국가/지역/언어를 선택한 다음 해당 위치에서 주문 지시사항을 따르십시오.
- 해당 지역의 IBM 담당자로부터 인쇄된 DB2 서적을 주문하려면 다음을 수행하십시오.
  1. 다음 웹 사이트 중 하나에서 해당 지역 담당자에 대한 문의처 정보를 찾으십시오.
     - www.ibm.com/planetwide에 있는 IBM 전세계 문의처 디렉토리
     - http://www.ibm.com/shop/publications/order의 IBM Publications 웹 사이트. 사용 지역의 해당 서적 홈 페이지에 액세스하려면 해당 국가, 지역 또는 언어를 선택해야 합니다. 이 페이지에서 "이 제품의 정보" 링크를 수행하십시오.
  2. 전화로 주문할 경우, 주문할 DB2 서적을 지정하십시오.
  3. 담당자에게 주문하려는 서적의 제목 및 문서 번호를 제공하십시오. 서적의 제목 및 문서 번호는 890 페이지의 『DB2 기술 라이브러리(하드카피 또는 PDF 형식)』를 참조하십시오.

명령행 처리기에서 SQL 상태 도움말 표시

DB2 제품은 SQL문의 결과로 나타나는 상태에 대한 SQLSTATE 값을 리턴합니다. SQLSTATE 도움말은 SQL 상태 및 SQL 상태 클래스 코드의 의미를 설명합니다.

SQL 상태 도움말을 시작하려면 명령행 처리기를 열고 다음을 입력하십시오.

```
? sqlstate or ? class code
```

여기서, *sqlstate*는 유효한 5자리 숫자로 된 SQL 상태이고 *class code*는 SQL 상태의 처음 2자리 숫자를 나타냅니다.
예를 들어, `? 08003`은 08003 SQL 상태에 대한 도움말을 표시하고, `? 08`은 08 클래스 코드에 대한 도움말을 표시합니다.

DB2 정보 센터의 다른 버전에 액세스

DB2 버전 9.7 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/입니다.

DB2 버전 9.5 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/입니다.

DB2 버전 9 주제에 대한 DB2 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v9/입니다.

DB2 버전 8 주제에 대한 버전 8 정보 센터 URL은 http://publib.boulder.ibm.com/infocenter/db2luw/v8/입니다.

DB2 정보 센터에서 원하는 언어로 항목 표시

DB2 정보 센터는 브라우저 환경 설정에 지정된 언어로 주제 항목을 표시합니다. 주제가 원하는 언어로 변환되지 않은 경우, DB2 정보 센터는 해당 주제 항목을 영어로 표시합니다.

- Internet Explorer 브라우저에서 원하는 언어로 항목을 표시하려면 다음을 수행하십시오.
  1. Internet Explorer에서 **도구 —> 인터넷 옵션 —> 언어** 단추를 누르십시오. 언어 환경 설정 창이 열립니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     - 목록에 새 언어를 추가하려면 **추가...** 단추를 누르십시오.

       주: 언어를 추가하더라도 원하는 언어로 항목을 표시하는 데 필요한 글꼴이 컴퓨터에 설치되지는 않습니다.
     - 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 해당 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.
  3. 브라우저 캐시를 지운 후 페이지를 새로 고치면 원하는 언어로 DB2 정보 센터가 표시됩니다.
- Firefox 또는 Mozilla 브라우저에서 원하는 언어로 주제 항목을 표시하려면 다음을 수행하십시오.
  1. **도구 —> 설정 —> 내용** 대화 상자의 **언어** 섹션에서 단추를 선택하십시오. 환경 설정 창에 언어 패널이 표시됩니다.
  2. 원하는 언어가 언어 목록의 첫 번째 항목으로 지정되었는지 확인하십시오.
     - 목록에 새 언어를 추가하려면 언어 선택 창에서 원하는 언어를 선택한 다음 **추가...** 단추를 누르십시오.

– 언어를 목록 맨위로 이동하려면, 언어를 선택한 후 해당 언어가 언어 목록의 첫 번째 항목이 될 때까지 **위로 이동** 단추를 누르십시오.

3. 브라우저 캐시를 지운 후 페이지를 새로 고치면 원하는 언어로 DB2 정보 센터가 표시됩니다.

일부 브라우저 및 운영 체제 조합에서는 운영 체제의 국가별 설정을 선택한 로케일 및 언어로 변경해야 합니다.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 갱신

로컬로 설치된 DB2 정보 센터는 주기적으로 갱신해야 합니다.

시작하기 전에

DB2 버전 9.7 정보 센터는 미리 설치된 상태여야 합니다. 자세한 내용은 *DB2 Server 설치*의 『DB2 설치 마법사를 사용하여 DB2 정보 센터 설치』 주제를 참조하십시오. 정보 센터 설치에 적용되는 모든 전제조건 및 제한사항은 정보 센터 갱신에도 적용됩니다.

이 태스크에 대한 정보

기존의 DB2 정보 센터는 자동 또는 수동으로 갱신할 수 있습니다.

- 자동 갱신 - 기존 정보 센터 기능 및 언어를 갱신합니다. 자동 갱신의 또 다른 이점으로는 갱신 동안 정보 센터를 사용할 수 없는 시간이 매우 짧다는 점입니다. 또한 자동 갱신은 주기적으로 실행되는 기타 일괄처리 작업의 일부로 실행되도록 설정할 수도 있습니다.
- 수동 갱신 - 갱신 프로세스 중에 기능이나 언어를 추가하려는 경우 사용하십시오. 예를 들어, 로컬 정보 센터는 기본적으로 영어와 프랑스어로 설치되어 있으며, 수동 갱신을 통해 기존 정보 센터의 기능 및 언어 갱신뿐만 아니라 독일어도 설치할 수 있습니다. 단, 수동 갱신을 수행하려면 정보 센터를 중지한 다음 갱신하고 재시작해야 합니다. 정보 센터는 갱신 프로세스 동안에는 사용할 수 없습니다.

프로시저

이 주제는 자동 갱신 프로세스에 대한 설명입니다. 수동 갱신에 대한 지시사항은 『컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신』 주제를 참조하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 자동으로 갱신하려면 다음을 수행하십시오.

1. Linux 운영 체제의 경우
   a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 `/opt/ibm/db2ic/V9.7` 디렉토리에 디폴트로 설치됩니다.

b. 설치 디렉토리에서 doc/bin 디렉토리로 이동하십시오.

c. 다음과 같이 ic-update 스크립트를 실행하십시오.

```
ic-update
```

2. Windows 운영 체제의 경우

a. 명령 창을 여십시오.

b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 <Program Files>₩IBM₩DB2 Information Center₩Version 9.7 디렉토리에 디폴트로 설치됩니다. 여기서 <Program Files>는 프로그램 파일 디렉토리의 위치를 나타냅니다.

c. 설치 디렉토리에서 doc₩bin 디렉토리로 이동하십시오.

d. 다음과 같이 ic-update.bat 파일을 실행하십시오.

```
ic-update.bat
```

결과

DB2 정보 센터가 자동으로 재시작됩니다. 갱신사항이 사용 가능한 경우, 정보 센터에는 새로 갱신된 주제가 표시됩니다. 정보 센터 갱신을 사용할 수 없는 경우, 메시지가 로그에 추가됩니다. 로그 파일은 doc₩eclipse₩configuration 디렉토리에 있습니다. 이 로그 파일 이름은 임의로 생성된 번호입니다. 예: 1239053440785.log

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터 수동 갱신

DB2 정보 센터를 로컬로 설치한 경우, IBM으로부터 문서 갱신사항을 받아 설치할 수 있습니다.

로컬로 설치된 DB2 정보 센터를 수동으로 갱신하려면 다음을 수행하십시오.

1. 컴퓨터에서 DB2 정보 센터를 중지한 후 독립형 모드에서 다시 시작하십시오. 독립형 모드에서 정보 센터를 실행하면 사용자의 네트워크와 연결된 다른 사용자는 정보 센터에 액세스할 수 없으므로 갱신사항을 적용할 수 있습니다. DB2 정보 센터의 워크스테이션 버전은 항상 독립형 모드에서 실행됩니다.
2. 사용 가능한 갱신사항을 확인하려면 갱신 기능을 사용하십시오. 설치해야 할 갱신사항이 있는 경우, 갱신 기능을 사용하여 이를 가져온 후 설치할 수 있습니다.

   주: 인터넷에 연결되지 않은 머신에 DB2 정보 센터 갱신사항을 설치해야 할 경우, 인터넷에 연결되고 DB2 정보 센터가 설치된 머신을 사용하여 갱신 사이트를 로컬 파일 시스템으로 미러하십시오. 네트워크 상에 문서 갱신사항을 설치하려는 사용자가 많을 경우에는 갱신 사이트를 로컬로 미러링하거나 갱신 사이트의 프록시를 작성하여 갱신을 수행하면 각 개인에게 필요한 시간을 줄일 수 있습니다.
   갱신 패키지가 사용 가능하면 갱신 기능을 사용하여 패키지를 가져오십시오. 그러나 갱신 기능은 독립형 모드에서만 사용할 수 있습니다.

3. 독립형 정보 센터를 중지한 후 컴퓨터에서 DB2 정보 센터를 재시작하십시오.

주: Windows 2008, Windows Vista 이상의 경우 이 절 다음에 나오는 명령은 관리자로 실행해야 합니다. 전체 관리자 권한으로 명령 프롬프트 또는 그래픽 도구를 열려면 단축 아이콘을 마우스 오른쪽 단추로 누른 후 **관리자로 실행**을 선택하십시오.

컴퓨터 또는 인트라넷 서버에 설치된 DB2 정보 센터를 갱신하려면 다음을 수행하십시오.

1. DB2 정보 센터를 중지하십시오.
   - Windows의 경우, **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 Information Center** 서비스를 마우스 오른쪽 단추로 누른 후 **중지**를 선택하십시오.
   - Linux의 경우, 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 stop
     ```
2. 독립형 모드에서 정보 센터를 시작하십시오.
   - Windows의 경우:
     a. 명령 창을 여십시오.
     b. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 `<Program Files>₩IBM₩DB2 Information Center₩Version 9.7` 디렉토리에 디폴트로 설치됩니다. 여기서 `<Program Files>`는 프로그램 파일 디렉토리의 위치를 나타냅니다.
     c. 설치 디렉토리에서 `doc₩bin` 디렉토리로 이동하십시오.
     d. 다음과 같이 `help_start.bat` 파일을 실행하십시오.

        ```
        help_start.bat
        ```
   - Linux의 경우:
     a. 정보 센터가 설치된 경로를 찾아가십시오. DB2 정보 센터는 `/opt/ibm/db2ic/V9.7` 디렉토리에 디폴트로 설치됩니다.
     b. 설치 디렉토리에서 `doc/bin` 디렉토리로 이동하십시오.
     c. 다음과 같이 `help_start` 스크립트를 실행하십시오.

        ```
        help_start
        ```

   시스템의 기본 웹 브라우저가 열리고 독립형 정보 센터가 표시됩니다.
3. **갱신** 단추( )를 누르십시오. (JavaScript™가 브라우저에서 사용 가능해야 합니다.) 정보 센터의 오른쪽 패널에서 **갱신사항 찾기**를 누르십시오. 기존 문서의 갱신사항 목록이 표시됩니다.
4. 설치 프로세스를 시작하려면 설치할 선택란을 체크한 후 **갱신사항 설치**를 누르십시오.
5. 설치 프로세스가 완료되면 **완료**를 누르십시오.

6. 독립형 정보 센터를 중지하십시오.
   - Windows의 경우, 설치 디렉토리의 doc¥bin 디렉토리로 이동한 후 다음과 같이 help_end.bat 파일을 실행하십시오.

     ```
     help_end.bat
     ```

     주: help_end 일괄처리 파일에는 help_start 일괄처리 파일로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start.bat를 중지할 때 Ctrl+C 또는 다른 메소드를 사용하지 마십시오.
   - Linux의 경우, 설치 디렉토리의 doc/bin 디렉토리로 이동한 후 다음과 같이 help_end 스크립트를 실행하십시오.

     ```
     help_end
     ```

     주: help_end 스크립트에는 help_start 스크립트로 시작된 프로세스를 안전하게 중지하는 데 필요한 명령이 포함되어 있습니다. help_start 스크립트를 중지할 때 다른 메소드를 사용하지 마십시오.
7. DB2 정보 센터를 재시작하십시오.
   - Windows의 경우, **시작 → 제어판 → 관리 도구 → 서비스**를 누르십시오. 그런 다음 **DB2 Information Center** 서비스를 마우스 오른쪽 단추로 누른 후 **시작**을 선택하십시오.
   - Linux의 경우, 다음 명령을 입력하십시오.

     ```
     /etc/init.d/db2icdv97 start
     ```

갱신된 DB2 정보 센터에는 새로 갱신된 주제가 표시됩니다.

DB2 자습서

DB2 자습서는 DB2 제품의 여러가지 측면을 학습하는 데 유용합니다. 각 레슨은 단계별 지시사항을 제공합니다.

시작하기 전에

정보 센터(http://publib.boulder.ibm.com/infocenter/db2help/)에서 XHTML 버전의 자습서를 볼 수 있습니다.

일부 레슨에서는 샘플 데이터나 코드를 사용합니다. 특정 태스크에 필요한 전제조건 설명은 자습서를 참조하십시오.

DB2 자습서

자습서를 보려면 제목을 누르십시오.

『**pureXML®**』(*pureXML Guide*)

DB2 데이터베이스를 설정하여 XML 데이터를 저장하고 원시 XML 데이터 스토어로 기본 조작을 수행할 수 있습니다.

*Visual Explain 자습서*의 『**Visual Explain**』

더 나은 성능을 위해 Visual Explain을 사용하여 SQL문을 분석, 최적화 및 조정할 수 있습니다.

DB2 문제점 해결 정보

DB2 데이터베이스 제품 사용 시 발생하는 광범위한 문제점을 판별하고 해결하는 데 도움이 되는 정보를 사용할 수 있습니다.

DB2 문서

문제점 해결 정보는 *DB2 문제점 해결 안내서* 또는 *DB2 정보 센터*의 데이터베이스 기본 절을 참조하십시오. DB2 진단 도구 및 유틸리티를 사용하여 문제점을 찾아내고 식별하는 방법, 가장 일반적인 문제점에 대한 솔루션 및 DB2 데이터베이스 제품에서 발생할 수 있는 문제점을 해결하는 방법 등에 관한 정보가 있습니다.

DB2 기술 지원 웹 사이트

문제점이 발생한 경우 해당 원인 및 솔루션을 찾으려면 DB2 기술 지원 웹 사이트를 참조하십시오. 기술 지원 사이트에는 최신 DB2 서적, 기술 노트, APAR(Authorized Program Analysis Report 또는 버그 수정), FixPack 및 기타 자원에 대한 링크가 있습니다. 이러한 기술 자료를 검색하여 문제에 대해 사용 가능한 솔루션을 찾을 수 있습니다.

다음은 DB2 기술 지원 웹 사이트의 주소입니다. http://www.ibm.com/software/data/db2/support/db2_9/

이용약관

다음 조건에 따라 이 책을 사용할 수 있습니다.

개인적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 이 책을 개인적, 비상업적 용도로 복제할 수 있습니다. IBM의 명시적인 동의 없이는 이 책 또는 그 일부를 배포 또는 전시하거나 2차적 저작물을 만들 수 없습니다.

상업적 사용: 모든 소유권 사항을 표시하는 경우에 한하여 귀하는 이 책을 귀하 기업 집단 내에서만 복제, 배포 및 전시할 수 있습니다. 귀하는 IBM의 명시적 동의 없이 이 책의 2차적 저작물을 만들거나 이 책 또는 그 일부를 복제, 배포 또는 전시할 수 없습니다.

본 허가에서 명시적으로 부여된 경우를 제외하고, 이 책이나 이 책에 포함된 정보, 데이터, 소프트웨어 또는 기타 지적 재산권에 대한 어떠한 허가나 라이센스 또는 권한도 명시적 또는 묵시적으로 부여되지 않습니다.

IBM은 이 책의 사용이 IBM의 이익을 해친다고 판단되거나 위에서 언급된 지시사항이 준수되지 않는다고 판단하는 경우 언제든지 이 사이트에서 부여한 허가를 철회할 수 있습니다.

귀하는 미국 수출법 및 관련 규정을 포함하여 모든 적용 가능한 법률 및 규정을 철저히 준수하는 경우에만 본 정보를 다운로드, 송신 또는 재송신할 수 있습니다.

IBM은 이 책의 내용에 대해 어떠한 보증도 제공하지 않습니다. 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증 없이 현 상태대로 제공합니다.

부록 D. 주의사항

이 정보는 미국에서 제공되는 제품 및 서비스용으로 작성된 것입니다. 비IBM 제품에 대한 정보는 이 책을 처음 발행할 때의 정보에 기초하고 있으며 변경될 수 있습니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이센스까지 부여하는 것은 아닙니다. 라이센스에 대한 의문사항은 다음으로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠 주식회사
고객만족센터
전화번호: 080-023-8080

2바이트 문자 세트(DBCS) 정보에 관한 라이센스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
3-2-12, Roppongi, Minato-ku, Tokyo 106-8711 Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여 (단, 이에 한하지 않음) 묵시적이든 명시적이든 어떠한 종류의 보증없이 이 책을 『현상태대로』 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 변경된 사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지 없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 언급되는 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

IBM은 귀하의 권리를 침해하지 않는 범위 내에서 적절하다고 생각하는 방식으로 귀하가 제공한 정보를 사용하거나 배포할 수 있습니다.

(i) 독자적으로 작성된 프로그램과 다른 프로그램(본 프로그램 포함) 간의 정보 교환 및 (ii) 교환된 정보의 상호 이용을 목적으로 본 프로그램에 관한 정보를 얻고자 하는 라이센스 사용자는 다음 주소로 문의하십시오.

135-700
서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩
한국 아이.비.엠. 주식회사
고객만족센터

이러한 정보는 해당 조건(예를 들면, 사용료 지불 등) 하에서 사용될 수 있습니다.

이 정보에 기술된 라이센스가 부여된 프로그램 및 프로그램에 대해 사용 가능한 모든 라이센스가 부여된 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이센스 계약(IPLA) 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 측정치는 개발 레벨 시스템에서 작성되었을 수 있으며, 따라서 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 책의 사용자는 해당 데이터를 본인의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 다른 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM이 제시하는 방향 또는 의도에 관한 모든 언급은 특별한 통지 없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위하여 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이센스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 되는 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 IBM에 추가 비용을 지불하지 않고 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 따라서 IBM은 이러한 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증하거나 진술하지 않습니다. 샘플 프로그램은 어떠한 보증없이 ″있는 그대로″ 제공됩니다. IBM은 샘플 프로그램의 사용으로 인해 발생하는 모든 손해에 대해 책임을 지지 않습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.

© (*귀하의 회사명*) (연도). 이 코드의 일부는 IBM Corp.의 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. *_enter* 연도_. All rights reserved.

상표

IBM, IBM 로고 및 ibm.com®은 여러 국가에 등록된 International Business Machines Corp.의 상표 또는 등록상표입니다. 기타 제품 및 서비스 이름은 IBM 또는 기타 회사의 상표입니다. 현재 IBM 상표 목록은 웹 “저작권 및 상표 정보”(www.ibm.com/legal/kr/copytrade.shtml)에 있습니다.

다음 용어는 기타 회사의 상표 또는 등록상표입니다.

- Linux는 미국 또는 기타 국가에서 사용되는 Linus Torvalds의 등록상표입니다.
- Java™ 및 모든 Java 기반 상표는 미국 또는 기타 국가에서 사용되는 Sun Microsystems, Inc.의 상표입니다.
- UNIX는 미국 또는 기타 국가에서 사용되는 The Open Group의 등록상표입니다.
- Intel®, Intel 로고, Intel Inside®, Intel Inside 로고, Intel® Centrino®, Intel Centrino 로고, Celeron®, Intel® Xeon®, Intel SpeedStep®, Itanium® 및 Pentium®은 미국 또는 기타 국가에서 사용되는 Intel Corporation의 상표 또는 등록상표입니다.
- Microsoft, Windows, Windows NT® 및 Windows 로고는 미국 또는 기타 국가에서 사용되는 Microsoft Corporation의 상표입니다.

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

색인

[가]

[나]

[다]

[라]

[마]

[바]

[사]

[아]

[자]

[차]

[카]

[타]

[파]

[하]

A

B

C

D

F

G

H

I

K

L

M

N

R

S

T

U

V

X

SA30-3958-00

Spine information:

Linux, UNIX 및 Windows용 IBM DB2 9.7

관리 API 참조서