

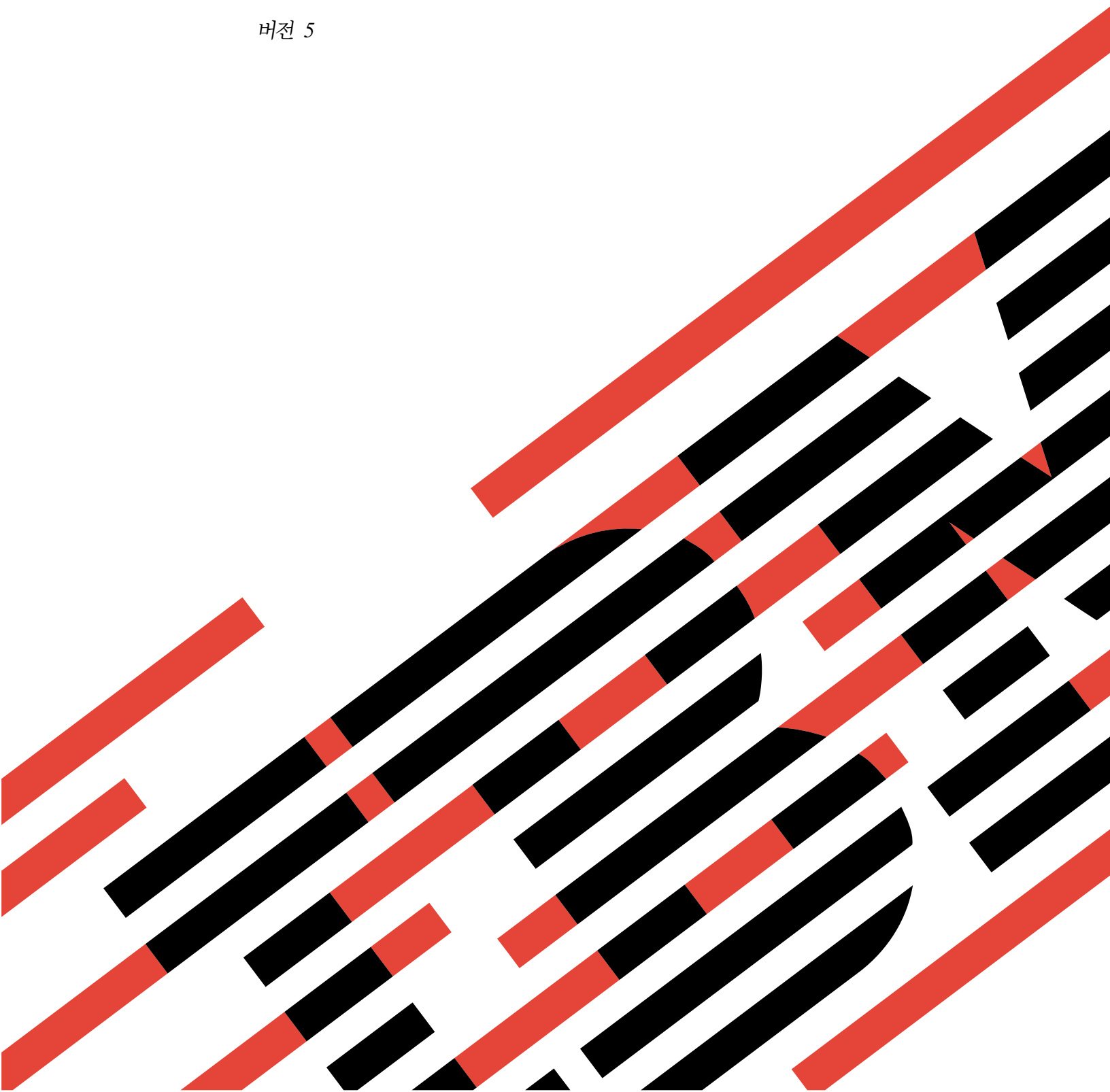
IBM

@server

iSeries

호스트 언어를 사용하는 iSeries용
DB2 Universal Database SQL 프로그래밍

버전 5





@server

iSeries

호스트 언어를 사용하는 iSeries용

DB2 Universal Database SQL 프로그래밍

버전 5

목차

호스트 언어를 사용하는 iSeries용 DB2 UDB SQL 프로그래밍 제품 정보 ix

호스트 언어를 사용하는 SQL 프로그래밍 책의 사용자. ix

호스트 언어를 사용하는 SQL 프로그래밍 책에서 SQL문의 예와 관련된 가정 ix

 코드 면책사항 정보 x

호스트 언어를 사용하는 SQL 프로그래밍 책에서 구문 다이어그램의 해석 방법 xi

호스트 언어를 사용하는 SQL 프로그래밍 버전 5 릴리스 2 책의 새로운 사항. xii

제 1 장 호스트 언어로 SQL 사용을 위한 일반적인 개념 및 규칙 1

SQL을 사용하는 어플리케이션 작성. 1

SQL문에서 호스트 변수 사용. 3

 SQL문에서 호스트 변수에 대한 지정 규칙 3

 SQL을 사용하는 어플리케이션에서 인디케이터 변수. 6

SQL 오류 리턴 코드 처리. 8

WHENEVER문에 대한 예외 조건 처리 9

제 2 장 C 및 C++ 어플리케이션에서 SQL문 코딩 13

SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL 통신 영역 정의. 13

SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL 설명자의 영역 정의. 15

SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL 문 삽입. 17

 SQL을 사용하는 C 및 C++ 어플리케이션의 주식 18

 SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL문의 계속 18

 SQL을 사용하는 C 및 C++ 어플리케이션의 포함 코드. 19

 SQL을 사용하는 C 및 C++ 어플리케이션의 여백 19

 SQL을 사용하는 C 및 C++ 어플리케이션의 이름 19

 SQL을 사용하는 C 및 C++ 어플리케이션의 NULL 및 NULS 19

 SQL을 사용하는 C 및 C++ 어플리케이션의 명령문 레이블. 19

 SQL을 사용하는 C 및 C++ 어플리케이션에 대한 사전처리기 순서. 20

SQL을 사용하는 C 및 C++ 어플리케이션의 삼중 문자. 20

SQL을 사용하는 C 및 C++ 어플리케이션의 WHENEVER문. 20

SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 변수 사용. 20

 SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 변수 선언 21

SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 사용. 32

 SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 선언 33

 SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 인디케이터 배열 36

SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조의 배열 사용 36

 SQL을 사용하는 C 및 C++ 어플리케이션의 호스트 구조 배열. 37

 SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 배열 인디케이터 구조 39

SQL을 사용하는 C 및 C++ 어플리케이션에서 포인터 자료 유형 사용 40

SQL을 사용하는 C 및 C++ 어플리케이션의 typedef 41

SQL을 사용하는 C 및 C++ 어플리케이션에서 ILE C 컴파일러 외부 파일 설명 사용 42

동등한 SQL 및 C 또는 C++ 자료 유형 판별. . . . 43

 C 및 C++ 변수 선언 및 사용에 대한 주 46

SQL을 사용하는 C 및 C++ 어플리케이션에서 인디케이터 변수 사용 46

제 3 장 COBOL 어플리케이션에서의 SQL문 코딩 49

SQL을 사용하는 COBOL 어플리케이션에서 SQL 통신 영역 정의. 49

SQL을 사용하는 COBOL 어플리케이션에서 SQL 설명자 영역 정의 51

SQL을 사용하는 COBOL 어플리케이션에서 SQL문 삽입. 52

 SQL을 사용하는 COBOL 어플리케이션의 주식 53

 SQL을 사용하는 COBOL 어플리케이션에서 SQL문의 연속 53

 SQL을 사용하는 COBOL 어플리케이션에서 포함하는 코드. 53

SQL을 사용하는 COBOL 어플리케이션의 여백	53
SQL을 사용하는 COBOL 어플리케이션에서 순번	54
SQL을 사용하는 COBOL 어플리케이션에서의 이름	54
SQL을 사용하는 COBOL 어플리케이션에서 COBOL 컴파일시 옵션	54
SQL을 사용하는 COBOL 어플리케이션에서 명령문 레이블	54
SQL을 사용하는 COBOL 어플리케이션에서 WHENEVER문	54
복수 소스 COBOL 프로그램 및 SQL COBOL 사전컴파일러	54
SQL을 사용하는 COBOL 어플리케이션에서 호스트 변수 사용	54
SQL을 사용하는 COBOL 어플리케이션에서 호스트 변수 선언	55
SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 사용	64
SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조	65
SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 인디케이터 배열	68
SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 배열 사용	68
SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 배열	69
SQL을 사용하는 COBOL 어플리케이션에서 호스트 배열 인디케이터 구조	72
SQL을 사용하는 COBOL 어플리케이션에서 외부 파일 설명 사용	73
SQL을 사용하는 COBOL 어플리케이션에서 외부 파일 설명 사용	73
동등한 SQL 및 COBOL 자료 유형 판별	74
COBOL 변수 선언 및 사용에 관한 주	77
SQL을 사용하는 COBOL 어플리케이션에서 인디케이터 변수 사용	78
제 4 장 PL/I 어플리케이션에서의 SQL문 코딩	79
SQL을 사용하는 PL/I 어플리케이션에서 SQL 통신 영역 정의	79
SQL을 사용하는 PL/I 어플리케이션에서 SQL 설명자 영역 정의	80
SQL을 사용하는 PL/I 어플리케이션에서 SQL문 삽입	81
예: SQL을 사용하는 PL/I 어플리케이션에서 SQL문 삽입	82
SQL을 사용하는 PL/I 어플리케이션에서 주석	82

SQL을 사용하는 PL/I 어플리케이션에서 SQL문의 연속	82
SQL을 사용하는 PL/I 어플리케이션에서 포함 코드	82
SQL을 사용하는 PL/I 어플리케이션에서 여백	82
SQL을 사용하는 PL/I 어플리케이션에서 이름	83
SQL을 사용하는 PL/I 어플리케이션에서 명령문 레이블	83
SQL을 사용하는 PL/I 어플리케이션에서 WHENEVER문	83
SQL을 사용하는 PL/I 어플리케이션에서 호스트 변수 사용	83
SQL을 사용하는 PL/I 어플리케이션에서 호스트 변수 선언	83
SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 사용	88
SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조	89
SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 인디케이터 배열	90
SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 배열 사용	91
SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 배열	92
SQL을 사용하는 PL/I 어플리케이션에서 외부 파일 설명 사용	93
동일한 SQL 및 PL/I 자료 유형 판별	94
SQL을 사용하는 PL/I 어플리케이션에서 인디케이터 변수 사용	96
구조 매개변수 전달 기법에 따른 PL/I의 차이점	97
제 5 장 iSeries용 RPG 어플리케이션에서의 SQL문 코딩	99
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL 통신 영역 정의	100
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL 설명자 영역 정의	100
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL문 삽입	101
예: SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL문 삽입	102
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 주석	102
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL문의 연속	102
SQL을 사용하는 iSeries용 RPG 어플리케이션에 코드 포함	102

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 순번	103
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 이름	103
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 명령문 레이블	103
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 WHENEVER문	103
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 변수 사용	103
SQL을 사용하는 iSeries용 RPG에서 호스트 변수 사용	103
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 구조 사용	104
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 구조 배열 사용	105
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 외부 파일 설명 사용	106
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 구조 배열에 대한 외부 파일 설명 고려사항.	107
동등한 SQL 및 iSeries용 RPG 자료 유형 판별	107
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 iSeries용 RPG 변수 선언 및 사용에 대한 주	110
SQL을 사용하는 iSeries용 RPG 어플리케이션에서 인디케이터 변수 사용.	110
예: SQL을 사용하는 iSeries용 RPG 어플리케이션에서 인디케이터 변수 사용	110
구조 매개변수 전달 기법에 따른 iSeries용 RPG의 차이점.	111
SQL을 사용하는 호출된 iSeries용 RPG 프로그램의 올바른 종료.	111
제 6 장 iSeries용 ILE RPG 어플리케이션에서의 SQL문 코딩	113
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL 통신 영역 정의.	114
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL 설명자 영역 정의	115
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL문 삽입.	116
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 주석	117
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL문의 연속.	117

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에 코드 포함	117
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 지시문 사용	118
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 순번	118
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 이름	118
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 명령문 레이블	118
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 WHENEVER문	118
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 변수 사용	118
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 변수 선언.	119
SQL을 사용하는 iSeries용 ILE RPG에서 호스트 구조 사용	120
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 구조 사용	121
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 호스트 변수 선언	122
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 호스트 변수	122
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 위치 지정자	123
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 파일 참조 변수	124
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 ROWID 변수	125
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 외부 파일 설명 사용	125
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 구조 배열에 대한 외부 파일 설명 고려사항.	126
동등한 SQL 및 RPG 자료 유형 판별	127
iSeries용 ILE RPG 변수 선언 및 사용에 대한 주	131
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 인디케이터 변수 사용.	131
예: SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 인디케이터 변수 사용	131
SQL에서 사용하는 iSeries용 ILE RPG 어플리케이션에서 복수 행 영역 페치에 대한 SQLDA 예	132
SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 동적 SQL 예	133

제 7 장 REXX 어플리케이션에서의 SQL문 코드	135
REXX 어플리케이션에서 SQL 통신 영역 사용	135
REXX 어플리케이션에서 SQL 설명자 영역 사용	136
REXX 어플리케이션에서 SQL문 삽입	139
SQL을 사용하는 REXX 어플리케이션에서 주석	140
SQL을 사용하는 REXX 어플리케이션에서 SQL문의 연속	140
SQL을 사용하는 REXX 어플리케이션에서 포함하는 코드	140
SQL을 사용하는 REXX 어플리케이션에서 여백	140
SQL을 사용하는 REXX 어플리케이션에서 이름	140
SQL을 사용하는 REXX 어플리케이션에서 널 (null)	140
SQL을 사용하는 REXX 어플리케이션에서 명령문 레이블	140
SQL을 사용하는 REXX 어플리케이션에서 오류 및 경고 처리	140
SQL을 사용하는 REXX 어플리케이션에서 호스트 변수 사용	141
SQL을 사용하는 REXX 어플리케이션에서 입력 호스트 변수의 자료 유형 판별	142
SQL을 사용하는 REXX 어플리케이션에서 출력 호스트 변수의 형식	143
SQL을 사용하는 REXX 어플리케이션에서 REXX 변환 회피	143
SQL을 사용하는 REXX 어플리케이션에서 인디케이터 변수 사용	144
제 8 장 SQL문을 사용하는 프로그램 준비 및 실행	145
SQL 사전컴파일러의 기본 처리	145
SQL 사전컴파일러에 대한 입력	146
SQL 사전컴파일러에서 소스 파일 CCSID	147
SQL 사전컴파일러로부터 출력	147
비ILE SQL 사전컴파일러 명령	153
SQL을 사용하는 비ILE 어플리케이션 프로그램 컴파일링	153
ILE SQL 사전컴파일러 명령	154
SQL을 사용하는 ILE 어플리케이션 프로그램 컴파일링	154
SQL을 사용하는 어플리케이션에서 컴파일 오류 해석	156
SQL을 사용하는 어플리케이션 프로그램의 컴파일시 오류 및 경고 메시지	156
SQL을 사용하는 어플리케이션 바인딩	157
SQL을 사용하는 어플리케이션에서 프로그램 참조	158

SQL 사전컴파일러 옵션 표시	159
삽입된 SQL을 사용하는 프로그램 실행	159
삽입된 SQL을 사용하는 프로그램 실행: OS/400 DDM 고려사항	159
삽입된 SQL을 사용하는 프로그램 실행: 대체 고려사항	159
삽입된 SQL을 사용하는 프로그램 실행: SQL 리턴 코드	160

부록 A. iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램	161
예: ILE C 및 C++ 프로그램에서 SQL문	163
예: COBOL과 ILE COBOL 프로그램에서 SQL문	168
예: PL/I에서 SQL문	176
예: iSeries용 RPG 프로그램에서 SQL문	181
예: iSeries용 ILE RPG 프로그램에서 SQL문	187
예: REXX 프로그램에서 SQL문	193
SQL을 사용하는 샘플 프로그램에 의해 작성된 보고서	197

부록 B. 호스트 언어 사전컴파일러에 대한 iSeries용 DB2 UDB CL 명령 설명	199
SQL 사전컴파일러 명령	199
CRTSQLCBL(구조화 조회 언어 COBOL 작성) 명령	199
CRTSQLCBLI(SQL ILE COBOL 오브젝트 작성) 명령	217
CRTSQLCI(구조화 조회 언어 ILE C 오브젝트 작성) 명령	236
CRTSQLCPPI(구조화 조회 언어 C++ 오브젝트 작성) 명령	256
CRTSQLPLI(구조화 조회 언어 PL/I 작성) 명령	274
CRTSQLRPG(구조화 조회 언어 RPG 작성) 명령	292
CRTSQLRPGI(SQL ILE RPG 오브젝트 작성) 명령	310

부록 C. iSeries용 FORTRAN 사전컴파일러 사용	331
FORTRAN/400 사전컴파일러 사용	331
CRTSQLFTN(구조화 조회 언어 FORTRAN 작성) 명령	331

부록 D. FORTRAN 어플리케이션에서 SQL문 코드	349
FORTRAN 어플리케이션에서 SQL 통신 영역 정의	349
FORTRAN 어플리케이션에서 SQL 설명자 영역 정의	351
FORTRAN 어플리케이션에서 SQL문 삽입	351

SQL을 사용하는 FORTRAN 어플리케이션에서 주석	352
SQL을 사용하는 FORTRAN 어플리케이션에서 행 디버그	352
SQL을 사용하는 FORTRAN 어플리케이션에서 SQL문의 연속	352
SQL을 사용하는 FORTRAN 어플리케이션에서 코드 포함	353
SQL을 사용하는 FORTRAN 어플리케이션에서 여백	353
SQL을 사용하는 FORTRAN 어플리케이션에서 이름	353
SQL을 사용하는 FORTRAN 어플리케이션에서 명령문 레이블	353

SQL을 사용하는 FORTRAN 어플리케이션에서 WHENEVER문	354
SQL 사전컴파일러에서 FORTRAN 컴파일시 옵션	354
FORTRAN 어플리케이션에서 호스트 변수 사용	354
FORTRAN 어플리케이션에서 호스트 변수 선언	355
동일한 SQL 및 FORTRAN 자료 유형 판별	356
FORTRAN 변수 선언 및 사용에 대한 주	357
FORTRAN 어플리케이션에서 인디케이터 변수 사 용	357
색인	359

호스트 언어를 사용하는 iSeries용 DB2 UDB SQL 프로그래밍 제품 정보

이 책에서는 프로그래머와 관리자가 iSeries용 DB2 UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성하는 방법을 설명합니다.

어플리케이션 프로그래밍 환경에서 iSeries용 DB2 UDB SQL 지침 및 구현의 예에 대한 자세한 정보는 Information Center의 데이터베이스 범주에서 다음 책을 참조하십시오.

- SQL 참조
- SQL 프로그래밍 개념
- 데이터베이스 성능 및 조회 최적화
- SQL 호출 레벨 인터페이스(ODBC)

이 안내서에 대한 자세한 정보는 다음 절을 참조하십시오.

- 『호스트 언어를 사용하는 SQL 프로그래밍 책의 사용자』
- 『호스트 언어를 사용하는 SQL 프로그래밍 책에서 SQL문의 예와 관련된 가정』
- xi 페이지의 『호스트 언어를 사용하는 SQL 프로그래밍 책에서 구문 다이어그램의 해석 방법』
- xii 페이지의 『호스트 언어를 사용하는 SQL 프로그래밍 버전 5 릴리스 2 책의 새로운 사항』

호스트 언어를 사용하는 SQL 프로그래밍 책의 사용자

이 책은 iSeries용 COBOL, iSeries용 ILE COBOL, iSeries PL/I, iSeries용 ILE C, ILE C++, REXX, RPG III(iSeries용 RPG의 일부) 또는 iSeries용 ILE RPG 언어에 익숙하여 프로그래밍할 수 있고 기본데이터베이스 어플리케이션을 이해할 수 있는 어플리케이션 프로그래머 및 데이터베이스 관리자가 사용해야 합니다.

호스트 언어를 사용하는 SQL 프로그래밍 책에서 SQL문의 예와 관련된 가정

이 책에 나오는 SQL문의 예는 iSeries Information Center에 있는 SQL 프로그래밍 개요 책의 부록 A, "iSeries용 DB2 UDB 샘플 표를 기초로 한 것으로서 다음을 가정합니다.

- SQL문은 대화식 SQL 환경에서 표시되거나 ILE C 또는 COBOL로 작성됩니다. EXEC SQL과 END-EXEC는 COBOL 프로그램 내에서 SQL문을 구분하는 데 사용됩니다. COBOL 프로그램에서 SQL문을 사용하는 방법에 대해서는 49 페이지의

제 3 장 『COBOL 어플리케이션에서의 SQL문 코딩』에 설명되어 있습니다. ILE C 프로그램에서 SQL문을 사용하는 방법에 대해서는 13 페이지의 제 2 장 『C 및 C++ 어플리케이션에서 SQL문 코딩』에 설명되어 있습니다.

- 각 SQL 예는 여러 행으로 표시되며 명령문의 각 절은 하나의 개별 행으로 표시됩니다.
- SQL 키워드는 강조표시됩니다.
- 샘플 표에 있는 표 이름은 컬렉션 CORPDATA를 사용합니다. 이 샘플 표에 없는 표 이름은 사용자가 작성한 컬렉션을 사용해야 합니다. 이러한 표 정의 및 표를 작성하는 방법에 대해서는 SQL 프로그래밍 개념 책의 부록 A, "iSeries용 DB2 UDB 샘플 표"를 참조하십시오.
- 계산되는 열은 괄호 ()와 브라켓 []으로 묶입니다.
- SQL 명명 규칙이 사용됩니다.
- COBOL에서는 APOST와 APOSTSQL 사전컴파일러 옵션을 디폴트 옵션으로 가정합니다. SQL 및 호스트 언어 명령문 내의 문자 스트링 리터럴은 작은 따옴표(')로 구분됩니다.
- 다른 주(note)가 없는 한, *HEX의 정렬 순서가 이용됩니다.
- 대부분의 예에는 SQL문의 완전한 구문이 표시되어 있지 않습니다. 이 안내서에 설명된 명령문에 대한 자세한 설명 및 구문에 대한 내용은 SQL 참조서를 참조하십시오.

이러한 가정은 예가 나올 때마다 언급됩니다.

이 책은 어플리케이션 프로그래머를 위한 것이므로 대부분의 예들이 어플리케이션 프로그램에서 작성된 것처럼 표시됩니다. 그러나 많은 예에 있어서 약간씩 변경한 후 대화식 SQL을 이용하여 대화식으로 수행시킬 수 있습니다. 대화식 SQL 사용시, SQL문의 구문은 동일한 명령문이 프로그램에 삽입될 때의 형식과 약간 차이가 납니다.

프로그램 예 사용에 대한 자세한 정보는 『코드 면책사항 정보』를 참조하십시오.

코드 면책사항 정보

이 자료에는 프로그래밍 예들이 수록되어 있습니다.

IBM®은 귀하가 자신의 특정 요구에 맞게 유사한 기능을 생성한 모든 프로그래밍 코드 예를 사용함에 있어서 비독점적 저작권을 귀하에게 부여합니다.

IBM이 제공하는 모든 샘플 코드는 예시용입니다. 이 예들은 모든 환경에서 완벽하게 테스트가 이루어진 것이 아닙니다. 따라서 IBM은 이 프로그램의 신뢰성, 서비스 가능성 또는 기능을 보증 또는 암시하지 않습니다.

여기에 수록된 모든 프로그램들은 어떠한 종류의 보증없이 "현상태대로" 제공합니다. 비침해, 상품성, 특정 목적에의 적합성에 대한 암묵적 보증을 명시적으로 부인합니다.

호스트 언어를 사용하는 SQL 프로그래밍 책에서 구문 다이어그램의 해석 방법

이 책에서는 다음과 같은 구조를 사용하여 구문을 서술합니다.

- 구문 다이어그램은 왼쪽에서 오른쪽으로, 위에서 아래로, 그리고 다음과 같은 선의 경로를 따라서 읽도록 하십시오.

▶▶ — 기호는 명령문의 시작을 나타냅니다.

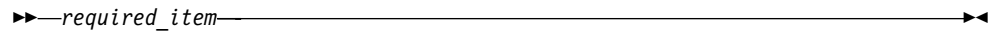
—▶ 기호는 명령문 구문이 다음 행에 계속됨을 나타냅니다.

▶ — 기호는 명령문이 이전 행에서 계속된 것임을 나타냅니다.

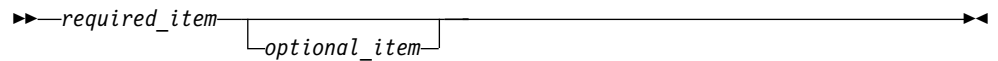
—▶▶ 기호는 명령문의 끝을 나타냅니다.

명령문 전체를 나타내는 경우가 아닌 구문 단위의 다이어그램에서는 ▶ — 기호로 시작하여 —▶ 기호로 끝납니다.

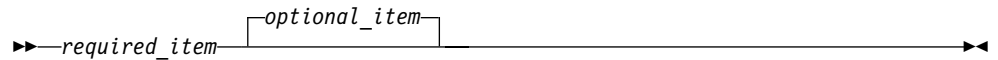
- 필수 항목은 수평선(기본 경로)상에 표시됩니다.



- 선택적인 항목은 기본 경로 밑에 표시됩니다.

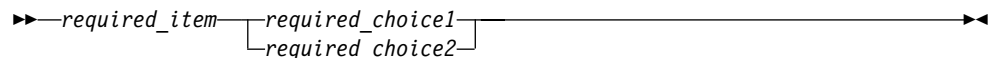


선택적인 항목이 기본 경로 위에 표시되면 그 항목은 명령문의 실행에 아무런 영향을 미치지 않으며 읽기용으로만 사용되는 것입니다.

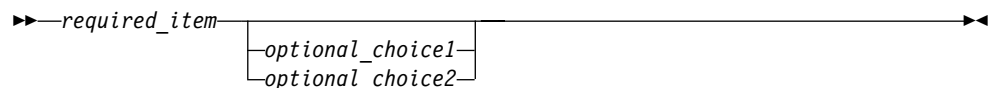


- 둘 이상의 항목에서 선택할 경우에는 항목들이 세로로 스택에 표시됩니다.

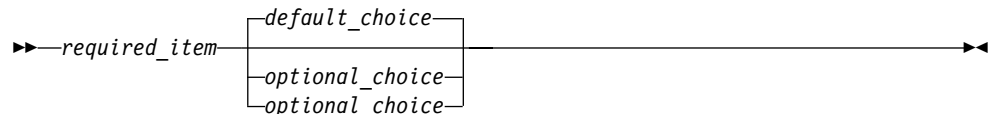
항목 중 하나를 선택해야 하는 경우 스택의 한 항목이 기본 경로에 표시됩니다.



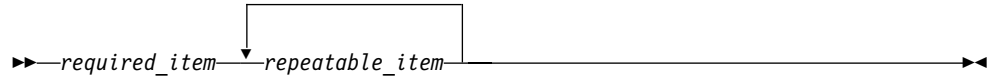
항목 중에서 반드시 하나를 선택하지 않아도 되는 경우에는 전체 스택이 기본 경로 밑에 표시됩니다.



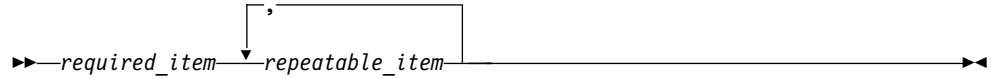
항목 중 하나가 디폴트이면 그 디폴트는 기본 경로 위에 표시되고 나머지 선택사항들은 기본 경로 밑에 표시됩니다.



- 회살표가 기본 행의 위를 통해 왼쪽으로 되돌아 오면 항목이 반복될 수 있음을 나타냅니다.



반복 화살표에 쉼표가 들어 있으면 반복되는 항목들을 쉼표로 분리해야 합니다.



스택 위의 반복 화살표는 스택 내의 항목들을 반복할 수 있음을 나타냅니다.

- 키워드는 대문자로 표시되며(예를 들어 FROM), 표시된 대로 철자를 정확히 입력해야 합니다. 변수는 모두 영문 소문자로 표시되며(예를 들어 *column-name*) 사용자 제공 이름이나 값을 나타냅니다.
- 구두점, 괄호, 산술 연산자 또는 기타 그와 같은 기호들이 표시되면 구문의 한 부분으로서 반드시 입력해야 합니다.

호스트 언어를 사용하는 SQL 프로그래밍 버전 5 릴리스 2 책의 새로운 사항

C, C++, COBOL, ILE COBOL, ILE RPG, 및 PL/I에 대한 ROWID 신규 호스트 변수 유형에 대해서는 다음을 참조하십시오.

- 32 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 ROWID 호스트 변수』
- 63 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 ROWID 호스트 변수』
- 87 페이지의 『SQL을 사용하는 PL/I 어플리케이션의 ROWID 호스트 변수』
- 125 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 ROWID 변수』

C/C++에 대한 SQL VARCHAR 유형에 관한 자세한 정보는 21 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 문자 호스트 변수』를 참조하십시오.

제 1 장 호스트 언어로 SQL 사용을 위한 일반적인 개념 및 규칙

이 장에서는 다음을 포함하는 호스트 언어로 SQL문 사용에 대한 일반적인 일부 개념 및 규칙을 설명합니다.

- SQL문에서 호스트 변수 사용
- SQL 오류 및 리턴 코드 처리
- WHENEVER문에 대한 예외 조건 처리

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

SQL을 사용하는 어플리케이션 작성

iSeries용 DB2 UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성할 수 있습니다. 각각의 호스트 언어에 대한 어플리케이션 및 코딩 요구 사항에 대한 자세한 내용은 다음을 선택하여 참조하십시오.

- 13 페이지의 제 2 장 『C 및 C++ 어플리케이션에서 SQL문 코딩』
- 49 페이지의 제 3 장 『COBOL 어플리케이션에서의 SQL문 코딩』
- 79 페이지의 제 4 장 『PL/I 어플리케이션에서의 SQL문 코딩』
- 99 페이지의 제 5 장 『iSeries용 RPG 어플리케이션에서의 SQL문 코딩』
- 113 페이지의 제 6 장 『iSeries용 ILE RPG 어플리케이션에서의 SQL문 코딩』
- 135 페이지의 제 7 장 『REXX 어플리케이션에서의 SQL문 코딩』
- 145 페이지의 제 8 장 『SQL문을 사용하는 프로그램 준비 및 실행』

주: 호스트 언어로 Java™ 사용에 대한 정보는 Java용 IBM Developer Kit를 참조하십시오.

SQL문에서 호스트 변수 사용

프로그램이 자료를 검색할 때 값은 프로그램에 의해 정의된 자료항목에 놓여지고 SELECT INTO문이나 FETCH문의 INTO절과 함께 지정됩니다. 자료 항목은 호스트 변수라고 합니다.

호스트 변수는 SQL문에 지정된 프로그램 내의 필드로서 일반적으로 열 값의 소스나 목표 표를 말합니다. 호스트 변수와 열은 반드시 호환되는 자료 유형이어야 합니다. 호스트 변수는 DESCRIBE TABLE문을 제외하고는 표나 뷰와 같은 SQL 오브젝트 식별에 사용될 수 없을 수 있습니다.

호스트 구조는 선택된 값 세트에 대한 소스나 목표로 사용되는 호스트 변수의 그룹입니다(예를 들어 한 행의 열에 대한 값의 세트). 호스트 구조 배열은 복수 행 FETCH 및 블록화 INSERT문에 사용되는 호스트 구조의 배열입니다.

주: SQL문 내에 리터럴 값 대신 호스트 변수를 사용하면 어플리케이션 프로그램이 표나 뷰 내의 다른 행을 처리하는 데 필요한 융통성을 갖게 됩니다.

예를 들어 WHERE절에 실제 부서 번호를 코딩하는 대신 부서 번호에 대한 호스트 변수 세트를 사용할 수 있습니다.

호스트 변수는 보통 다음과 같은 방법으로 SQL문에 사용됩니다.

1. **WHERE 절에서:** 호스트 변수를 사용하여 탐색 조건의 술부 값을 지정하거나 표현식의 리터럴 값을 대체할 수 있습니다. 예를 들어 사원 번호가 들어 있는 EMPID라는 필드를 정의한 경우 사원 번호가 000110인 직원명을 검색할 수 있습니다.

```
MOVE '000110' TO EMPID.  
EXEC SQL  
SELECT LASTNAME  
INTO :PGM-LASTNAME  
FROM CORPDATA.EMPLOYEE  
WHERE EMPNO = :EMPID  
END-EXEC.
```

2. **(INTO절에서 명명된) 열 값에 대한 수신 영역으로서:** 호스트 변수를 사용하여 검색된 행의 열 값을 포함할 프로그램 자료 영역을 지정할 수 있습니다. INTO절은 SQL에 의해 리턴된 열 값을 포함시키려는 하나 이상의 호스트 변수를 명명합니다. 예를 들어 EMPNO, LASTNAME 및 WORKDEPT 열 값을 CORPDATA.EMPLOYEE 표의 행에서 검색중이라고 생각해 보겠습니다. 사용자는 각 열을 보유하기 위해 프로그램에서 호스트 변수를 정의할 수 있으며 이때 INTO절로 호스트 변수를 명명할 수 있습니다. 예를 들면 다음과 같습니다.

```
EXEC SQL  
SELECT EMPNO, LASTNAME, WORKDEPT  
INTO :CBLEMPNO, :CBLNAME, :CBLDEPT  
FROM CORPDATA.EMPLOYEE  
WHERE EMPNO = :EMPID  
END-EXEC.
```

이 예에서 호스트 변수 CBLEMPNO는 EMPNO로부터 값을 수신하고 CBLNAME은 LASTNAME으로부터 값을 수신하며 CBLDEPT는 WORKDEPT로부터 값을 수신합니다.

3. **SELECT절의 값으로서:** SELECT절에 항목 리스트를 지정할 때 뷰의 열 이름으로 제한되지 않습니다. 사용자 프로그램은 호스트 변수값 및 리터럴 상수로 혼합된 열 값 세트를 리턴할 수 있습니다. 예를 들면 다음과 같습니다.

```
MOVE '000220' TO PERSON.  
EXEC SQL  
SELECT "A", LASTNAME, SALARY, :RAISE,  
SALARY + :RAISE
```



```

INTO :PROCESS, :PERSON-NAME, :EMP-SAL,
      :EMP-RAISE, :EMP-TTL
FROM CORPDATA.EMPLOYEE
WHERE EMPNO = :PERSON
END-EXEC.

```

결과는 다음과 같습니다.

PROCESS	PERSON-NAME	EMP-SAL	EMP-RAISE	EMP-TTL
A	LUTZ	29840	4476	34316

4. SQL문의 다른 절에 있는 값으로서:

UPDATE문의 SET절

INSERT문의 VALUES절

CALL문

이러한 명령문에 대한 자세한 정보는 SQL 참조서를 참조하십시오.

호스트 변수 사용에 대한 자세한 정보는 다음 절을 참조하십시오.

- 『SQL문에서 호스트 변수에 대한 지정 규칙』
- 6 페이지의 『SQL을 사용하는 어플리케이션에서 인디케이터 변수』

SQL문에서 호스트 변수에 대한 지정 규칙

SQL 값은 FETCH, SELECT INTO, SET 및 VALUES INTO문 실행시 호스트 변수로 설정(또는 할당)됩니다. SQL 값은 INSERT, UPDATE 및 CALL문의 수행 중에 호스트 변수로부터 설정(또는 할당)됩니다. 모든 지정 조작은 다음 규칙을 준수해야 합니다.

- 숫자와 스트링은 호환할 수 없습니다.
 - 숫자는 스트링 열 또는 스트링 호스트 변수에 할당될 수 없습니다.
 - 스트링은 숫자 열 또는 숫자 호스트 변수에 할당될 수 없습니다.
- 모든 문자와 DBCS 그래픽 스트링은 CCSID간에 변환이 지원되는 경우 UCS-2 그래픽 열과 호환될 수 있습니다. 코드화 문자 세트 ID(CCSID)가 호환될 수 있으면 모든 그래픽 스트링도 호환될 수 있습니다. 숫자값은 모두 호환될 수 있습니다. 변환은 SQL에 의해 필요할 때마다 수행됩니다. 모든 문자와 DBCS 그래픽 스트링은 CCSID간 변환이 지원되는 경우 지정 조작에 있어서 UCS-2 그래픽 문자열과 호환될 수 있습니다. CALL문의 경우에 변환이 지원되면 문자 및 DBCS 그래픽 매개변수가 UCS-2 매개변수와 호환될 수 있습니다.
- 널(null)은 연관된 인디케이터 변수가 없는 호스트 변수에 할당될 수 없습니다.

- 다른 유형의 날짜/시간값은 호환되지 않습니다. 날짜는 날짜 또는 날짜의 스트링 표시에 대해서만 호환됩니다. 시간은 시간 또는 시간의 스트링 표시에 대해서만 호환됩니다. 시간소인은 시간소인 또는 시간소인의 스트링 표시에 대해서만 호환됩니다. 날짜는 날짜 열, 문자 열, DBCS 개방 열, DBCS 선택 열, 변수 또는 문자 변수에만 할당될 수 있습니다¹. 날짜 열의 삽입 또는 갱신 값은 날짜 또는 날짜의 문자열 표시이어야 합니다.

시간은 시간 열, 문자 열, DBCS 개방 열, DBCS 선택 열, 변수 또는 문자 변수에만 할당될 수 있습니다. 시간 열의 삽입 또는 갱신 값은 시간 또는 시간의 문자열 표시이어야 합니다.

시간소인은 시간소인 열, 문자 열, DBCS 개방 열, DBCS 선택 열이나 변수 또는 문자 변수에만 할당될 수 있습니다. 시간소인 열의 삽입 또는 갱신 값은 시간소인 또는 시간소인의 문자열 표시이어야 합니다.

SQL문에서 호스트 변수의 스트링 지정에 대한 규칙

문자 스트링 할당에 관한 규칙은 다음과 같습니다.

- 스트링이 열에 할당되면 스트링 값의 길이가 열의 길이 속성보다 클 수 없습니다. (후미 공백은 대개 스트링의 길이에 포함됩니다. 그러나 스트링 할당 후미 공백은 스트링의 길이에 포함되지 않습니다.)
- MIXED 문자 결과 열이 MIXED 열에 할당되면 MIXED 문자 결과 열 값이 유효한 MIXED 문자 스트링이어야 합니다.
- 결과 열 값이 호스트 변수에 할당되고 결과 열의 스트링 값이 호스트 변수의 길이 속성보다 길면 스트링은 필요한 문자 수 만큼 오른쪽에서 절단됩니다. 이 경우 SQLWARN0와 SQLWARN1(SQLCA 내의)이 W로 설정됩니다.
- 결과 열 값이 고정 길이 호스트 변수로 할당되거나 호스트 변수가 고정 길이 CHAR 결과 열에 할당되고 스트링 값의 길이가 목표의 길이 속성보다 작으면 스트링은 오른쪽이 필요한 수만큼 공백으로 채워집니다.
- 할당중인 호스트 변수의 길이가 스트링 길이보다 작아서 MIXED 문자 결과 열이 절단되지만 스트링 끝에 있는 SI 문자는 보존됩니다. 따라서 그 결과는 계속 유효한 MIXED 문자 스트링입니다.

SQL문에서 호스트 변수의 CCSID에 대한 규칙

한 문자 또는 그래픽 값을 다른 값에 할당할 때 CCSID를 고려해야 합니다. 여기에는 호스트 변수 할당이 포함됩니다. 데이터베이스 관리자는 SBCS, DBCS, MIXED 및 그래픽 자료에 대해 일련의 공동 시스템 서비스를 사용합니다.

CCSID에 대한 규칙은 다음과 같습니다.

1. DBCS 개방 또는 DBCS 선택 변수는 외부 서술 파일의 정의를 포함하여 호스트 언어에 선언된 변수입니다. 작업 CCSID가 혼합 자료를 표시하거나 DECLARE VARIABLE문이 사용되고 MIXED CCSID 또는 FOR MIXED DATA절이 지정되면 DBCS 개방 변수도 선언됩니다. SQL 참조서에서 DECLARE VARIABLE을 참조하십시오.

- 소스의 CCSID가 목표의 CCSID와 일치하면 변환없이 값이 할당됩니다.
- 소스나 목표의 부속유형이 BIT이면 변환 없이 값이 할당됩니다.
- 값이 널(null)이거나 공백 스트링이면 변환 없이 값이 할당됩니다.
- 특정 CCSID 사이에 변환이 정의되지 않으면 값이 할당되지 않고 오류 메시지가 발행됩니다.
- 변환이 정의되고 필요하면 소스 값은 할당이 수행되기 전에 목표의 CCSID로 변환됩니다.

CCSID에 대한 자세한 정보는 Information Center의 Globalization 주제를 참조하십시오.

SQL문에서 호스트 변수의 숫자 지정에 대한 규칙

숫자 지정에 관한 규칙은 다음과 같습니다.

- 숫자의 정수 부분이 부동 소수점으로 변환시 변경될 수도 있습니다. 단정밀도 부동 소수점 필드에는 7개의 소수 자릿수만 포함될 수 있습니다. 8 자릿수 이상이 들어 있는 수의 정수 부분은 반올림 때문에 변경됩니다. 배정밀도 부동 소수점 필드에는 16개의 소수 자릿수만 포함될 수 있습니다. 17 자릿수 이상이 들어 있는 수의 정수 부분은 반올림 때문에 변경됩니다.
- 숫자의 정수 부분은 절대로 절단되지 않습니다. 필요할 경우 수의 소수 부분은 절단됩니다. 변환된 숫자가 목표 호스트 변수 또는 열에 들어 가지 않으면 음의 SQLCODE가 리턴됩니다.
- 소수, 분수 또는 2진 숫자가 소수, 분수 또는 2진 열이나 호스트 변수에 지정될 때마다, 이 숫자는 필요에 따라 목표의 정밀도와 스케일로 변환됩니다. 선행 0으로 필요한 수는 추가되거나 삭제됩니다. 수의 소수 부분에서는 후미 0으로 필요한 수가 추가되거나 후미 숫자의 필요한 수가 제거됩니다.
- 2진 또는 부동 소수점 숫자가 소수 또는 분수 열이나 호스트 변수에 지정되는 경우 이 숫자는 먼저 임시 소수 또는 분수로 변환된 다음 필요할 경우 목표의 정밀도와 스케일로 변환됩니다.
 - 스케일이 0인 2진 반단어 정수(SMALLINT)가 소수 또는 분수로 변환되는 경우 임시 결과는 정밀도 5와 스케일 0을 가집니다.
 - 2진 전단어 정수(INTEGER)가 소수 또는 분수로 변환되는 경우 임시 결과는 정밀도 11과 스케일 0을 가집니다.
 - 2배 전단어 2진 정수(BIGINT)는 소수 또는 분수로 변환되는 경우 임시 결과는 정밀도 19와 스케일 0을 가집니다.
 - 부동 소수점 숫자가 소수 또는 분수로 변환되는 경우 임시 결과는 숫자의 정수 부분이 유효 자리나 정확성을 잃지 않고 절단될 수 있는 스케일과 정밀도 31을 가집니다.

SQL문에서 호스트 변수의 날짜, 시간 및 시간소인 지정에 대한 규칙

날짜가 호스트 변수로 지정되는 경우 이 날짜는 CRTSQLxxx 명령의 DATFMT 및 DATSEP 매개변수에 의해 지정된 스트링 표현으로 변환됩니다. 선행 0은 날짜 표시의 어느 부분에서도 생략되지 않습니다. 호스트 변수는 그 길이가 *USA, *EUR, *JIS 또는 *ISO 날짜 형식에 대해서는 최소한 10바이트이고 *MDY, *DMY 또는 *YMD 날짜 형식에 대해서는 최소한 8바이트이며 *JUL 날짜 형식에 대해서는 최소한 6바이트 인 고정 또는 가변 길이 문자 스트링 변수이어야 합니다. 길이가 10보다 크면 스트링은 오른쪽이 공백으로 채워집니다. ILE RPG 및 ILE COBOL에서 호스트 변수는 날짜 변수가 될 수도 있습니다.

시간이 호스트 변수로 지정되는 경우 이 시간은 CRTSQLxxx 명령의 TIMFMT 및 TIMSEP 매개변수에 의해 지정된 스트링 표현으로 변환됩니다. 선행 0은 생략되지 않습니다. 호스트 변수는 고정 또는 가변 길이 문자 스트링 변수이어야 합니다. 호스트 변수의 길이가 시간의 스트링 표시보다 길 경우 스트링은 오른쪽이 공백으로 채워집니다. ILE RPG 및 ILE COBOL에서 호스트 변수는 시간 변수가 될 수도 있습니다.

- *USA 형식이 사용되면 호스트 변수의 길이가 8보다 작아서는 안됩니다.
- *HMS, *ISO, *EUR 또는 *JIS 형식이 사용되면 호스트 변수의 길이는 추가 포함되는 경우 최소한 8바이트, 시간과 분만 필요한 경우 최소한 5바이트이어야 합니다. 이 경우 SQLWARN0와 SQLWARN1(SQLCA 내의)이 W로 지정되며 인디케이터 변수가 지정된 경우에는 절단된 실제 초 수로 설정됩니다.

시간소인이 호스트 변수로 지정되는 경우 시간소인은 해당 스트링 표시로 변환됩니다. 선행 0은 어느 부분에서도 생략되지 않습니다. 호스트 변수는 길이가 최소한 19바이트 인 고정 또는 가변 길이 문자 스트링 변수이어야 합니다. 길이가 26보다 작으면 호스트 변수에는 모든 마이크로초의 자릿수가 포함되지 않습니다. 길이가 26보다 크면 호스트 변수는 오른쪽이 공백으로 채워집니다. ILE RPG 및 ILE COBOL에서 호스트 변수는 시간소인 변수가 될 수도 있습니다.

SQL을 사용하는 어플리케이션에서 인디케이터 변수

인디케이터 변수는 해당되는 연관된 호스트 변수가 널값으로 지정되었는지 여부를 나타내는데 사용되는 단언어 정수 변수입니다.

- 결과 열의 값이 널(null)이면 SQL이 인디케이터 변수에 -1을 기록합니다.
- 인디케이터 변수를 사용하지 않았으며 결과 열이 널(null)이면 음의 SQLCODE가 리턴됩니다.
- 결과 열의 값이 자료 맵핑 오류를 발생시키면 SQL은 인디케이터 변수를 -2로 설정합니다.

인디케이터 변수를 사용해 검색된 스트링 값이 절단되지 않았는지 확인할 수도 있습니다. 절단이 발생하면 인디케이터 변수에는 스트링의 원래 길이를 지정하는 양의 정수가

포함됩니다. 스트링이 큰 오브젝트(LOB)를 표시하거나 스트링의 원래 길이가 32767보다 클 경우, 반단어 정수에서 변수는 32767보다 큰 수를 저장할 수 없으므로 변수는 인디케이터 변수에서 32767로 저장됩니다.

데이터베이스 관리자가 결과 열에서 값을 리턴시킬 때 인디케이터 변수를 테스트할 수 있습니다. 인디케이터 변수의 값이 음수이면 결과 열 값이 널(null)임을 알 수 있습니다. 데이터베이스 관리자가 널을 리턴시키면 호스트 변수가 열 호스트 변수에 대한 디폴트 값으로 설정됩니다.

호스트 변수 또는 키워드 INDICATOR 바로 다음에 인디케이터 변수(앞에 콜론(:)을 붙임)를 지정합니다. 예를 들면 다음과 같습니다.

```
EXEC SQL
  SELECT COUNT(*), AVG(SALARY)
  INTO :PLICNT, :PLISAL:INDNULL
  FROM CORPDATA.EMPLOYEE
  WHERE EDLEVEL < 18
END-EXEC.
```

그런 다음 INDNULL을 테스트하여 여기에 음의 값이 들어 있는지를 알 수 있습니다. 음의 값이 들어 있는 경우 SQL은 널(null)을 리턴시킵니다.

항상 **IS NULL** 술어를 사용하여 한 열 안에서 널을 테스트하십시오. 예를 들면 다음과 같습니다.

```
WHERE expression IS NULL
```

NULL에 대해 다음 방식으로 테스트하지 마십시오.

```
MOVE -1 TO HUIND.
EXEC SQL...WHERE column-name = :HUI :HUIND
```

널과 비교시 EQUAL 술부는 항상 거짓으로 평가됩니다. 이 예의 결과, 행은 선택되지 않습니다.

호스트 구조에 사용된 인디케이터 변수

또한 인디케이터 구조(반단어 정수 변수의 배열로 정의됨)를 지정하여 호스트 구조를 지원할 수도 있습니다. 호스트 구조에 리턴된 결과 열 값이 널이 될 수 있으면 인디케이터 구조명을 호스트 구조명에 추가할 수 있습니다. 이로 인해 SQL은 호스트 구조 내의 호스트 변수에 리턴된 각 널에 대한 프로그램을 사용자에게 알리게 됩니다.

예를 들어 COBOL에서는 다음과 같습니다.

```
01 SAL-REC.
   10 MIN-SAL          PIC S9(6)V99 USAGE COMP-3.
   10 AVG-SAL          PIC S9(6)V99 USAGE COMP-3.
   10 MAX-SAL          PIC S9(6)V99 USAGE COMP-3.
01 SALTABLE.
02 SALIND              PIC S9999 USAGE COMP-4 OCCURS 3 TIMES.
01 EDUC-LEVEL         PIC S9999 COMP-4.
...
```

```

MOVE 20 TO EDUC-LEVEL.
...
EXEC SQL
SELECT MIN(SALARY), AVG(SALARY), MAX(SALARY)
INTO :SAL-REC:SALIND
FROM CORPDATA.EMPLOYEE
WHERE EDLEVEL>:EDUC-LEVEL
END-EXEC.

```

이 예에서 SALIND는 3개의 값이 들어 있는 배열이며 각각 음의 값에 대해 테스트될 수 있습니다. 예를 들어 SALIND(1)에 음의 값이 들어 있으면 호스트 구조 내의 해당 호스트 변수(즉, MIN-SAL)는 선택된 행에 대해 변경되지 않습니다.

위의 예에서 SQL은 행의 열 값을 호스트 구조로 선택합니다. 따라서 선택된 열 값(있는 경우)이 널인지를 결정하는 인디케이터 변수에 대해 해당 구조를 사용해야 합니다.

널(null) 값 설정에 사용된 인디케이터 변수

인디케이터 변수를 사용하여 열 내의 널(null)을 설정할 수 있습니다. UPDATE 또는 INSERT문 처리시, SQL은 인디케이터 변수(있는 경우)를 테스트합니다. 인디케이터 변수에 음의 값이 들어 있으면 열 값은 널로 설정됩니다. 인디케이터 변수에 -1보다 큰 값이 들어 있으면 연관된 호스트 변수에는 열에 대한 값이 들어 있습니다.

예를 들어 (INSERT 또는 UPDATE문을 사용하여)값이 열에 기록될 것을 지정할 수 있으나 그 값이 입력 자료로 지정되었는지는 확인할 수 없습니다. 열을 널로 설정하는 기능을 제공하기 위해 다음 명령문을 작성할 수 있습니다.

```

EXEC SQL
UPDATE CORPDATA.EMPLOYEE
SET PHONENO = :NEWPHONE:PHONEIND
WHERE EMPNO = :EMPID
END-EXEC.

```

NEWPHONE에 널 이외의 값이 들어 있으면 다음과 같이 명령문 앞에 PHONEIND를 0으로 설정하십시오.

```
MOVE 0 to PHONEIND.
```

그렇지 않은 경우 NEWPHONE에 널이 들어 있음을 SQL에 알려려면 다음과 같이 PHONEIND에 음의 값을 설정하십시오.

```
MOVE -1 TO PHONEIND.
```

SQL 오류 리턴 코드 처리

SQL문이 사용자의 프로그램에서 처리될 때 SQL은 리턴 코드를 SQLCODE 및 SQLSTATE 필드에 위치시킵니다. 리턴 코드는 명령문 수행의 성공이나 실패를 표시합니다. SQL이 명령문 처리 중에 오류를 발견할 경우 SQLCODE는 음수이며 SUBSTR(SQLSTATE,1,2)은 '00', '01' 또는 '02'가 아닙니다. SQL이 명령문의 처리 중에 유효한 조건이 아닌 예외를 발견할 경우 SQLCODE는 양수이며

SUBSTR(SQLSTATE,1,2)은 '01' 또는 '02'입니다. SQL문이 오류나 경고 조건을 발생하지 않고 처리되는 경우 SQLCODE는 0이며 SQLSTATE는 '00000'입니다.

주: 0인 SQLCODE가 사용자 프로그램에 리턴되었으나 그 결과가 만족스럽지 않을 수도 있습니다. 예를 들어 프로그램 수행 결과 값이 절단되었으면 프로그램에 리턴된 SQLCODE는 0입니다. 그러나 SQL 경고 플래그(SQLWARN1) 중 하나가 절단을 표시합니다. 이 경우 SQLSTATE는 '00000'이 아닙니다.

경고: 음의 SQLCODE를 테스트하지 않거나 WHENEVER SQLERROR문을 지정하면 프로그램이 다음 명령문으로 계속됩니다. 오류가 발생된 후에도 계속 수행되는 경우 예측할 수 없는 결과가 발생할 수 있습니다.

SQLSTATE의 주요 목적은 다른 IBM 관계형 데이터베이스 시스템간 공통 리턴 조건에 대해 공통 리턴 코드를 제공하는 것입니다. SQLSTATE는 분산 데이터베이스 조작으로 문제점을 처리할 때 특히 유용합니다. 자세한 내용은 SQL 참조서를 참조하십시오.

SQLCA는 유용한 문제점 진단 툴이기 때문에 SQLCA에 들어 있는 일부 정보를 표시하는 데 필수적인 명령어를 어플리케이션 프로그램에 포함시키는 것이 좋습니다. 다음 SQLCA 필드는 매우 중요합니다.

SQLCODE	리턴 코드.
SQLSTATE	리턴 코드.
SQLERRD(3)	SQL에 의해 갱신, 삽입 또는 삭제된 행 수.
SQLWARN0	W로 설정되면 SQL 경고 플래그(SQLWARN1 - SQLWARNA) 중 적어도 하나가 설정됩니다.

SQLCA에 대한 자세한 내용은 SQL 참조서의 『SQL 통신 영역』의 부록 B를 참조하십시오. 특정 SQLCODE 및 SQLSTATE를 찾으려면, SQL Message finder를 사용하십시오. iSeries용 DB2 UDB SQLCODE 및 SQLSTATE의 리스트는 iSeries Information Center의 SQL 메시지 및 코드를 참조하십시오.

WHENEVER문에 대한 예외 조건 처리

WHENEVER문은 SQL이 SQLSTATE와 SQLCODE를 체크하여 프로그램을 계속해서 처리하도록 하거나 SQL문을 실행한 결과 오류, 예외 및 경고가 존재하면 프로그램의 다른 곳으로 분기가 일어나도록 합니다. 그런 후 예외 조건 처리 서브루틴(프로그램의 일부)은 SQLCODE 필드를 체크하여 오류나 예외 조건에 대해 특정 조치를 취할 수 있습니다.

주: WHENEVER문은 REXX 프로시저어에서는 허용되지 않습니다. REXX에서의 예외 조건 처리에 대해 자세한 사항은 제 7 장 『REXX 어플리케이션에서의 SQL문 코딩』을 참조하십시오.

WHENEVER문을 사용해 일반 조건이 참일 때마다 수행할 작업을 지정할 수 있습니다. 동일 조건에 대해 둘 이상의 WHENEVER문을 지정할 수 있습니다. 이 경우 첫 번째 WHENEVER문은 다른 WHENEVER문이 지정될 때까지 소스 프로그램 내의 모든 후속 SQL문에 적용됩니다.

WHENEVER문은 다음과 같습니다.

```
EXEC SQL
WHENEVER condition action
END-EXEC.
```

다음과 같은 3가지 조건을 지정할 수 있습니다.

SQLWARNING

SQLWARN0 = W이거나 SQLCODE에 100이 아닌 양의 값(SUBSTR(SQLSTATE,1,2)='0')이 들어 있는 경우 수행할 작업을 표시하려면 SQLWARNING을 지정하십시오.

주: 몇 가지 다른 이유에 대해서도 SQLWARN0를 설정할 수 있습니다. 예를 들어 호스트 변수로 이동할 때 열 값이 절단되었으면 프로그램이 이를 오류로 간주합니다.

SQLERROR

SQLERROR을 지정하여 오류 코드가 SQL문 (SQLCODE < 0) (SUBSTR(SQLSTATE,1,2) > '02')의 결과로서 리턴될 때 수행하려는 것을 표시하십시오.

NOT FOUND

다음과 같은 이유로 +100인 SQLCODE와 SQLSTATE '02000'가 리턴된 경우 수행할 작업을 표시하려면 NOT FOUND를 지정하십시오.

- 커서에 대해 단일 행 SELECT가 발행되거나 첫 번째 FETCH가 발행된 후, 프로그램이 지정하는 자료가 존재하지 않습니다.
- 차후의 FETCH 후, 커서 SELECT문을 만족시키는 추가 행을 검색할 수 없습니다.
- UPDATE, DELETE 또는 INSERT 실행 후, 탐색 조건과 일치하는 행이 없습니다.

또한 사용할 조치를 지정할 수 있습니다.

CONTINUE

이는 프로그램이 다음 명령문으로 계속하도록 합니다.

GO TO label

이는 프로그램이 프로그램 내의 영역으로 분기되도록 합니다. 이 영역에 대한 레이블 앞에 콜론(:)을 표시할 수 있습니다. **WHENEVER ... GO TO**문은 다음과 같습니다.

- COBOL에서는 섹션명 또는 규정되지 않은 단락명 이어야 합니다.
- PL/I와 C에서는 레이블입니다.
- RPG에서는 TAG 레이블입니다.

예를 들어 커서를 사용해 행을 검색중이면 **FETCH**문 발행시 **SQL**이 결국 다른 행을 찾을 수 없게 됩니다. 이 상황에 대처하기 위해 **WHENEVER NOT FOUND GO TO...**문을 지정하여 커서를 올바르게 닫기 위해 **CLOSE**문을 발행하는 프로그램 내의 특정 위치로 **SQL**을 분기시키십시오.

주: **WHENEVER**문은 다른 **WHENEVER**가 발견될 때까지 모든 후속 소스 **SQL**문에 영향을 줍니다.

다시 말하면 두 개의 **WHENEVER**문 사이에서 작성된 모든 **SQL**문(또는 만약 하나밖에 없을 경우 첫 번째 다음에)은 프로그램이 지정하는 경로에 관계없이 첫 번째 **WHENEVER**문에 의해 관리됩니다.

이로 인해 **WHENEVER**문은 영향을 받게 되는 첫 번째 **SQL**문 앞에 와야 합니다. **WHENEVER**가 **SQL**문 다음에 오는 경우, 분기는 **SQL**문에 의해 설정된 **SQLCODE**와 **SQLSTATE**의 값에 기초하지 않습니다. 그러나 사용자의 프로그램이 **SQLCODE** 또는 **SQLSTATE**를 직접 체크할 경우 이 체크는 **SQL**문 수행 후에 실행되어야 합니다.

WHENEVER문은 서브루틴 옵션에 **CALL**을 제공하지 않습니다. 이런 이유로 사용자는 각 **SQL**문이 수행된 후에 **SQLCODE** 또는 **SQLSTATE** 값을 검토하고 **WHENEVER**문을 사용하기보다는 서브루틴을 호출하려고 할 수도 있을 것입니다.

제 2 장 C 및 C++ 어플리케이션에서 SQL문 코딩

이 장에서는 C 또는 C++ 프로그램에 SQL문을 삽입하기 위한 고유의 어플리케이션과 코딩 요구사항을 설명합니다. C 프로그램은 iSeries용 ILE C 프로그램을 말합니다. C++ 프로그램은 ILE C++ 프로그램을 말합니다. 또한 이 장에서는 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다. 자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL 통신 영역 정의』
- 15 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL 설명자의 영역 정의』
- 17 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL문 삽입』
- 20 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 변수 사용』
- 32 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 사용』
- 36 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조의 배열 사용』
- 40 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 포인터 자료 유형 사용』
- 41 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 typedef』
- 42 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 ILE C 컴파일러 외부 파일 설명 사용』
- 43 페이지의 『동등한 SQL 및 C 또는 C++ 자료 유형 판별』
- 46 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 인디케이터 변수 사용』

SQL문을 사용하는 방법을 보여주는 샘플 C 프로그램에 대한 자세한 내용은 부록 A 『iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램』을 참조하십시오.

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL 통신 영역 정의

SQL문이 들어 있는 C 또는 C++ 프로그램에는 반드시 다음 중 하나 또는 모두가 포함되어 있어야 합니다.

- long SQLCODE로서 선언된 SQLCODE 변수
- Char SQLSTATE[6]로서 선언된 SQLSTATE 변수

또는

- SQLCA(SQLCODE 및 SQLSTATE 변수가 들어 있는)

SQLCODE 값과 SQLSTATE 값은 각각의 SQL문이 실행된 후에 데이터베이스 관리자에 의해 설정됩니다. 어플리케이션은 SQLCODE 값을 체크하여 최종 SQL문이 제대로 수행되었는지를 판별합니다.

C나 C++ 프로그램에서 SQLCA를 직접 또는 SQL INCLUDE문을 사용해서 코딩할 수 있습니다. SQL INCLUDE문을 사용할 때에는 다음의 표준 선언 부분이 포함되어야 합니다.

```
EXEC SQL INCLUDE SQLCA ;
```

표준 선언에는 구조 정의와 'sqlca'라는 정적 자료 영역이 포함됩니다.

SQLCODE, SQLSTATE 및 SQLCA 변수는 실행 가능한 모든 명령문 앞에 표시되어야 합니다. 선언 부분의 범위에는 프로그램 내의 모든 SQL문의 범위가 포함되어야 합니다.

SQLCA에 대한 C 또는 소스문은 다음과 같습니다.

```
#ifndef SQLCODE
struct sqlca {
    unsigned char sqlcaid[8];
    long          sqlcabc;
    long          sqlcode;
    short         sqlerrml;
    unsigned char sqlerrmc[70];
    unsigned char sqlerrp[8];
    long          sqlerrd[6];
    unsigned char sqlwarn[11];
    unsigned char sqlstate[5];
};
#define SQLCODE sqlca.sqlcode
#define SQLWARN0 sqlca.sqlwarn[0]
#define SQLWARN1 sqlca.sqlwarn[1]
#define SQLWARN2 sqlca.sqlwarn[2]
#define SQLWARN3 sqlca.sqlwarn[3]
#define SQLWARN4 sqlca.sqlwarn[4]
#define SQLWARN5 sqlca.sqlwarn[5]
#define SQLWARN6 sqlca.sqlwarn[6]
#define SQLWARN7 sqlca.sqlwarn[7]
#define SQLWARN8 sqlca.sqlwarn[8]
#define SQLWARN9 sqlca.sqlwarn[9]
#define SQLWARNA sqlca.sqlwarn[10]
#define SQLSTATE sqlca.sqlstate
#endif
struct sqlca sqlca;
```

프로그램에 SQLCODE에 대한 선언이 있고 사전컴파일러에 의해 SQLCA가 제공될 때 SQLCODE는 SQLCADE로 대체됩니다. 프로그램에 SQLSTATE에 대한 선언이 있고 사전컴파일러에 의해 SQLCA가 제공될 때 SQLSTATE는 SQLSTOTE로 대체됩니다.

주: 많은 SQL 오류 메시지는 가변 길이로 이루어진 메시지 자료가 있습니다. 이와 같은 자료 필드의 길이는 SQLCA sqlerrmc 필드 값에 삽입됩니다. 이러한 길이로 인해 C나 C++ 프로그램으로부터 sqlerrmc 값을 인쇄하는 경우 예상 밖의 결과가 나타나기도 합니다.

SQLCA에 대한 자세한 정보는 SQL 참조서의 부록 B, SQL 통신 영역을 참조하십시오.

SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL 설명자의 영역 정의

다음 명령문은 SQLDA를 필요로 합니다.

EXECUTE...USING DESCRIPTOR *descriptor-name*

FETCH...USING DESCRIPTOR *descriptor-name*

OPEN...USING DESCRIPTOR *descriptor-name*

DESCRIBE *statement-name* INTO *descriptor-name*

DESCRIBE TABLE *host-variable* INTO *descriptor-name*

PREPARE *statement-name* INTO *descriptor-name*

CALL...USING DESCRIPTOR *descriptor-name*

SQLCA와는 달리 두 개 이상의 SQLDA가 프로그램에 있을 수 있으며 SQLDA는 유효한 모든 이름을 가질 수 있습니다. SQLDA는 직접 또는 SQL INCLUDE문을 사용해 C나 C++ 프로그램에서 코드화될 수 있습니다. SQL INCLUDE문 사용시 표준 SQLDA 선언 부분을 포함시켜야 합니다.

```
EXEC SQL INCLUDE SQLDA;
```

표준 선언 부분에는 이름이 'sqlda'인 구조 정의만 포함됩니다.

SQLDA에 대해 포함된 C 및 C++ 선언 부분은 다음과 같습니다.

```
#ifndef SQLDASIZE
struct sqlda {
    unsigned char sqldaaid[8];
    long sqldabc;
    short sqln;
    short sqld;
    struct sqlvar {
        short sqltype;
        short sqlen;
        unsigned char *sqldata;
        short *sqlind;
        struct sqlname {
            short length;
            unsigned char data[30];
        } sqlname;
    }
};
#endif
```

```

        } sqlvar[1];
    };
#define SQLDASIZE(n) (sizeof(struct sqlda) + (n-1)* sizeof(struct sqlvar))
#endif

```

INCLUDE SQLDA SQL문을 사용하는 한 가지 장점은 다음의 매크로 정의도 얻을 수 있다는 점입니다.

```
#define SQLDASIZE(n) (sizeof(struct sqlda) + (n-1)* sizeof(struct sqlvar))
```

이 매크로를 사용하면 지정된 수의 SQLVAR 요소를 가진 SQLDA에 대해 기억영역을 쉽게 할당할 수 있습니다. 다음 예에서는 SQLDASIZE 매크로를 사용하여 SQLDA의 기억영역에 20 SQLVAR 요소를 할당합니다.

```

#include <stdlib.h>
EXEC SQL INCLUDE SQLDA;

struct sqlda *mydaptr;
short numvars = 20;
.
.
mydaptr = (struct sqlda *) malloc(SQLDASIZE(numvars));
mydaptr->sqln = 20;

```

다음과 같이 INCLUDE SQLDA문에 포함된 다른 매크로 정의가 있습니다.

GETSQLDOUBLED(daptr) daptr이 지정한 SQLDA가 2배가 되면 1을 리턴하고 2배가 되지 않으면 0을 리턴합니다. SQLDA는 SQLDAID 필드에 7번째 바이트가 '2'로 설정되는 경우 2배가 됩니다.

SETSQLDOUBLED(daptr, newvalue) SQLDAID의 7번째 바이트를 새로운 값으로 설정합니다.

GETSQLDALONGLEN(daptr,n) daptr이 지정한 SQLDA의 n번째 항목의 길이 속성을 리턴합니다. SQLDA가 2배가 되었고 n번째 SQLVAR 항목이 LOB 자료 유형을 갖는 경우에만 이것을 사용하십시오.

SETSQLDALONGLEN(daptr,n,len) daptr이 n번째 항목에 대한 길이에 지정된 SQLDA의 SQLLONGLEN 필드를 설정합니다. SQLDA가 2배가 되었고 n번째 SQLVAR 항목이 LOB 자료 유형을 갖는 경우에만 이것을 사용하십시오.

GETSQLDALENPTR(daptr,n) daptr이 지정한 SQLDA의 n번째 항목에 대한 자료의 실제 길이로 포인터를 리턴합니다. SQLDATALEN 포

인터 필드는 포인터를 (4바이트) 정수 길이로 리턴합니다. SQLDATALEN 포인터가 0이면 NULL 포인터가 리턴됩니다. SQLDA가 2배가 된 경우에만 이것을 사용하십시오.

SETSQLDALENPTR(daptr,n,ptr)

daptr이 지정한 SQLDA의 n번째 항목에 대한 자료의 실제 길이로 포인터를 설정합니다. SQLDA가 2배가 된 경우에만 이것을 사용하십시오.

SQLDA를 포인터로 선언했을 때 포인터로 선언된 호스트 변수에서와 마찬가지로 그것은 SQL문에서 사용될 때 정확히 선언되어야 합니다. 컴파일러 오류를 피하려면 SQLDA의 sqldata 필드에 할당된 값의 유형은 부호 없는 문자의 포인터이어야 합니다. 이는 컴파일러 오류를 방지해줍니다. 유형 캐스팅은 어플리케이션 프로그램이 프로그램상의 호스트 변수 주소를 전달해 주는 EXECUTE, OPEN, CALL 및 FETCH문에 대해서만 필요합니다. 예를 들어 madaptr라는 SQLDA에 대한 포인터를 선언할 때 다음과 같이 PREPARE문에서 사용할 수 있습니다.

```
EXEC SQL PREPARE myname INTO :*mydaptr FROM :mysqlstring;
```

SQLDA 선언 부분은 구조 정의가 허용되는 위치마다 표시될 수 있습니다. 정상 C 범위 규칙을 적용합니다.

동적 SQL은 iSeries용 DB2® UDB 프로그램 개념 정보의 동적 SQL 어플리케이션에 설명되어 있는 첨단 프로그래밍 기술입니다. 동적 SQL을 사용하면 프로그램 수행 중 사용자 프로그램이 SQL문을 개발 및 수행할 수 있습니다. 동적으로 수행하는 가변 SELECT 리스트(조회)의 한 부분으로 리턴될 자료의 리스트가 있는 SELECT문은 SQL 설명자 영역(SQLDA)을 필요로 합니다. 왜냐하면 사용자가 SELECT의 결과를 받아들이기 위해 지정된 변수의 수와 유형을 미리 알 수 없기 때문입니다.

SQLDA에 대한 자세한 정보는 SQL 참조서의 "SQL 설명자 영역"을 참조하십시오.

SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL문 삽입

SQL문은 C나 C++ 명령문이 실행될 수 있는 모든 곳에 위치할 수 있습니다.

각 SQL문은 EXEC SQL로 시작하고 세미콜론(;)으로 끝나야 합니다. EXEC SQL 키워드는 한 행 안에 있어야 합니다. SQL문의 나머지 부분은 두 개 이상의 행에 있을 수 있습니다.

예: C나 C++ 프로그램에서 코딩된 UPDATE문은 다음과 같을 수 있습니다.

```
EXEC SQL
    UPDATE DEPARTMENT
SET MGRNO = :MGR_NUM
WHERE DEPTNO = :INT_DEPT ;
```

자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 C 및 C++ 어플리케이션의 주석』
- 『SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL문의 계속』
- 19 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 포함 코드』
- 19 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 여백』
- 19 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 이름』
- 19 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 NULL 및 NULS』
- 19 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 명령문 레이블』
- 20 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에 대한 사전처리기 순서』
- 20 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 삼중 문자』
- 20 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션의 WHENEVER문』

SQL을 사용하는 C 및 C++ 어플리케이션의 주석

SQL 주석(--) 외에도 공백이 허용될 때는 언제든지 키워드 EXEC와 SQL을 제외하고 삽입된 SQL문 내에 C 주석(* ...*)이 포함될 수 있습니다. 주석은 여러 행에 걸쳐 있을 수 있습니다. 주석들이 중첩될 수는 없습니다. C++에서는 단일행 주석(//로 시작하는 주석)을 사용할 수 있지만 C에서는 사용할 수 없습니다.

SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL문의 계속

SQL문은 하나 이상의 행에 포함될 수 있습니다. 공백이 있는 곳에서 SQL문을 분할할 수 있습니다. 역슬래시(\)는 스트링 상수나 분리 ID를 연속시키는데 사용될 수 있습니다. 구분되지 않는 식별자는 계속될 수 없습니다.

DBCS 자료가 들어 있는 상수는 다음 두 방법으로 여러 행에 걸쳐 계속될 수 있습니다.

- 계속되는 행의 오른쪽 여백에 있는 문자가 SI 문자이고 계속되는 행의 왼쪽 여백에 있는 문자가 SO 문자이면 좌우 여백에 위치한 시프트 문자들은 삭제됩니다.

이 예에서 SQL문에 G'<AABBCCDDEEFFGGHHIIJJKK>'의 유효한 그래픽 상수가 있습니다. 여백의 중복된 시프트는 삭제됩니다.

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....*....8
EXEC SQL SELECT * FROM GRAPHTAB          WHERE GRAPHCOL = G'<AABBCCDDEEFFGGHH
<IIJJKK>';
```

- 여백 외부에 시프트 문자를 놓을 수 있습니다. 이 예의 경우 여백을 5-75로 가정하십시오. 이 예에서 SQL문에 G'<AABBCCDDEEFFGGHHIIJJKK>'의 유효한 그래픽 상수가 있습니다.

```
*...(...1....+....2....+....3....+....4....+....5....+....6....+....7....)....8
EXEC SQL SELECT * FROM GRAPHTAB          WHERE GRAPHCOL = G'<AABBCCDD
<EEFFGGHHIIJJKK>';
```


SQL을 사용하는 C 및 C++ 어플리케이션의 포함 코드

SQL문, C문 또는 C++문은 소스 코드에 다음 SQL문을 삽입시키면 포함될 수 있습니다.

```
EXEC SQL INCLUDE member-name ;
```

참조된 SQL문 또는 C 및 C++ 호스트 변수의 선언을 SQL문에 포함시키기 위하여 C #include문을 사용할 수는 없습니다.

SQL을 사용하는 C 및 C++ 어플리케이션의 여백

CRTSQCLI, CRTSQLCPPI 또는 CRTSQLCPPI 명령의 MARGINS 매개변수에 의해 지정된 여백 내에 SQL문을 작성하십시오. MARGINS 매개변수가 *SRCFILE로 지정되면 소스 파일의 레코드 길이가 사용됩니다. 값이 오른쪽 여백에 지정되고 해당 값이 소스 레코드 길이보다 길면, 전체 레코드가 읽혀집니다. 값이 포함된 멤버에도 적용됩니다. 예를 들어, 오른쪽 여백 200이 지정되고 소스 파일의 레코드 길이가 80이면 자료의 80열만이 소스 파일에서 읽혀집니다. 동일한 사전컴파일에 포함된 소스 멤버의 레코드 길이가 200이면 포함에서 전체 200이 읽혀집니다.

EXEC SQL이 지정된 여백에서 시작하지 않으면 SQL 사전컴파일러는 SQL문을 인식하지 못합니다. CRTSQCLI 또는 CRTSQLCPPI에 대한 자세한 정보는 부록 B 『호스트 언어 사전컴파일러에 대한 iSeries용 DB2 UDB CL 명령 설명』을 참조하십시오.

SQL을 사용하는 C 및 C++ 어플리케이션의 이름

호스트 변수에 대해 유효한 C 또는 C++ 변수명을 사용할 수 있습니다. 제한 사항은 다음과 같습니다.

대문자 또는 소문자 조합으로 'SQL', 'RDI' 또는 'DSN'으로 시작하는 호스트 변수명이나 외부 항목명을 사용하지 마십시오. 이러한 이름들은 데이터베이스 관리자 예약어입니다. 호스트 변수명의 길이는 128자로 제한됩니다.

SQL을 사용하는 C 및 C++ 어플리케이션의 NULL 및 NULS

C, C++ 및 SQL 모두는 단어 널(null)을 사용하지만 다른 의미를 갖습니다. C와 C++ 언어에는 널 문자, 널 포인터 및 널 명령문(세미콜론)이 있습니다. C NUL은 0과 동일하게 비교되는 단일 문자입니다. C NULL은 유효한 자료 오브젝트를 지시하지 않는 특수 예약 포인터 값입니다. SQL 널값은 널이 아닌 모든 것과 구별되는 특수값으로서 값(널이 아님)이 없음을 나타냅니다.

SQL을 사용하는 C 및 C++ 어플리케이션의 명령문 레이블

실행 가능 SQL문은 레이블 앞에 놓일 수 있습니다.

SQL을 사용하는 C 및 C++ 어플리케이션에 대한 사전처리기 순서

C나 C++을 사용하기 전에 SQL 사전처리기를 실행하십시오. SQL문 내에서 C나 C++ 사전처리기 지시문은 사용할 수 없습니다.

SQL을 사용하는 C 및 C++ 어플리케이션의 삼중 문자

C와 C++ 문자 세트 중 일부 문자가 모든 키보드에 사용될 수는 없습니다. 이러한 문자들은 삼중 문자라는 세 문자 순서를 사용하여 C나 C++ 소스 프로그램에 입력될 수 있습니다. 다음 삼중 문자 순서는 호스트 변수 선언 부분 내에서 지원됩니다.

- ??(왼쪽 괄호
- ??) 오른쪽 괄호
- ??< 왼쪽 꺾쇠
- ??> 오른쪽 꺾쇠
- ??= 파운드
- ??/ 역슬래시

SQL을 사용하는 C 및 C++ 어플리케이션의 WHENEVER문

SQL WHENEVER문의 GOTO절에 대한 목표는 WHENEVER문에 의해 영향을 받는 모든 SQL문의 범위 내에 있어야 합니다.

SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 변수 사용

SQL문에서 사용된 모든 호스트 변수는 명시적으로 선언되어야 합니다. SQL문에서 사용된 호스트 변수는 SQL문에서 호스트 변수를 처음 사용하기 전에 선언되어야 합니다.

호스트 변수 정의에 사용된 C문 앞에는 BEGIN DECLARE SECTION문이 오고 뒤에는 END DECLARE SECTION문이 와야 합니다. BEGIN DECLARE SECTION 및 END DECLARE SECTION이 지정되면 SQL문에서 사용된 모든 호스트 변수 선언은 BEGIN DECLARE SECTION과 END DECLARE SECTION문 사이에 있어야 합니다. typedef 식별자를 사용하여 선언된 호스트 변수에는 BEGIN DECLARE SECTION 및 END DECLARE SECTION도 필요합니다. typedef 선언은 이러한 두 섹션 사이에 필요 없습니다.

호스트 변수 정의에 사용된 C++문 앞에는 BEGIN DECLARE SECTION문이 오고 뒤에는 END DECLARE SECTION문이 와야 합니다. BEGIN DECLARE SECTION문과 END DECLARE SECTION문 사이에 있지 않은 변수는 모두 호스트 변수로 사용될 수 없습니다.

SQL문 내의 모든 호스트 변수 앞에는 콜론(:)이 와야 합니다.

호스트 변수명은 호스트 변수가 다른 블록이나 프로시저에 존재 하더라도 프로그램 내에서는 고유해야 합니다.

호스트 변수를 사용하는 SQL문은 변수가 선언된 명령문의 범위 내에 있어야 합니다.

호스트 변수는 통합 요소가 될 수 없습니다.

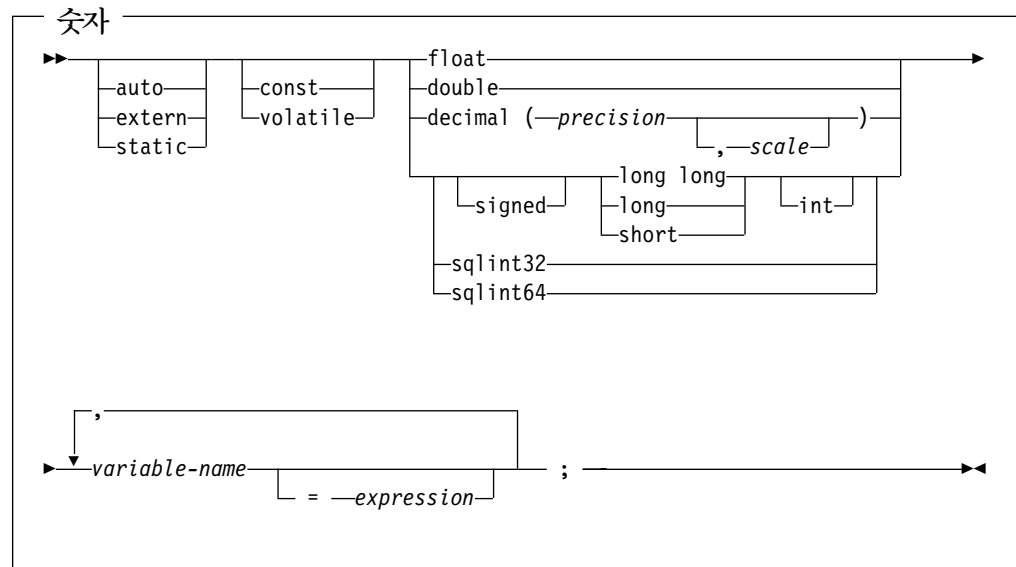
자세한 정보는 『SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 변수 선언』을 참조하십시오.

SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 변수 선언

C 및 C++ 사전컴파일러는 유효한 호스트 변수 선언으로서 유효한 C와 C++ 선언의 서브세트만을 인식합니다.

SQL을 사용하는 C 및 C++ 어플리케이션의 숫자 호스트 변수

다음 그림은 유효한 숫자 호스트 변수 선언 부분에 대한 구문을 표시합니다.



주:

1. precision 및 scale은 정수 상수이어야 합니다. precision의 범위는 1-31 사이입니다. scale은 0에서 정밀도 사이일 수 있습니다.
2. 10진 자료 유형을 사용할 경우 헤더 파일 decimal.h가 포함되어야 합니다.
3. sqlint32 또는 sqlint64를 사용 중이면 sqlsystem.h 헤더 파일이 포함되어야 합니다.

SQL을 사용하는 C 및 C++ 어플리케이션의 문자 호스트 변수

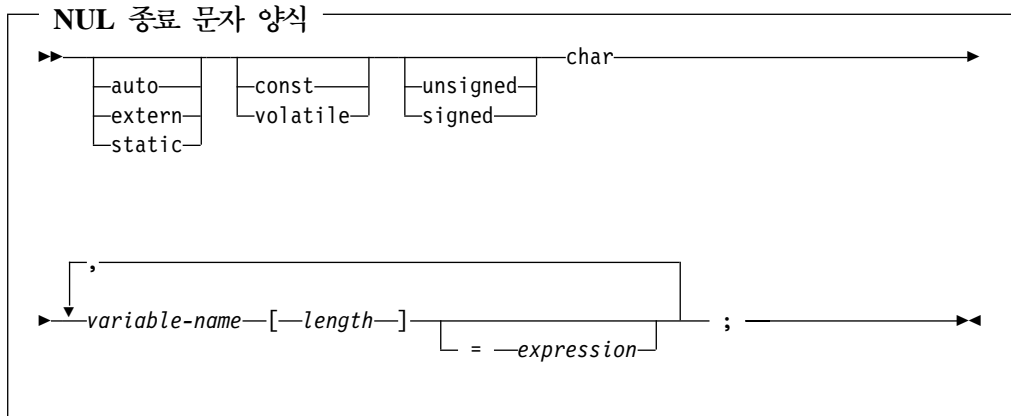
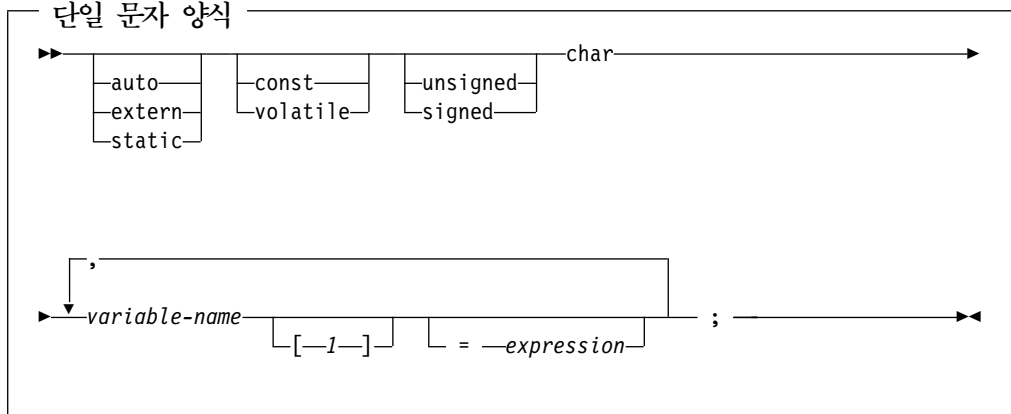
문자 호스트 변수에는 다음 세 가지의 유효 양식이 있습니다.

- 단일 문자 양식

- NUL 종료 문자 양식
- VARCHAR 구조화 양식

또한, SQL VARCHAR 선언은 varchar 호스트 변수를 정의하는데 사용될 수 있습니다.

모든 문자 유형은 부호 없이 처리됩니다.



주:

1. length는 1보다 크고 32741이하의 정수 상수이어야 합니다.
2. *CNULRQD 옵션이 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정될 경우 입력 호스트 변수에는 NUL 종료자가 포함되어야 합니다. 출력 호스트 변수는 공백 사이에 있으며 마지막 문자는 NUL 종료자입니다. 출력 호스트 변수가 너무 작아서 자료 및 NUL 종료자를 포함할 수 없다면 다음과 같은 경우가 발생합니다.
 - 자료가 절단됩니다.
 - 마지막 문자가 NUL 종료자입니다.

2. *var-1*과 *var-2*는 단순 변수 참조여야 하며 정수와 문자 호스트 변수로 개별적으로 사용될 수 없습니다.
3. struct tag는 다른 자료 영역을 정의하는 데 사용될 수 있으나 호스트 변수로는 사용될 수 없습니다.
4. VARCHAR 구조화 양식은 널(null) 문자를 포함할 수 있는 비트 자료에 사용되어야 합니다. VARCHAR 구조 양식은 끝에 널 종료자를 사용하지 않습니다.
5. `_Packed`는 C++에서 사용될 수 없습니다. 대신, 선언 이전에 `#pragma pack(1)`를 지정하고 선언 후 `#pragma pack()`를 지정하십시오.

주: `#pragma pack()` 대신 `#pragma pack(재설정)`을 사용할 수 있습니다.

```
#pragma pack(1)
struct VARCHAR {
    short len;
    char s[10];
} vstring;
#pragma pack()
```

예:

```
EXEC SQL BEGIN DECLARE SECTION;

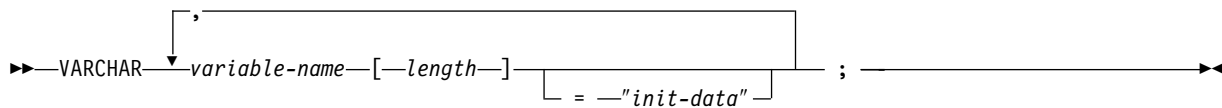
/* valid declaration of host variable vstring */

struct VARCHAR {
    short len;
    char s[10];
} vstring;

/* invalid declaration of host variable wstring */

struct VARCHAR wstring;
```

SQL VARCHAR 양식



주:

1. VARCHAR은 대소문자를 혼합하여 사용할 수 있습니다.
2. length는 0보다 크고 32740 이하의 정수 상수여야 합니다.
3. SQL VARCHAR 양식은 널(null) 문자를 포함할 수 있는 비트 자료에 사용되어야 합니다. SQL VARCHAR 양식은 끝에 널 종료자를 사용하지 않습니다.

예:

다음 선언을 사용할 경우:

```
VARCHAR vstring[528]="mydata";
```

다음과 같은 구조가 생성됩니다.

```
_Packed struct { short len;  
                  char data[528];}  
vstring={6, "mydata"};
```

다음 선언을 사용할 경우:

```
VARCHAR vstring1[111],  
        vstring2[222]="mydata",  
        vstring3[333]="more data";
```

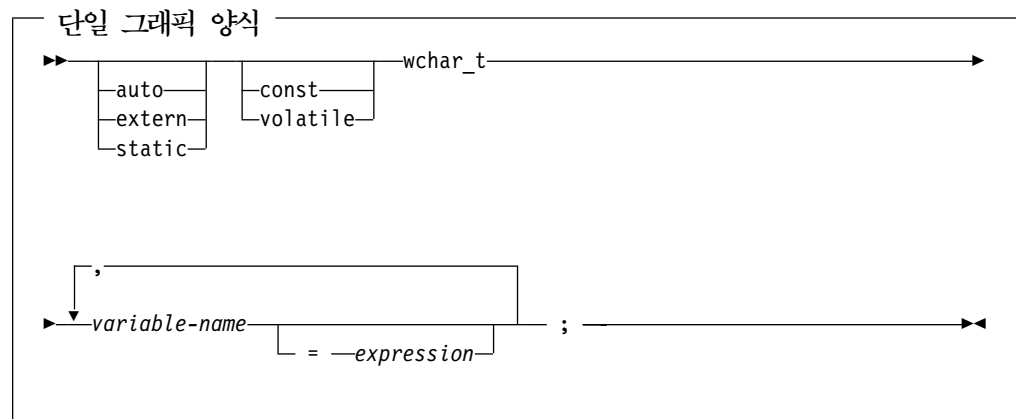
다음과 같은 구조가 생성됩니다.

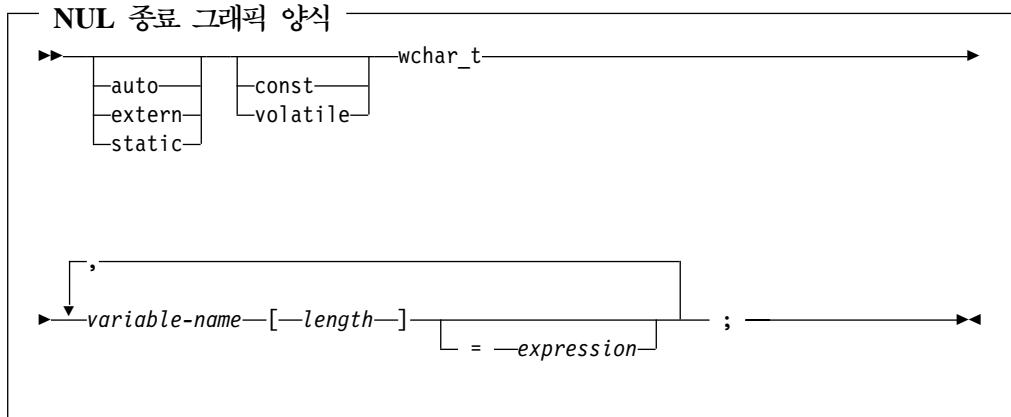
```
_Packed struct { short len;  
                  char data[111];}  
vstring1;  
  
_Packed struct { short len;  
                  char data[222];}  
vstring2={6,"mydata"};  
  
_Packed struct { short len;  
                  char data[333];}  
vstring3={9,"more data"};
```

SQL을 사용하는 C 및 C++ 어플리케이션의 그래픽 호스트 변수

그래픽 호스트 변수에는 다음 세 가지의 유효 양식이 있습니다.

- 단일 그래픽 양식
- NUL 종료 그래픽 양식
- VARGRAPHIC 구조화 양식





주:

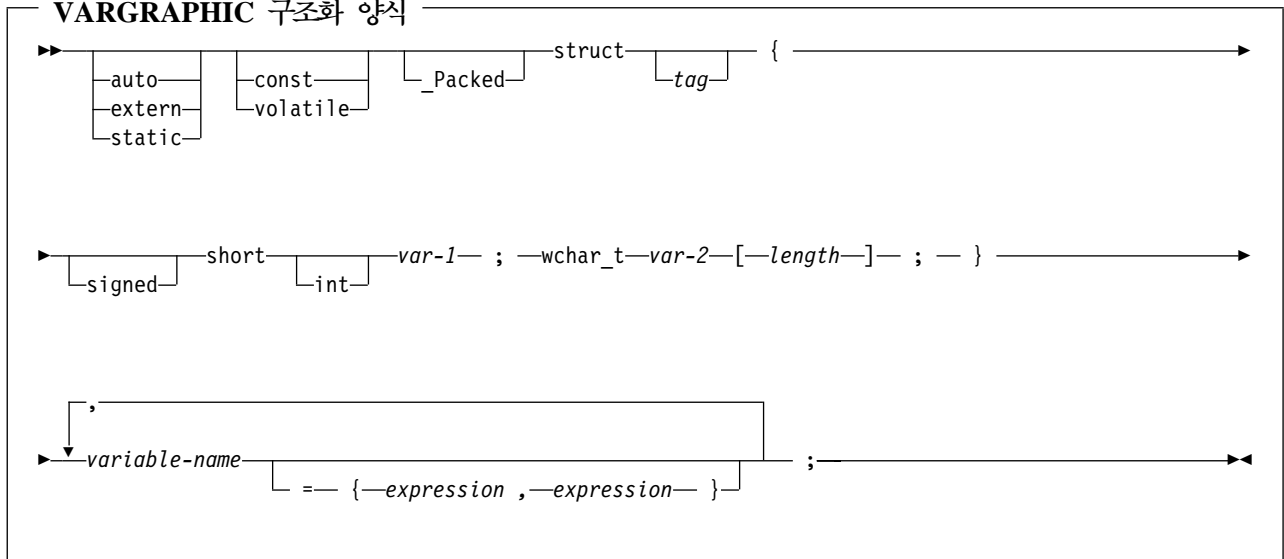
1. *length*는 1보다 크고 16371 이하의 정수 상수이어야 합니다.
2. *CNULRQD 옵션이 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정될 경우 입력 호스트 변수에는 그래픽 NUL 종료자(/0/0)가 포함되어야 합니다. 출력 호스트 변수는 DBCS 공백 사이에 놓이며 최종 문자는 그래픽 NUL 종료자입니다. 출력 호스트 변수가 너무 작아서 자료 및 NUL 종료자를 포함할 수 없다면 다음과 같은 경우가 발생합니다.

- 자료가 절단됩니다.
- 최종 문자로 그래픽 NUL 종료자가 사용됩니다.
- SQLWARN1이 'W'에 설정됩니다.

*CNULRQD 옵션이 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정될 경우 입력 호스트 변수에 그래픽 NUL 종료자를 포함시킬 필요가 없습니다. 다음은 출력 호스트 변수에 대한 것입니다.

- 만일 호스트 변수가 자료 및 그래픽 NUL 종료자를 포함하기에 충분히 크다면 다음과 같은 경우가 발생합니다.
 - 자료가 복귀되지만 DBCS 공백 사이에 들어가지는 않습니다.
 - 자료 뒤에 그래픽 NUL 종료자가 바로 따라 나옵니다.
- 만일 호스트 변수가 자료를 포함할 만큼 충분히 크지만 그래픽 NUL 종료자는 포함할 수 없다면 다음과 같은 경우가 발생합니다.
 - 자료가 리턴됩니다.
 - 그래픽 NUL 종료자가 리턴되지 않습니다.
 - SQLWARN1이 'N'에 설정됩니다.
- 만일 호스트 변수가 자료를 포함할 만큼 충분히 크지 않다면 다음과 같은 경우가 발생합니다.
 - 자료가 절단됩니다.

- 그래픽 NUL 종료자가 리턴되지 않습니다.
- SQLWARN1이 'W'에 설정됩니다.



주:

1. *length*는 0보다 크고 16370 이하의 정수 상수이어야 합니다.
2. *var-1*과 *var-2*는 단순 변수 참조이어야 하며 호스트 변수로 사용될 수 없습니다.
3. *struct tag*는 다른 자료 영역을 정의하는 데 사용될 수 있으나 호스트 변수로는 사용될 수 없습니다.
4. *_Packed*는 C++에서 사용될 수 없습니다. 대신, 선언 이전에 `#pragma pack(1)`를 지정하고 선언 후 `#pragma pack()`를 지정하십시오.

```

#pragma pack(1)
struct VARGRAPH {
    short len;
    wchar_t s[10];
} vstring;
#pragma pack()

```

예:

EXEC SQL **BEGIN DECLARE SECTION;**

```

/* valid declaration of host variable graphic string */

struct VARGRAPH {
    short len;
    wchar_t s[10];
} vstring;

```

```

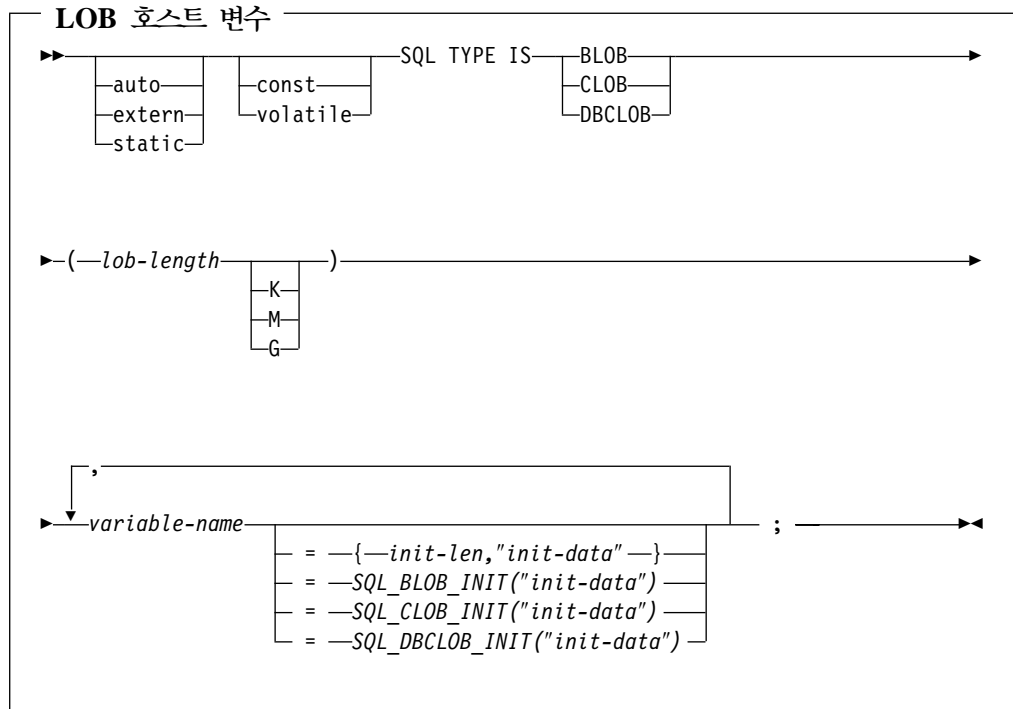
/* invalid declaration of host variable wstring */

struct VARGRAPH wstring;

```

SQL을 사용하는 C 및 C++ 어플리케이션의 LOB 호스트 변수

C 및 C++에는 LOB(대형 오브젝트)에 대한 SQL 자료 유형에 대응하는 변수가 없습니다. 이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQL TYPE IS 절을 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버에서 이 선언을 C 언어 구조로 대체합니다.



주:

1. K는 *lob-length*에 1024를 곱합니다. M은 *lob-length*에 1,048,576을 곱합니다. G는 *lob-length*에 1,073,741,824를 곱합니다.
2. BLOB 및 CLOB의 경우 $1 \leq lob-length \leq 2,147,483,647$
3. DBCLOB의 경우 $1 \leq lob-length \leq 1,073,741,823$
4. SQL TYPE IS, BLOB, CLOB, DBCLOB, K, M, G는 대소문자를 혼합하여 사용할 수 있습니다.
5. 초기화 스트링에 대해 허용되는 최대 길이는 32,766바이트입니다.
6. 초기화 길이인 *init-len*은 숫자 상수(즉, K, M 또는 G를 포함할 수 없음)여야 합니다.
7. LOB의 길이를 지정해야 합니다. 즉, 다음 선언은 허용되지 않습니다.

```
SQL TYPE IS BLOB my_blob;
```

8. 선언 내에서 LOB가 초기화되지 않은 경우 사전컴파일러 생성 코드 내에서 어떤 초기화도 수행되지 않습니다.
9. 사전컴파일러는 호스트 변수 유형에 대해 캐스트하는 데 사용될 수 있는 구조 태 그를 생성합니다.
10. LOB 호스트 변수에 대한 포인터는 다른 호스트 변수 유형에 대한 포인터와 같은 규칙 및 제한사항을 적용하여 선언할 수 있습니다.
11. LOB 호스트 변수에 대한 CCSID 처리는 다른 문자 및 그래픽 호스트 변수 유형에 대한 처리와 같습니다.
12. DBCLOB가 초기화되면 사용자가 스트링 앞에 접두부로 'L'을 붙여야 합니다(와이드 문자 스트링을 나타냄).

BLOB 예

다음 선언을 사용할 경우:

```
static SQL TYPE IS BLOB(128K)  
my_blob=SQL_BLOB_INIT("mydata");
```

다음과 같은 구조가 생성됩니다.

```
static struct my_blob_t {  
    unsigned long length;  
    char data[131072];  
} my_blob=SQL_BLOB_INIT("my_data");
```

CLOB 예

다음 선언을 사용할 경우:

```
SQL TYPE IS CLOB(128K) var1, var2 = {10, "data2data2"};
```

사전컴파일러는 C에 대해 다음을 생성합니다.

```
_Packed struct var1_t {  
    unsigned long length;  
    char data[131072];  
} var1,var2={10,"data2data2"};
```

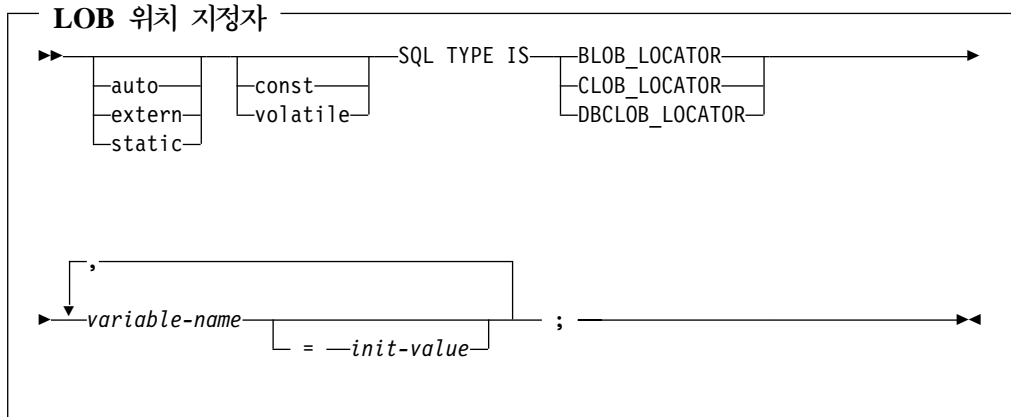
DBCLOB 예

다음 선언을 사용할 경우:

```
SQL TYPE IS DBCLOB(128K) my_dbclob;
```

사전컴파일러는 다음을 생성합니다.

```
_Packed struct my_dbclob_t {  
    unsigned long length;  
    wchar_t data[131072]; } my_dbclob;
```



주:

1. SQL TYPE IS, BLOB_LOCATOR, CLOB_LOCATOR, DBCLOB_LOCATOR 는 대소문자를 혼합하여 사용할 수 있습니다.
2. *init-value*는 포인터 위치 지정자 변수의 초기화를 허용합니다. 다른 초기화 유형들은 의미가 없습니다.
3. LOB 위치 지정자에 대한 포인터를 선언할 수 있습니다. 이 때 다른 호스트 변수 유형에 대한 포인터의 경우와 같은 규칙 및 제한사항이 적용됩니다.

CLOB 위치 지정자 예

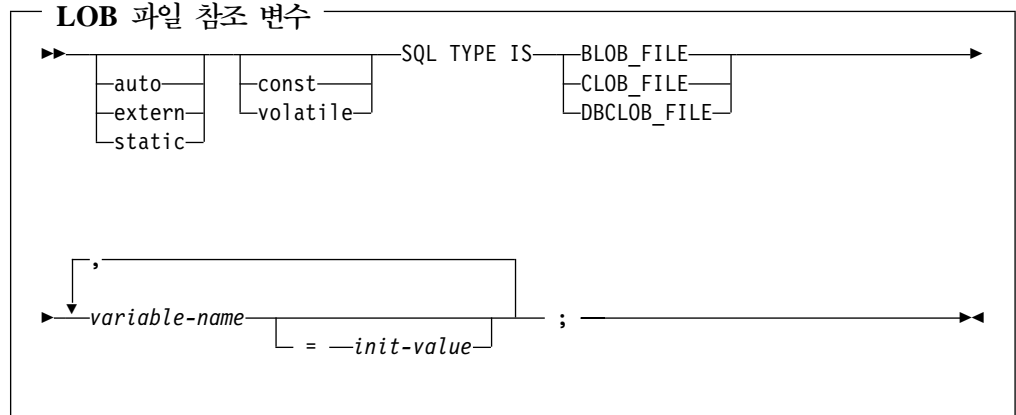
다음 선언을 사용할 경우:

```
static SQL TYPE IS CLOB_LOCATOR my_locator;
```

다음과 같이 생성됩니다.

```
static long int unsigned my_locator;
```

BLOB 및 DBCLOB 위치 지정자의 구문은 비슷합니다.



주:

1. SQL TYPE IS, BLOB_FILE, CLOB_FILE, DBCLOB_FILE은 대소문자를 혼합하여 사용할 수 있습니다.
2. LOB 파일 참조 변수에 대한 포인터를 선언할 수 있습니다. 이 때 다른 호스트 변수 유형에 대한 포인터의 경우와 같은 규칙 및 제한사항이 적용됩니다.

CLOB 파일 참조 예

다음 선언을 사용할 경우:

```
static SQL TYPE IS CLOB_FILE my_file;
```

다음과 같은 구조가 생성됩니다.

```
static _Packed struct {
    unsigned long    name_length;
    unsigned long    data_length;
    unsigned long    file_options;
    char             name[255];
} my_file;
```

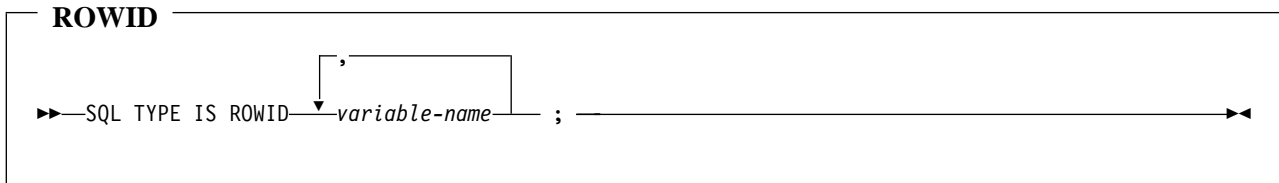
BLOB 및 DBCLOB 파일 참조 변수의 구문은 비슷합니다.

사전컴파일러는 다음의 파일 옵션 상수에 대한 선언을 생성합니다. 파일 참조 호스트 변수를 사용할 때 이러한 상수를 사용하여 file_options 변수를 설정할 수 있습니다. 이러한 값에 대한 자세한 정보는 SQL 프로그래밍 개념의 LOB 파일 참조 변수를 참조하십시오.

- SQL_FILE_READ (2)
- SQL_FILE_CREATE (8)
- SQL_FILE_OVERWRITE (16)
- SQL_FILE_APPEND (32)

SQL을 사용하는 C 및 C++ 어플리케이션의 ROWID 호스트 변수

C 및 C++에는 ROWID의 SQL 자료 유형에 대응하는 변수가 없습니다. 이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQL TYPE IS 절을 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버에서 이 선언을 C 언어 구조로 대체합니다.



주:

1. SQL TYPE IS ROWID는 대소문자를 혼합하여 사용할 수 있습니다.

ROWID 예

다음 선언을 사용할 경우:

```
SQL TYPE IS ROWID myrowid, myrowid2;
```

다음과 같은 구조가 생성됩니다.

C에는 다음이 생성됩니다.

```
_Packed struct { short len;  
                  char data[40];}  
myrowid1, myrowid2;
```

SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 사용

C와 C++ 프로그램에서는 명명된 기본 C 또는 C++ 변수 세트인 호스트 구조를 정의할 수 있습니다. 호스트 구조 자체가 복수 레벨 구조 내에서 발생하는 경우에도 최대 두 개의 레벨을 갖습니다. 다른 구조를 필요로 하는 가변 길이 스트링의 선언 부분은 예외입니다.

호스트 구조명은 종속 레벨이 기본 C 또는 C++ 변수를 명명하는 그룹명이 될 수 있습니다. 예를 들면 다음과 같습니다.

```
struct {  
  struct {  
    char c1;  
    char c2;  
  } b_st;  
} a_st;
```

이 예에서 b_st는 기본 항목 c1 및 c2로 구성되는 호스트 구조의 이름입니다.

구조명을 스칼라 리스트에 대한 단축 표기법으로 사용할 수 있으나 두 레벨 구조에만 사용할 수 있습니다. 호스트 변수를 구조명으로 규정할 수 있습니다(예를 들어 구조 필드). 호스트 구조는 두 레벨로 제한됩니다(예를 들어, 위의 호스트 구조 예에서 a_st는 SQL에서 참조될 수 없습니다). 구조는 중간 레벨 구조를 포함할 수 없습니다. 이전 예에서 a_st는 호스트 변수로 사용될 수 없거나 SQL문에서 참조할 수 없습니다. SQL 자료에 대한 호스트 구조에는 두 레벨이 있으며 명명된 세트의 호스트 변수로서 간주됩니다. 호스트 구조가 정의된 후에는 여러 개의 호스트 변수(즉, 호스트 구조를 구성하는 호스트 변수명)를 리스트하는 대신 SQL문에서 호스트 구조를 참조할 수 있습니다.

예를 들어 다음과 같이 표 CORPDATA.EMPLOYEE의 선택된 행으로부터 모든 열 값을 검색할 수 있습니다.

```

struct { char empno[7];
        struct          { short int firstname_len;
                          char  firstname_text[12];
                          }  firstname;
        char midint,
        struct          { short int lastname_len;
                          char  lastname_text[15];
                          }  lastname;
        char workdept[4];
        } pemp1;
.....
strcpy("000220",pemp1.empno);
.....
exec sql
  SELECT *
  INTO :pemp1
  FROM corpdata.employee
  WHERE empno=:pemp1.empno;

```

pemp1의 선언 부분에서 이름과 성의 두 개의 가변 길이 스트링 요소가 구조에 포함되어 있음에 주의하십시오.

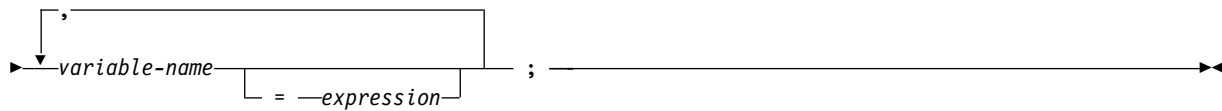
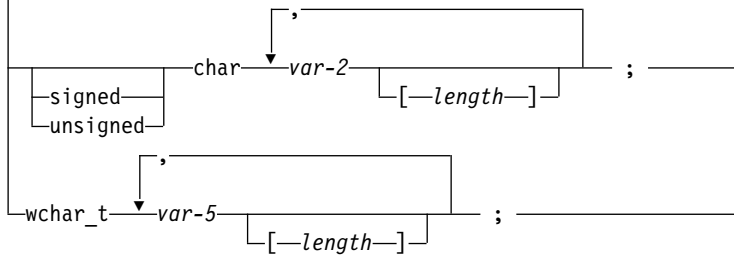
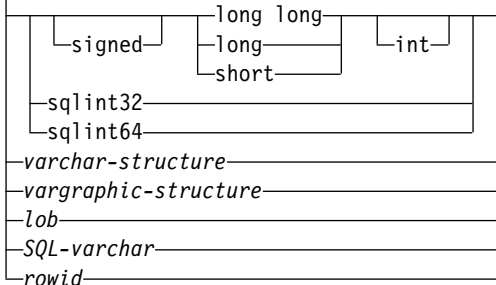
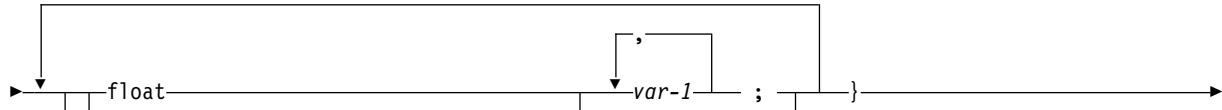
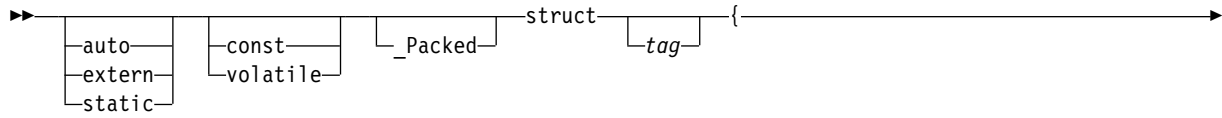
자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 선언』
- 36 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 인디케이터 배열』

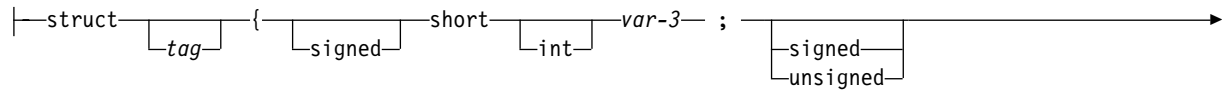
SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 선언

다음 그림은 호스트 구조 선언에 대한 유효한 구문을 보여줍니다.

호스트 구조



varchar-structure:



호스트 구조(계속)

vargraphic-structure:

```
|—struct—  
|—tag— { —signed— short —int—  
|—var-6— ; —wchar_t—var-7—[—length—]— ; —}
```

lob:

```
|—SQL TYPE IS—  
|—BLOB— (—lob-length—)  
|—CLOB—  
|—DBCLOB—  
|—K—  
|—M—  
|—G—  
|—BLOB_LOCATOR—  
|—CLOB_LOCATOR—  
|—DBCLOB_LOCATOR—  
|—BLOB_FILE—  
|—CLOB_FILE—  
|—DBCLOB_FILE—
```

SQL-varchar:

```
|—VARCHAR—variable-name—[—length—]
```

rowid:

```
|—SQL TYPE IS ROWID—
```

주:

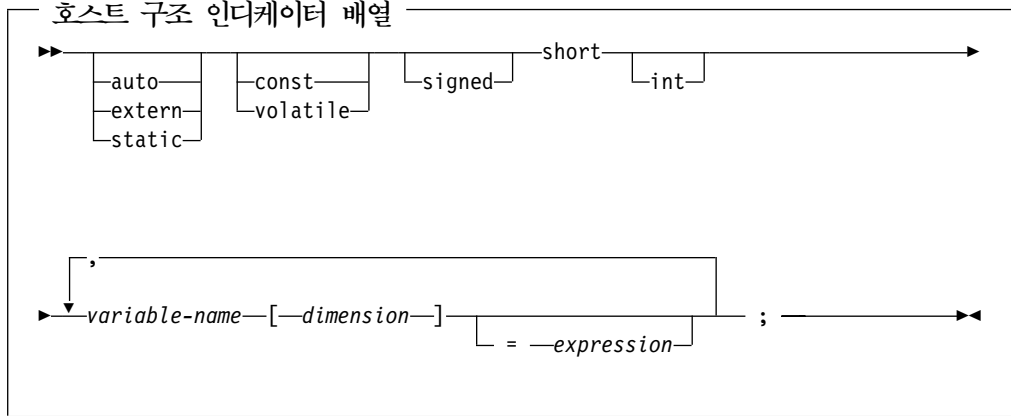
1. 숫자, 문자, 그래픽, LOB 및 ROWID 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트, 그래픽 호스트 변수, LOB 호스트 변수 및 ROWID 호스트 변수 아래에 있는 주를 참조하십시오.
2. char이나 wchar_t의 뒤에 오는 short int 배열의 구조는 SQL C 및 C++ 사전컴파일러에 의해 VARCHAR나 VARGRAPHIC 구조로 해석됩니다.
3. _Packed는 C++에서 사용될 수 없습니다. 대신, 선언 이전에 #pragma pack(1)를 지정하고 선언 후 #pragma pack()를 지정하십시오.

```
#pragma pack(1)  
struct {  
    short myshort;  
    long mylong;  
    char mychar[5];  
} a_st;  
#pragma pack()
```

4. sqlint32 또는 sqlint64를 사용 중이면 sqlsystem.h 헤더 파일이 포함되어야 합니다.

SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 인디케이터 배열

다음 그림은 호스트 구조 인디케이터 배열 선언에 대한 유효한 구문을 보여줍니다.



주: dimension은 1-32767 사이의 정수 상수이어야 합니다.

SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조의 배열 사용

C와 C++ 프로그램에서는 차원 속성을 가진 호스트 구조 배열을 정의할 수 있습니다. 호스트 구조 배열에는 배열이 복수 레벨 구조 내에서 발생하는 경우에도 최대 두 개의 레벨을 갖습니다. 가변 길이 문자 스트링 또는 가변 길이 그래픽 스트링이 사용되지 않으면 다른 구조는 필요하지 않습니다.

C 예:

```
struct {
    _Packed struct{
        char c1_var[20];
        short c2_var;
    } b_array[10];
} a_struct;
```

C++ 예:

```
#pragma pack(1)
struct {
    struct {
        char c1_var[20];
        short c2_var;
    } b_array[10];
} a_struct;
#pragma pack()
```

다음 사항이 참입니다.

- b_array의 모든 멤버가 유효한 변수 선언이어야 합니다.
- _Packed 속성은 struct 태그에 지정되어야 합니다.

- b_array는 c1_var 및 c2_var 멤버를 포함하는 호스트 구조의 배열 이름입니다.
- b_array는 FETCH문 및 INSERT문의 블록 양식에서만 사용할 수 있습니다.
- c1_var 및 c2_var는 SQL문에서 유효한 호스트 변수가 아닙니다.
- 구조는 중간 레벨 구조를 포함할 수 없습니다.

예를 들면 C에서 다음을 사용하여 커서로부터 10행을 검색할 수 있습니다.

```
_Packed struct {char first_initial;
                 char middle_initial;
                 _Packed struct {short lastname_len;
                                 char lastname_data[15];
                                 } lastname;
                 double total_salary;
                 } employee_rec[10];
struct { short inds[4];
        } employee_inds[10];
...
EXEC SQL DECLARE C1 CURSOR FOR
  SELECT SUBSTR(FIRSTNME,1,1), MIDINIT, LASTNAME,
         SALARY+BONUS+COMM
  FROM CORPDATA.EMPLOYEE;
EXEC SQL OPEN C1;
EXEC SQL FETCH C1 FOR 10 ROWS INTO :employee_rec:employee_inds;
...
```

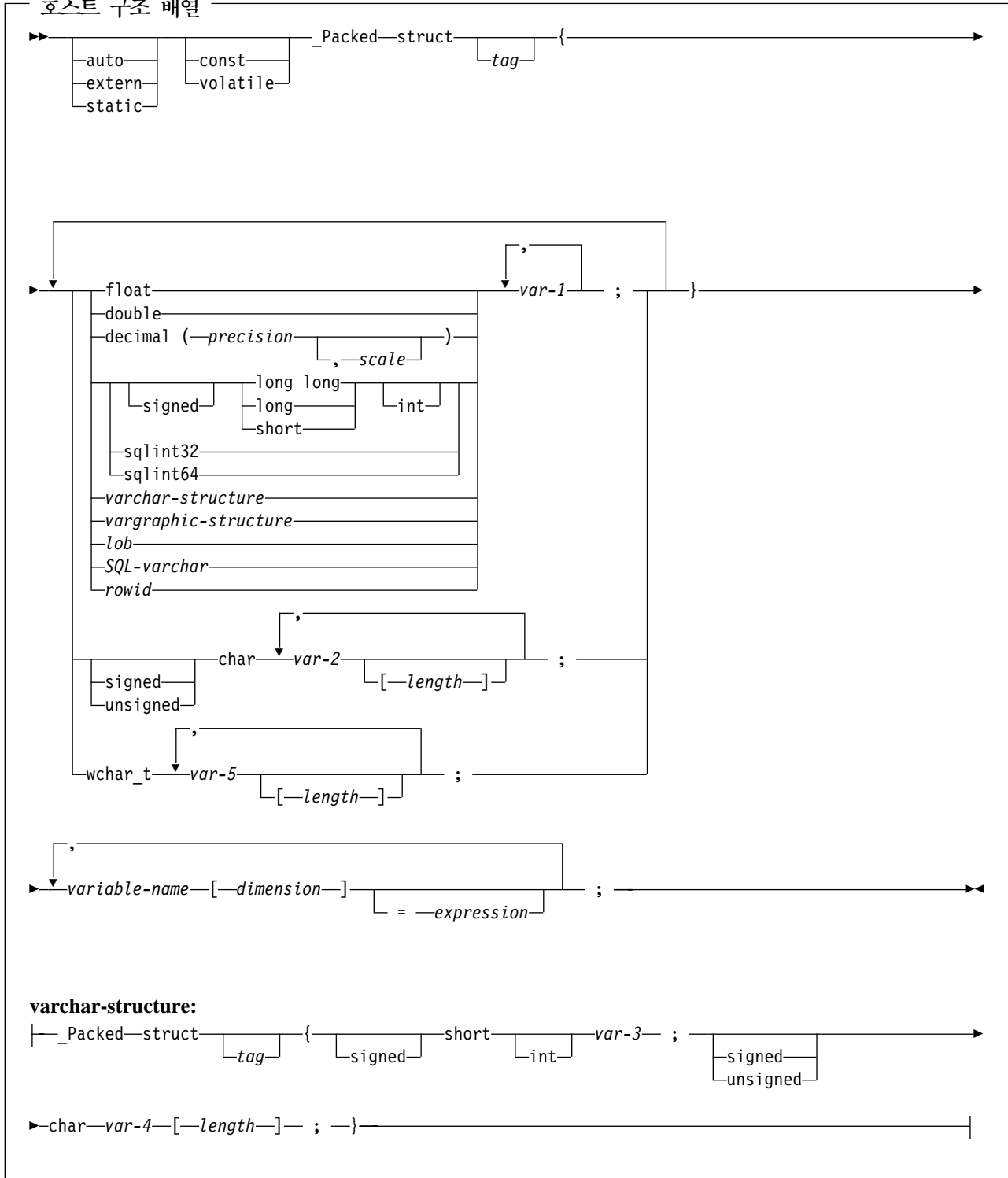
자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 C 및 C++ 어플리케이션의 호스트 구조 배열』
- 39 페이지의 『SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 배열 인 디케이터 구조』

SQL을 사용하는 C 및 C++ 어플리케이션의 호스트 구조 배열

다음 그림은 호스트 구조 배열 선언에 대한 유효한 구문을 보여줍니다.

호스트 구조 배열



- 호스트 변수가 포인터로 선언되는 경우에는 동일한 소스 파일에서 동일명을 가진 다른 호스트 변수를 선언할 수 없습니다. 예를 들어 다음 중 두 번째 선언 부분은 유효하지 않습니다.

```
char *mychar;           /* This declaration is valid           */
char mychar;          /* But this one is invalid           */
```

- 호스트 변수가 SQL문 내에서 참조될 때 NUL 종료 문자 배열 포인터를 제외하고는 그 호스트 변수가 선언된 그대로 참조되어야 합니다. 예를 들어 다음 선언 부분에는 괄호가 필요합니다.

```
char (*mychara)[20];   /* ptr to char array of 20 bytes     */
```

그러나 SELECT문과 같은 SQL문에서 호스트 변수를 참조할 때는 괄호가 허용되지 않습니다.

```
EXEC SQL SELECT name INTO :*mychara FROM mytable;
```

- 별표(*)만이 호스트 변수명에 대한 연산자로 사용될 수 있습니다.
- 별표(*)가 이름의 일부로 간주되므로 호스트 변수명의 최대 길이는 지정된 별표 수에 영향을 받습니다.
- 구조에 대한 포인터는 가변 문자 구조를 제외하고는 호스트 변수로 사용할 수 없습니다. 또한 구조 내의 포인터 필드는 호스트 변수로는 사용할 수 없습니다.
- SQL은 기본 호스트 변수에 대해 지정된 모든 기억영역이 할당되도록 요구합니다. 기억영역이 할당되지 않을 경우에는 예측 불가능한 결과가 발생할 수 있습니다.

SQL을 사용하는 C 및 C++ 어플리케이션의 typedef

typedef 선언을 사용하여 short, float 및 double과 같은 C 유형 지정자 대신에서 사용할 사용자 고유의 식별자를 정의할 수도 있습니다. 호스트 변수를 선언하는 데 사용되는 typedef 식별자는 typedef 선언이 다른 블록이나 프로시저에 있는 경우에도 프로그램 내에서 고유해야 합니다. 프로그램에 BEGIN DECLARE SECTION 및 END DECLARE SECTION문이 포함되면 typedef 선언에는 BEGIN DECLARE SECTION 및 END DECLARE SECTION이 포함될 필요가 없습니다. typedef 식별자는 BEGIN DECLARE SECTION 내의 SQL 사전컴파일러에서 인식됩니다. C 및 C++ 사전컴파일러는 호스트 변수 선언에서와 동일하게 typedef 선언의 서브세트만 인식합니다.

유효한 typedef문의 예:

- long typedef 선언 후 typedef를 참조하는 호스트 변수 선언.

```
typedef long int LONG_T;
LONG_T I1, *I2;
```

- 문자 배열 길이는 typedef나 호스트 변수 선언에 지정될 수 있으나 둘 모두에는 지정될 수 없습니다.

```

typedef char NAME_T[30];
typedef char CHAR_T;
CHAR_T name1[30]; /* Valid */
NAME_T name2; /* Valid */
NAME_T name3[10]; /* Not valid for SQL use */

```

- SQL TYPE IS문은 typedef에 사용될 수 있습니다.

```

typedef SQL TYPE IS CLOB(5K) CLOB_T;
CLOB_T clob_var1;

```

- 기억장치 클래스(auto, extern, static), 휘발성 또는 상주 규정자는 호스트 변수 선언에 지정될 수 있습니다.

```

typedef short INT_T;
typedef short INT2_T;
static INT_T i1;
volatile INT2_T i2;

```

- 구조의 typedef가 지원됩니다.

```

typedef _Packed struct {char dept[3];
                        char deptname[30];
                        long Num_employees;} DEPT_T;

DEPT_T dept_rec;
DEPT_T dept_array[20]1 /* use for blocked insert or fetch */

```

SQL을 사용하는 C 및 C++ 어플리케이션에서 ILE C 컴파일러 외부 파일 설명 사용

C 또는 C++ #pragma mapinc 지시문과 #include 지시문을 사용하여 프로그램에 외부 파일을 포함시킬 수 있습니다. SQL과 함께 사용할 때 특정 형식의 #pragma mapinc 지시문만 SQL에 의해 인식됩니다. 필요한 모든 요소가 지정되지 않으면 사전컴파일러는 지시문을 무시하고 호스트 변수 구조를 생성하지 않습니다. 필수 요소는 다음과 같습니다.

- Include명
- 외부 서술 파일명
- 형식명 또는 형식명의 리스트
- 옵션
- 변환 옵션

라이브러리명, 유니언명, 변환 옵션 및 접두부명은 옵션입니다. 사용자가 작성한 typedef 문이 사전컴파일러에 의해 인식되지 않아도 #pragma mapinc와 #include 지시문에 의해 작성된 명령문은 인식됩니다. SQL은 옵션 매개변수에 대한 SQL, 출력, 양쪽 모두 및 키 값을 지원합니다. 변환 옵션의 경우 지원되는 값은 D, p, z, _P 및 1BYTE_CHAR입니다. 이들 옵션은 D와 p가 함께 지정될 수 없는 경우를 제외하고는 어떤 순서로든 지정될 수 있습니다. #pragma mapinc와 #include 지시문에 의해 작성된 typedef union을 사용해 선언된 unions는 SQL문에서 호스트 변수로 사용될 수 없

습니다. typedef 구조가 들어 있는 구조는 SQL문에서는 사용될 수 없습니다. typedef 를 사용하여 선언된 구조는 사용될 수 있습니다.

iSeries용 DB2 UDB 프로그래밍 개념 정보의 iSeries용 DB2 UDB 샘플 표에 설명된 샘플 표 DEPARTMENT의 정의를 검색하기 위해 다음을 코딩할 수 있습니다.


```
#pragma mapinc ("dept","CORPDATA/DEPARTMENT(*ALL)","both")
#include "dept"
CORPDATA_DEPARTMENT_DEPARTMENT_both_t Dept_Structure;
```

Dept_Structure라는 호스트 구조는 DEPTNO, DEPTNAME, MGRNO 및 ADMRDEPT 요소로 정의됩니다. 이러한 필드명은 SQL문에서 호스트 변수로 사용될 수 있습니다.

주: DATE, TIME 및 TIMESTAMP 열은 문자 호스트 변수 정의를 생성합니다. 이들은 동일한 비교와 할당 규칙을 가진 SQL에 의해 DATE, TIME 및 TIMESTAMP 열로 처리됩니다. 예를 들어 날짜 호스트 변수는 날짜가 유효하게 표시된 DATE 열이나 문자 스트링에 대해서만 비교될 수 있습니다.

GRAPHIC 또는 VARGRAPHIC 열에 UCS-2 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UCS-2 CCSID를 갖게 됩니다.

존, 2진(비존 소수 자릿수 필드) 및 선택적으로 십진수가 iSeries용 ILE C 내의 문자 필드에 맵핑되어도, SQL에서는 이들 필드를 숫자로 처리합니다. 확장 프로그램 모델(EPM) 루틴을 사용하여 존 및 팩 10진 자료를 변환하기 위해 이들 필드를 조

작할 수 있습니다. 자세한 정보는 ILE C for iSeries™ Language Reference  책을 참조하십시오.

동등한 SQL 및 C 또는 C++ 자료 유형 판별

사전컴파일러는 다음 표에 있는 호스트 변수의 기본 SQLTYPE 및 SQLLEN을 판별합니다. 호스트 변수가 인디케이터 변수로 표시되면 SQLTYPE은 기본 SQLTYPE에 1을 더한 값이 됩니다.

표 1. 일반 SQL 자료 유형에 맵핑되는 C나 C++ 선언

C 또는 C++ 자료 유형	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
short int	500	2	SMALLINT
long int	496	4	INTEGER
long long int	492	8	BIGINT
십진수(p,s)	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s)
float	480	4	FLOAT(단정밀도)
double	480	8	FLOAT(배정밀도)
단일 문자 양식	452	1	CHAR(1)

표 1. 일반 SQL 자료 유형에 맵핑되는 C나 C++ 선언 (계속)

C 또는 C++ 자료 유형	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
NUL 종료 문자 양식	460	length	VARCHAR(길이 - 1)
VARCHAR 구조화 양식, 길이 < 255	448	length	VARCHAR(길이)
VARCHAR 구조 양식, 길이 > 254	456	length	VARCHAR(길이)
단일 그래픽 양식	468	1	GRAPHIC(1)
NUL 종료 단일 그래픽 양식	400	length	VARGRAPHIC(길이 - 1)
VARGRAPHIC 구조화 양식, 길이 < 128	464	length	VARGRAPHIC(길이)
VARGRAPHIC 구조화 양식, 길이 > 127	472	length	VARGRAPHIC(길이)

다음 표는 제공된 SQL 자료 유형과 동일한 C 또는 C++ 자료 유형을 판별하는 데 사용될 수 있습니다.

표 2. 일반 C 또는 C++ 선언 부분에 맵핑되는 SQL 자료 유형

SQL 자료 유형	C 또는 C++ 자료 유형	주
SMALLINT	short int	
INTEGER	long int	
BIGINT	long long int	
DECIMAL(p,s)	십진수(p,s)	p는 1 - 31의 양수이고, s는 0 - 31의 양수입니다.
NUMERIC(p,s) 또는 0이 아닌 스케일 2진	같은 것이 없습니다.	십진수 사용(p,s).
FLOAT(단정밀도)	float	
FLOAT(배정밀도)	double	
CHAR(1)	단일 문자 양식	
CHAR(n)	같은 것이 없습니다.	n>1인 경우 NUL 종료 문자 양식을 사용하지십시오.
VARCHAR(n)	NUL 종료 문자 양식	NUL 종료자를 수용하려면 최소한 n+1을 허용하십시오. 자료가 문자 NUL(\0)을 포함할 수 있는 경우 VARCHAR 구조화 양식 또는 SQL VARCHAR을 사용하지십시오. n은 양의 정수입니다. n의 최대값은 32740입니다.
	VARCHAR 구조화 양식	n의 최대값은 32740입니다. 또한 SQL VARCHAR 양식도 사용됩니다.

표 2. 일반 C 또는 C++ 선언 부분에 맵핑되는 SQL 자료 유형 (계속)

SQL 자료 유형	C 또는 C++ 자료 유형	주
BLOB	없음	C 또는 C++로 BLOB를 선언하려면 SQL TYPE IS를 사용하십시오.
CLOB	없음	C 또는 C++로 CLOB를 선언하려면 SQL TYPE IS를 사용하십시오.
GRAPHIC(1)	단일 그래픽 양식	
GRAPHIC(n)	같은 것이 없습니다.	$n > 1$ 인 경우 NUL 종료 그래픽 양식을 사용하십시오.
VARGRAPHIC(n)	NUL 종료 그래픽 양식	자료가 그래픽 NUL 값(0/0)을 포함할 수 있는 경우 VARGRAPHIC 구조화 양식을 사용하십시오. NUL 종료자를 수용하려면 최소한 $n+1$ 을 허용하십시오. n 은 양의 정수입니다. n 의 최대값은 16370입니다.
	VARGRAPHIC 구조화 양식	n 은 양의 정수입니다. n 의 최대값은 16370입니다.
DBCLOB	없음	C 또는 C++로 DBCLOB를 선언하려면 SQL TYPE IS를 사용하십시오.
DATE	NUL 종료 문자 양식	형식이 *USA, *ISO, *JIS 또는 *EUR인 경우 NUL 종료자를 수용하려면 최소한 11자를 허용하십시오. 형식이 *MDY, *YMD 또는 *DMY인 경우 NUL 종료 기호를 수용하려면 최소한 9자를 허용하십시오. 형식이 *JUL인 경우 NUL 종료 기호를 수용하려면 최소한 7자를 허용하십시오.
	VARCHAR 구조화 양식	형식이 *USA, *ISO, *JIS 또는 *EUR인 경우 최소한 10자를 허용하십시오. 형식이 *MDY, *YMD 또는 *DMY인 경우 최소한 8자를 허용하십시오. 형식이 *JUL이면 최소한 6자를 허용합니다.
TIME	NUL 종료 문자 양식	NUL 종료자를 수용하려면 최소한 7자(초까지 포함하려면 9자)를 허용하십시오.
	VARCHAR 구조화 양식	최소한 6자를 허용합니다. 초를 포함할 경우에는 8자를 허용합니다.

표 2. 일반 C 또는 C++ 선언 부분에 맵핑되는 SQL 자료 유형 (계속)

SQL 자료 유형	C 또는 C++ 자료 유형	주
TIMESTAMP	NUL 종료 문자 양식	NUL 종료자를 수용하려면 최소한 20자(전체 정밀도의 마이크로초까지 포함하려면 27)를 허용하십시오. n이 27보다 작으면 마이크로초 부분에서 잘립니다.
	VARCHAR 구조화 양식	최소한 19자를 허용하십시오. 전체 정밀도 마이크로초를 포함하려면 26자를 허용하십시오. 문자의 수가 26보다 작을 경우 마이크로초 부분에서 잘립니다.
DATALINK	지원되지 않습니다.	
ROWID	없음	C 또는 C++로 ROWID를 선언하려면 SQL TYPE IS를 사용하십시오.

자세한 내용은 『C 및 C++ 변수 선언 및 사용에 대한 주』를 참조하십시오.

C 및 C++ 변수 선언 및 사용에 대한 주

작은 따옴표와 큰 따옴표는 C, C++ 및 SQL에서 서로 다른 의미를 가집니다. C 및 C++는 큰 따옴표를 사용하여 스트링 상수를 구분하고 작은 따옴표로 문자 상수를 구분합니다. SQL에서는 이러한 구분이 없으나 큰 따옴표는 분리 식별자에 사용되고 작은 따옴표는 문자 스트링 상수를 구분하는 데 사용됩니다. SQL에서 문자 자료는 정수 자료와 구분됩니다.

SQL을 사용하는 C 및 C++ 어플리케이션에서 인디케이터 변수 사용

인디케이터 변수는 2바이트 정수입니다. 또한 인디케이터 구조(반단어 정수 변수의 배열로 정의됨)를 지정하여 호스트 구조를 지원할 수도 있습니다. 검색시 연관 호스트 변수에 널(null)이 할당되었는지 여부를 나타내기 위해 인디케이터 변수를 사용합니다. 음의 인디케이터 변수는 열에 할당할 때 널이 할당되어야 함을 나타내기 위해 사용됩니다.

자세한 정보는 SQL 참조서의 인디케이터 변수 주제를 참조하십시오.

인디케이터 변수는 호스트 변수와 동일한 방식으로 선언됩니다. 사용자에게 적절하게 보이도록 두 가지 선언 부분을 여러 가지 방법으로 혼합할 수 있습니다.

예:

다음 명령문을 보십시오.

```
EXEC SQL FETCH CLS_CURSOR INTO :ClsCd,  
                                     :Day :DayInd,  
                                     :Bgn :BgnInd,  
                                     :End :EndInd;
```

이 경우 다음과 같이 변수를 선언할 수 있습니다.

```
EXEC SQL BEGIN DECLARE SECTION;  
char ClsCd[8];  
char Bgn[9];  
char End[9];  
short Day, DayInd, BgnInd, EndInd;  
EXEC SQL END DECLARE SECTION;
```

제 3 장 COBOL 어플리케이션에서의 SQL문 코딩

iSeries 시스템은 둘 이상의 COBOL 컴파일러를 지원합니다. DB2 UDB 조회 관리자 및 SQL 개발 킷 사용권 프로그램만이 iSeries용 COBOL 및 iSeries용 COBOL 언어를 지원합니다. 이 장에서는 COBOL 프로그램에 SQL문을 삽입할 때의 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.

자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 COBOL 어플리케이션에서 SQL 통신 영역 정의』
- 51 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 SQL 설명자 영역 정의』
- 52 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 SQL문 삽입』
- 54 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 호스트 변수 사용』
- 64 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 사용』
- 73 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 외부 파일 설명 사용』
- 74 페이지의 『동등한 SQL 및 COBOL 자료 유형 판별』
- 78 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 인디케이터 변수 사용』

SQL문의 사용 방법을 보여 주는 샘플 COBOL 프로그램에 대해 자세한 사항은 부록 A 『iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램』을 참조하십시오.

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

SQL을 사용하는 COBOL 어플리케이션에서 SQL 통신 영역 정의

SQL문이 들어 있는 COBOL 프로그램은 반드시 다음 중 하나 또는 모두를 포함하고 있어야 합니다.

- PICTURE S9(9) BINARY, PICTURE S9(9) COMP-4 또는 PICTURE S9(9) COMP로 선언된 SQLCODE 변수
- PICTURE X(5)로 선언된 SQLSTATE 변수

또는

- SQLCA(SQLCODE 및 SQLSTATE 변수가 들어 있는)

SQLCODE 값과 SQLSTATE 값은 각각의 SQL문이 실행된 후에 데이터베이스 관리자에 의해 설정됩니다. 어플리케이션은 SQLCODE 값을 체크하여 최종 SQL문이 제대로 수행되었는지를 판별합니다.

SQLCA는 COBOL 프로그램에서 직접 또는 SQL INCLUDE문을 사용하여 작성될 수 있습니다. SQL INCLUDE문을 사용할 때에는 다음의 표준 선언 부분이 포함되어야 합니다.

```
EXEC SQL INCLUDE SQLCA END-EXEC.
```

SQLCODE, SQLSTATE 및 SQLCA 변수 선언은 사용자 프로그램의 WORKING-STORAGE SECTION 또는 LINKAGE SECTION에 표시되어야 하며 그러한 섹션에 레코드 서술 항목이 지정될 수 있는 곳이면 어디든지 위치될 수 있습니다.

INCLUDE문 사용시 SQL COBOL 사전컴파일러는 SQLCA에 대한 COBOL 소스 명령문을 포함합니다.

```
01 SQLCA.
  05 SQLCAID      PIC X(8).
  05 SQLCABC      PIC S9(9) BINARY.
  05 SQLCODE      PIC S9(9) BINARY.
  05 SQLERRM.
    49 SQLERRML   PIC S9(4) BINARY.
    49 SQLERRMC   PIC X(70).
  05 SQLERRP      PIC X(8).
  05 SQLERRD      OCCURS 6 TIMES
                  PIC S9(9) BINARY.

  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).
```

iSeries용 ILE COBOL의 경우 SQLCA는 GLOBAL 절을 사용하여 선언됩니다. 프로그램에 SQLCODE에 대한 선언이 있고 사전컴파일러에 의해 SQLCA가 제공될 때 SQLCODE는 SQLCADE로 대체됩니다. 프로그램에 SQLCODE에 대한 선언이 있고 사전컴파일러에 의해 SQLCA가 제공될 때 SQLCODE는 QLCQTOT로 대체됩니다.

SQLCA에 대한 자세한 정보는 SQL 참조서의 SQL 통신 영역을 참조하십시오.

SQL을 사용하는 COBOL 어플리케이션에서 SQL 설명자 영역 정의

다음 명령문은 SQLDA를 필요로 합니다.

```
EXECUTE...USING DESCRIPTOR descriptor-name
FETCH...USING DESCRIPTOR descriptor-name
OPEN...USING DESCRIPTOR descriptor-name
CALL...USING DESCRIPTOR descriptor-name
DESCRIBE statement-name INTO descriptor-name
DESCRIBE TABLE host-variable INTO descriptor-name
PREPARE statement-name INTO descriptor-name
```

프로그램에는 SQLCA와는 달리 두 개 이상의 SQLDA가 있을 수 있습니다. SQLDA는 모든 유효한 이름을 가질 수 있습니다. SQLDA는 COBOL 프로그램에서 직접 또는 INCLUDE문에 추가되어 작성될 수 있습니다. SQL INCLUDE문 사용시 표준 SQLDA 선언 부분을 포함시켜야 합니다.

```
EXEC SQL INCLUDE SQLDA END-EXEC.
```

SQLDA에 대해 포함된 COBOL 선언은 다음과 같습니다.

```
1 SQLDA.
  05 SQLDAID      PIC X(8).
  05 SQLDABC      PIC S9(9) BINARY.
  05 SQLN         PIC S9(4) BINARY.
  05 SQLD        PIC S9(4) BINARY.
  05 SQLVAR OCCURS 0 TO 409 TIMES DEPENDING ON SQLD.
    10 SQLTYPE   PIC S9(4) BINARY.
    10 SQLLEN    PIC S9(4) BINARY.
    10 FILLER    REDEFINES SQLLEN.
      15 SQLPRECISION PIC X.
      15 SQLSCALE    PIC X.
    10 SQLRES    PIC X(12).
    10 SQLDATA   POINTER.
    10 SQLIND    POINTER.
    10 SQLNAME.
      49 SQLNAMEL PIC S9(4) BINARY.
      49 SQLNAMEC PIC X(30).
```

그림 1. COBOL에 대한 INCLUDE SQLDA 선언 부분

SQLDA 선언 부분은 사용자 프로그램의 WORKING-STORAGE SECTION 또는 LINKAGE SECTION에 표시되어야 하며 그러한 섹션에 레코드 서술 항목이 지정될 수 있는 곳이면 어디든지 자리잡을 수 있습니다. iSeries용 ILE COBOL의 경우 SQLDA는 GLOBAL 절을 사용하여 선언됩니다.

동적 SQL은 iSeries용 DB2 UDB 프로그램 개념 정보의 동적 SQL 어플리케이션에 설명되어 있는 첨단 프로그래밍 기술입니다. 동적 SQL을 사용하면 프로그램 수행 중 사

용자 프로그램이 SQL문을 개발 및 수행할 수 있습니다. 동적으로 수행하는 가변 SELECT 리스트(조회된 한 부분으로 리턴될 자료의 리스트임)가 있는 SELECT문은 SQL 설명자 영역(SQLDA)을 필요로 합니다. 그 이유는 사용자가 SELECT의 결과를 수신하기 위해 지정된 변수의 수와 유형을 미리 알 수 없기 때문입니다.

SQLDA에 대한 자세한 정보는 SQL 참조서의 SQL 설명자 영역을 참조하십시오.

SQL을 사용하는 COBOL 어플리케이션에서 SQL문 삽입

SQL문은 다음과 같이 COBOL 프로그램 섹션에서 작성될 수 있습니다.

SQL문	프로그램 섹션
BEGIN DECLARE SECTION	WORKING-STORAGE SECTION 또는
END DECLARE SECTION	LINKAGE SECTION
DECLARE VARIABLE	
DECLARE STATEMENT	
INCLUDE SQLCA	WORKING-STORAGE SECTION 또는
INCLUDE SQLDA	LINKAGE SECTION
INCLUDE Member-Name	DATA DIVISION 또는 PROCEDURE DIVISION
기타	PROCEDURE DIVISION

COBOL 프로그램에서 각 SQL문은 EXEC SQL로 시작하고 END-EXEC로 끝나야 합니다. SQL문이 두 개의 COBOL문 사이에 표시되면 마침표는 선택 가능하며 적절하지 않을 수도 있습니다. EXEC SQL 키워드는 모두 한 행에 표시되어야 하나 명령문의 나머지 부분이 다음 및 후속 행에 표시될 수도 있습니다.

예:

COBOL 프로그램에서 작성된 UPDATE문은 다음과 같습니다.

```
EXEC SQL
    UPDATE DEPARTMENT
    SET MGRNO = :MGR-NUM
    WHERE DEPTNO = :INT-DEPT
END-EXEC.
```

자세한 내용은 다음 절을 참조하십시오.

- 53 페이지의 『SQL을 사용하는 COBOL 어플리케이션의 주석』
- 53 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 SQL문의 연속』
- 53 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 포함하는 코드』
- 53 페이지의 『SQL을 사용하는 COBOL 어플리케이션의 여백』
- 54 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 순번』
- 54 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서의 이름』

- 54 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 COBOL 컴파일시 옵션』
- 54 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 명령문 레이블』
- 54 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 WHENEVER문』
- 54 페이지의 『복수 소스 COBOL 프로그램 및 SQL COBOL 사전컴파일러』

SQL을 사용하는 COBOL 어플리케이션의 주석

SQL 주석(--) 이외에도 키워드 EXEC와 SQL 사이를 제외하고 삽입된 SQL문 내에 COBOL 주석 행(7열의 * 또는 /)을 포함시킬 수 있습니다. COBOL 수정 작업 행(7열의 D)은 사전컴파일러에 의해 주석 행으로 처리됩니다.

SQL을 사용하는 COBOL 어플리케이션에서 SQL문의 연속

SQL문의 행 연속 규칙은 EXEC SQL을 한 행 내에 지정해야 하는 것을 제외하고는 다른 COBOL문의 규칙과 동일합니다.

한 행에서 다음 행으로 문자열 상수를 계속할 경우 다음 행의 첫 번째 비공백 문자는 작은 따옴표 또는 큰 따옴표 중의 하나이어야 합니다. 분리 식별자를 한 행에서 다음 행으로 계속할 경우 다음 행의 첫 번째 비공백 문자는 작은 따옴표나 큰 따옴표 중의 하나이어야 합니다.

DBCS 자료를 포함하는 상수는 계속되는 행의 72열에 SI 문자를 위치시키고 계속행의 첫 번째 스트링 분리문자 다음에 SO문자를 위치시킴으로써 복수 행에 걸쳐서 계속될 수 있습니다.

이 예에서 SQL문에 G'<AABBCCDDEEFFGGHHIIJJKK>'의 유효한 그래픽 상수가 있습니다. 중복되는 시프트는 삭제됩니다.

```
*...+...1...+...2...+...3...+...4...+...5...+...6...+...7...+...8
EXEC SQL
SELECT * FROM GRAPHTAB          WHERE GRAPHCOL = G'<AABB>
- '<CCDDEEFFGGHHIIJJKK>'
END-EXEC.
```

SQL을 사용하는 COBOL 어플리케이션에서 포함하는 코드

SQL문이나 COBOL 호스트 변수 선언문은 명령문이 삽입될 소스 코드 내의 한 지점에 다음 SQL문을 삽입하여 포함될 수 있습니다.

```
EXEC SQL INCLUDE member-name END-EXEC.
```

COBOL COPY문을 사용하여 SQL문이나 SQL문에서 참조된 COBOL 호스트 변수 선언 부분을 포함시킬 수 없습니다.

SQL을 사용하는 COBOL 어플리케이션의 여백

12-72열에 SQL문 작성. EXEC SQL을 지정된 여백 앞(즉, 12열 앞)에서 시작하면 SQL 사전컴파일러는 명령문을 인식하지 못합니다.

SQL을 사용하는 COBOL 어플리케이션에서 순번

SQL 사전컴파일러에 의해 생성된 소스 명령문은 SQL문과 동일한 순서 번호로 생성됩니다.

SQL을 사용하는 COBOL 어플리케이션에서의 이름

호스트 변수에 대해 유효한 모든 COBOL 변수명을 사용할 수 있으며 다음 제한사항에 따라야 합니다.

'SQL', 'RDI' 또는 'DSN'으로 시작하는 호스트 변수명이나 외부 항목명을 사용하지 마십시오. 이러한 이름들은 데이터베이스 관리자 예약어입니다.

FILLER를 포함하는 구조를 사용하면 SQL문에서 예상한 대로 작동되지 않을 수 있습니다. 예기치 못한 결과를 피하기 위해 COBOL 구조 내의 모든 필드를 명명하는 것이 좋습니다.

SQL을 사용하는 COBOL 어플리케이션에서 COBOL 컴파일시 옵션

COBOL PROCESS문을 사용하여 COBOL 컴파일러에 대한 컴파일시 옵션을 지정할 수 있습니다. PROCESS문은 프로그램을 작성하기 위해 사전컴파일러에 의해 호출될 때 COBOL 컴파일러에 의해 인식되지만 SQL 사전컴파일러 자체는 PROCESS문을 인식하지 못합니다. 따라서 APOST 및 QUOTE와 같은 COBOL 소스의 구문에 영향을 주는 옵션을 PROCESS문에 지정해서는 안됩니다. 대신 *APOST와 *QUOTE를 CRTSQLCBL 명령의 옵션 매개변수에 지정해야 합니다.

SQL을 사용하는 COBOL 어플리케이션에서 명령문 레이블

PROCEDURE DIVISION의 실행 가능한 SQL문은 단락명 앞에 올 수 있습니다.

SQL을 사용하는 COBOL 어플리케이션에서 WHENEVER문

SQL WHENEVER문의 GOTO절의 목표는 PROCEDURE DIVISION 내의 섹션명이나 규정되지 않은 단락명이어야 합니다.

복수 소스 COBOL 프로그램 및 SQL COBOL 사전컴파일러

SQL COBOL 사전컴파일러는 PROCESS문으로 분리된 복수 소스 프로그램의 사전컴파일을 지원하지 않습니다.

SQL을 사용하는 COBOL 어플리케이션에서 호스트 변수 사용

SQL문에서 사용된 모든 호스트 변수는 명시적으로 선언되어야 합니다. SQL문에서 사용된 호스트 변수는 SQL문에서 호스트 변수를 처음 사용하기 전에 선언되어야 합니다.

호스트 변수를 정의하기 위해 사용된 COBOL문 앞에는 BEGIN DECLARE SECTION문이 오고 뒤에는 END DECLARE SECTION문이 와야 합니다. BEGIN DECLARE

SECTION 및 END DECLARE SECTION이 지정되면 SQL문에서 사용된 모든 호스트 변수 선언은 BEGIN DECLARE SECTION과 END DECLARE SECTION문 사이에 있어야 합니다.

SQL문 내의 모든 호스트 변수 앞에는 콜론(:)이 와야 합니다.

호스트 변수는 레코드나 요소가 될 수 없습니다.

COBOL 호스트 변수명 내에서 대시(-)를 사용하려면 공백이 앞에 와야 하며 마이너스 부호가 뒤에 와야 합니다.

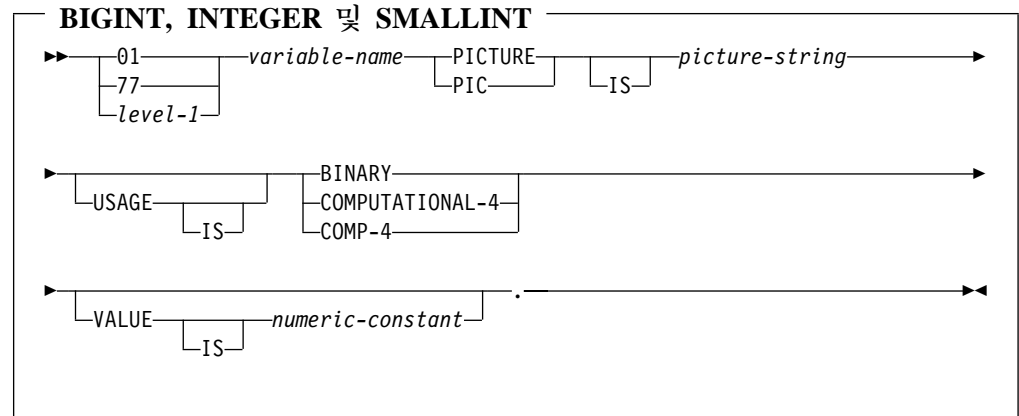
자세한 내용은 『SQL을 사용하는 COBOL 어플리케이션에서 호스트 변수 선언』을 참조하십시오.

SQL을 사용하는 COBOL 어플리케이션에서 호스트 변수 선언

COBOL 사전컴파일러만이 유효한 호스트 변수 선언 부분으로서 유효한 COBOL 선언 부분의 서브세트를 인식합니다.

SQL을 사용하는 COBOL 어플리케이션에서 숫자 호스트 변수

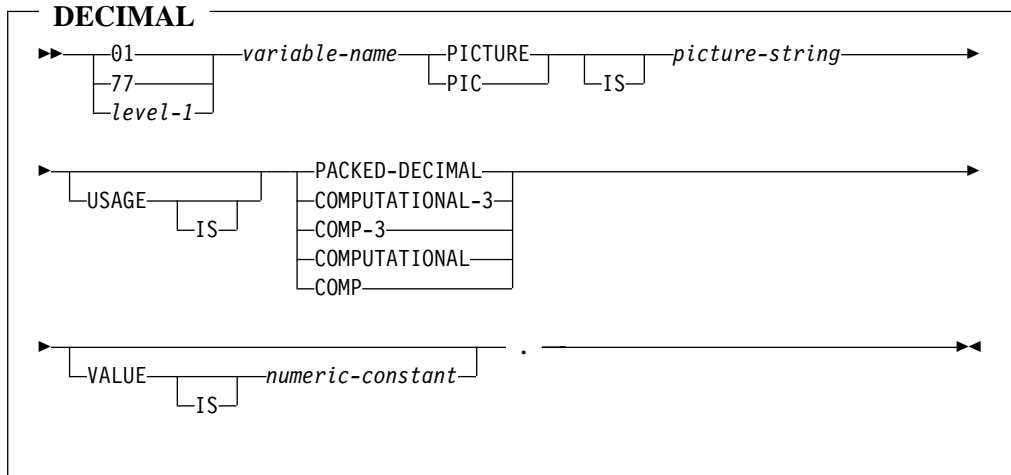
다음 그림은 유효한 정수 호스트 변수 선언 부분에 대한 구문을 표시합니다.



주:

1. BINARY, COMPUTATIONAL 및 COMP-4는 동등합니다. COMPUTATIONAL-4와 COMP-4는 ISO(표준화 국제 조직)/ANSI COBOL에서 지원하지 않는 IBM 확장자이므로 휴대용 어플리케이션은 BINARY로 코딩되어야 합니다. 이 유형과 연관된 픽처 스트링은 양식 S9(i)V9(d)(또는 S9...9V9...9, 9의 *i* 및 *d* 인스턴스)을 가져야 합니다. $i + d$ 는 18 이하이어야 합니다.
2. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

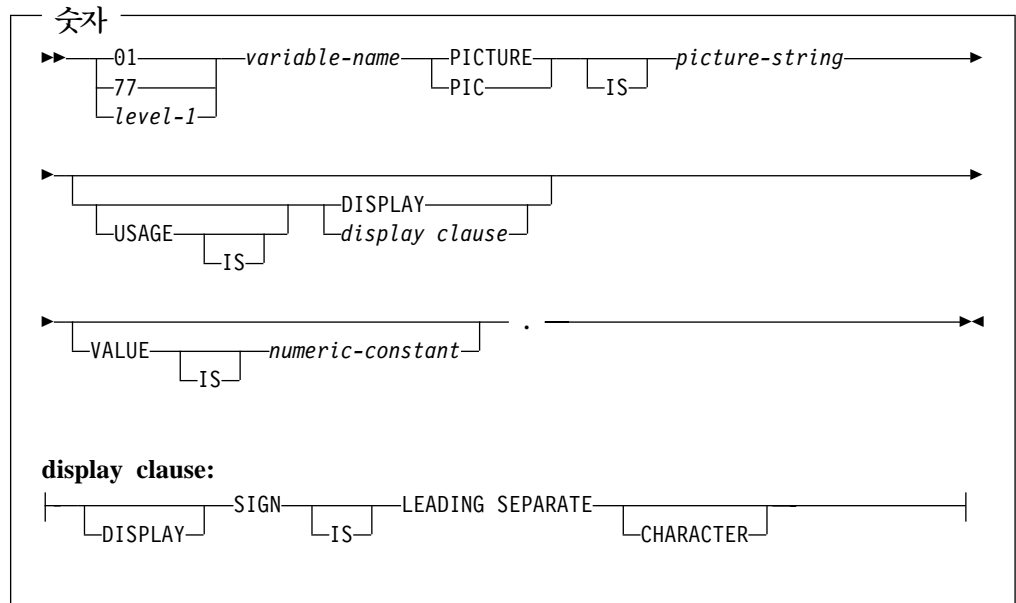
다음 그림은 유효한 10진 호스트 변수 선언 부분에 대한 구문을 표시합니다.



주:

1. PACKED-DECIMAL, COMPUTATIONAL-3 및 COMP-3은 동등합니다. COMPUTATIONAL-3과 COMP-3은 ISO/ANS COBOL에서 지원되지 않은 IBM 확장자이기 때문에 이식 가능한 어플리케이션은 PACKED-DECIMAL로 작성되어야 합니다. 이 유형과 연관된 픽처 스트링은 양식 S9(i)V9(d)(또는 S9...9V9...9, 9의 *i* 및 *d* 인스턴스)을 가져야 합니다. $i + d$ 는 18 이하이어야 합니다.
2. COMPUTATIONAL과 COMP는 동등합니다. 형상 스트링은 이와 연관되어 있으며 이들이 표시되는 자료 유형은 제품 특성입니다. 그러므로 COMP와 COMPUTATIONAL은 간단한 어플리케이션에 사용될 수 없습니다. iSeries용 COBOL 프로그램에서 이러한 유형과 관련된 픽처 스트링은 S9(i)V9(d)(또는 S9...9V9...9, 9의 *i* 및 *d* 인스턴스가 있는) 양식이어야 합니다. $i + d$ 는 18 이하이어야 합니다.
3. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

다음 그림은 유효한 숫자 호스트 변수 선언 부분에 대한 구문을 표시합니다.

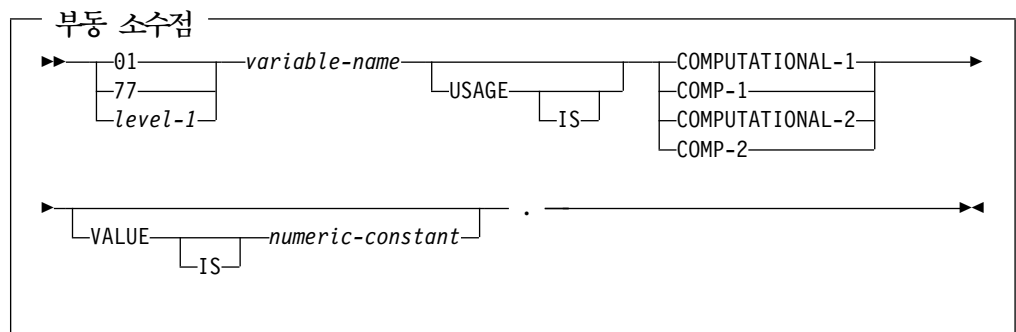


주:

1. SIGN LEADING SEMAPHORE 및 DISPLAY와 연관된 *picture-string*은 양식 S9(i)V9(d)(또는 S9...9V9...9, 9의 *i* 및 *d* 인스턴스)을 가져야 합니다. $i + d$ 는 18 이하이어야 합니다.
2. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

SQL을 사용하는 COBOL 어플리케이션에서 부동 소수점 호스트 변수

다음 그림에서는 유효한 부동 소수점 호스트 변수 선언 부분에 대한 구문을 보여줍니다. 부동 소수점 호스트 변수는 iSeries용 ILE COBOL에만 지원됩니다.



주:

1. COMPUTATIONAL-1과 COMP-1는 동등합니다. COMPUTATIONAL-2와 COMP-2는 동등합니다.
2. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

주:

1. 이러한 양식과 연관된 *picture-string-1*은 양식 S9(m) 또는 S9...9, 9의 m 인스턴스이어야 합니다. m은 1 - 4이어야 합니다.

iSeries의 COBOL이 최대 지정된 정밀도까지 값만 인식한다고 해도 데이터베이스 관리자는 S9(m) 변수의 전체 크기를 사용한다는 점에 유의하십시오. COBOL문이 수행 중이고 가변 길이 문자 스트링의 최대 길이를 효과적으로 지정된 정밀도로 제한하는 경우 이는 자료 절단 오류를 발생시킬 수 있습니다.

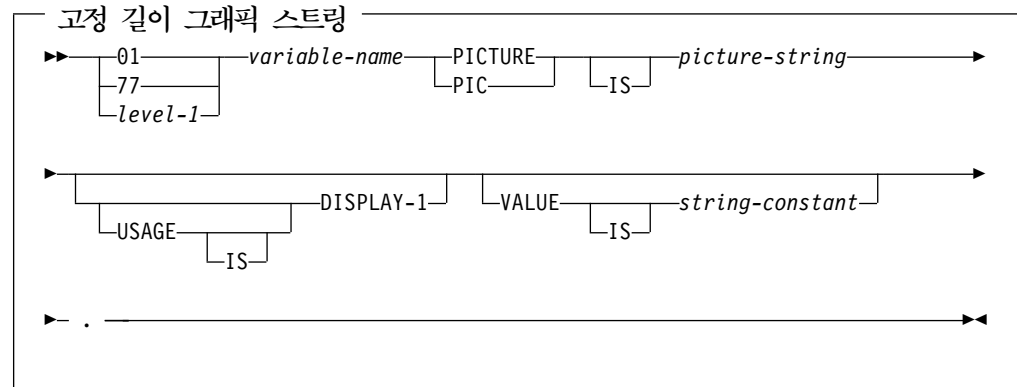
2. 이러한 양식과 연관된 *picture-string-2*는 X(m) 또는 XX...X, X의 m 인스턴스이어야 하고 여기서 $1 \leq m \leq 32\,740$ 입니다.
3. *var-1*과 *var-2*는 호스트 변수로 사용될 수 없습니다.
4. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

SQL을 사용하는 COBOL 어플리케이션에서 그래픽 호스트 변수

그래픽 호스트 변수는 iSeries용 ILE COBOL에서만 지원됩니다.

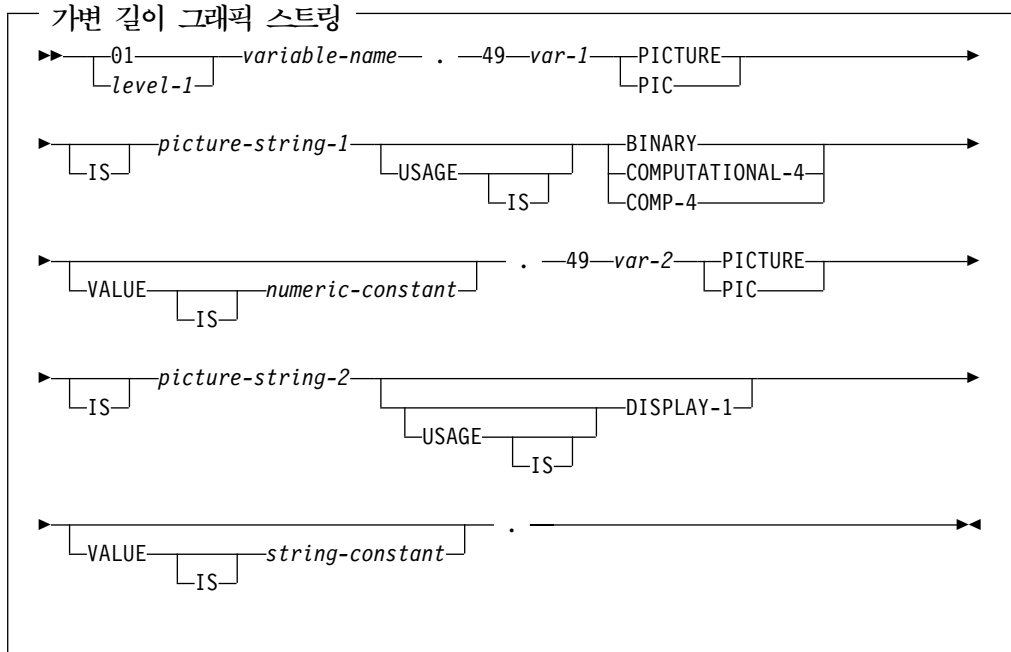
문자 호스트 변수에는 다음의 두 가지 유효한 양식이 있습니다.

- 고정 길이 그래픽 스트링
- 가변 길이 그래픽 스트링



주:

1. 이러한 양식과 연관된 픽처 스트링은 G(m)(또는 GGG...G, G의 m 인스턴스) 또는 N(m)(또는 NNN...N, N의 m 인스턴스)이어야 하며 여기서 $1 \leq m \leq 16\,383$ 입니다.
2. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.



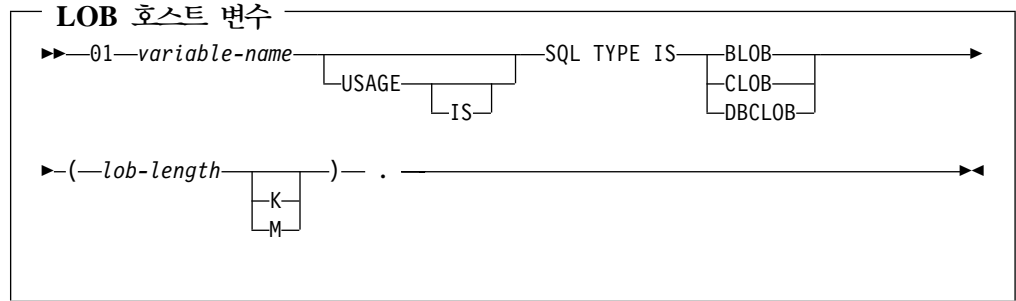
주:

1. 이러한 양식과 연관된 *picture-string-1*은 양식 S9(m) 또는 S9...9, 9의 m 인스턴스이어야 합니다. m은 1 - 4이어야 합니다.
 iSeries의 COBOL이 최대 지정된 정밀도까지 값만 인식한다고 해도 데이터베이스 관리자는 S9(m) 변수의 전체 크기를 사용한다는 점에 유의하십시오. COBOL문이 수행 중이고 가변 길이 스트링의 최대 길이를 효과적으로 지정된 정밀도로 제한하는 경우 이는 자료 절단 오류를 발생시킬 수 있습니다.
2. 이러한 양식과 연관된 *picture-string-2*는 G(m), GG...G G의 m 인스턴스, N(m) 또는 NN...N N의 m 인스턴스이어야 합니다(여기서 $1 \leq m \leq 16\ 370$).
3. *var-1*과 *var-2*는 호스트 변수로 사용될 수 없습니다.
4. *level-1*은 COBOL 레벨이 2-48 사이임을 나타냅니다.

SQL을 사용하는 COBOL 어플리케이션에서 LOB 호스트 변수

COBOL에는 LOB(대형 오브젝트)에 대한 SQL 자료 유형에 대응하는 변수가 없습니다. 이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQL TYPE IS 절을 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버에서 이 선언을 COBOL 언어 구조로 대체합니다.

LOB 호스트 변수는 iSeries용 ILE COBOL에서만 지원됩니다.



주:

1. BLOB 및 CLOB의 경우 $1 \leq \text{lob-length} \leq 15,728,640$
2. DBCLOB의 경우 $1 \leq \text{lob-length} \leq 7,864,320$
3. SQL TYPE IS, BLOB, CLOB, DBCLOB는 대소문자를 혼합하여 사용할 수 있습니다.

BLOB 예

다음 선언을 사용할 경우:

```
01 MY-BLOB SQL TYPE IS BLOB(16384).
```

다음과 같은 구조가 생성됩니다.

```
01 MY-BLOB.
  49 MY-BLOB-LENGTH PIC 9(9) BINARY.
  49 MY-BLOB-DATA PIC X(16384).
```

CLOB 예

다음 선언을 사용할 경우:

```
01 MY-CLOB SQL TYPE IS CLOB(16384).
```

다음과 같은 구조가 생성됩니다.

```
01 MY-CLOB.
  49 MY-CLOB-LENGTH PIC 9(9) BINARY.
  49 MY-CLOB-DATA PIC X(16384).
```

DBCLOB 예

다음 선언을 사용할 경우:

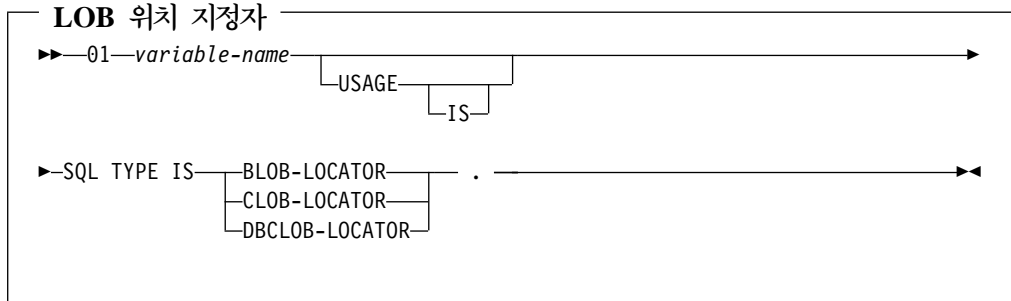
```
01MY-DBCLOB SQL TYPE IS DBCLOB(8192).
```

다음과 같은 구조가 생성됩니다.

```

01 MY-DBCLOB.
  49 MY-DBCLOB-LENGTH PIC 9(9) BINARY.
  49 MY-DBCLOB-DATA PIC G(8192) DISPLAY-1.

```



주:

1. SQL TYPE IS, BLOB-LOCATOR, CLOB-LOCATOR, DBCLOB-LOCATOR 는 대소문자를 혼합하여 사용할 수 있습니다.
2. LOB 위치 지정자는 SQL TYPE IS문으로 초기설정될 수 없습니다.

BLOB 위치 지정자 예

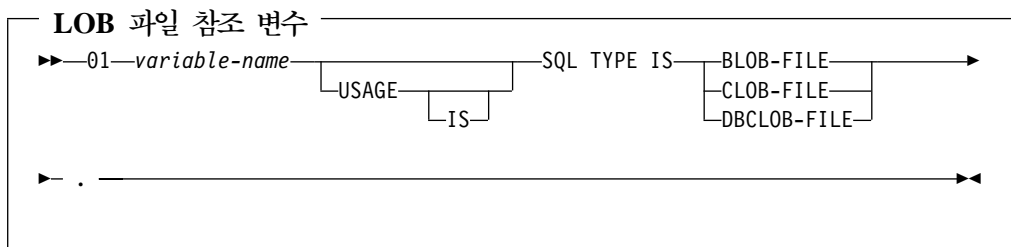
다음 선언을 사용할 경우:

```
01 MY-LOCATOR SQL TYPE IS BLOB_LOCATOR.
```

다음과 같이 생성됩니다.

```
01 MY-LOCATOR PIC 9(9) BINARY.
```

CLOB 및 DBCLOB 위치 지정자의 구문은 비슷합니다.



주: SQL TYPE IS, BLOB-FILE, CLOB-FILE, DBCLOB-FILE은 대소문자를 혼합하여 사용할 수 있습니다.

BLOB 파일 참조 예

다음 선언을 사용할 경우:

```
01 MY-FILE SQL TYPE IS BLOB-FILE.
```

다음과 같은 구조가 생성됩니다.

```
01 MY-FILE.  
  49 MY-FILE-NAME-LENGTH PIC S9(9) COMP-5.  
  49 MY-FILE-DATA-LENGTH PIC S9(9) COMP-5.  
  49 MY-FILE-FILE-OPTIONS PIC S9(9) COMP-5.  
  49 MY-FILE-NAME PIC X(255).
```

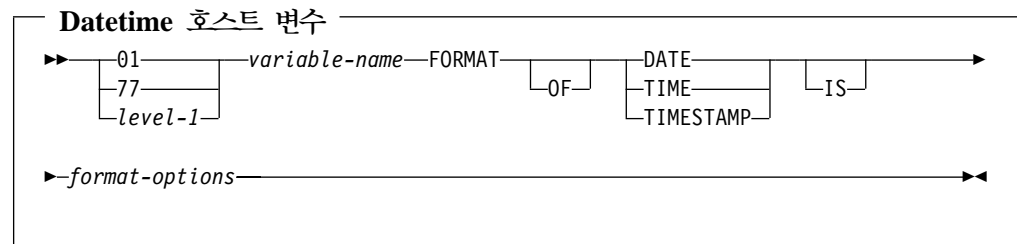
CLOB 및 DBCLOB 파일 참조 변수의 구문은 비슷합니다.

사전컴파일러는 다음의 파일 옵션 상수에 대한 선언을 생성합니다. 파일 참조 호스트 변수를 사용할 때 이러한 상수를 사용하여 xxx-FILE-OPTIONS 변수를 설정할 수 있습니다. 이러한 값에 대한 자세한 정보는 SQL 프로그래밍 개념의 LOB 파일 참조 변수를 참조하십시오.


- SQL_FILE_READ (2)
- SQL_FILE_CREATE (8)
- SQL_FILE_OVERWRITE (16)
- SQL_FILE_APPEND (32)

SQL을 사용하는 COBOL 어플리케이션에서 Datetime 호스트 변수

다음 그림은 유효한 날짜, 시간 및 시간소인 호스트 변수 선언에 대한 구문을 나타냅니다. Datetime 호스트 변수는 iSeries용 ILE COBOL에서만 지원됩니다.



주:

1. *level-1*은 COBOL 레벨이 2-48 사이임을 나타냅니다.
2. *format-options*은 COBOL 컴파일러에서 지원되는 유효한 datetime 옵션을 나타냅니다. 자세한 정보는 ILE COBOL Reference  를 참조하십시오.

SQL을 사용하는 COBOL 어플리케이션에서 ROWID 호스트 변수

COBOL에는 ROWID의 SQL 자료 유형에 대응하는 변수가 없습니다. 이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQL TYPE IS 절을 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버에서 이 선언을 COBOL 언어 구조로 대체합니다.

ROWID

▶—01—*variable-name*—SQL TYPE IS ROWID— . —————▶▶

주: SQL TYPE IS ROWID는 대소문자를 혼합하여 사용할 수 있습니다.

ROWID 예

다음 선언을 사용할 경우:

```
01 MY-ROWID SQL TYPE IS ROWID.
```

다음과 같은 구조가 생성됩니다.

```
01 MY-ROWID.  
 49 MY-ROWID-LENGTH PIC 9(2) BINARY.  
 49 MY-ROWID-DATA PIC X(40).
```

SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 사용

호스트 구조는 사용자 프로그램의 DATA DIVISION에 정의된 호스트 변수 세트입니다. 호스트 구조 자체가 복수 레벨 구조 내에서 발생하는 경우에도 최대 두 개의 레벨을 갖습니다. 다른 레벨이 49여야 하는 가변 길이 스트링의 선언 부분은 예외입니다.

호스트 구조명은 종속되는 레벨이 기본 자료항목을 명명한 그룹명이 될 수 있습니다. 예를 들면 다음과 같습니다.

```
01 A  
 02 B  
    03 C1 PICTURE ...  
    03 C2 PICTURE ...
```

이 예에서 B는 기본 항목 C1과 C2로 구성된 호스트 구조명입니다.

규정된 호스트 변수명(예를 들어 구조 내의 필드를 식별하기 위해)을 사용하여 SQL문을 작성할 때에는 구조명과 그 뒤에 마침표 및 필드명(즉, PL/I 양식)을 사용하십시오. 예를 들어 C1 OF B 또는 C1 IN B가 아닌 B.C1을 지정하십시오. 그러나 PL/I 스타일은 SQL문에 규정된 이름에만 적용됩니다. 이 방법은 COBOL문에서 규정된 이름을 작성하는 데 사용할 수 없습니다.

호스트 구조는 다음 항목 가운데 어느 것이라도 발견되면 완전한 것으로 간주됩니다.

- 영역 A에서 시작해야 하는 COBOL 항목
- 모든 SQL문(SQL INCLUDE 제외)

호스트 구조가 정의된 후에는 여러 호스트 변수를 리스트하는 대신(즉, 호스트 구조를 구성하는 자료 항목명 대신) SQL문에서 호스트 구조를 참조할 수 있습니다.

예를 들어 다음과 같이 표 CORPDATA.EMPLOYEE의 선택된 행으로부터 모든 열 값을 검색할 수 있습니다.

```
01 PEMPL.
   10 EMPNO                PIC X(6).
   10 FIRSTNME.
       49 FIRSTNME-LEN    PIC S9(4) USAGE BINARY.
       49 FIRSTNME-TEXT  PIC X(12).
   10 MIDINIT             PIC X(1).
   10 LASTNAME.
       49 LASTNAME-LEN    PIC S9(4) USAGE BINARY.
       49 LASTNAME-TEXT  PIC X(15).
   10 WORKDEPT            PIC X(3).
...
MOVE "000220" TO EMPNO.
...
      EXEC SQL
      SELECT *
      INTO :PEMPL
      FROM CORPDATA.EMPLOYEE
      WHERE EMPNO = :EMPNO
      END-EXEC.
```

PEMPL의 선언에서 FIRSTNME 및 LASTNAME의 두 개의 가변 길이 스트링 요소가 구조에 포함되어 있음에 주의하십시오.

자세한 내용은 다음 절을 참조하십시오.

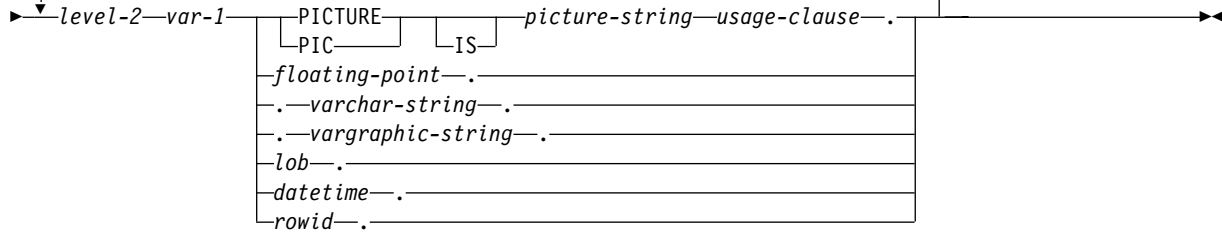
- 『SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조』
- 68 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 인디케이터 배열』
- 68 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 배열 사용』
- 69 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 배열』
- 72 페이지의 『SQL을 사용하는 COBOL 어플리케이션에서 호스트 배열 인디케이터 구조』

SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조

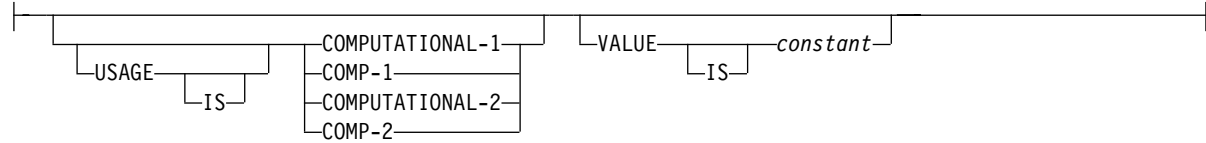
다음 그림은 올바른 호스트 구조에 대한 구문을 보여줍니다.

호스트 구조

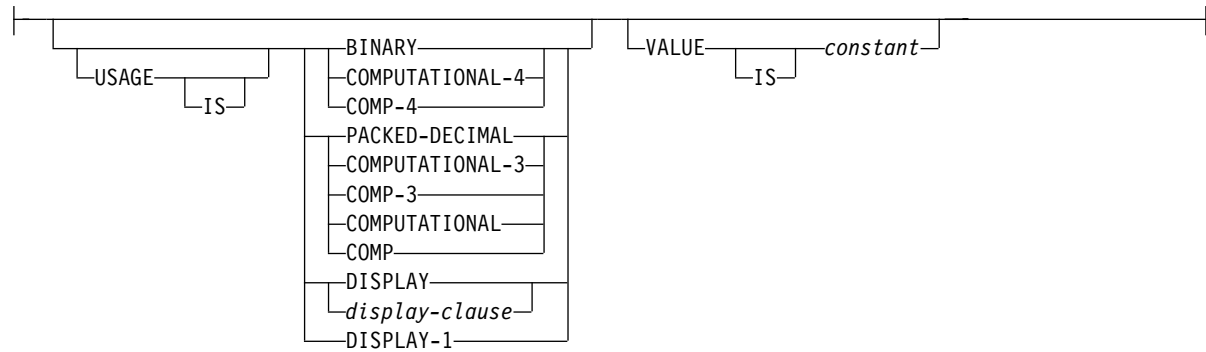
▶ *level-1-variable-name* .



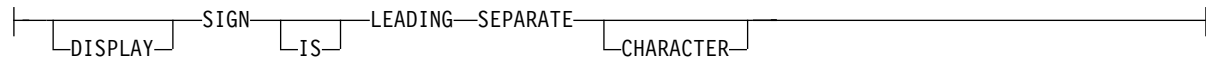
floating-point:



usage-clause:

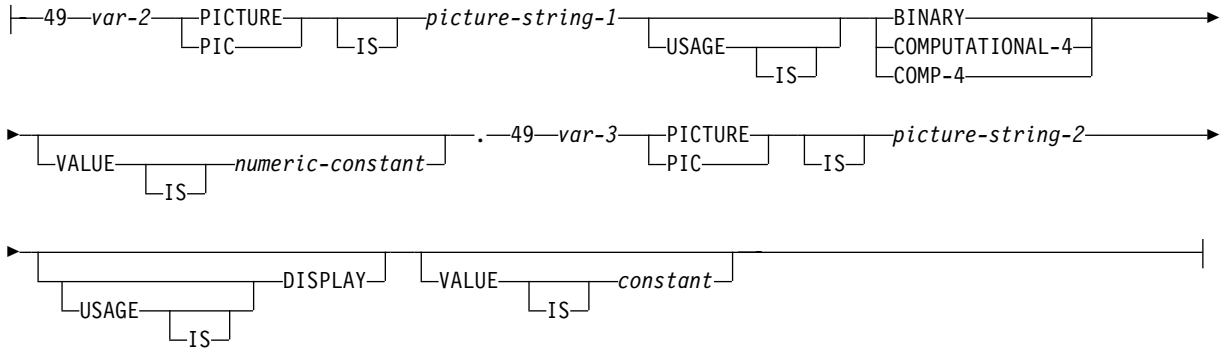


display-clause:

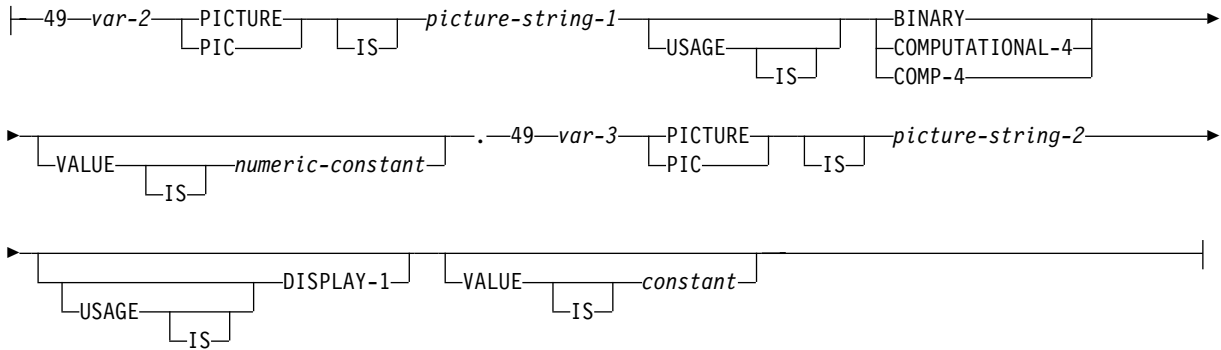


호스트 구조(계속)

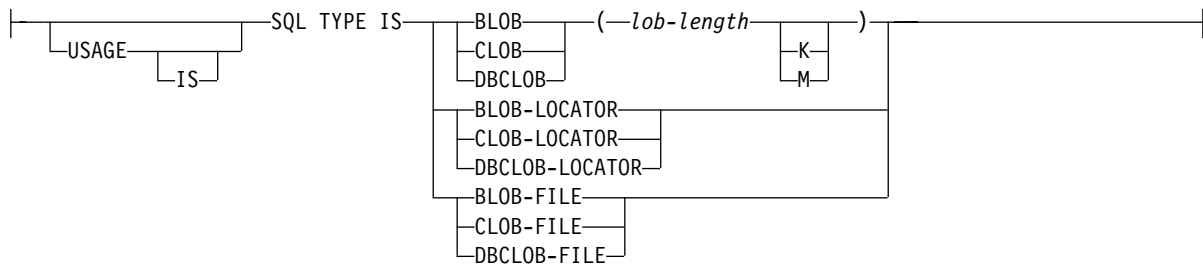
varchar-string:



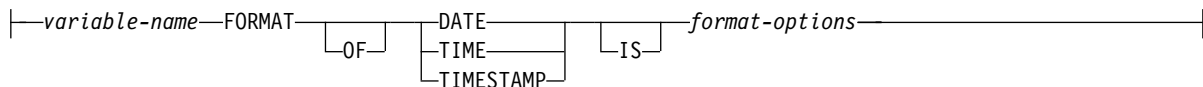
vargraphic-string:



lob:




datetime:



rowid:

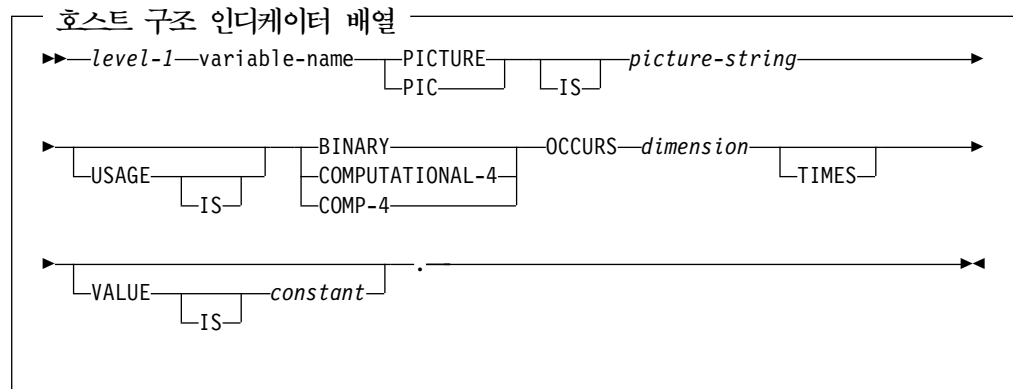


주:

1. level-1은 COBOL 레벨이 1-47 사이임을 나타냅니다.
2. level-2는 COBOL 레벨이 2-48 사이(level-2 > level-1)임을 나타냅니다.
3. 그래픽 호스트 변수, LOB 호스트 변수 및 부동 소수점 호스트 변수는 iSeries용 ILE COBOL에만 지원됩니다.
4. 숫자, 문자, 그래픽, LOB 및 ROWID 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트 변수, 그래픽 호스트 변수, LOB 호스트 변수 및 ROWID 호스트 변수 아래에 있는 주를 참조하십시오.
5. *format-options*은 COBOL 컴파일러에서 지원되는 유효한 datetime 옵션을 나타냅니다. 자세한 정보는 ILE COBOL Reference  를 참조하십시오.

SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 인디케이터 배열

다음 그림은 유효한 인디케이터 배열 선언에 대한 구문을 보여줍니다.



주:

1. dimension은 1-32767 사이의 정수이어야 합니다.
2. 레벨 1은 2-48 사이의 정수이어야 합니다.
3. BINARY, COMPUTATIONAL 및 COMP-4는 동등합니다. COMPUTATIONAL-4와 COMP-4는 ISO/ANSI COBOL에서 지원되지 않는 IBM 확장자이므로 이식 가능한 어플리케이션은 BINARY로 작성되어야 합니다. 이 유형과 연관된 *picture-string*은 형식 S9(i)(또는 9가 i만큼의 S9...9, 9)를 가져야 합니다. i는 4 이하이어야 합니다.

SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 배열 사용

호스트 구조 배열은 프로그램의 Data Division에 정의된 호스트 변수 세트이며, OCCURS 절을 갖습니다. 호스트 구조가 복수 레벨 구조 내에서 발생되더라도 호스트

구조 배열에는 최고 두 개의 레벨이 있습니다. 가변 길이 스트링은 다른 레벨 즉, 레벨 49를 필요로 합니다. 호스트 구조 배열명은 종속되는 레벨이 기본 자료 항목을 명명한 그룹명이 될 수 있습니다.

이 예에서는 다음 사항이 참입니다.

- B-ARRAY의 모든 멤버가 유효해야 합니다.
- B-ARRAY는 규정될 수 없습니다.
- B-ARRAY는 FETCH 및 INSERT문의 블록 양식으로만 사용될 수 있습니다.
- B-ARRAY는 C1-VAR 및 C2-VAR을 포함하는 호스트 구조의 배열명입니다.
- SYNCHRONIZED 속성이 지정되어서는 안됩니다.
- C1-VAR 및 C2-VAR은 SQL문에서 유효한 호스트 변수가 아닙니다. 구조는 중간 레벨 구조를 포함할 수 없습니다.

```
01 A-STRUCT.
  02 B-ARRAY OCCURS 10 TIMES.
    03 C1-VAR PIC X(20).
    03 C2-VAR PIC S9(4).
```

CORPDATA.DEPARTMENT 표에서 10개 행을 검색할 경우 다음 예를 사용하십시오.

```
01 TABLE-1.
  02 DEPT OCCURS 10 TIMES.
  05 DEPTNO PIC X(3).
  05 DEPTNAME.
    49 DEPTNAME-LEN PIC S9(4) BINARY.
    49 DEPTNAME-TEXT PIC X(29).
05 MGRNO PIC X(6).
05 ADMRDEPT PIC X(3).
01 TABLE-2.
  02 IND-ARRAY OCCURS 10 TIMES.
    05 INDS PIC S9(4) BINARY OCCURS 4 TIMES.
....
EXEC SQL
  DECLARE C1 CURSOR FOR
  SELECT *
  FROM CORPDATA.DEPARTMENT
  END-EXEC.
....
EXEC SQL
  FETCH C1 FOR 10 ROWS INTO :DEPT :IND-ARRAY
  END-EXEC.
```

SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 배열

다음 그림은 유효한 호스트 구조 배열 선언 부분에 대한 구문을 표시합니다.

호스트 구조 배열

▶ *level-1*—*variable-name*—OCCURS—*dimension*—TIMES—.

▶ *level-2*—*var-1*—
 PICTURE—*picture-string-1*—usage-clause—.
 PIC— IS
 floating-point—.
 .—varchar-string—.
 .—vargraphic-string—.
 lob—.
 datetime—.
 rowid—.

floating-point:

— USAGE— IS— COMPUTATIONAL-1— VALUE— IS— constant—
 COMP-1—
 COMPUTATIONAL-2—
 COMP-2—

usage-clause:

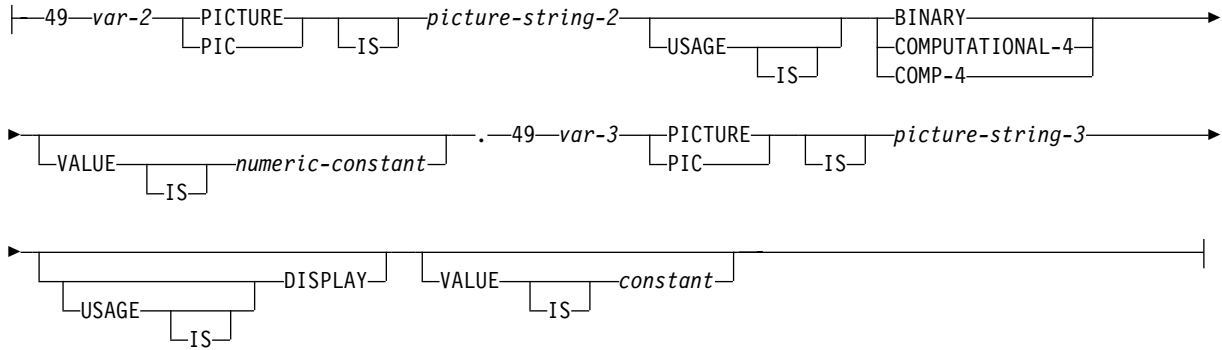
— USAGE— IS— BINARY— VALUE— IS— constant—
 COMPUTATIONAL-4—
 COMP-4—
 PACKED-DECIMAL—
 COMPUTATIONAL-3—
 COMP-3—
 COMPUTATIONAL—
 COMP—
 DISPLAY—
 display-clause—
 DISPLAY-1—

display-clause:

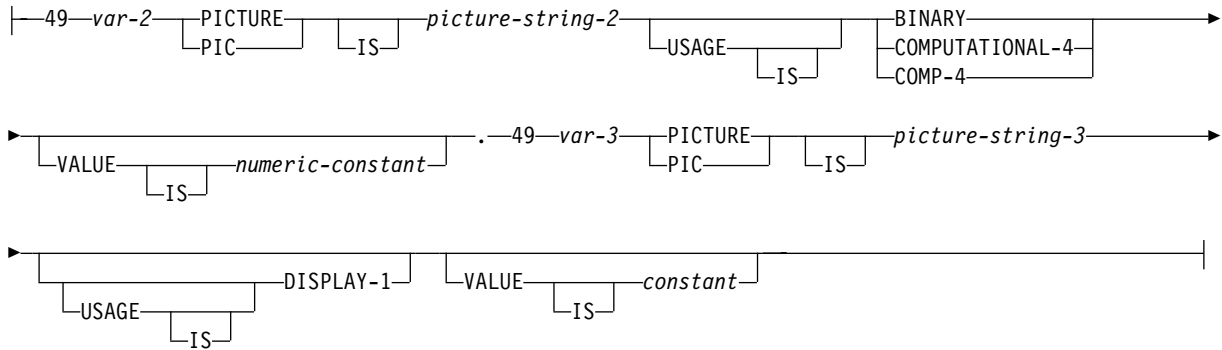
— DISPLAY— SIGN— IS— LEADING— SEPARATE— CHARACTER—

호스트 구조 배열(계속)

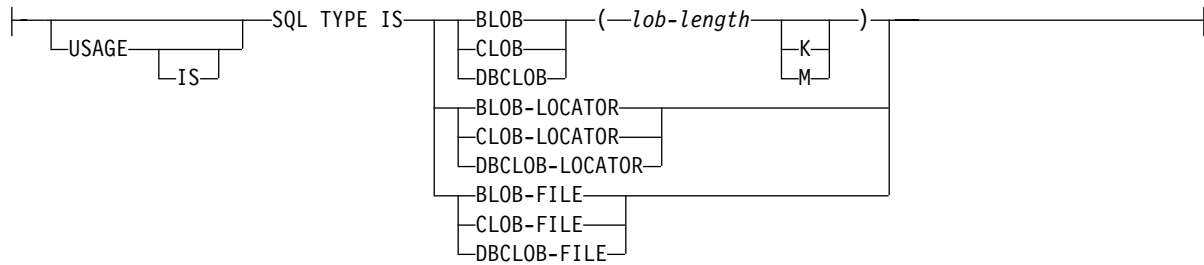
varchar-string:



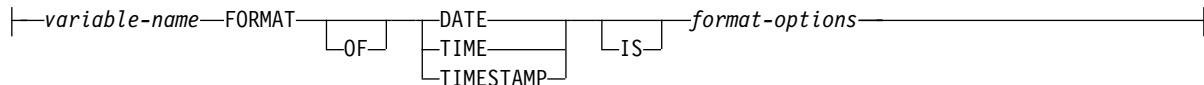
vargraphic-string:



lob:




datetime:



rowid:

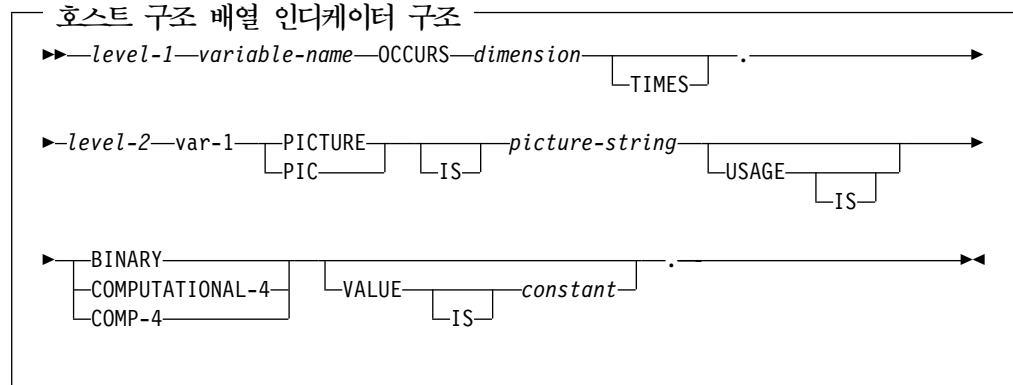


주:

1. level-1은 COBOL 레벨이 2-47 사이임을 나타냅니다.
2. level-2는 COBOL 레벨이 3-48 사이(level-2 > level-1)임을 나타냅니다.
3. 그래픽 호스트 변수, LOB 호스트 변수 및 부동 소수점 호스트 변수는 iSeries용 ILE COBOL에만 지원됩니다.
4. 숫자, 문자, 그래픽, LOB 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트, 그래픽 호스트 변수 및 LOB 호스트 변수 아래에 있는 주를 참조하십시오.
5. dimension은 1-32767 사이의 정수 상수이어야 합니다.
6. *format-options*는 COBOL 컴파일러에서 지원되는 유효한 datetime 옵션을 나타냅니다. 자세한 정보는 ILE COBOL Reference  를 참조하십시오.

SQL을 사용하는 COBOL 어플리케이션에서 호스트 배열 인디케이터 구조

이 그림은 호스트 구조 배열 인디케이터에 대한 유효한 구문을 표시합니다.



주:

1. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.
2. level-2는 COBOL 레벨이 3-48 사이(level-2 > level-1)임을 나타냅니다.
3. dimension은 1-32767 사이의 정수 상수이어야 합니다.
4. BINARY, COMPUTATIONAL 및 COMP-4는 동등합니다. COMPUTATIONAL-4와 COMP-4는 ISO/ANSI COBOL에서 지원되지 않는 IBM 확장자이므로 이식 가능한 어플리케이션은 BINARY로 작성되어야 합니다. 이 유형과 연관된 *picture-string*은 형식 S9(i)(또는 9가 i만큼의 S9...9, 9)를 가져야 합니다. i는 4 이하이어야 합니다.

SQL을 사용하는 COBOL 어플리케이션에서 외부 파일 설명 사용



SQL은 COPY DD-Format-Name, COPY DD-ALL-FORMATS, COPY DDS-Format-Name, COPY DDR-Format-Name, COPY DDR-ALL-FORMATS, COPY DDSR-Format-Name, COPY DDS-ALL-FORMATS 및 COPY DDSR-ALL-FORMATS를 사용하여 파일 정의로부터 호스트 변수를 검색합니다. REPLACING 옵션이 지정되면 완전명만 대체됩니다. var-1은 형식명과 필드명에 대해 비교됩니다. 형식명과 필드명이 같으면 var-2는 새로운 이름으로 사용됩니다.

주: COBOL 예약어인 필드명을 가진 파일 정의로부터는 호스트 변수를 검색할 수 없습니다. COPY DDx 형식 명령문을 COBOL 호스트 구조 내에 두어야 합니다.

iSeries용 DB2 UDB 프로그래밍 개념 정보의 iSeries용 DB2 UDB 샘플 표에 설명된 샘플 표 DEPARTMENT의 정의를 검색하기 위해 다음을 코딩할 수 있습니다.

```
01 DEPARTMENT-STRUCTURE.  
   COPY DDS-ALL-FORMATS OF DEPARTMENT.
```

DEPARTMENT-STRUCTURE로 명명된 호스트 구조는 DEPTNO, DEPTNAME, MGRNO 및 ADMRDEPT로 명명된 4개의 06 레벨 필드가 들어 있는 DEPART-RECORD로 명명된 05 레벨 필드로 정의됩니다. 이러한 필드명은 SQL문에서 호스트 변수로 사용될 수 있습니다. COBOL COPY 명령에 대한 자세한 정보는

COBOL/400® User's Guide  및 ILE COBOL Reference  를 참조하십시오.

외부 파일 설명에 대한 자세한 내용은 『SQL을 사용하는 COBOL 어플리케이션에서 외부 파일 설명 사용』을 참조하십시오.

SQL을 사용하는 COBOL 어플리케이션에서 외부 파일 설명 사용

COBOL은 외부 서술 자료를 포함할 때 특수 레벨을 작성하기 때문에 OCCURS절은 04 레벨 앞에 위치해야 합니다. 구조는 05 레벨에서 추가 선언 부분을 포함할 수 없습니다.

파일이 FILLER로 생성된 필드를 포함하면 구조는 호스트 구조 배열로 사용될 수 없습니다.

장치 파일의 경우 INDARA가 지정되지 않고 파일이 인디케이터를 포함하면 선언 부분은 호스트 구조 배열로 사용될 수 없습니다. 인디케이터 영역은 생성된 구조에 포함되며 레코드에 대한 기억영역이 인접할 수 없는 원인이 됩니다.

예를 들어, 다음은 COPY-DDS를 사용하여 호스트 구조 배열을 생성하고 10 행을 호스트 구조 배열에 폐치하는 방법을 보여줍니다.

```

01 DEPT.
   04 DEPT-ARRAY OCCURS 10 TIMES.
   COPY DDS-ALL-FORMATS OF DEPARTMENT.
   :

   EXEC SQL DECLARE C1 CURSOR FOR
     SELECT * FROM CORPDATA.DEPARTMENT
END EXEC.

EXEC SQL OPEN C1
   END-EXEC.

EXEC SQL FETCH C1 FOR 10 ROWS INTO :DEPARTMENT
   END-EXEC.

```

주: DATE, TIME 및 TIMESTAMP 열은 동일한 비교와 할당 규칙을 가진 SQL에 의해 DATE, TIME 또는 TIMESTAMP 열로 처리되는 호스트 변수 정의를 생성합니다. 예를 들어 날짜 호스트 변수는 DATE 열 또는 자료의 유효한 표시인 문자 스트링에 대해서만 비교될 수 있습니다.

GRAPHIC 및 VARGRAPHIC이 iSeries용 COBOL의 문자 변수에 맵핑되더라도, SQL은 이러한 GRAPHIC 및 VARGRAPHIC 변수를 고려합니다. GRAPHIC 또는 VARGRAPHIC 열에 UCS-2 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UCS-2 CCSID를 갖게 됩니다.

동등한 SQL 및 COBOL 자료 유형 판별

사전컴파일러는 다음 표에 있는 호스트 변수의 기본 SQLTYPE 및 SQLLEN을 판별합니다. 호스트 변수가 인디케이터 변수로 표시되면 SQLTYPE은 기본 SQLTYPE에 1을 더한 값이 됩니다.

표 3. 일반 SQL 자료 유형에 맵핑되는 COBOL 선언

COBOL 자료 유형	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
S9(i)V9(d) COMP-3 또는 S9(i)V9(d) COMP 또는 S9(i)V9(d) PACKED-DECIMAL	484	바이트 1에서 i+d, 바이트 2에서 d	DECIMAL(i+d,d)
S9(i)V9(d) DISPLAY SIGN LEADING SEPARATE	504	바이트 1에서 i+d, 바이트 2에서 d	정확히 같은 것은 없습니다. DECIMAL(i+d,d) 또는 NUMERIC(i+d,d)을 사용합니다.
S9(i)V9(d)DISPLAY	488	바이트 1에서 i+d, 바이트 2에서 d	NUMERIC (i+d,d)
S9(i) BINARY 또는 S9(i) COMP-4, i는 1 - 4 임.	500	2	SMALLINT
S9(i) BINARY 또는 S9(i) COMP-4, i는 5 - 9	496	4	INTEGER

표 3. 일반 SQL 자료 유형에 맵핑되는 COBOL 선언 (계속)

COBOL 자료 유형	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
S9(i) BINARY 또는 S9(i) COMP-4, i는 10 - 18 iSeries용 COBOL에 지원되지 않습니다.	492	8	BIGINT
S9(i)V9(d) BINARY 또는 S9(i)V9(d) COMP-4, i+d ≤ 4	500	바이트 1에서 i+d, 바이트 2에서 d	정확히 같은 것은 없습니다. DECIMAL(i+d,d) 또는 NUMERIC(i+d,d) 을 사용합니다.
S9(i)V9(d) BINARY 또는 S9(i)V9(d) COMP-4, 4 < i+d ≤ 9	496	바이트 1에서 i+d, 바이트 2에서 d	정확히 같은 것은 없습니다. DECIMAL(i+d,d) 또는 NUMERIC(i+d,d) 을 사용합니다.
COMP-1 iSeries용 COBOL에 지원되지 않습니다.	480	4	FLOAT(단정밀도)
COMP-2 iSeries용 COBOL에 지원되지 않습니다.	480	8	FLOAT(배정밀도)
고정 길이 문자 자료	452	m	CHAR(m)
가변 길이 문자 자료, m < 255	448	m	VARCHAR(m)
가변 길이 문자 자료, m > 254	456	m	VARCHAR(m)
고정 길이 그래픽 자료 iSeries용 COBOL에 지원되지 않습니다.	468	m	GRAPHIC(m)
가변 길이 그래픽 자료, m < 128 iSeries용 COBOL에 지원되지 않습니다.	464	m	VARGRAPHIC(m)
가변 길이 문자 자료, m > 127 iSeries용 COBOL에 지원되지 않습니다.	472	m	VARGRAPHIC(m)
DATE iSeries용 COBOL에 지원되지 않습니다.	384		DATE
TIME iSeries용 COBOL에 지원되지 않습니다.	388		TIME
TIMESTAMP iSeries용 COBOL에 지원되지 않습니다.	392	26	TIMESTAMP

다음 표는 제공된 SQL 자료 유형과 동일한 COBOL 자료 유형 판별에 사용될 수 있습니다.

표 4. 일반적인 COBOL 선언 부분에 맵핑된 SQL 자료 유형

SQL 자료 유형	COBOL 자료 유형	주
SMALLINT	S9(m) COMP-4	m은 1-4 사이입니다.
INTEGER	S9(m) COMP-4	m은 5-9 사이입니다.

표 4. 일반적인 COBOL 선언 부분에 맵핑된 SQL 자료 유형 (계속)

SQL 자료 유형	COBOL 자료 유형	주
BIGINT	iSeries용 ILE COBOL의 S9(m) COMP-4. iSeries용 COBOL에 지원되지 않습니다.	m은 10 - 18
DECIMAL(p,s)	p<19: S9(p-s)V9(s) PACKED-DECIMAL이나 S9(p-s)V9(s) COMP 또는 S9(p-s)V9(s) COMP-3, p>18인 경우는 지원되지 않습니다.	p는 정밀도이고 s는 스케일입니다. 0<=s<=p<=18. s=0인 경우 S9(p) 또는 S9(p)V를 사용하고 s=p인 경우는 SV9(s)를 사용합니다.
NUMERIC(p,s)	p<19: S9(p-s)V9(s) DISPLAY, p>18인 경우는 지원되지 않습니다.	p는 정밀도이고 s는 스케일입니다. 0<=s<=p<=18. s=0인 경우 S9(p) 또는 S9(p)V를 사용하고 s=p인 경우는 SV9(s)를 사용합니다.
FLOAT(단정밀도)	iSeries용 ILE COBOL의 COMP-1. iSeries용 COBOL에 지원되지 않습니다.	
FLOAT(배정밀도)	iSeries용 ILE COBOL의 COMP-2. iSeries용 COBOL에 지원되지 않습니다.	
CHAR(n)	고정 길이 문자 스트링	32766≥n≥1
VARCHAR(n)	가변 길이 문자 스트링	32740≥n≥1
BLOB	없음	BLOB를 선언하려면 SQL TYPE IS를 사용하십시오. iSeries용 ILE COBOL의 경우. iSeries용 COBOL에 지원되지 않습니다.
CLOB	없음	CLOB를 선언하려면 SQL TYPE IS를 사용하십시오. iSeries용 ILE COBOL의 경우. iSeries용 COBOL에 지원되지 않습니다.
GRAPHIC(n)	iSeries용 ILE COBOL의 고정 길이 그래픽 스트링. iSeries용 COBOL에 지원되지 않습니다.	16383≥n≥1
VARGRAPHIC(n)	iSeries용 ILE COBOL의 가변 길이 그래픽 스트링. iSeries용 COBOL에 지원되지 않습니다.	16370≥n≥1

표 4. 일반적인 COBOL 선언 부분에 맵핑된 SQL 자료 유형 (계속)

SQL 자료 유형	COBOL 자료 유형	주
DBCLOB	없음	DBCLOB를 선언하려면 SQL TYPE IS를 사용하십시오. iSeries용 ILE COBOL의 경우. iSeries용 COBOL에 지원되지 않습니다.
DATE	고정 길이 문자 스트링 또는 DATE(iSeries용 ILE COBOL의 경우)	형식이 *USA, *JIS, *EUR 또는 *ISO이면 최소한 10자를 허용합니다. 형식이 *YMD, *DMY 또는 *MDY이면 최소한 8자를 허용합니다. 형식이 *JUL이면 최소한 6자를 허용합니다.
TIME	고정 길이 문자 스트링 또는 TIME(iSeries용 ILE COBOL의 경우)	최소한 6자를 허용합니다. 초를 포함할 경우에는 8자를 허용합니다.
TIMESTAMP	고정 길이 문자 스트링 또는 TIMESTAMP(iSeries용 ILE COBOL의 경우)	n은 최소한 19이어야 합니다. 전체 정밀도에 마이크로초를 포함시키려면 n은 26이어야 합니다. n이 26보다 작으면 마이크로초 부분에서 잘립니다.
DATALINK	지원되지 않습니다.	
ROWID	없음	ROWID를 선언하려면 SQL TYPE IS를 사용하십시오.

자세한 내용은 『COBOL 변수 선언 및 사용에 관한 주』를 참조하십시오.

COBOL 변수 선언 및 사용에 관한 주

레벨 77인 모든 자료 서술 항목 뒤에 하나 이상의 REDEFINES 항목이 있을 수 있습니다. 그러나 이러한 항목의 이름은 SQL문에서 사용할 수 없습니다.

FILLER 항목 아래에 정의된 레벨이 구조 안에 포함되어 있으면 예기치 못한 결과가 발생할 수도 있습니다.

SMALLINT, BIGINT 및 INTEGER 자료 유형에 대한 COBOL 선언 부분은 소수 자릿수로 표현됩니다. 데이터베이스 관리자는 정수 전체 크기를 사용하며 COBOL 선언 부분 내의 지정된 자릿수에 허용된 값보다 호스트 변수 내에 더 큰 값을 위치시킬 수 있습니다. 그러나 이는 COBOL문 수행시 자료 절단이나 크기 오류를 발생시킬 수 있습니다. 어플리케이션 내의 숫자 크기가 선언된 자릿수 내에 있는지 확인하십시오.

SQL을 사용하는 COBOL 어플리케이션에서 인디케이터 변수 사용

인디케이터 변수는 2바이트 정수(PIC S9(m) USAGE BINARY, 여기서 m은 1-4임)입니다. 또한 인디케이터 구조(반단어 정수 변수의 배열로 정의됨)를 지정하여 호스트 구조를 지원할 수도 있습니다. 검색시 연관 호스트 변수에 널(null)이 할당되었는지 여부를 나타내기 위해 인디케이터 변수를 사용합니다. 음의 인디케이터 변수는 열에 할당할 때 널이 할당되어야 함을 나타내기 위해 사용됩니다.

자세한 정보는 SQL 참조서의 인디케이터 변수 주제를 참조하십시오.

인디케이터 변수는 호스트 변수와 동일한 방식으로 선언되며 필요에 따라 두 가지 선언 부분을 여러 가지 방법으로 혼합할 수 있습니다.

예:

다음 명령문을 보십시오.

```
EXEC SQL FETCH CLS_CURSOR INTO :CLS-CD,  
                                :NUMDAY :NUMDAY-IND,  
                                :BGN :BGN-IND,  
                                :ENDCLS :ENDCLS-IND  
END-EXEC.
```

이 경우 변수는 다음과 같이 선언될 수 있습니다.

```
EXEC SQL BEGIN DECLARE SECTION END-EXEC.  
77 CLS-CD      PIC X(7).  
77 NUMDAY     PIC S9(4) BINARY.  
77 BGN        PIC X(8).  
77 ENDCLS     PIC X(8).  
77 NUMDAY-IND PIC S9(4) BINARY.  
77 BGN-IND    PIC S9(4) BINARY.  
77 ENDCLS-IND PIC S9(4) BINARY.  
EXEC SQL END DECLARE SECTION END-EXEC.
```

제 4 장 PL/I 어플리케이션에서의 SQL문 코딩

이 장에서는 iSeries PL/I 프로그램에서 SQL문을 삽입할 때의 고유 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.

자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 PL/I 어플리케이션에서 SQL 통신 영역 정의』
- 80 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 SQL 설명자 영역 정의』
- 81 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 SQL문 삽입』
- 83 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 호스트 변수 사용』
- 88 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 사용』
- 91 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 배열 사용』
- 93 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 외부 파일 설명 사용』
- 94 페이지의 『동일한 SQL 및 PL/I 자료 유형 판별』
- 96 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 인디케이터 변수 사용』
- 97 페이지의 『구조 매개변수 전달 기법에 따른 PL/I의 차이점』

SQL문의 사용 방법을 보여주는 PL/I 프로그램 샘플에 대해 자세한 사항은 부록 A 『iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램』을 참조하십시오.

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

SQL을 사용하는 PL/I 어플리케이션에서 SQL 통신 영역 정의

SQL문이 들어 있는 PL/I 프로그램은 반드시 다음 중 하나 또는 모두를 포함하고 있어야 합니다.

- FIXED BINARY(31)로 선언된 SQLCODE 변수
- CHAR(5)로 선언된 SQLSTATE 변수

또는

- SQLCA(SQLCODE 및 SQLSTATE 변수가 들어 있는)

SQLCODE 값과 SQLSTATE 값은 각각의 SQL문이 실행된 후에 데이터베이스 관리자에 의해 설정됩니다. 어플리케이션은 SQLCODE 값을 체크하여 최종 SQL문이 제대로 수행되었는지를 판별합니다.

SQLCA는 직접 또는 SQL INCLUDE문을 사용하여 PL/I 프로그램에서 작성될 수 있습니다. SQL INCLUDE문을 사용할 때에는 다음의 표준 SQLCA 선언 부분을 포함시켜야 합니다.

```
EXEC SQL INCLUDE SQLCA ;
```

SQLCODE, SQLSTATE 및 SQLCA 변수의 범위에는 프로그램 내의 모든 SQL문이 포함되어야 합니다.

SQLCA에 대해 포함된 PL/I 소스문은 다음과 같습니다.

```
DCL 1 SQLCA,  
  2 SQLCAID      CHAR(8),  
  2 SQLCABC      FIXED(31) BINARY,  
  2 SQLCODE      FIXED(31) BINARY,  
  2 SQLERRM      CHAR(70) VAR,  
  2 SQLERRP      CHAR(8),  
  2 SQLERRD(6)   FIXED(31) BINARY,  
  2 SQLWARN,  
    3 SQLWARN0   CHAR(1),  
    3 SQLWARN1   CHAR(1),  
    3 SQLWARN2   CHAR(1),  
    3 SQLWARN3   CHAR(1),  
    3 SQLWARN4   CHAR(1),  
    3 SQLWARN5   CHAR(1),  
    3 SQLWARN6   CHAR(1),  
    3 SQLWARN7   CHAR(1),  
    3 SQLWARN8   CHAR(1),  
    3 SQLWARN9   CHAR(1),  
    3 SQLWARNA   CHAR(1),  
  2 SQLSTATE     CHAR(5);
```

프로그램에 SQLCODE에 대한 선언이 있고 사전컴파일러에 의해 SQLCA가 제공될 때 SQLCODE는 SQLCADE로 대체됩니다. 프로그램에 SQLCODE에 대한 선언이 있고 사전컴파일러에 의해 SQLCA가 제공될 때 SQLCODE는 QLCQTOT로 대체됩니다.

SQLCA에 대한 자세한 정보는 SQL 참조서의 SQL 통신 영역을 참조하십시오.

SQL을 사용하는 PL/I 어플리케이션에서 SQL 설명자 영역 정의

다음 명령문은 SQLDA를 필요로 합니다.

```
EXECUTE...USING DESCRIPTOR descriptor-name  
FETCH...USING DESCRIPTOR descriptor-name  
OPEN...USING DESCRIPTOR descriptor-name  
CALL...USING DESCRIPTOR descriptor-name  
DESCRIBE statement-name INTO descriptor-name  
DESCRIBE TABLE host-variable INTO descriptor-name  
PREPARE statement-name INTO descriptor-name
```

SQLCA와는 달리 프로그램에 두 개 이상의 SQLDA가 있을 수 있으며 SQLDA는 유효한 모든 이름을 가질 수 있습니다. SQLDA는 PL/I 프로그램에 직접 코딩되거나 또는 SQL INCLUDE문을 사용하여 코딩될 수 있습니다. SQL INCLUDE문 사용시 표준 SQLDA 선언 부분을 포함시켜야 합니다.

```
EXEC SQL INCLUDE SQLDA;
```

SQLDA에 대해 포함된 PL/I 소스문은 다음과 같습니다.

```
DCL 1 SQLDA BASED(SQLDAPTR),
    2 SQLDAID      CHAR(8),
    2 SQLDABC      FIXED(31) BINARY,
    2 SQLN         FIXED(15) BINARY,
    2 SQLD         FIXED(15) BINARY,
    2 SQLVAR(99),
    3 SQLTYPE      FIXED(15) BINARY,
    3 SQLLEN       FIXED(15) BINARY,
    3 SQLRES       CHAR(12),
    3 SQLDATA      PTR,
    3 SQLIND       PTR,
    3 SQLNAME      CHAR(30) VAR;
DCL SQLDAPTR PTR;
```

동적 SQL은 iSeries용 DB2 UDB 프로그램 개념 정보의 동적 SQL 어플리케이션에 설명되어 있는 첨단 프로그래밍 기술입니다. 동적 SQL을 사용하면 프로그램 수행 중 사용자 프로그램이 SQL문을 개발 및 수행할 수 있습니다. 동적으로 수행하는 가변 SELECT 리스트(조회된 한 부분으로 리턴될 자료의 리스트임)가 있는 SELECT문은 SQL 설명자 영역(SQLDA)을 필요로 합니다. 그 이유는 사용자가 SELECT의 결과를 수신하기 위해 지정된 변수의 수와 유형을 미리 알 수 없기 때문입니다.

SQLDA에 대한 자세한 정보는 SQL 참조서의 SQL 설명자 영역을 참조하십시오.

SQL을 사용하는 PL/I 어플리케이션에서 SQL문 삽입

PL/I 프로그램의 첫 번째 명령문은 PROCEDURE문이어야 합니다.

SQL문은 실행 가능한 명령문이 표시되는 모든 PL/I 프로그램에서 작성할 수 있습니다.

PL/I 프로그램의 각 SQL문은 EXEC SQL로 시작하고 세미콜론(;)으로 끝나야 합니다. 키워드 EXEC SQL은 모두 한 행에 표시되어야 하나 명령문의 나머지 부분은 다음 행 및 후속 행에 표시될 수 있습니다.

자세한 내용은 다음 절을 참조하십시오.

- 82 페이지의 『예: SQL을 사용하는 PL/I 어플리케이션에서 SQL문 삽입』
- 82 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 주석』
- 82 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 SQL문의 연속』

- 『SQL을 사용하는 PL/I 어플리케이션에서 포함 코드』
- 『SQL을 사용하는 PL/I 어플리케이션에서 여백』
- 83 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 이름』
- 83 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 명령문 레이블』
- 83 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 WHENEVER문』

예: SQL을 사용하는 PL/I 어플리케이션에서 SQL문 삽입

PL/I 프로그램에서 작성된 UPDATE문은 다음과 같습니다.

```
EXEC SQL UPDATE DEPARTMENT
        SET MGRNO = :MGR_NUM
        WHERE DEPTNO = :INT_DEPT ;
```

SQL을 사용하는 PL/I 어플리케이션에서 주석

SQL 주석(--) 이외에도 키워드 EXEC와 SQL 사이를 제외하고 공백이 허용되는 모든 삽입 SQL문 사이에는 PL/I 주석(/*...*/)을 포함시킬 수 있습니다.

SQL을 사용하는 PL/I 어플리케이션에서 SQL문의 연속

SQL문의 행 연속 규칙은 EXEC SQL을 한 행 내에 지정해야 하는 것을 제외하고는 다른 PL/I문의 규칙과 동일합니다.

DBCS 자료가 들어 있는 상수는 여백 밖에 SI 및 SO 문자를 위치시켜 복수 행에 걸쳐 계속될 수 있습니다. 다음 예에서는 여백을 2-72로 가정하겠습니다. 이 예에서 SQL 문에 G'<AABBCCDDEEFFGGHHIIJJKK>'의 유효한 그래픽 상수가 있습니다.

```
*(...+....1....+....2....+....3....+....4....+....5....+....6....+....7.)...
EXEC SQL SELECT * FROM GRAPHTAB WHERE GRAPHCOL = G'<AABBCCDD>
<EEFFGGHHIIJJKK>';
```

SQL을 사용하는 PL/I 어플리케이션에서 포함 코드

SQL문 또는 PL/I 호스트 변수 선언문은 명령문이 삽입될 소스 코드 내의 한 지점에 다음 SQL문을 삽입하여 포함시킬 수 있습니다.

```
EXEC SQL INCLUDE member-name ;
```

SQL문에서는 PL/I 사전처리 지시문이 허용되지 않습니다. PL/I %INCLUDE문을 사용하여 SQL문이나 SQL문에서 참조될 PL/I 호스트 변수 선언 부분을 포함시킬 수 없습니다.

SQL을 사용하는 PL/I 어플리케이션에서 여백

CRTSQLPLI 명령의 MARGINS 매개변수에 의해 지정된 여백 내에 SQL문을 작성하십시오. EXEC SQL이 지정된 여백에서 시작하지 않으면 SQL 컴파일러는 SQL문을 인식하지 못합니다. CRTSQLPLI 명령은 부록 B 『호스트 언어 사전컴파일러에 대한 iSeries용 DB2 UDB CL 명령 설명』을 참조하십시오.

SQL을 사용하는 PL/I 어플리케이션에서 이름

유효한 모든 PL/I 변수명은 호스트 변수에 사용될 수 있으며 다음 제한사항에 따라야 합니다.

'SQL', 'RDI' 또는 'DSN'으로 시작하는 호스트 변수명이나 외부 항목명을 사용하지 마십시오. 이러한 이름들은 데이터베이스 관리자 예약어입니다.

SQL을 사용하는 PL/I 어플리케이션에서 명령문 레이블

PL/I문과 같이 실행 가능한 모든 SQL문은 레이블 접두부를 가질 수 있습니다.

SQL을 사용하는 PL/I 어플리케이션에서 WHENEVER문

SQL WHENEVER문 내의 GOTO절의 목표는 PL/I 소스 코드 내의 레이블이어야 하며 WHENEVER문에 의해 영향을 받는 모든 SQL문의 범위 내에 있어야 합니다.

SQL을 사용하는 PL/I 어플리케이션에서 호스트 변수 사용

SQL문에서 사용된 모든 호스트 변수는 명시적으로 선언되어야 합니다.

호스트 변수 정의에 사용된 PL/I문 앞에는 BEGIN DECLARE SECTION문이 오고 뒤에는 END DECLARE SECTION문이 와야 합니다. BEGIN DECLARE SECTION 및 END DECLARE SECTION이 지정되면 SQL문에서 사용된 모든 호스트 변수 선언은 BEGIN DECLARE SECTION과 END DECLARE SECTION문 사이에 있어야 합니다.

SQL문 내의 모든 호스트 변수 앞에는 콜론(:)이 와야 합니다.

호스트 변수명은 호스트 변수가 다른 블록이나 프로시저에 존재 하더라도 프로그램 내에서는 고유해야 합니다.

호스트 변수를 사용하는 SQL문은 변수가 선언된 명령문의 범위 내에 있어야 합니다.

호스트 변수는 스칼라 변수이어야 하며 배열의 요소가 될 수 없습니다.

자세한 내용은 『SQL을 사용하는 PL/I 어플리케이션에서 호스트 변수 선언』를 참조하십시오.

SQL을 사용하는 PL/I 어플리케이션에서 호스트 변수 선언

PL/I 사전컴파일러는 유효한 호스트 변수 선언 부분으로서 유효한 PL/I 선언 부분의 서브세트만 인식합니다.

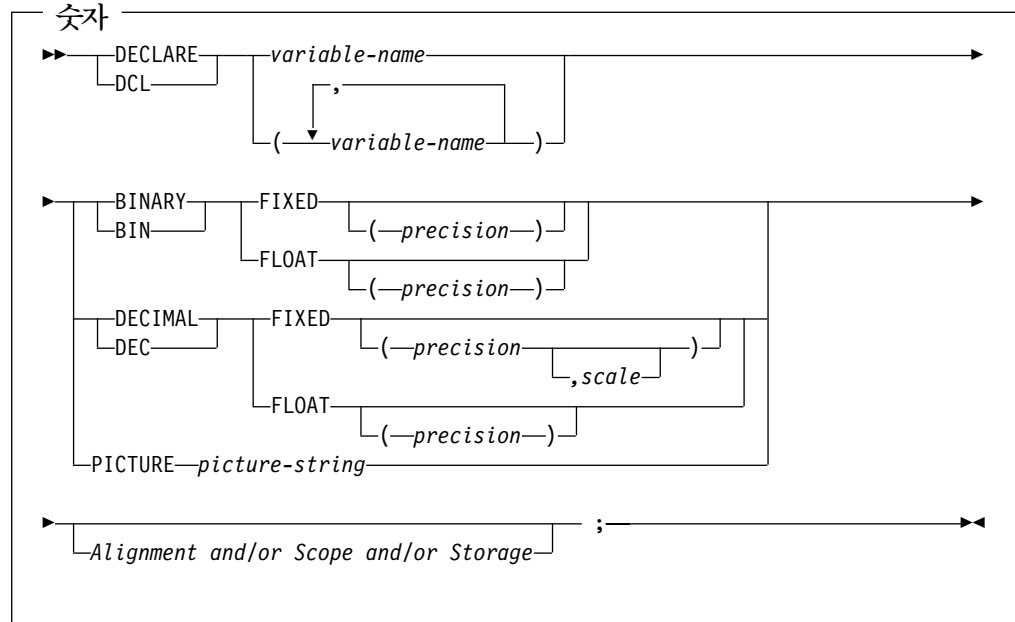
변수의 이름과 자료 속성만이 사전컴파일러에 의해 사용됩니다. 정렬, 범위 및 기억영역 속성은 무시됩니다. 정렬, 범위 및 기억영역이 무시되더라도 이들을 사용할 때는 몇 가

지 제한사항이 있으므로 무시될 경우 사전컴파일러에 의해 생성된 PL/I 소스 코드 컴파일시 문제점이 발생할 수 있습니다. 제한사항은 다음과 같습니다.

- EXTERNAL 범위 속성과 STATIC 기억영역 속성의 선언은 INITIAL 기억영역 속성도 가져야 합니다.
- BASED 기억영역 속성이 작성되면 뒤에 PL/I 요소 지정자 표현식이 와야 합니다.

SQL을 사용하는 PL/I 어플리케이션에서 숫자 호스트 변수

다음 그림은 유효한 스칼라 숫자 호스트 변수 선언 부분에 대한 구문을 보여줍니다.

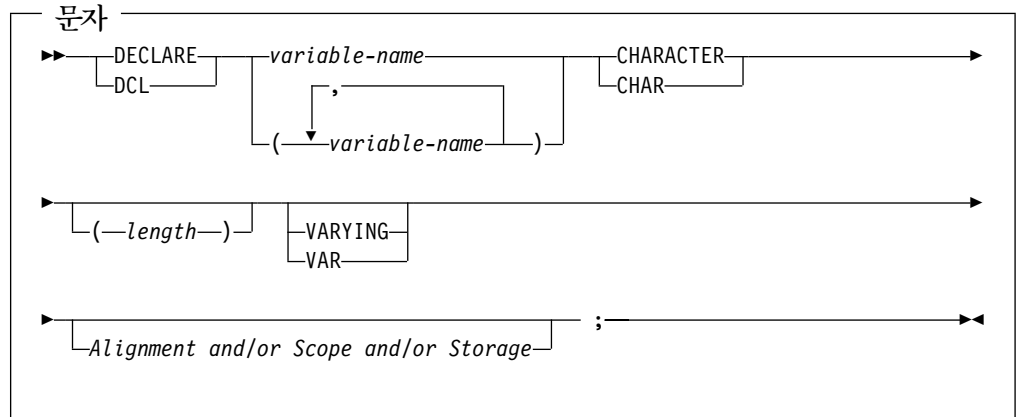


주:

1. (BINARY, BIN, DECIMAL 또는 DEC)와 (FIXED 또는 FLOAT) 및 (정밀도, 소수 자릿수)를 임의 순서로 지정할 수 있습니다.
2. 형식 '9...9V9...R'의 picture-string은 그것이 숫자 호스트 변수임을 나타냅니다. R 이 필요합니다. 옵션 V는 내포된 소수점임을 나타냅니다.
3. 형식 'S9...9V9...9'의 picture-string은 부호가 앞에 있는 개별 분리된 호스트 변수임을 나타냅니다. S가 필요합니다. 옵션 V는 내포된 소수점임을 나타냅니다.

SQL을 사용하는 PL/I 어플리케이션에서 문자 호스트 변수

다음 그림은 유효한 스칼라 문자 호스트 변수에 대한 구문을 표시합니다.



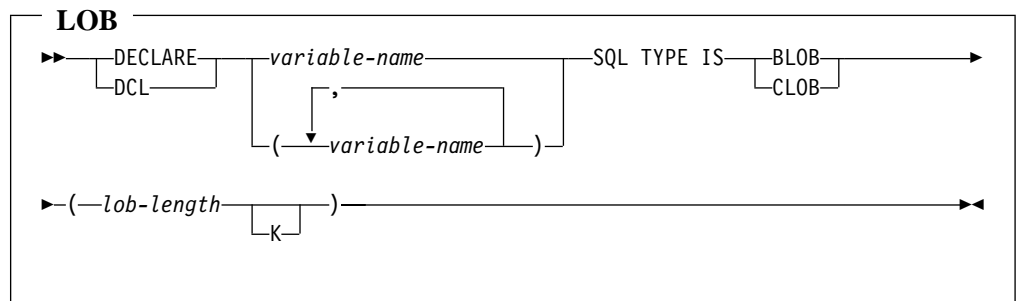
주:

1. *length*는 VARYING 또는 VAR이 지정되지 않은 경우 32766 이하의 정수 상수 이어야 합니다.
2. VARYING 또는 VAR이 지정되면 *length*는 32740 이하의 상수이어야 합니다.

SQL을 사용하는 PL/I 어플리케이션의 LOB 호스트 변수

PL/I에는 LOB(대형 오브젝트)에 대한 SQL 자료 유형에 대응하는 변수가 없습니다. 이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQL TYPE IS 절을 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버에서 이 선언을 PL/I 언어 구조로 대체합니다.

다음 그림은 유효한 LOB 호스트 변수에 대한 구문을 나타냅니다.



주:

1. BLOB 및 CLOB의 경우 $1 \leq \text{lob-length} \leq 32,766$
2. SQL TYPE IS, BLOB, CLOB는 대소문자를 혼용하여 사용할 수 있습니다.

BLOB 예:

다음 선언을 사용할 경우:

```
DCL MY_BLOB SQL TYPE IS BLOB(16384);
```

다음과 같은 구조가 생성됩니다.

```
DCL 1 MY_BLOB,  
    3 MY_BLOB_LENGTH BINARY FIXED (31) UNALIGNED,  
    3 MY_BLOB_DATA CHARACTER (16384);
```

CLOB 예:

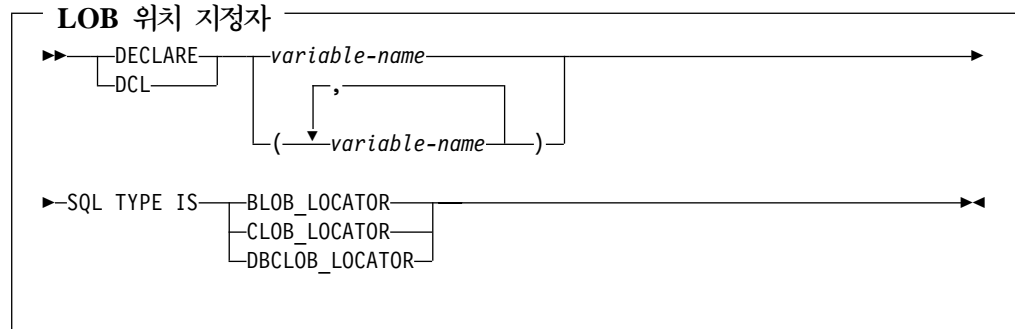
다음 선언을 사용할 경우:

```
DCL MY_CLOB SQL TYPE IS CLOB(16384);
```

다음과 같은 구조가 생성됩니다.

```
DCL 1 MY_CLOB,  
    3 MY_CLOB_LENGTH BINARY FIXED (31) UNALIGNED,  
    3 MY_CLOB_DATA CHARACTER (16384);
```

다음 그림은 유효한 LOB 위치 지정자에 대한 구문을 보여줍니다.



주: SQL TYPE IS, BLOB_LOCATOR, CLOB_LOCATOR, DBCLOB_LOCATOR
는 대소문자를 혼합하여 사용할 수 있습니다.

CLOB 위치 지정자 예:

다음 선언을 사용할 경우:

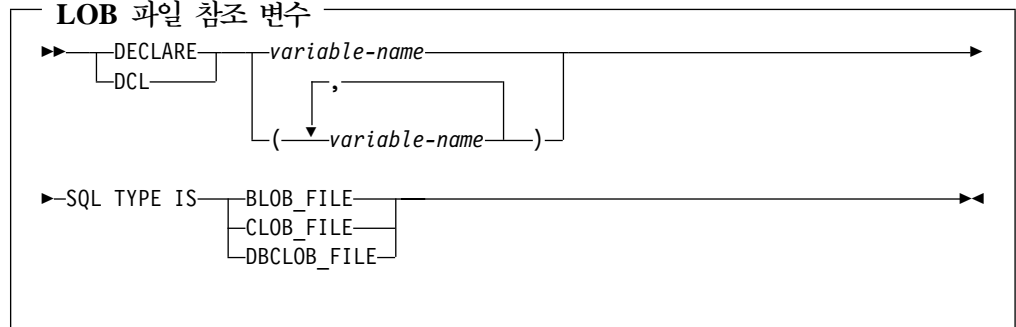
```
DCL MY_LOCATOR SQL TYPE IS CLOB_LOCATOR;
```

다음과 같이 생성됩니다.

```
DCL MY_LOCATOR BINARY FIXED(31) UNALIGNED;
```

BLOB 및 DBCLOB 위치 지정자의 구문은 비슷합니다.

다음 그림은 유효한 LOB 파일 참조 변수에 대한 구문을 보여줍니다.



주: SQL TYPE IS, BLOB_FILE, CLOB_FILE 및 DBCLOB_FILE은 대소문자를 혼합하여 사용할 수 있습니다.

CLOB 파일 참조 예:

다음 선언을 사용할 경우:

```
DCL MY_FILE SQL TYPE IS CLOB_FILE;
```

다음과 같은 구조가 생성됩니다.

```
DCL 1 MY_FILE,
      3 MY_FILE_NAME_LENGTH BINARY FIXED(31) UNALIGNED,
      3 MY_FILE_DATA_LENGTH BINARY FIXED(31) UNALIGNED,
      3 MY_FILE_FILE_OPTIONS BINARY FIXED(31) UNALIGNED,
      3 MY_FILE_NAME_CHAR(255);
```

BLOB 및 DBCLOB 위치 지정자의 구문은 비슷합니다.

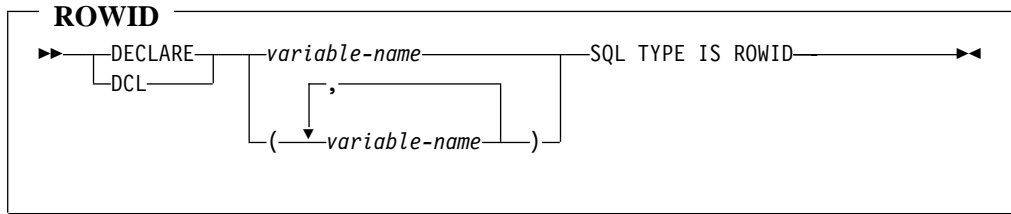
사전컴파일러는 다음의 파일 옵션 상수에 대한 선언을 생성합니다.

- SQL_FILE_READ(2)
- SQL_FILE_CREATE(8)
- SQL_FILE_OVERWRITE(16)
- SQL_FILE_APPEND(32)

이러한 값에 대한 자세한 정보는 SQL 프로그래밍 개념의 LOB 파일 참조 변수를 참조하십시오.

SQL을 사용하는 PL/I 어플리케이션의 ROWID 호스트 변수

PL/I에는 ROWID의 SQL 자료 유형에 대응하는 변수가 없습니다. 이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQL TYPE IS 절을 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버에서 이 선언을 PL/I 언어 구조로 대체합니다.



주: SQL TYPE IS ROWID는 대소문자를 혼용하여 사용할 수 있습니다.

ROWID 예

다음 선언을 사용할 경우:

```
DCL MY_ROWID SQL TYPE IS ROWID;
```

다음과 같이 생성됩니다.

```
DCL MY_ROWID CHARACTER(40) VARYING;
```

SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 사용

PL/I 프로그램에는 명명된 PL/I 변수 세트인 호스트 구조를 정의할 수 있습니다. 호스트 구조명은 종속 레벨이 기본 PL/I 변수를 명명하는 그룹명이 될 수 있습니다. 예를 들면 다음과 같습니다.

```
DCL 1 A,
    2 B,
    3 C1 CHAR(...),
    3 C2 CHAR(...);
```

이 예에서 B는 기본 항목 C1 및 C2로 구성된 호스트 구조명입니다.

스칼라 리스트에 대한 단축 표기법으로 구조명을 사용할 수 있습니다. 호스트 변수는 구조명으로(예를 들어 STRUCTURE.FIELD) 규정할 수 있습니다. 호스트 구조는 두 레벨로 제한됩니다(예를 들어 위의 호스트 구조 예에서 A는 SQL에서 참조될 수 없습니다). 구조는 중간 레벨 구조를 포함할 수 없습니다. 이전 예에서 A는 호스트 변수로 사용할 수 없으며 SQL문에서도 참조할 수 없습니다. 그러나 B는 첫 번째 레벨 구조로 SQL문에서 참조될 수 있습니다. SQL 자료에 대한 호스트 구조는 두 레벨로 되어 있고 호스트 변수의 명명된 세트로 생각할 수 있습니다. 호스트 구조가 정의된 후에는 여러 개의 호스트 변수(즉, 호스트 구조를 구성하는 호스트 변수명)를 리스트하는 대신 SQL문에서 호스트 구조를 참조할 수 있습니다.

예를 들어 다음과 같이 표 CORPDATA.EMPLOYEE의 선택된 행으로부터 모든 열 값을 검색할 수 있습니다.

```

DCL 1 PEMPL,
      5 EMPNO    CHAR(6),
      5 FIRSTNME CHAR(12) VAR,
      5 MIDINIT  CHAR(1),
      5 LASTNAME CHAR(15) VAR,
      5 WORKDEPT CHAR(3);

...
EMPID = '000220';
...
EXEC SQL
SELECT *
INTO :PEMPL
FROM CORPDATA.EMPLOYEE
WHERE EMPNO = :EMPID;

```

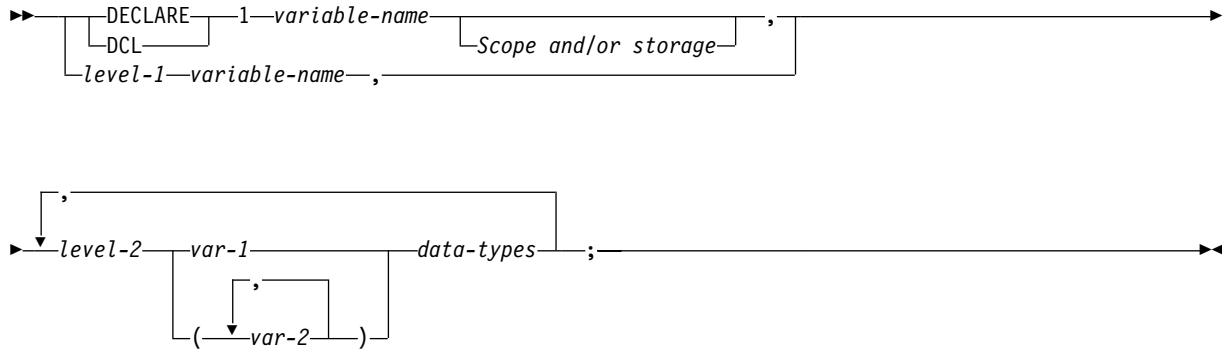
자세한 정보는 다음 절을 참조하십시오.

- 『SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조』
- 90 페이지의 『SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 인디케이터 배열』

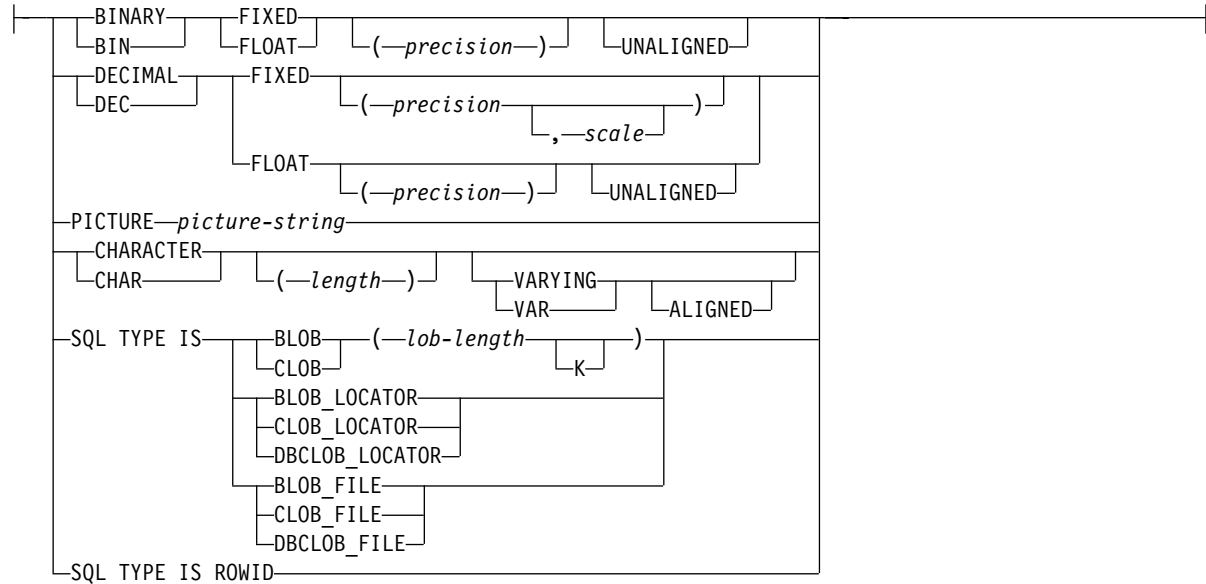
SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조

다음 그림은 유효한 호스트 구조 선언에 대한 구문을 보여줍니다.

호스트 구조



data-types:

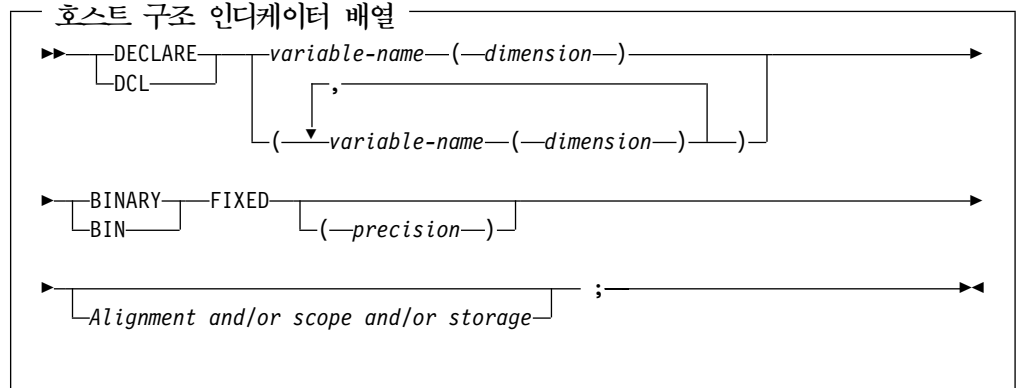


주:

1. level-1은 중간 레벨 구조가 있음을 나타냅니다.
2. level-1은 1-254 사이의 정수 상수이어야 합니다.
3. level-2는 2-255 사이의 정수 상수이어야 합니다.
4. 숫자, 문자, LOB 및 ROWID 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트, 그래픽 호스트 변수, LOB 호스트 변수 및 ROWID 호스트 변수 아래에 있는 주를 참조하십시오.

SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 인디케이터 배열

다음 그림은 유효한 인디케이터 배열에 대한 구문을 보여줍니다.



주: dimension은 1과 32766 사이의 정수 상수이어야 합니다.

SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 배열 사용

PL/I 프로그램에서 호스트 구조 배열을 정의할 수 있습니다. 이 예에서는 다음 사항이 참입니다.

- B_ARRAY는 C1_VAR 및 C2_VAR 항목을 포함하는 호스트 구조 배열명입니다.
- B_ARRAY는 규정될 수 없습니다.
- B_ARRAY는 FETCH 및 INSERT문의 블록 양식에만 사용될 수 있습니다.
- B_ARRAY의 모든 항목은 유효한 호스트 변수이어야 합니다.
- C1_VAR 및 C2_VAR는 SQL문에서 유효한 호스트 변수가 아닙니다. 구조는 중간 레벨 구조를 포함할 수 없습니다. A_STRUCT는 차원 속성을 포함할 수 없습니다.

```

DCL 1 A_STRUCT,
    2 B_ARRAY(10),
    3 C1_VAR CHAR(20),
    3 C2_FIXED BIN(15) UNALIGNED;
  
```

CORPDATA.DEPARTMENT 표에서 10개 행을 검색할 경우 다음과 같이 하십시오.

```

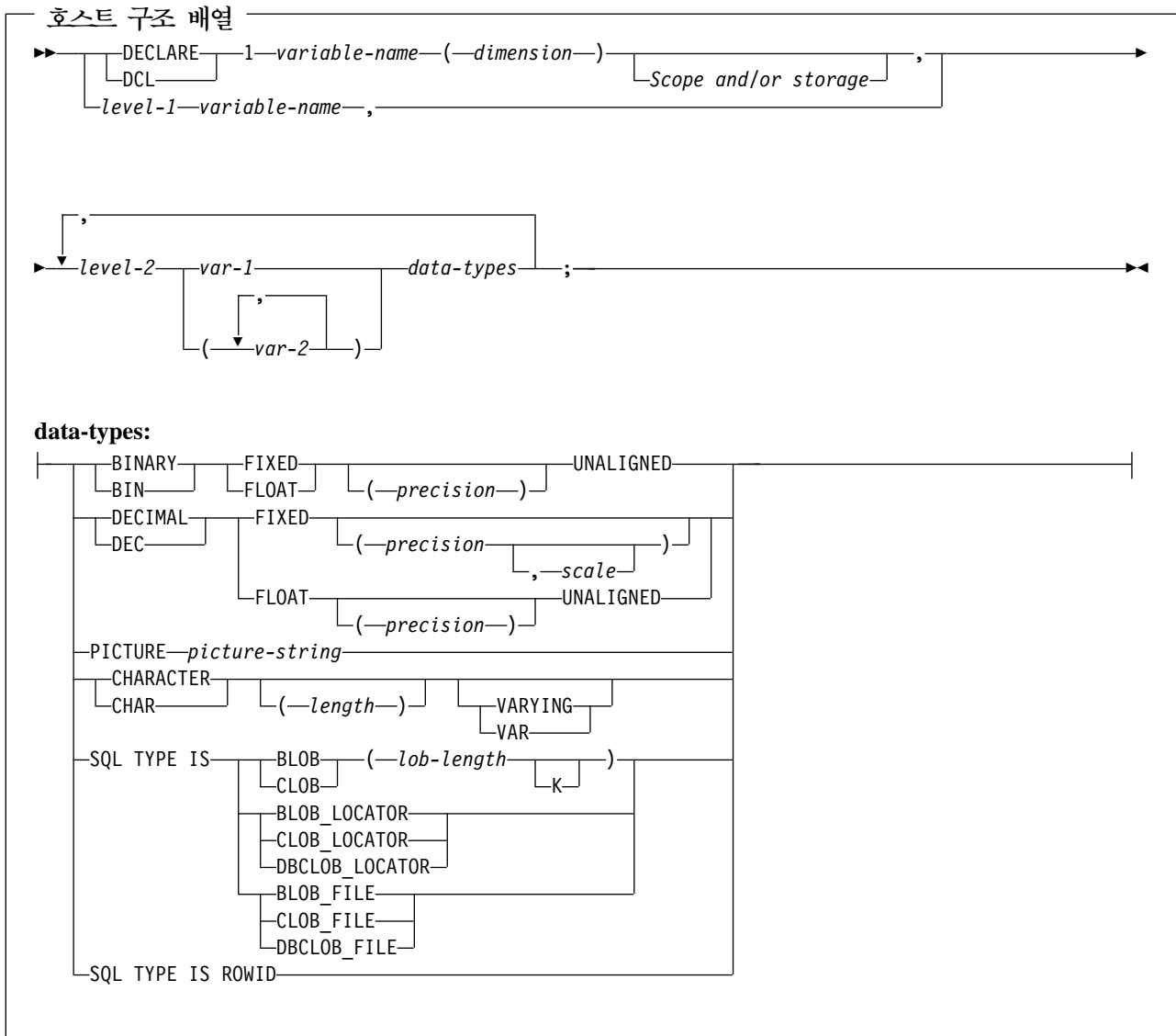
DCL 1 DEPT(10),
    5 DEPTPNO CHAR(3),
    5 DEPTNAME CHAR(29) VAR,
    5 MGRNO CHAR(6),
    5 ADMRDEPT CHAR(3);
DCL 1 IND_ARRAY(10),
    5 INDS(4) FIXED BIN(15);
EXEC SQL
    DECLARE C1 CURSOR FOR
    SELECT *
    FROM CORPDATA.DEPARTMENT;

EXEC SQL
    FETCH C1 FOR 10 ROWS INTO :DEPT :IND_ARRAY;
  
```

자세한 내용은 『SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 배열』을 참조하십시오.

SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 배열

다음의 구문 다이어그램은 유효한 구조 배열 선언에 대한 구문을 보여줍니다.

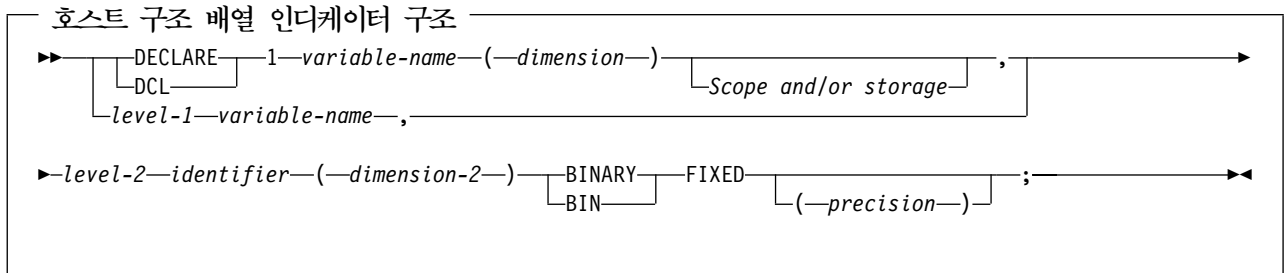


주:

1. level-1은 중간 레벨 구조가 있음을 나타냅니다.
2. level-1은 1-254 사이의 정수 상수이어야 합니다.
3. level-2는 2-255 사이의 정수 상수이어야 합니다.

4. 숫자, 문자, LOB 및 ROWID 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트, 그래픽 호스트 변수, LOB 호스트 변수 및 ROWID 호스트 변수 아래에 있는 주를 참조하십시오.
5. dimension은 1-32767 사이의 정수 상수이어야 합니다.

SQL을 사용하는 PL/I 어플리케이션에서 호스트 구조 배열 인디케이터
 다음 그림은 유효한 호스트 구조 배열에 대한 구문 다이어그램을 보여줍니다.



주:

1. level-1은 중간 레벨 구조가 있음을 나타냅니다.
2. level-1은 1-254 사이의 정수 상수이어야 합니다.
3. level-2는 2-255 사이의 정수 상수이어야 합니다.
4. dimension-1 및 dimension-2는 1과 32767 사이의 정수 상수이어야 합니다.

SQL을 사용하는 PL/I 어플리케이션에서 외부 파일 설명 사용

PL/I %INCLUDE 지시문을 사용하여 소스 프로그램에 외부 서술 파일의 정의를 포함시킬 수 있습니다. SQL과 함께 사용할 때는 특정 형식의 %INCLUDE 지시문이 SQL 컴파일러에 의해 인식됩니다. 그 지시문 형식은 다음의 세 가지 요소나 매개변수 값을 가져야 하며 그렇지 않으면 사전컴파일러가 그 지시문을 무시합니다. 필수 요소는 파일명, 형식 이름 그리고 요소 유형입니다. SQL 사전컴파일러에 의해 지원되는 두 가지 선택 가능한 요소로는 접두부명과 COMMA가 있습니다.

구조는 레코드나 키 구조의 최종 자료 요소에 의해 정상적으로 종료됩니다. 그러나 %INCLUDE 지시문에 COMMA 요소가 지정되어 있으면 구조가 종료되지 않습니다.

iSeries용 DB2 UDB 프로그래밍 개념 정보의 iSeries용 DB2 UDB 샘플 표에 설명된 샘플 표 DEPARTMENT의 정의를 포함하려면 다음과 같이 코딩할 수 있습니다.

```
DCL 1 TDEPT_STRUCTURE,
  %INCLUDE DEPARTMENT(DEPARTMENT,RECORD);
```

위의 예에서 TDEPT_STRUCTURE라는 호스트 구조는 4개의 필드를 갖도록 정의됩니다. 그 필드는 DEPTNO, DEPTNAME, MGRNO 및 ADMRDEPT입니다.

장치 파일의 경우 INDARA가 지정되지 않고 파일이 인디케이터를 포함하면 선언 부분은 호스트 구조 배열로 사용될 수 없습니다. 인디케이터 영역은 생성된 구조에 포함되며 기억영역을 인접하지 않도록 합니다.

```
DCL 1 DEPT_REC(10),
    %INCLUDE DEPARTMENT(DEPARTMENT,RECORD);
    :

    EXEC SQL DECLARE C1 CURSOR FOR
    SELECT * FROM CORPDATA.DEPARTMENT;

    EXEC SQL OPEN C1;

EXEC SQL FETCH C1 FOR 10 ROWS INTO :DEPT_REC;
```

주: DATE, TIME 및 TIMESTAMP 열은 DATE, TIME 및 TIMESTAMP 열과 동일한 비교 및 할당 규칙을 가진 SQL에 의해 처리된 호스트 변수 정의를 생성합니다. 예를 들어 날짜 호스트 변수는 날짜가 유효하게 표시된 DATE 열 또는 문자 스트링에 대해서만 비교됩니다.

유효 숫자가 16 이상인 10진 및 존(zoned) 필드와 소수 자릿수가 있는 정수 필드가 PL/I에서 문자 필드와 맵핑되어도 SQL은 이러한 필드를 숫자로 간주합니다.

GRAPHIC 및 VARGRAPHIC이 PL/I의 문자 변수에 맵핑되더라도 SQL은 이들을 GRAPHIC 및 VARGRAPHIC 호스트 변수로 간주합니다. GRAPHIC 또는 VARGRAPHIC 열에 UCS-2 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UCS-2 CCSID를 갖게 됩니다.

동일한 SQL 및 PL/I 자료 유형 판별

사전컴파일러는 다음 표에 있는 호스트 변수의 기본 SQLTYPE 및 SQLLEN을 판별합니다. 호스트 변수가 인디케이터 변수로 표시되면 SQLTYPE은 기본 SQLTYPE에 1을 더한 값이 됩니다.

표 5. 일반 SQL 자료 유형에 맵핑되는 PL/I 선언

PL/I 자료 유형	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
BIN FIXED(p), p는 1 - 15	500	2	SMALLINT
BIN FIXED(p), p는 16 - 31	496	4	INTEGER
DEC FIXED(p,s)	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s)
BIN FLOAT(p), p는 1 - 24	480	4	FLOAT(단정밀도)

표 5. 일반 SQL 자료 유형에 맵핑되는 PL/I 선언 (계속)

PL/I 자료 유형	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
BIN FLOAT(p), p는 25 - 53	480	8	FLOAT(배정밀도)
DEC FLOAT(m), m은 1 - 7	480	4	FLOAT(단정밀도)
DEC FLOAT(m), m은 8 - 16	480	8	FLOAT(배정밀도)
PICTURE 형상 문자(숫자)	488	바이트 1에는 p, 바이트 2에는 s	NUMERIC(p,s)
PICTURE 형상 문자(선행 부호 분리)	504	바이트 1에는 p, 바이트 2에는 s	정확히 같은 것은 없습니다. NUMERIC(p,s)을 사용합니다.
CHAR(n)	452	n	CHAR(n)
CHAR(n) VARYING, n < 255	448	n	VARCHAR(n)
CHAR(n) VARYING, n > 254	456	n	VARCHAR(n)

다음 표는 제공된 SQL 자료 유형과 동일한 PL/I 자료 유형 판별에 사용될 수 있습니다.

표 6. 일반 PL/I 선언 부분에 맵핑되는 SQL 자료 유형

SQL 자료 유형	PL/I 자료 유형	주석
SMALLINT	BIN FIXED(p)	p는 1 - 15 사이의 양의 정수입니다.
INTEGER	BIN FIXED(p)	p는 16 - 31 사이의 양의 정수입니다.
BIGINT	같은 것이 없습니다.	DEC FIXED(18)를 사용하십시오.
DECIMAL(p,s) 또는 NUMERIC(p,s)	DEC FIXED(p) 또는 DEC FIXED(p,s) 또는 PICTURE 스트링	s(스케일 인수)와 p(정밀도)는 양수입니다. p는 1 - 31 사이의 양의 정수입니다. s는 0 - p 범위의 양의 정수입니다.
FLOAT(단정밀도)	BIN FLOAT(p) 또는 DEC FLOAT(m)	p는 1 - 24 사이의 양의 정수입니다. m은 1 - 7 사이의 양의 정수입니다.
FLOAT(배정밀도)	BIN FLOAT(p) 또는 DEC FLOAT(m)	p는 25 - 53 사이의 양의 정수입니다. m은 8 - 16 사이의 양의 정수입니다.
CHAR(n)	CHAR(n)	n은 1 - 32766 사이의 양의 정수입니다.
VARCHAR(n)	CHAR(n) VAR	n은 1 - 32740 사이의 양의 정수입니다.
BLOB	없음	BLOB를 선언하려면 SQL TYPE IS를 사용하십시오.

표 6. 일반 PL/I 선언 부분에 맵핑되는 SQL 자료 유형 (계속)

SQL 자료 유형	PL/I 자료 유형	주석
CLOB	없음	CLOB를 선언하려면 SQL TYPE IS를 사용하십시오.
GRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
VARGRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
DBCLOB	없음	DBCLOB를 선언하려면 SQL TYPE IS를 사용하십시오.
DATE	CHAR(n)	형식이 *USA, *JIS, *EUR 또는 *ISO인 경우 n은 최소한 10자이어야 합니다. 형식이 *YMD, *DMY 또는 *MDY인 경우 n은 최소한 8자이어야 합니다. 형식이 *JUL인 경우 n은 최소한 6자이어야 합니다.
TIME	CHAR(n)	n은 최소한 6자이어야 하며 초를 포함시키려면 n은 최소한 8자이어야 합니다.
TIMESTAMP	CHAR(n)	n은 최소한 19이어야 합니다. 전체 정밀도에 마이크로초를 포함시키려면 n은 26이어야 하고 n이 26 미만이면 마이크로초 부분에서 잘립니다.
DATALINK	지원되지 않습니다.	지원되지 않습니다.
ROWID	없음	ROWID를 선언하려면 SQL TYPE IS를 사용하십시오.

SQL을 사용하는 PL/I 어플리케이션에서 인디케이터 변수 사용

인디케이터 변수는 2바이트 정수(BIN FIXED(p), p= 1-15)입니다. 또한 인디케이터 구조(반단어 정수 변수의 배열로 정의됨)를 지정하여 호스트 구조를 지원할 수도 있습니다. 검색시 연관 호스트 변수에 널(null)이 할당되었는지 여부를 나타내기 위해 인디케이터 변수를 사용합니다. 음의 인디케이터 변수는 열에 할당할 때 널이 할당되어야 함을 나타내기 위해 사용됩니다.

자세한 정보는 SQL 참조서의 인디케이터 변수 주제를 참조하십시오.

인디케이터 변수는 호스트 변수와 동일한 방법으로 선언되며 프로그래머는 적절한 방식으로 그 두 가지 선언 부분을 혼합할 수 있습니다.

예:

다음 명령문을 보십시오.

```
EXEC SQL FETCH CLS_CURSOR INTO :CLS_CD,
                                :DAY :DAY_IND,
                                :BGN :BGN_IND,
                                :END :END_IND;
```

이 경우 다음과 같이 변수를 선언할 수 있습니다.

```
EXEC SQL BEGIN DECLARE SECTION;
DCL CLS_CD CHAR(7);
DCL DAY BIN FIXED(15);
DCL BGN CHAR(8);
DCL END CHAR(8);
DCL (DAY_IND, BGN_IND, END_IND) BIN FIXED(15);
EXEC SQL END DECLARE SECTION;
```

구조 매개변수 전달 기법에 따른 PL/I의 차이점

PL/I 사전컴파일러는 가능하면 구조 매개변수 전달 기법을 사용하려고 시도합니다. 이 구조 매개변수 전달 기법은 SQL을 사용하는 대부분의 PL/I의 뛰어난 성능을 향상시켜 줍니다. 사전컴파일러는 다음 조건에 모두 참일 경우 각 호스트 변수가 별도의 매개변수인 코드를 생성합니다.

- PL/I %INCLUDE 컴파일러 지시문은 외부 텍스트가 소스 프로그램으로 복사된 것을 발견합니다.
- 명령문에서 참조된 호스트 변수의 자료 길이는 32703보다 큼니다. SQL이 구조의 64 바이트를 사용하므로 자료 구조의 최대 길이는 32767(32703 + 64)입니다.
- PL/I 사전컴파일러는 사용자 정의명에 대하여 PL/I 제한을 초과할 수 있는지를 평가합니다.
- 선행 부호 분리 호스트 변수가 SQL문에 대한 호스트 변수 리스트에서 발견됩니다.

구조 매개변수 전달 기법에 대한 자세한 정보는 *iSeries*용 DB2 UDB 데이터베이스 성능 및 조회 최적화 정보의 데이터베이스 어플리케이션 설계 추가 정보: 구조 매개변수 전달 기법 사용을 참조하십시오.

제 5 장 iSeries용 RPG 어플리케이션에서의 SQL문 코딩

iSeries용 RPG 사용권 프로그램에서는 RPG II 및 RPG III 프로그램을 모두 지원합니다. SQL문은 RPG III 프로그램에서만 사용될 수 있습니다. RPG II와 AutoReport는 지원되지 않습니다. 이 책의 RPG에 관한 모든 언급은 RPG III에만 적용됩니다.



이 장에서는 iSeries용 RPG 프로그램에서 SQL문을 삽입할 때 고유한 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 변수에 대한 요구사항도 정의됩니다.

자세한 내용은 다음 절을 참조하십시오.

- 100 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL 통신 영역 정의』
- 100 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL 설명자 영역 정의』
- 101 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL문 삽입』
- 103 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 변수 사용』
- 104 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 구조 사용』
- 105 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 구조 배열 사용』
- 106 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 외부 파일 설명 사용』
- 107 페이지의 『동등한 SQL 및 iSeries용 RPG 자료 유형 판별』
- 110 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 인디케이터 변수 사용』
- 111 페이지의 『구조 매개변수 전달 기법에 따른 iSeries용 RPG의 차이점』
- 111 페이지의 『SQL을 사용하는 호출된 iSeries용 RPG 프로그램의 올바른 종료』

SQL문의 사용 방법을 보여주는 자세한 샘플 iSeries용 RPG 프로그램이 부록 A 『iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램』에 나와 있습니다.

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

RPG를 사용한 프로그래밍에 대한 자세한 정보는 RPG/400® User's Guide  및 RPG/400 Reference  를 참조하십시오.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL 통신 영역 정의

SQL 사전컴파일러는 첫 번째 연산 스펙 이전의 iSeries용 RPG 프로그램의 입력 스펙에 SQLCA를 자동으로 위치시킵니다. 소스 프로그램에 INCLUDE SQLCA를 코딩해서는 안됩니다. 소스 프로그램이 INCLUDE SQLCA를 지정하면 명령문이 허용되기는 하지만 중복됩니다. iSeries용 RPG에 정의된 SQLCA는 다음과 같습니다.

ISQLCA	DS	SQL
I*	SQL Communications area	SQL
I	1 8 SQLAID	SQL
I	B 9 120SQLABC	SQL
I	B 13 160SQLCOD	SQL
I	B 17 180SQLERL	SQL
I	19 88 SQLERM	SQL
I	89 96 SQLERP	SQL
I	97 120 SQLERR	SQL
I	B 97 1000SQLER1	SQL
I	B 101 1040SQLER2	SQL
I	B 105 1080SQLER3	SQL
I	B 109 1120SQLER4	SQL
I	B 113 1160SQLER5	SQL
I	B 117 1200SQLER6	SQL
I	121 131 SQLWRN	SQL
I	121 121 SQLWN0	SQL
I	122 122 SQLWN1	SQL
I	123 123 SQLWN2	SQL
I	124 124 SQLWN3	SQL
I	125 125 SQLWN4	SQL
I	126 126 SQLWN5	SQL
I	127 127 SQLWN6	SQL
I	128 128 SQLWN7	SQL
I	129 129 SQLWN8	SQL
I	130 130 SQLWN9	SQL
I	131 131 SQLWNA	SQL
I	132 136 SQLSTT	SQL
I*	End of SQLCA	SQL

주: iSeries용 RPG의 변수명은 6자로 제한됩니다. 표준 SQLCA 이름의 길이는 6으로 변경되었습니다. iSeries용 RPG에서는 배열을 확장 스펙에 정의하지 않고 자료 구조에 정의하는 방법이 없습니다. SQLERR은 요소명으로 사용된 SQLER1에서 SQLER6까지의 문자로 정의됩니다.

자세한 정보는 SQL 참조서의 SQL 통신 영역을 참조하십시오.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL 설명자 영역 정의

다음 명령문은 SQLDA를 필요로 합니다.

EXECUTE...USING DESCRIPTOR *descriptor-name*

FETCH...USING DESCRIPTOR *descriptor-name*

OPEN...USING DESCRIPTOR *descriptor-name*

CALL...USING DESCRIPTOR *descriptor-name*
DESCRIBE *statement-name* INTO *descriptor-name*
DESCRIBE TABLE *host-variable* INTO *descriptor-name*
PREPARE *statement-name* INTO *descriptor-name*

SQLCA와는 달리 프로그램에는 두 개 이상의 SQLDA가 있을 수 있으며 SQLDA는 유효한 모든 이름을 가질 수 있습니다.

동적 SQL은 iSeries용 DB2 UDB 프로그램 개념 정보의 동적 SQL 어플리케이션에 설명되어 있는 첨단 프로그래밍 기술입니다. 동적 SQL을 사용하면 프로그램 수행 중 사용자 프로그램이 SQL문을 개발 및 수행할 수 있습니다. 동적으로 수행하는 가변 SELECT 리스트(조회)의 한 부분으로 리턴될 자료의 리스트임)가 있는 SELECT문은 SQL 설명자 영역(SQLDA)을 필요로 합니다. 그 이유는 사용자가 SELECT의 결과를 수신하기 위해 지정된 변수의 수와 유형을 미리 알 수 없기 때문입니다.

SQLDA는 iSeries용 RPG에서 지원되지 않은 포인터 변수를 사용하므로 INCLUDE SQLDA문을 iSeries용 RPG 프로그램에 지정할 수 없습니다. SQLDA는 C, COBOL, PL/I 또는 ILE RPG 프로그램에 의해 설정된 후에 사용하기 위해 RPG 프로그램으로 전달되어야 합니다.

SQLDA에 대한 자세한 정보는 SQL 참조서의 SQL 설명 영역을 참조하십시오.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL문 삽입

iSeries용 RPG 프로그램에서 코딩된 SQL문은 연산 구역에 위치해야 합니다. 이 경우 C가 위치 6에 있어야 합니다. SQL문은 상세 계산, 합계 또는 iSeries용 RPG 서브루틴에 위치할 수 있어야 합니다. SQL문은 iSeries용 RPG문의 논리에 따라 실행됩니다.

키워드 EXEC SQL은 SQL문의 시작을 표시합니다. EXEC SQL은 소스문의 위치 8에서 16까지를 차지하며 그 앞의 위치 7에는 슬래시(/)가 옵니다. SQL문은 위치 17에서 시작하여 위치 74까지 계속됩니다.

키워드 END-EXEC는 SQL문의 수행을 종료합니다. END-EXEC는 소스문의 위치 8에서 16까지를 차지하며 그 앞의 위치 7에는 /가 옵니다. 위치 17에서 74까지는 비어 있어야 합니다.

SQL문에는 대문자와 소문자 모두 사용될 수 있습니다.

자세한 내용은 다음 절을 참조하십시오.

- 102 페이지의 『예: SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL문 삽입』
- 102 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 주석』

- 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL문의 연속』
- 『SQL을 사용하는 iSeries용 RPG 어플리케이션에 코드 포함』
- 103 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 순번』
- 103 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 이름』
- 103 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 명령문 레이블』
- 103 페이지의 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 WHENEVER문』

예: SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL문 삽입

iSeries용 RPG 프로그램에서 코딩된 UPDATE문은 다음과 같이 코딩될 수 있습니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....*
C/EXEC SQL UPDATE DEPARTMENT
C+          SET MANAGER = :MGRNUM
C+          WHERE DEPTNO = :INTDEP
C/END-EXEC
```

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 주석

SQL 주석(--) 외에도 iSeries용 RPG 주석을 키워드 EXEC 및 SQL 사이를 제외하고 공백이 허용되는 모든 SQL문에 포함시킬 수 있습니다. SQL문 내에 iSeries용 RPG 주석을 삽입하려면 위치 7에 별표(*)를 배치하십시오.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 SQL문의 연속

SQL문이 들어갈 추가 레코드가 필요하면 9-74열까지 사용할 수 있습니다. 7열은 반드시 +(더하기 부호)이고 8열은 반드시 공백이어야 합니다.

DBCS 자료가 들어 있는 상수는 계속되는 행의 75열에 SI 문자를 위치시키고 계속행 8열에 SO 문자를 위치시킴으로 복수 행에 걸쳐 계속될 수 있습니다. 이 예에서 SQL 문에 G'<AABBCCDDEEFFGGHHIIJJKK>'의 유효한 그래픽 상수가 있습니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
C/EXEC SQL SELECT * FROM GRAPHTAB          WHERE GRAPHCOL = G'<AABB>
C+<CCDDEEFFGGHHIIJJKK>'
C/END-EXEC
```

SQL을 사용하는 iSeries용 RPG 어플리케이션에 코드 포함

SQL문과 iSeries용 RPG 연산 스펙은 SQL문을 삽입하여 포함시킬 수 있습니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
C/EXEC SQL INCLUDE member-name
C/END-EXEC
```

/COPY문을 사용하여 SQL문이나 iSeries용 RPG 스펙을 포함시킬 수 있습니다.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 순번

SQL 사전컴파일러에 의해 생성된 소스문의 순서 번호는 CRTSQLRPG 명령에 있는 옵션 매개변수의 *NOSEQSRC/*SEQSRC 키워드의 기본이 됩니다. *NOSEQSRC 지정시 입력 소스 멤버에서 순서 번호가 사용됩니다. *SEQSRC에 대해 순서 번호는 000001에서 시작하며 1씩 증가합니다.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 이름

호스트 변수에 대한 유효한 모든 RPG 변수명을 사용할 수 있으며 다음 제한사항에 따라야 합니다.

'SQ', 'SQL', 'RDI' 또는 'DSN'으로 시작하는 호스트 변수명이나 외부 항목명을 사용하지 마십시오. 이러한 이름들은 데이터베이스 관리자 예약어입니다.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 명령문 레이블

TAG문은 모든 SQL문 앞에 올 수 있습니다. EXEC SQL 앞 행에 TAG문을 코딩하십시오.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 WHENEVER문

GOTO절의 목표는 TAG문의 레이블이어야 합니다. GOTO/TAG에 대한 범위 규칙을 준수해야 합니다.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 변수 사용

SQL문에서 사용된 모든 호스트 변수는 명시적으로 선언되어야 합니다. LOB 및 ROWID 호스트 변수는 iSeries용 RPG에서 지원되지 않습니다.

iSeries용 RPG에 삽입된 SQL은 호스트 변수를 식별하기 위해 SQL BEGIN DECLARE SECTION 및 END DECLARE SECTION문을 사용하지 않습니다. 이러한 명령문은 소스 프로그램에 작성하지 마십시오.

SQL문 내의 모든 호스트 변수 앞에는 콜론(:)이 와야 합니다.

호스트 변수명은 프로그램 내에서 고유해야 합니다.

자세한 내용은 『SQL을 사용하는 iSeries용 RPG에서 호스트 변수 사용』을 참조하십시오.

SQL을 사용하는 iSeries용 RPG에서 호스트 변수 사용

iSeries용 SQL RPG 사전컴파일러는 유효한 호스트 변수 선언으로 iSeries용 RPG 선언의 서브세트만 인식합니다.

iSeries용 RPG에 정의된 모든 변수는 다음을 제외하고는 SQL문에서 사용할 수 있습니다.

인디케이터 필드 이름(*INxx)

표

UPDATE

UDAY

UMONTH

UYEAR

선행 참조 필드

명명된 상수

호스트 변수로 사용되는 필드는 iSeries용 RPG의 CALL/PARM 함수를 사용하여 SQL에 전달됩니다. PARM의 결과 필드에 사용될 수 없는 필드는 호스트 변수로서 사용할 수 없습니다.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 구조 사용

iSeries용 RPG 자료 구조명은 서브필드가 자료 구조에 존재하는 경우 호스트 구조명으로 사용될 수 있습니다. SQL문에 자료 구조명을 사용하면 서브필드명의 리스트가 자료 구조를 구성한다는 의미가 됩니다.

서브필드가 자료 구조 내에 존재하지 않으면 자료 구조명은 문자 유형의 호스트 변수입니다. 자료 구조가 최대 9999자가 될 수 있으므로 문자 변수의 길이는 256보다 클 수 있습니다.

다음 예에서 BIGCHR은 서브필드가 없는 iSeries용 RPG 자료 구조입니다. SQL은 BIGCHR에 대한 모든 참조를 길이가 642인 문자 스트링으로 간주합니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7...*
IBIGCHR      DS                      642
```

다음 예에서 PEMPL은 서브필드 EMPNO, FIRSTNME, MIDINIT, LASTNAME 및 DEPTNO로 구성된 호스트 구조명입니다. PEMPL에 대한 참조는 서브필드를 사용합니다. 예를 들어 EMPLOYEE의 첫 번째 열은 EMPNO에 위치하고 두 번째 열은 FIRSTN에 위치하는 식입니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7. ...*
IPEMPL      DS
I              01 06 EMPNO
I              07 18 FIRSTN
I              19 19 MIDINT
I              20 34 LASTNA
I              35 37 DEPTNO
...
C              MOVE      '000220'      EMPNO
```

```

...
C/EXEC SQL
C+ SELECT * INTO :PEMPL
C+ FROM CORPDATA.EMPLOYEE
C+ WHERE EMPNO = :EMPNO
C/END-EXEC

```

SQL문을 작성할 때 서브필드에 대한 참조를 규정할 수 있습니다. 뒤에 마침표와 서브필드명이 있는 자료 구조명을 사용하십시오. 예를 들어 PEMPL.MIDINT는 MIDINT만 지정하는 것과 동일합니다.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 구조 배열 사용

호스트 구조 배열은 발생 자료 구조로 정의됩니다. 발생 자료 구조는 복수 행을 폐치할 때 SQL FETCH문에 사용됩니다. 이 예에서는 다음 사항이 참입니다.

- B_ARRAY 내의 모든 항목은 유효한 호스트 변수이어야 합니다.
- BARRAY 내의 모든 항목은 인접해야 합니다. 첫 번째 FROM 열은 1이어야 하고 TO와 FROM 열에는 중복이 있을 수 없습니다.
- 발생 자료 구조가 사용된다면 복수 행 FETCH 및 블록 INSERT 이외의 모든 명령문에 대해 현재 발생이 사용됩니다. 복수 행 FETCH 및 블록 INSERT의 경우 발생이 1에 설정됩니다.

```

*...1....+....2....+....3....+....4....+....5....+....6....+....7. ...*
IBARRAY      DS                      10
I              01 20 C1VAR
I              B 21 220C2VAR

```

다음의 예는 DEPARTMENT 표에서 10행을 검색하기 위하여 DEPT 및 복수 행 FETCH문을 호출한 호스트 구조 배열을 사용합니다.

```

*...1....+....2....+....3....+....4....+....5....+....6....+....7...*
E              INDS                   4 4 0
IDEPT          DS                      10
I              01 03 DEPTNO
I              04 32 DEPTNM
I              33 38 MGRNO
I              39 41 ADMRD
IINDARR        DS                      10
I              B 1 80INDS

```

```

...
C/EXEC SQL
C+ DECLARE C1 CURSOR FOR
C+ SELECT *
C+ FROM CORPDATA.DEPARTMENT
C/END-EXEC
C/EXEC SQL
C+ OPEN C1
C/END-EXEC
C/EXEC SQL
C+ FETCH C1 FOR 10 ROWS INTO :DEPT:INDARR
C/END-EXEC

```

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 외부 파일 설명 사용

SQL 사전컴파일러는 iSeries용 ILE RPG 컴파일러와 거의 같은 방식으로 iSeries용 RPG 소스를 처리합니다. 즉, 사전컴파일러는 호스트 변수의 정의를 위해 /COPY문을 처리합니다. 다른 이름이 지정되면 외부 서술 파일에 대한 필드 정의의 사용을 지정하고 재명명합니다. 자료 구조의 외부 정의 형식을 사용하여 호스트 변수로 사용될 열명의 사본을 확보할 수 있습니다.

다음 예에서 샘플 표 DEPARTMENT는 iSeries용 ILE RPG 프로그램에서 파일로 사용됩니다. SQL 사전컴파일러는 호스트 변수로 사용될 DEPARTMENT에 대한 필드(열) 정의를 검색합니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....*
FTDEPT  IP  E                                DISK
F          TDEPT                            KRENAMEDPTREC
IDEPTREC
I          DEPTNAME                          DEPTN
I          ADMRDEPT                          ADMRD
```

주: 파일에 대한 I/O 조작을 수행하려면 iSeries용 RPG 명령문을 사용하는 경우에만 사용자의 RPG 프로그램에서 파일에 대한 F 스펙을 코딩하십시오. SQL문만을 사용하여 파일에 대한 I/O 조작을 수행하는 경우에는 외부 자료 구조를 사용하여 외부 정의를 포함시킬 수 있습니다.

다음 예에서는 샘플 표가 외부 자료 구조로서 지정되어 있습니다. SQL 사전컴파일러는 필드(열) 정의를 자료 구조의 서브필드로서 검색합니다. 서브필드명은 호스트 변수명으로 사용될 수 있으며, 자료 구조명 TDEPT는 호스트 구조명으로 사용될 수 있습니다. 필드명은 7자 이상이기 때문에 변경되어야 합니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....*
ITDEPT  E DSDEPARTMENT
I          DEPTNAME                          DEPTN
I          ADMRDEPT                          ADMRD
```

주: DATE, TIME 및 TIMESTAMP 열은 DATE, TIME 및 TIMESTAMP 열과 동일한 비교 및 할당 규칙을 가진 SQL에 의해 처리된 호스트 변수 정의를 생성합니다. 예를 들어 날짜 호스트 변수는 DATE 열 또는 자료의 유효한 표시인 문자 스트링에 대해서만 비교될 수 있습니다.

가변 길이 열이 고정 길이 문자 호스트 변수 정의를 생성해도 SQL에 대해서는 가변 길이 문자 변수입니다.

GRAPHIC 및 VARGRAPHIC 열이 iSeries용 RPG의 문자 변수에 맵핑되더라도, SQL은 이러한 GRAPHIC 및 VARGRAPHIC 변수를 고려합니다. GRAPHIC 또는 VARGRAPHIC 열에 UCS-2 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UCS-2 CCSID를 갖게 됩니다.

다른 예는 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 구조 배열에 대한 외부 파일 설명 고려사항』을 참조하십시오.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 호스트 구조 배열에 대한 외부 파일 설명 고려사항

파일이 부동 소수점 필드를 포함하는 경우 이는 호스트 구조 배열로 사용될 수 없습니다. 장치 파일의 경우 INDARA가 지정되지 않고 파일이 인디케이터를 포함하면 선언 부분은 호스트 구조 배열로 사용되지 않습니다. 인디케이터 영역은 생성된 구조 내로 포함되며 기억영역이 인접하지 못하는 원인이 됩니다.

다음 예에서 DEPARTMENT 표는 iSeries용 RPG 프로그램에 포함되어 있으며 호스트 구조 배열을 선언하는 데 사용됩니다. 그러면 복수 행 FETCH문이 10행을 검색하기 위하여 호스트 구조 배열로 사용됩니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....*
ITDEPT      E DSDEPARTMENT          10
I           DEPARTMENT              DEPTN
I           ADMRDEPT                 ADMRD

...

C/EXEC SQL
  C+ DECLARE C1 CURSOR FOR
C+   SELECT *
C+   FROM CORPDATA.DEPARTMENT
  C/END-EXEC

...

C/EXEC SQL
C+ FETCH C1 FOR 10 ROWS INTO :TDEPT
  C/END-EXEC
```

동등한 SQL 및 iSeries용 RPG 자료 유형 판별

사전컴파일러는 다음 표에 있는 호스트 변수의 기본 SQLTYPE 및 SQLLEN을 판별합니다. 호스트 변수가 인디케이터 변수로 표시되면 SQLTYPE은 기본 SQLTYPE에 1을 더한 값이 됩니다.

표 7. 일반 SQL 자료 유형에 맵핑되는 iSeries용 RPG 선언

iSeries용 RPG 자료 유형	43열	52열	기타 iSeries용 RPG 코딩	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
자료 구조 서브필드	공백	공백	길이 = n, n ≤ 256	452	n	CHAR(n)
자료 구조(서브필드 없음)	n/a	n/a	길이 = n, n ≤ 9999	452	n	CHAR(n)
입력 필드	공백	공백	길이 = n, n ≤ 256	452	n	CHAR(n)
연산 결과 필드	n/a	공백	길이 = n, n ≤ 256	452	n	CHAR(n)

표 7. 일반 SQL 자료 유형에 맵핑되는 iSeries용 RPG 선언 (계속)

iSeries용 RPG 자료 유형	43열	52열	기타 iSeries용 RPG 코딩	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
자료 구조 서브필드	B	0	길이 = 2	500	2	SMALLINT
자료 구조 서브필드	B	0	길이 = 4	496	4	INTEGER
자료 구조 서브필드	B	1-4	길이 = 2	500	2	DECIMAL(4,s), s = 52열
자료 구조 서브필드	B	1-9	길이 = 4	496	4	DECIMAL(9,s), s = 52열
자료 구조 서브필드	P	0-9	길이 = n, n= 1-16	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s) , p = n*2-1이고 s = 52열
입력 필드	P	0-9	길이 = n, n= 1-16	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s) , p = n*2-1이고 s = 52열
입력 필드	공백	0-9	길이 = n, n= 1-30	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s), p = n이고 s = 52열
입력 필드	B	0-4이면 n = 2; 0-9이면 n = 4	길이 = 2 또는 4	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s), n=2이면 p=4 또는 n=4이고 s = 52열이면 9
연산 결과 필드	n/a	0-9	길이 = n, n= 1-30	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s), p = n이고 s = 52열
자료 구조 서브필드	공백	0-9	길이 = n, n= 1-30	488	바이트 1에는 p, 바이트 2에는 s	NUMERIC(p,s), p = n이고 s = 52열

다음 표의 정보를 사용하여 제공된 SQL 자료 유형과 동일한 iSeries용 RPG 자료 유형을 판별할 수 있습니다.

표 8. 일반 iSeries용 RPG 선언에 맵핑되는 SQL 자료 유형

SQL 자료 유형	iSeries용 RPG 자료 유형	주
SMALLINT	자료 구조의 서브필드. 43열은 B, 서브필드 스펙의 52열은 길이가 2와 0이어야 합니다.	
INTEGER	자료 구조의 서브필드. 43열은 B, 서브필드 스펙의 52열은 길이가 4와 0이어야 합니다.	
BIGINT	같은 것이 없습니다.	서브필드 스펙의 위치 43열에는 P, 52열에는 0을 사용하십시오.

표 8. 일반 iSeries용 RPG 선언에 맵핑되는 SQL 자료 유형 (계속)

SQL 자료 유형	iSeries용 RPG 자료 유형	주
DECIMAL	자료 구조의 서브필드. 43열은 P, 서브필드 스펙의 53열은 0-9이어야 합니다. 또는 숫자로 정의되며 자료 구조의 서브필드가 아닙니다.	최대 길이 16(정밀도 30)과 최대 소수 자릿수 9.
NUMERIC	자료 구조의 서브필드. 43열은 공백, 서브필드의 52열은 0-9이어야 합니다.	최대 길이 30(정밀도 30)과 최대 소수 자릿수 9.
FLOAT(단정밀도)	같은 것이 없습니다.	아래에 설명된 숫자 자료 유형 중 하나를 사용하십시오.
FLOAT(배정밀도)	같은 것이 없습니다.	아래에 설명된 숫자 자료 유형 중 하나를 사용하십시오.
CHAR(n)	자료 구조 또는 입력 필드의 서브필드. 스펙의 43열과 52열은 공백. 또는 소수 자리 없이 정의된 연산 결과 필드.	n은 1-256 사이일 수 있습니다.
CHAR(n)	자료 구조에 서브필드가 없는 자료 구조명.	n은 1 - 9999입니다.
VARCHAR(n)	같은 것이 없습니다.	예측된 가장 큰 VARCHAR 값을 포함시키기에 충분히 큰 문자 호스트 변수를 사용하십시오.
BLOB	지원되지 않습니다.	지원되지 않습니다.
CLOB	지원되지 않습니다.	지원되지 않습니다.
GRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
VARGRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
DBCLOB	지원되지 않습니다.	지원되지 않습니다.
DATE	자료 구조의 서브필드. 서브필드 스펙의 52열은 공백. 또는 소수 자릿수 없이 정의된 필드.	형식이 *USA, *JIS *EUR 또는 *ISO이면 길이는 최소한 10이어야 합니다. 형식이 *YMD, *DMY 또는 *MDY이면 길이는 최소한 8이어야 합니다. 형식이 *JUL이면 길이는 최소한 6이어야 합니다.
TIME	자료 구조의 서브필드. 서브필드 스펙의 52열은 공백. 또는 소수 자릿수 없이 정의된 필드.	길이는 최소한 6이어야 합니다. 초를 포함하려면 최소한 8이 되어야 합니다.
TIMESTAMP	자료 구조의 서브필드. 서브필드 스펙의 52열은 공백. 또는 소수 자릿수 없이 정의된 필드.	길이는 최소한 19여야 하며 전체 정밀도에 마이크로 초를 포함하려면 길이가 26이 되어야 합니다. 길이가 26보다 적다면 마이크로초 부분에서 잘립니다.
DATALINK	지원되지 않습니다.	지원되지 않습니다.
ROWID	지원되지 않습니다.	지원되지 않습니다.

자세한 정보는 『SQL을 사용하는 iSeries용 RPG 어플리케이션에서 iSeries용 RPG 변수 선언 및 사용에 대한 주』를 참조하십시오.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 iSeries용 RPG 변수 선언 및 사용에 대한 주

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 지정 규칙

iSeries용 RPG는 정밀도와 소수 자릿수를 모든 숫자 유형과 연관시킵니다. iSeries용 RPG는 자료가 팩 형식이라고 가정하고 숫자 연산을 정의합니다. 즉, 2진 변수가 포함된 연산에는 연산 수행 전에 팩된 형식으로(필요하면 다시 2진수로) 내재 변환을 포함시킵니다. SQL 연산 수행시 자료는 내포된 소수점에 정렬됩니다.

SQL을 사용하는 iSeries용 RPG 어플리케이션에서 인디케이터 변수 사용

인디케이터 변수는 2바이트 정수입니다. (107 페이지의 표 7에 있는 SMALLINT SQL 자료 유형 항목을 참조하십시오.)

인디케이터 구조는 변수를 요소 길이가 4,0인 배열로 선언하고 배열명을 43열이 B인 자료 구조의 서브필드로 선언함으로써 정의할 수 있습니다. 검색시 연관 호스트 변수에 널(null)이 할당되었는지 여부를 나타내기 위해 인디케이터 변수를 사용합니다. 음의 인디케이터 변수는 열에 값을 할당할 때 널이 할당되어야 함을 나타내는데 사용됩니다.

자세한 정보는 SQL 참조서의 인디케이터 변수 주제를 참조하십시오.

인디케이터 변수는 호스트 변수와 동일한 방법으로 선언되며 프로그래머는 적절한 방식으로 그 두 가지 선언 부분을 혼합할 수 있습니다.

인디케이터 변수 사용 예는 『예: SQL을 사용하는 iSeries용 RPG 어플리케이션에서 인디케이터 변수 사용』을 참조하십시오.

예: SQL을 사용하는 iSeries용 RPG 어플리케이션에서 인디케이터 변수 사용

다음 명령문을 보십시오.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7...*
C/EXEC SQL FETCH CLS_CURSOR INTO :CLSCD,
C+                               :DAY :DAYIND,
C+                               :BGN :BGNIND,
C+                               :END :ENDIND
C/END-EXEC
```

이 경우 다음과 같이 변수를 선언할 수 있습니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7...*
I          DS
I                               1   7 CLSCD
I                               B   8   90DAY
I                               B  10 110DAYIND
```

I	12	19	BGN
I	B	20	210BGNIND
I		22	29 END
I	B	30	310ENDIND

구조 매개변수 전달 기법에 따른 iSeries용 RPG의 차이점

iSeries용 SQL RPG 사전컴파일러는 가능한 한, 구조 매개변수 전달 기법을 사용하고 합니다. 사전컴파일러는 다음 조건에 모두 참일 경우 각 호스트 변수가 별도의 매개 변수인 코드를 생성합니다.

- 명령문에서 참조된 호스트 변수의 자료 길이는 9935보다 큼니다. SQL이 구조의 64 바이트를 사용하므로 자료 구조의 최대 길이는 9999(9935 + 64)입니다.
- 색인화 인디케이터명의 길이에 필요한 색인값을 더한 값이 6자를 넘는 명령문에 인디케이터를 지정합니다. 사전컴파일러는 인디케이터명을 갖는 인디케이터에 대한 지정문을 6자로 제한한 결과 필드에 생성해야만 합니다("INDIC,1"은 7자를 필요로 합니다).
- 호스트 변수의 길이는 256보다 큼니다. 이것은 서브필드가 없는 자료 구조가 호스트 변수로서 사용된 경우이며 길이가 256을 초과합니다. 서브필드의 길이는 256보다 크게 정의될 수 없습니다.

구조 매개변수 전달 기법에 대한 자세한 정보는 *iSeries용 DB2 UDB 데이터베이스 성능 및 조회 최적화* 정보의 데이터베이스 어플리케이션 설계 추가 정보: 구조 매개변수 전달 기법 사용을 참조하십시오.

SQL을 사용하는 호출된 iSeries용 RPG 프로그램의 올바른 종료

SQL 수행시 SQL은 호스트 변수가 들어 있는 각각의 SQL문에 대해 자료 영역(내부 SQLDA)을 빌드하고 유지보수합니다. 이 내부 SQLDA는 명령문이 맨처음 수행될 때 설정되어 명령문의 후속 실행에 계속 재사용되므로 성능 향상에 도움이 됩니다. 내부 SQLDA는 최소 하나의 SQL 프로그램이 활동하고 있는 한 재사용될 수 있습니다. SQL 사전컴파일러는 내부 SQLDA를 올바르게 관리하기 위해 SQL 수행시 사용되는 정적 기억영역을 할당합니다.

SQL을 포함하는 iSeries용 RPG 프로그램이 SQL이 포함된 또 다른 프로그램에서 호출되면 iSeries용 RPG 프로그램은 최종 레코드(LR) 인디케이터를 on으로 설정해서는 안됩니다. LR 인디케이터를 on으로 설정하면 다음 번에 iSeries용 RPG 프로그램이 실행될 때 정적 기억장치가 다시 초기화됩니다. 정적 기억영역이 다시 초기설정되면 내부 SQLDA가 재설정되므로 성능이 떨어지게 됩니다.

SQL문을 포함하는 프로그램에서 호출된 SQL문을 포함하는 iSeries용 RPG 프로그램은 다음 두 방법 중 하나로 종료되어야 합니다.

- RETRN문에 의해

- RT 인디케이터를 on으로 설정하여
이는 내부 SQLDA를 다시 사용할 수 있게 하며 총 수행 시간을 감소시킵니다.

제 6 장 iSeries용 ILE RPG 어플리케이션에서의 SQL문 코딩



이 장에서는 iSeries용 ILE RPG 프로그램에서 SQL문을 삽입할 때 고유한 응용프로그램 및 코딩 요구사항에 대해 설명합니다. 호스트 변수에 대한 코딩 요구사항도 정의됩니다.

자세한 내용은 다음 절을 참조하십시오.

- 114 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL 통신 영역 정의』
- 115 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL 설명자 영역 정의』
- 116 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL문 삽입』
- 118 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 변수 사용』
- 120 페이지의 『SQL을 사용하는 iSeries용 ILE RPG에서 호스트 구조 사용』
- 121 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 구조 사용』
- 122 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 호스트 변수 선언』
- 125 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 ROWID 변수』
- 125 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 외부 파일 설명 사용』
- 127 페이지의 『동등한 SQL 및 RPG 자료 유형 판별』
- 131 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 인디케이터 변수 사용』
- 132 페이지의 『SQL에서 사용하는 iSeries용 ILE RPG 어플리케이션에서 복수 행 영역 페치에 대한 SQLDA 예』
- 133 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 동적 SQL 예』

SQL문을 사용하는 방법을 보여주는 자세한 ILE RPG 프로그램에 대해서는 187 페이지의 『예: iSeries용 ILE RPG 프로그램에서 SQL문』을 참조하십시오.

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

ILE RPG를 사용한 프로그래밍에 대한 자세한 정보는 ILE RPG Programmer's Guide  및 ILE RPG Reference  를 참조하십시오.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL 통신 영역 정의

SQL 사전컴파일러는 첫 번째 연산 스펙 이전의 iSeries용 ILE RPG 프로그램의 정의 스펙에 SQLCA를 위치시킵니다. 소스 프로그램에 INCLUDE SQLCA를 코딩해서는 안됩니다. 소스 프로그램이 INCLUDE SQLCA를 지정하면 명령문이 허용되기는 하지만 중복됩니다. iSeries용 ILE RPG에 정의된 SQLCA는 다음과 같습니다.

```
D*      SQL Communications area
D SQLCA      DS
D  SQLAID      1      8A
D  SQLABC      9      12B 0
D  SQLCOD     13      16B 0
D  SQLERL     17      18B 0
D  SQLERM     19      88A
D  SQLERP     89      96A
D  SQLERRD    97      120B 0 DIM(6)
D  SQLERR     97      120A
D  SQLER1     97      100B 0
D  SQLER2    101      104B 0
D  SQLER3    105      108B 0
D  SQLER4    109      112B 0
D  SQLER5    113      116B 0
D  SQLER6    117      120B 0
D  SQLWRN    121      131A
D  SQLWN0    121      121A
D  SQLWN1    122      122A
D  SQLWN2    123      123A
D  SQLWN3    124      124A
D  SQLWN4    125      125A
D  SQLWN5    126      126A
D  SQLWN6    127      127A
D  SQLWN7    128      128A
D  SQLWN8    129      129A
D  SQLWN9    130      130A
D  SQLWNA    131      131A
D  SQLSTT    132      136A
D*      End of SQLCA
```

주: iSeries용 RPG의 변수명은 6자로 제한됩니다. 표준 SQLCA명은 iSeries용 RPG에 대해 길이가 6으로 변경되었습니다. iSeries용 ILE RPG로 변환되는 iSeries용 RPG 프로그램과의 호환성을 유지보수하기 위해 SQLCA의 이름은 iSeries용 RPG에 사용된 대로 남게 됩니다. iSeries용 ILE RPG에 정의된 SQLCA는 6개의 정수 배열로 정의된 SQLERRD 필드를 추가합니다. SQLERRD는 SQLERR 정의를 오버레이하도록 정의합니다.

SQLCA에 대한 자세한 정보는 SQL 참조서의 SQL 통신 영역을 참조하십시오.


```

D  SQLLEN          3      4B 0
D  SQLRES          5      16A
D  SQLDATA        17     32*
D  SQLIND         33     48*
D  SQLNAMELEN     49     50B 0
D  SQLNAME        51     80A
D*  End of SQLDA

```

SQL_NUM은 사용자가 정의해야 합니다. SQL_NUM은 SQL_VAR에 필요한 차원을 갖는 숫자 상수로서 정의되어야 합니다.

INCLUDE SQLDA는 두 개의 자료 구조를 생성합니다. 2차 자료 구조는 필드 설명이 들어 있는 SQLDA의 부분을 설정/참조하는 데 사용됩니다.

SQLDA의 필드 설명을 설정하기 위해 프로그램은 SQLVAR의 서브필드에서 필드 설명을 설정한 후 SQLVAR의 MOVEA를 SQL_VAR, n으로 설정합니다. 여기서 n은 SQLDA의 필드 수입입니다. 이것은 모든 필드 설명이 설정될 때까지 반복됩니다.

SQLDA 필드 설명이 참조될 때 사용자는 SQL_VAR, n의 MOVEA를 SQLVAR로 설정합니다. 여기서 n은 처리되는 필드 설명의 수입입니다.

SQLDA에 대한 자세한 정보는 SQL 참조서의 SQL 설명자 영역을 참조하십시오.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL문 삽입

ILE RPG 프로그램에서 작성된 SQL문은 연산 섹션에 위치해야 합니다. 이 경우 C가 위치 6에 있어야 합니다. SQL문은 자세한 계산, 합계 또는 RPG/400 서브루틴에 위치할 수 있습니다. SQL문은 RPG문의 논리에 따라 수행됩니다.

키워드 EXEC SQL은 SQL문의 시작을 표시합니다. EXEC SQL은 소스문의 위치 8에서 16까지를 차지하며 그 앞의 위치 7에는 슬래시(/)가 옵니다. SQL문은 위치 17에서 시작하여 위치 80까지 계속됩니다.

키워드 END-EXEC는 SQL문의 수행을 종료합니다. END-EXEC는 소스문의 위치 8에서 16까지를 차지하며 그 앞의 위치 7에는 /가 옵니다. 위치 17에서 80까지는 비어 있어야 합니다.

SQL문에는 대문자와 소문자 모두 사용될 수 있습니다.

iSeries용 ILE RPG에서 코딩된 UPDATE문은 다음과 같이 코딩될 수 있습니다.

```

*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8.
C/EXEC SQL UPDATE DEPARTMENT
C+           SET MANAGER = :MGRNUM
C+           WHERE DEPTNO = :INTDEP
C/END-EXEC

```

자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 주석』
- 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL문의 연속』
- 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에 코드 포함』
- 118 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 지시문 사용』
- 118 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 순번』
- 118 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 이름』
- 118 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 명령문 레이블』
- 118 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 WHENEVER 문』

SELECT 및 UPDATE문 사이의 행 잠금에 대한 정보는 *SQL 프로그래밍 개념*의 *확약 제어*를 참조하십시오.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 주석

SQL 주석(--)
외에도 iSeries용 ILE RPG 주석을 SQL에서 공백 문자를 허용하는 모든 SQL문에 포함시킬 수 있습니다. SQL문 내에 iSeries용 ILE RPG 주석을 삽입하려면 위치 7에 별표(*)를 배치하십시오.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 SQL문의 연속

SQL문을 포함시키기 위한 추가 레코드가 필요할 때에는 9-80열이 사용될 수 있습니다. 7열은 반드시 +(더하기 부호)이고 8열은 반드시 공백이어야 합니다. 계속된 행의 80열은 계속 행의 9열로 연결됩니다.

DBCS 자료가 들어 있는 상수는 계속된 행의 81열에 DBCS 끝(SI) 제어 문자를 위치시키고 계속된 행의 8열에 DBCS 시작(SO) 제어 문자를 위치시킴으로써 복수 행에 걸쳐 계속됩니다.

이 예에서 SQL문은 G'<AABBCCDDEEFFGGHHIIJJKK>'의 유효한 그래픽 상수를 가집니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8.
C/EXEC SQL      SELECT * FROM GRAPHTAB WHERE GRAPHCOL =  G'<AABBCCDDEE>
C+<FFGGHHIIJJKK>'
C/END-EXEC
```

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에 코드 포함

SQL문 및 RPG 연산 스펙은 SQL문을 사용하여 포함시킬 수 있습니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8  
C/EXEC SQL INCLUDE member-name  
C/END-EXEC
```

RPG /COPY 지시문은 SQL문이나 RPG 스펙을 포함시키는 데 사용할 수 있습니다. 내포된 /COPY문은 사전컴파일러에서 지원하지 않습니다. RPG /INCLUDE 지시문은 사전컴파일러에서 인식하지 못합니다. SQL에서 처리되어서는 안되는 RPG 코드를 포함시키는 데 사용할 수 있습니다. 이는 조건부 지시문을 포함하는 코드와 기타 /COPY 블록에서의 내포에 유용할 수 있습니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 지시문 사용

/COPY 이외의 지시문은 SQL 사전컴파일러에서 무시됩니다. 이들은 처리할 컴파일러와 함께 전달됩니다. 이는 조건부 논리 블록 내의 모든 RPG 및 SQL문이 사전컴파일러에서 무조건적으로 처리됩니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 순번

SQL 사전컴파일러에 의해 생성된 소스문의 순서 번호는 CRTSQLRPGI 명령에 있는 OPTION 매개변수 *NOSEQSRC/*SEQSRC 키워드의 기본이 됩니다. *NOSEQSRC 지정시 입력 소스 맴버에서 순서 번호가 사용됩니다. *SEQSRC에 대해 순서 번호는 000001에서 시작하며 1씩 증가합니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 이름

유효한 모든 iSeries용 ILE RPG 변수명은 호스트 변수에 사용될 수 있으며, 다음 제한사항을 따라야 합니다.

문자 'SQ', 'SQL', 'RDI' 또는 'DSN'으로 시작하는 호스트 변수명이나 외부 항목명을 사용하지 마십시오. 이러한 이름들은 데이터베이스 관리자 예약어입니다. 호스트 변수명의 길이는 64자로 제한됩니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 명령문 레이블

TAG문은 모든 SQL문 앞에 올 수 있습니다. EXEC SQL 앞 행에 TAG문을 코딩하십시오.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 WHENEVER문

GOTO절의 목표는 TAG문의 레이블이어야 합니다. GOTO/TAG에 대한 범위 규칙을 준수해야 합니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 변수 사용

SQL문에서 사용된 모든 호스트 변수는 명시적으로 선언되어야 합니다.

iSeries용 ILE RPG에 삽입된 SQL은 호스트 변수를 식별하기 위해 SQL BEGIN DECLARE SECTION 및 END DECLARE SECTION문을 사용하지 않습니다. 이러한 명령문은 소스 프로그램에 작성하지 마십시오.

SQL문 내의 모든 호스트 변수 앞에는 콜론(:)이 와야 합니다.

호스트 변수명은 호스트 변수가 다른 블록이나 프로시저에 존재 하더라도 프로그램 내에서는 고유해야 합니다.

호스트 변수를 사용하는 SQL문은 변수가 선언된 명령문의 범위 내에 있어야 합니다.

자세한 내용은 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 변수 선언』을 참조하십시오.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 변수 선언

iSeries용 SQL ILE RPG 사전컴파일러는 유효한 호스트 변수 선언으로 iSeries용 ILE RPG 선언의 서브세트만 인식합니다.

iSeries용 ILE RPG에 정의된 대부분의 변수는 SQL문에서 사용할 수 있습니다. 지원되지 않는 일부 변수에 대한 목록은 다음과 같습니다.

포인터

표

UPDATE

UDAY

UMONTH

UYEAR

선행 참조 필드

명명된 상수

복수 차원 배열

*SIZE 또는 *ELEM의 맵핑을 필요로 하는 정의

상수가 OCCURS 또는 DIM에서 사용되지 않을 경우에 상수의 맵핑을 필요로 하는 정의

호스트 변수로 사용되는 필드는 iSeries용 ILE RPG의 CALL/PARM 함수를 사용하여 SQL에 전달됩니다. PARM의 결과 필드에 사용될 수 없는 필드는 호스트 변수로서 사용할 수 없습니다.

날짜와 시간 호스트 변수는 항상 SQL 사전컴파일러에 의해 생성된 구조에서 해당 날짜와 시간 서브필드에 할당됩니다. 생성된 날짜 및 시간 서브필드는 CRTSQLRPGI 명령상의 DATFMT, DATSEP, TIMFMT 및 TIMSEP 매개변수에 의해 지정된 형식과 분리문자를 사용함으로써 선언됩니다. 사용자 선언 호스트 변수 형식에서 사전컴파일 지

정 형식으로서의 변환은 SQL 생성 구조로의 지정 및 SQL 생성 구조로부터의 할당시에 발생합니다. DATFMT 매개변수 값이 시스템 형식(*MDY, *YMD, *DMY 또는 *JUL) 일 경우 모든 입/출력 호스트 변수에 1940-2039 범위 내의 날짜값이 들어 있어야 합니다. 날짜 값이 이 범위 밖에 있을 경우 사전컴파일에서의 DATFMT는 IBM SQL 형식인 *ISO, *USA, *EUR 또는 *JIS 중 하나로 지정되어야 합니다.

SQL을 사용하는 iSeries용 ILE RPG에서 호스트 구조 사용

iSeries용 ILE RPG 자료 구조명은 서브필드가 자료 구조에 존재하는 경우 호스트 구조명으로 사용될 수 있습니다. SQL문에 자료 구조명을 사용하면 서브필드명의 리스트가 자료 구조를 구성한다는 의미가 됩니다.

서브필드가 자료 구조 내에 존재하지 않으면 자료 구조명은 문자 유형의 호스트 변수입니다. 이는 257개 이상의 문자 변수를 허용합니다. 필드가 최대 길이 32766으로 정의될 수 있으므로 이 지원이 추가 기능을 제공하지 않는 동안 이는 iSeries용 RPG 프로그램과의 호환을 위해 필수입니다.

다음 예에서 BIGCHR은 서브필드가 없는 iSeries용 ILE RPG 자료 구조입니다. SQL은 BIGCHR에 대한 모든 참조를 길이가 642인 문자 스트링으로 간주합니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
DBIGCHR          DS          642
```

다음 예에서 PEMPL은 서브필드 EMPNO, FIRSTNME, MIDINIT, LASTNAME 및 DEPTNO로 구성된 호스트 구조명입니다. PEMPL에 대한 참조는 서브필드를 사용합니다. 예를 들어 CORPDATA.EMPLOYEE의 첫 번째 열은 EMPNO에 위치하고 두 번째 열은 FIRSTN에 위치하는 식입니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
DPEMPL          DS
D EMPNO          01      06A
D FIRSTN        07      18A
D MIDINT        19      19A
D LASTNA        20      34A
D DEPTNO        35      37A
```

```
...
C              MOVE      '000220'      EMPNO
```

```
...
C/EXEC SQL
C+ SELECT * INTO :PEMPL
C+ FROM CORPDATA.EMPLOYEE
C+ WHERE EMPNO = :EMPNO
C/END-EXEC
```

SQL문을 작성할 때 서브필드에 대한 참조를 규정할 수 있습니다. 뒤에 마침표와 서브필드명이 있는 자료 구조명을 사용하십시오. 예를 들어 PEMPL.MIDINT는 MIDINT만 지정하는 것과 동일합니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 구조 사용

호스트 구조 배열은 발생 자료 구조로 정의됩니다. 발생 자료 구조는 복수 행 처리시 SQL FETCH 또는 INSERT문에서 사용될 수 있습니다. 복수 행 블로킹 지원을 가진 자료 구조를 사용할 때는 반드시 다음을 고려해야 합니다.

- 모든 서브필드는 유효한 호스트 변수이어야 합니다.
- 모든 서브필드는 연속적이어야 합니다. 첫 번째 FROM 열은 1이어야 하고 TO와 FROM 열에는 중복이 있을 수 없습니다.
- 날짜 및 시간 형식과 호스트 구조 내의 날짜 및 시간 서브필드의 분리문자가 CRTSQLRPGI 명령에서의 DATFMT, DATSEP, TIMFMT 및 TIMSEP 매개변수와 같지 않을 경우 호스트 구조 배열은 사용될 수 없습니다.

블록 FETCH 및 블록 INSERT 이외의 모든 명령문에 대해 발생 자료 구조가 사용될 경우 현재의 발생이 사용됩니다. 블록 FETCH 및 블록 INSERT에 대해 발생은 1로 설정됩니다.

다음 예는 DEPARTMENT 표로부터 10개의 행을 검색하기 위해 DEPT로 불리는 호스트 구조 및 블록 FETCH문을 사용합니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
DDEPARTMENT          DS                      OCCURS(10)
D DEPTNO              01      03A
D DEPTNM              04      32A
D MGRNO               33      38A
D ADMRD               39      41A

DIND_ARRAY           DS                      OCCURS(10)
D INDS                4B 0 DIM(4)

...
C/EXEC SQL
      C+ DECLARE C1 CURSOR FOR
C+      SELECT *
C+      FROM   CORPDATA.DEPARTMENT
      C/END-EXEC

...

C/EXEC SQL
C+  FETCH C1 FOR 10 ROWS
C+  INTO :DEPARTMENT:IND_ARRAY
      C/END-EXEC
```

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 호스트 변수 선언

iSeries용 ILE RPG에는 LOB(대형 오브젝트)에 대한 SQL 자료 유형에 해당하는 변수가 없습니다. 이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQLTYPE 키워드를 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버에서 이 선언을 iSeries용 ILE RPG 언어 구조로 대체합니다. LOB 선언은 독립적이거나 자료 구조 내에 있습니다.

자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 호스트 변수』
- 123 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 위치 지정자』
- 124 페이지의 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 파일 참조 변수』

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 호스트 변수

BLOB 예

다음 선언을 사용할 경우:

```
D MYBLOB          S          SQLTYPE(BLOB:500)
```

다음과 같은 구조가 생성됩니다.

```
D MYBLOB          DS
D MYBLOB_LEN      10U
D MYBLOB_DATA     500A
```

CLOB 예

다음 선언을 사용할 경우:

```
D MYCLOB          S          SQLTYPE(CLOB:1000)
```

다음과 같은 구조가 생성됩니다.

```
D MYCLOB          DS
D MYCLOB_LEN      10U
D MYCLOB_DATA     1000A
```

DBCLOB 예

다음 선언을 사용할 경우:

```
D MYDBCLOB        S          SQLTYPE(DBCLOB:400)
```

다음과 같은 구조가 생성됩니다.

D MYDBCLOB	DS	
D MYDBCLOB_LEN		10U
D MYDBCLOB_DATA		400G

주:

1. BLOB, CLOB의 경우 $1 \leq \text{lob-length} \leq 32,766$
2. DBCLOB의 경우 $1 \leq \text{lob-length} \leq 16,383$
3. LOB 호스트 변수는 호스트 구조에서 선언될 수 있습니다.
4. LOB 호스트 변수는 호스트 구조 배열에서 허용되지 않습니다. 대신 LOB 위치 지정자가 사용됩니다.
5. 구조 배열에서 선언된 LOB 호스트 변수는 독립형 호스트 변수로서 사용될 수 없습니다.
6. SQLTYPE, BLOB, CLOB, DBCLOB는 대소문자를 혼합하여 사용할 수 있습니다.
7. SQLTYPE은 위치 44-80열 사이에 있어야 합니다.
8. LOB이 독립형 호스트 변수로서 선언되면 위치 24열은 'S'자를 포함하여야 하며 25열은 공백이어야 합니다.
9. 24열의 독립형 필드 인디케이터 'S'는 LOB가 호스트 구조에서 선언되면 생략됩니다.
10. LOB 호스트 변수는 초기화될 수 없습니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 위치 지정자

BLOB 위치 지정자 예

다음 선언을 사용할 경우:

D MYBLOB	S	SQLTYPE(BLOB_LOCATOR)
----------	---	-----------------------

다음과 같이 생성됩니다.

D MYBLOB	S	10U
----------	---	-----

CLOB 및 DBCLOB 위치 지정자의 구문은 비슷합니다.

주:

1. LOB 위치 지정자는 호스트 구조에서 선언될 수 있습니다.
2. SQLTYPE, BLOB_LOCATOR, CLOB_LOCATOR, DBCLOB_LOCATOR는 대소문자를 혼합하여 사용할 수 있습니다.
3. SQLTYPE은 위치 44-80열 사이에 있어야 합니다.
4. LOB 위치 지정자가 독립형 호스트 변수로서 선언되면 위치 24열은 'S'자를 포함하며 25열은 공백이어야 합니다.

5. 24열의 독립형 필드 인디케이터 'S'는 LOB 위치 지정자가 호스트 구조에서 선언되면 생략됩니다.
6. LOB 위치 지정자는 초기화될 수 없습니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 LOB 파일 참조 변수

CLOB 파일 참조 예

다음 선언을 사용할 경우:

```
D MY_FILE          S          SQLTYPE(CLOB_FILE)
```

다음과 같은 구조가 생성됩니다.

```
D MY_FILE          DS
D MY_FILE_NL              10U
D MY_FILE_DL              10U
D MY_FILE_FO              10U
D MY_FILE_NAME           255A
```

BLOB 및 DBCLOB 위치 지정자의 구문은 비슷합니다.

주:

1. LOB 파일 참조 변수는 호스트 구조에서 선언될 수 있습니다.
2. SQLTYPE, BLOB_FILE, CLOB_FILE, DBCLOB_FILE은 대소문자를 혼합하여 사용할 수 있습니다.
3. SQLTYPE은 위치 44-80열 사이에 있어야 합니다.
4. LOB 파일 참조가 독립형 호스트 변수로서 선언되면 위치 24열은 'S'자를 포함하며 25열은 공백이어야 합니다.
5. 24열의 독립형 필드 인디케이터 'S'는 LOB 파일 참조 변수가 호스트 구조에서 선언되면 생략됩니다.
6. LOB 파일 참조 변수는 초기화될 수 없습니다.

사전컴파일러는 다음의 파일 옵션 상수에 대한 선언을 생성합니다. 파일 참조 호스트 변수를 사용할 때 이러한 상수를 사용하여 xxx_FO 변수를 설정할 수 있습니다. 이러한 값에 대한 자세한 정보는 SQL 프로그래밍 개념의 LOB 파일 참조 변수를 참조하십시오.

- SQFRD(2)
- SQFCRT(8)
- SQFOVR(16)
- SQFAPP(32)

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 ROWID 변수

iSeries용 ILE RPG에는 ROWID의 SQL 자료 유형에 대응하는 변수가 없습니다. 이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQLTYPE 키워드를 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버에서 이 선언을 iSeries용 ILE RPG 언어 선언으로 대체합니다. ROWID 선언은 독립적이거나 자료 구조 내에 있습니다.

ROWID 예

다음 선언을 사용할 경우:

```
D MY_ROWID          S          SQLTYPE(ROWID)
```

다음과 같이 생성됩니다.

```
D MYROWID           S          40A  VARYING
```

주:

1. SQLTYPE, ROWID는 대소문자를 혼합하여 사용할 수 있습니다.
2. ROWID 호스트 변수는 호스트 구조에서 선언될 수 있습니다.
3. SQLTYPE은 위치 44 - 80열 사이에 있어야 합니다.
4. ROWID가 독립형 호스트 변수로서 선언되면 위치 24열은 'S'자를 포함하며 25열은 공백이어야 합니다.
5. 24열의 독립형 필드 인디케이터 'S'는 ROWID가 호스트 구조에서 선언되면 생략됩니다.
6. ROWID 호스트 변수는 초기화될 수 없습니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 외부 파일 설명 사용

SQL 사전컴파일러는 iSeries용 ILE RPG 컴파일러와 거의 같은 방식으로 iSeries용 ILE RPG 소스를 처리합니다. 즉, 사전컴파일러는 호스트 변수의 정의를 위해 /COPY문을 처리합니다. 다른 이름이 지정되면 외부 서술 파일에 대한 필드 정의의 사용을 지정하고 재명명합니다. 자료 구조의 외부 정의 형식을 사용하여 호스트 변수로 사용될 열명의 사본을 확보할 수 있습니다.

날짜 및 시간 필드 정의가 SQL 사전컴파일러에 의해 검색되고 처리되는 방법은 CRTSQLRPGI 명령의 OPTION 매개변수에서 *NOCVTDT 또는 *CVTDT 중 어느 것이 지정되는지에 따라 달라집니다. *NOCVTDT가 지정될 경우 날짜 및 시간 필드 정의는 형식과 분리문자를 포함하여 검색됩니다. *CVTDT가 지정될 경우 날짜 및 시간 정의 검색시 형식과 분리문자가 무시되고 사전컴파일러는 변수 선언이 문자 형식의 날짜/시간 호스트 변수인 것으로 가정합니다. *CVTDT는 iSeries용 RPG 사전컴파일러에 대한 호환성 옵션입니다.

다음 예에서 샘플 표 DEPARTMENT는 iSeries용 ILE RPG 프로그램에서 파일로 사용됩니다. SQL 사전컴파일러는 호스트 변수로 사용될 DEPARTMENT에 대한 필드(열) 정의를 검색합니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
FDEPARTMENTIP E DISK RENAME(ORIGREC:DEPTREC)
```

주: 파일에 대한 I/O 조작을 수행하기 위해 iSeries용 ILE RPG문을 사용하는 경우에만 사용자의 iSeries용 ILE RPG 프로그램에서 파일에 대한 F 스펙을 코딩하십시오. SQL문만을 사용하여 파일에 대한 I/O 조작을 수행하는 경우에는 외부 자료 구조를 사용하여 외부 정의를 포함시킬 수 있습니다.

다음 예에서는 샘플 표가 외부 자료 구조로서 지정되어 있습니다. SQL 사전컴파일러는 필드(열) 정의를 자료 구조의 서브필드로서 검색합니다. 서브필드명은 호스트 변수명으로 사용될 수 있으며, 자료 구조명 TDEPT는 호스트 구조명으로 사용될 수 있습니다. 이 예는 프로그램이 필요로 할 경우 필드명이 재명명될 수 있음을 나타냅니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
DTDEPT E DS EXTNAME(DEPARTMENT)
D DEPTN E EXTFLD(DEPTNAME)
D ADMRD E EXTFLD(ADMRDEPT)
```

GRAPHIC 또는 VARGRAPHIC 열에 UCS-2 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UCS-2 CCSID를 갖게 됩니다.

자세한 내용은 『SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 구조 배열에 대한 외부 파일 설명 고려사항』을 참조하십시오.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 호스트 구조 배열에 대한 외부 파일 설명 고려사항

장치 파일의 경우 INDARA가 지정되지 않고 파일이 인디케이터를 포함하면 선언 부분은 호스트 구조 배열로 사용되지 않습니다. 인디케이터 영역은 생성된 구조 내로 포함되며 기억영역이 분리되는 원인이 됩니다.

OPTION(*NOCVTDT)이 지정되고 파일 내의 날짜 및 시간 형식과 날짜 및 시간 필드 정의의 분리문자가 CRTSQLRPGI 명령에서의 DATFMT, DATSEP, TIMFMT 및 TIMSEP 매개변수와 같지 않을 경우 호스트 구조 배열은 사용될 수 없습니다.

다음 예에서 DEPARTMENT 표는 iSeries용 ILE RPG 프로그램에 포함되어 있으며 호스트 구조 배열을 선언하는 데 사용됩니다. 그런 후 블록 FETCH문은 호스트 구조 배열 내로 10개의 행을 검색하는 데 사용됩니다.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
DDEPARTMENT E DS OCCURS(10)
```

...

C/EXEC SQL

```

C+ DECLARE C1 CURSOR FOR
C+   SELECT *
C+   FROM CORPDATA.DEPARTMENT
C/END-EXEC

...

C/EXEC SQL
C+   FETCH C1 FOR 10 ROWS
C+   INTO :DEPARTMENT
C/END-EXEC

```

동등한 SQL 및 RPG 자료 유형 판별

사전컴파일러는 다음 표에 따라 호스트 변수의 기초 SQLTYPE 및 SQLLEN을 결정합니다. 호스트 변수가 인디케이터 변수로 표시되면 SQLTYPE은 기본 SQLTYPE에 1을 더한 값이 됩니다.

표 9. 일반 SQL 자료 유형에 맵핑되는 iSeries용 ILE RPG 선언

RPG 자료 유형	D 스펙 위치 40	D 스펙 위치 41, 42	기타 RPG 코딩	SQLTYPE 호스트 변수	SQLLEN 호스 트 변수	SQL 자료 유형
자료 구조(서브필드 없음)	공백	공백	길이 = n, n ≤ 32766	452	n	CHAR(n)
연산 결과 필드 (위치 69,70 = 공백)	n/a	n/a	길이 = n, n ≤ 32766(위치 59-63)	452	n	CHAR(n)
정의 스펙	A	공백	길이=n, n= 1 - 254. 44-80열의 VARYING.	448	n	VARCHAR(n)
정의 스펙	A	공백	길이=n, n > 254. 44-80 열의 VARYING.	456	n	VARCHAR(n)
정의 스펙	B	0	길이 ≤ 4	500	2	SMALLINT
정의 스펙	I	0	길이 = 5	500	2	SMALLINT
정의 스펙	B	0	길이 ≤ 9 및 ≥ 5	496	4	INTEGER
정의 스펙	I	0	길이 = 10	496	4	INTEGER
정의 스펙	I	0	길이 = 20	492	8	BIGINT
정의 스펙	B	1-4	길이 = 2	500	2	DECIMAL(4,s) s= 41, 42열
정의 스펙	B	1-9	길이 = 4	496	4	DECIMAL(9,s) s= 41, 42열
정의 스펙	P	0-30	길이 = n, n= 1-16	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s), p = n*2-1 및 s = pos 41, 42
정의 스펙	F	공백	길이 = 4	480	4	FLOAT(단정밀도)
정의 스펙	F	공백	길이 = 8	480	8	FLOAT(배정밀도)
서브필드가 아닌 정의 스펙	공백	0-30	길이 = n, n= 1-16	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s), p = n*2-1 및 s = pos 41, 42

표 9. 일반 SQL 자료 유형에 매핑되는 iSeries용 ILE RPG 선언 (계속)

RPG 자료 유형	D 스펙 위치 40	D 스펙 위치 41, 42	기타 RPG 코딩	SQLTYPE 호스트 변수	SQLLEN 호스 트 변수	SQL 자료 유형
입력 필드(위치 36 = P)	n/a	n/a	길이 = n, n= 1-16(위치 37-46)	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s), p = n*2-1 및 s = pos 47, 48
입력 필드(위치 36 = 공백 또는 S)	n/a	n/a	길이 = n, n= 1-30(위치 37-46)	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s), p = n 및 s = 위치 47, 48
입력 필드(위치 36 = B)	n/a	n/a	길이 = n, n= 2 또는 4(위치 37-46)	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s), n=2일 경우 p=4 또 는 n=4, s = 위치 47, 48일 경우 p=9
연산 결과 필드 (위치 69,70 ≠ 공백)	n/a	n/a	길이 = n, n= 1-30(위치 59-63)	484	바이트 1에는 p, 바이트 2에는 s	DECIMAL(p,s), p = n 및 s = 위치 64, 65
자료 구조 서브필 드	공백	0-30	길이 = n, n= 1-30	488	바이트 1에는 p, 바이트 2에는 s	NUMERIC(p,s), p = n 및 s = 위치 41, 42
정의 스펙	S	0-30	길이 = n, n= 1-30	488	바이트 1에는 p, 바이트 2에는 s	NUMERIC(p,s), p = n 및 s = 위치 41, 42
입력 필드(위치 36 = G)	n/a	n/a	길이 = n, n= 1-32766(위 치 37-46)	468	m	GRAPHIC(m), m = n/2 m = (TO-FROM-1)/2
정의 스펙	G	공백	길이 = n, n= 1 - 127. 44-80 열의 VARYING.	464	n	VARGRAPHIC(n)
정의 스펙	C	공백	길이 = n, n<16383	468	n	CCSID 13488 와 GRAPHIC(n)
정의 스펙	G	공백	길이 = n, n > 127. 44-80 열의 VARYING.	472	n	VARGRAPHIC(n)
정의 스펙	D	공백	길이 = n, n= 6, 8 또는 10	384	n	DATE(위치 44-80 에서 지정된 DATFMT, DATSEP)
입력 필드(위치 36 = D)	n/a	n/a	길이 = n, n= 6, 8 또는 10(위치 37-46)	384	n	DATE(위치 31-34 에서 지정된 형식)
정의 스펙	T	공백	길이 = n, n= 8	388	n	TIME(위치 44-80 에서 지정된 TIMFMT, TIMSEP)
입력 필드(위치 36 = T)	n/a	n/a	길이 = n, n= 8(위치 37-46)	388	n	TIME(위치 31-34 에서 지정된 형식)
정의 스펙	Z	공백	길이 = n, n= 26	392	n	TIMESTAMP
입력 필드(위치 36 = Z)	n/a	n/a	길이 = n, n= 26(위치 37-46)	392	n	TIMESTAMP

주:

1. 명시적으로 달리 언급되지 않는 한 첫 번째 열에서의 용어 "정의 스펙"에는 자료 구조 서브필드가 포함됩니다.
2. 정의 스펙에서 2진 필드(위치 40에서의 B)의 길이는 다음에 의해 결정됩니다.
 - FROM(26-32열)이 공백이 아니면 길이 = TO-FROM+1.
 - FROM(26-32열)이 공백이면 33-39열 < 5인 경우 길이 = 2, 33-39열 > 4인 경우 길이 = 4.
3. SQL은 CRTSQLRPGI 명령에서 지정된 DATE/TIME 형식을 사용하여 날짜/시간 서브필드를 작성합니다. 호스트 변수와 SQL 생성 서브필드 사이에서 맵핑이 수행될 때 호스트 변수 DATE/TIME 형식으로의 변환이 발생합니다.

다음 표는 제공된 SQL 자료 유형과 같은 RPG 자료 유형 판별에 사용할 수 있습니다.

표 10. 일반 RPG 선언 부분에 맵핑되는 SQL 자료 유형

SQL 자료 유형	RPG 자료 유형	주
SMALLINT	정의 스펙. 위치 40에서 I, 길이는 위치 42에서 5와 0. 또는 정의 스펙. 위치 40에서 B, 길이는 위치 42에서 ≤ 4와 0.	
INTEGER	정의 스펙. 위치 40에서 I, 길이는 위치 42에서 10과 0. 또는 정의 스펙. 위치 40에서 B, 길이는 위치 42에서 ≤ 9와 ≥ 5와 0.	
BIGINT	정의 스펙. 위치 40에서 I, 길이는 위치 42에서 20과 0.	
DECIMAL	정의 스펙. 위치 40에서 P 또는 비서브필드에 대한 위치 40에서 공백, 위치 41, 42에는 0-30. 또는 비정의 스펙에 숫자로 정의됨.	최대 길이 16(정밀도 30) 및 최대 소수 자리 수 30.
NUMERIC	정의 스펙. 위치 40에서 공백 또는 서브필드에 대한 위치 40에서 S, 위치 41, 42에는 0-30.	최대 길이 30(정밀도 30) 및 최대 소수 자리 수 30.
FLOAT(단정밀도)	정의 스펙. 위치 40에서 F, 길이는 4.	
FLOAT(배정밀도)	정의 스펙. 위치 40에서 F, 길이는 8.	

표 10. 일반 RPG 선언 부분에 맵핑되는 SQL 자료 유형 (계속)

SQL 자료 유형	RPG 자료 유형	주
CHAR(n)	정의 스펙. 위치 40에서 A 또는 위치 40에서 공백 및 위치 41, 42에서 공백. 또는 소수 자리 없이 정의된 입력 필드. 또는 소수 자리 없이 정의된 연산 결과 필드.	n은 1-32766 사이일 수 있습니다.
CHAR(n)	자료 구조에 서브필드가 없는 자료 구조명.	n은 1-32766 사이일 수 있습니다.
VARCHAR(n)	정의 스펙. 40 위치의 A 또는 공백과 44-80 위치의 VARYING.	n은 1 - 32740 사이일 수 있습니다.
BLOB	지원되지 않습니다.	BLOB를 선언하려면 SQLTYPE 키워드를 사용하십시오.
CLOB	지원되지 않습니다.	CLOB를 선언하려면 SQLTYPE 키워드를 사용하십시오.
GRAPHIC(n)	정의 스펙. 위치 40이 G. 또는 위치 36이 G로 정의된 입력 필드.	n은 1-16383 사이일 수 있습니다.
VARGRAPHIC(n)	정의 스펙. 40 위치의 G와 44-80 위치의 VARYING.	n은 1 - 16370 사이일 수 있습니다.
DBCLOB	지원되지 않습니다.	DBCLOB를 선언하려면 SQLTYPE 키워드를 사용하십시오.
DATE	문자 필드 또는 위치 40이 D로 정의된 스펙 또는 위치 36이 D로 정의된 입력 필드	형식이 *USA, *JIS *EUR 또는 *ISO이면 길이는 최소한 10이어야 합니다. 형식이 *YMD, *DMY 또는 *MDY이면 길이는 최소한 8이어야 합니다. 형식이 *JUL이면 길이는 최소한 6이어야 합니다.
TIME	문자 필드 또는 위치 40이 T로 정의된 스펙 또는 위치 36이 T로 정의된 필드	길이는 최소한 6이어야 합니다. 초를 포함하려면 최소한 8이 되어야 합니다.
TIMESTAMP	문자 필드 또는 위치 40이 Z로 정의된 스펙 또는 위치 36이 Z로 정의된 입력 필드	길이는 최소한 19이어야 합니다. 마이크로 초를 포함하려면 최소한 26이 되어야 합니다. 길이가 25 이하일 경우 마이크로초 부분에서 잘립니다.
DATALINK	지원되지 않습니다.	
ROWID	지원되지 않습니다.	ROWID를 선언하려면 SQLTYPE 키워드를 사용하십시오.

자세한 내용은 『iSeries용 ILE RPG 변수 선언 및 사용에 대한 주』를 참조하십시오.

iSeries용 ILE RPG 변수 선언 및 사용에 대한 주

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 지정 규칙

iSeries용 ILE RPG는 정밀도와 소수 자릿수를 모든 숫자 유형과 연관시킵니다. iSeries용 ILE RPG는 자료가 팩 형식이라고 가정하고 숫자 연산을 정의합니다. 즉, 2진 변수가 포함된 연산에는 연산 수행 전에 팩된 형식으로(필요하면 다시 2진수로) 내재 변환을 포함시킵니다. SQL 연산 수행시 자료는 내포된 소수점에 정렬됩니다.

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 인디케이터 변수 사용

인디케이터 변수는 길이가 5(2바이트) 이하인 2진 필드입니다.

인디케이터 배열은 변수 요소 길이 4, 0을 선언하고 정의 스펙에 DIM을 지정함으로써 정의될 수 있습니다.

검색시 연관 호스트 변수에 널(null)이 할당되었는지 여부를 나타내기 위해 인디케이터 변수를 사용합니다. 음의 인디케이터 변수는 열에 할당할 때 널이 할당되어야 함을 나타내기 위해 사용됩니다.

자세한 정보는 SQL 참조서의 인디케이터 변수 주제를 참조하십시오.

인디케이터 변수는 호스트 변수와 동일한 방법으로 선언되며 프로그래머는 적절한 방식으로 그 두 가지 선언 부분을 혼합할 수 있습니다.

ILE RPG에서 인디케이터 변수 사용 예는 『예: SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 인디케이터 변수 사용』을 참조하십시오.

예: SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 인디케이터 변수 사용

다음 명령문을 보십시오.

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
C/EXEC SQL FETCH CLS_CURSOR INTO :CLSCD,
C+                               :DAY :DAYIND,
C+                               :BGN :BGNIND,
C+                               :END :ENDIND
      C/END-EXEC
```

이 경우 다음과 같이 변수를 선언할 수 있습니다.

```

*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
D CLSCD          S          7
D DAY            S          2B 0
D DAYIND        S          2B 0
D BGN           S          8A
D BGNIND        S          2B 0
D END           S          8
D ENDIND        S          2B 0

```

SQL에서 사용하는 iSeries용 ILE RPG 어플리케이션에서 복수 행 영역 페치에 대한 SQLDA 예

```

*...1....+....2....+....3....+....4....+....5....+....6....+....7....+....8.
C/EXEC SQL INCLUDE SQLDA
C/END-EXEC
DDEPARTMENT      DS          OCCURS(10)
D DEPTNO         01         03A
D DEPTNM        04         32A
D MGRNO         33         38A
D ADMRD         39         41A
...

DIND_ARRAY       DS          OCCURS(10)
D INDS           4B 0 DIM(4)
...
C* setup number of sqlda entries and length of the sqlda
C          eval      sqld = 4
C          eval      sqln = 4
C          eval      sqldabc = 336
C*
C* setup the first entry in the sqlda
C*
C          eval      sqltype = 453
C          eval      sqlllen = 3
C          eval      sql_var(1) = sqlvar
C*
C* setup the second entry in the sqlda
C*
C          eval      sqltype = 453
C          eval      sqlllen = 29
C          eval      sql_var(2) = sqlvar
...
C*
C* setup the forth entry in the sqlda
C*
C          eval      sqltype = 453
C          eval      sqlllen = 3
C          eval      sql_var(4) = sqlvar
...
C/EXEC SQL
C+ DECLARE C1 FOR
C+   SELECT *
C+   FROM CORPDATA.DEPARTMENT
C/END-EXEC
...

```

```

C/EXEC SQL
C+   FETCH C1 FOR 10 ROWS
C+   USING DESCRIPTOR :SQLDA
C+   INTO :DEPARTMENT:IND_ARRAY
C/END-EXEC

```

SQL을 사용하는 iSeries용 ILE RPG 어플리케이션에서 동적 SQL 예

```

D*****
D* Declare program variables.                *
D* STMT initialized to the                   *
D* listed SQL statement.                    *
D*****
D EMPNUM          S           6A
D NAME            S           15A
D STMT            S           500A  INZ('SELECT LASTNAME      -
D                                     FROM CORPDATA.EMPLOYEE WHERE -
D                                     EMPNO = ?')

...

C*****
C* Prepare STMT as initialized in declare section *
C*****
C/EXEC SQL
C+ PREPARE S1 FROM :STMT
C/END-EXEC
C*

C*****
C* Declare Cursor for STMT                    *
C*****
C/EXEC SQL
C+ DECLARE C1 CURSOR FOR S1
C/END-EXEC
C*

C*****
C* Assign employee number to use in select statement *
C*****
C          eval      EMPNUM = '000110'

C*****
C* Open Cursor                                *
C*****
C/EXEC SQL
C+ OPEN C1 USING :EMPNUM
C/END-EXEC
C*

C*****
C* Fetch record and put value of              *
C* LASTNAME into NAME                        *
C*****
C/EXEC SQL
C+ FETCH C1 INTO :NAME
C/END-EXEC

...

```

```
C*****  
C* Program processes NAME here *  
C*****  
...  
C*****  
C* Close cursor *  
C*****  
C/EXEC SQL  
C+ CLOSE C1  
C/END-EXEC
```


SQLERRMC	오류 및 경고 메시지 토큰
SQLERRP	제품 코드 및 오류가 있을 경우 오류를 리턴한 모듈명
SQLERRD.n	진단 정보가 들어 있는 6개의 변수(<i>n</i> 은 1 - 6 사이의 숫자).
SQLWARN.n	경고 플래그가 들어 있는 11개의 변수(<i>n</i> 은 0 - 10 사이의 숫자).
SQLSTATE	대체 SQL 리턴 코드

SQLCA에 대한 자세한 정보는 *SQL 참조서의 SQL 통신 영역을 참조하십시오.*

REXX 어플리케이션에서 SQL 설명자 영역 사용

다음 명령문은 SQLDA를 필요로 합니다.

```
EXECUTE...USING DESCRIPTOR descriptor-name
FETCH...USING DESCRIPTOR descriptor-name
OPEN...USING DESCRIPTOR descriptor-name
CALL...USING DESCRIPTOR descriptor-name
DESCRIBE statement-name INTO descriptor-name
DESCRIBE TABLE host-variable INTO descriptor-name
```

SQLCA와는 달리, 프로시저어에 두 개 이상의 SQLDA가 있을 수 있으며 SQLDA는 유효한 모든 이름을 가질 수 있습니다. 각각의 SQLDA는 공통 어간의 REXX 변수 세트 구성되며 여기서 어간 이름은 해당 SQL문의 *descriptor-name*입니다. 이는 단순 어간이어야 합니다. 즉, 어간 자체에 마침표가 들어 있어서는 안 됩니다. SQL/REXX 인터페이스는 각 고유 설명자명에 대해 자동으로 SQLDA의 필드를 제공합니다. INCLUDE SQLDA문은 필요없으며 허용되지 않습니다.

SQL/REXX 인터페이스는 일반적인 SQL 사용과 똑같은 방법으로 SQLDA를 사용합니다. 그러나 SQL/REXX 인터페이스는 연속 자료 영역보다는 분리된 변수에 SQLDA의 필드를 유지보수합니다.

SQLDA에 대한 자세한 정보는 *SQL 참조서의 SQL 설명자 영역을 참조하십시오.*

다음 변수는 DESCRIBE, DESCRIBE TABLE 또는 PREPARE INTO문 뒤의 어플리케이션에 리턴됩니다.

```
stem.n.SQLNAME
    결과표의 n번째 열명
```

다음과 같은 변수들이 EXECUTE...USING DESCRIPTOR, OPEN...USING DESCRIPTOR, CALL...USING DESCRIPTOR 또는 FETCH...USING DESCRIPTOR

문 앞의 어플리케이션에 의해 제공되어야 합니다. 이 변수들은 DESCRIBE, DESCRIBE TABLE 또는 PREPARE INTO문 뒤의 어플리케이션에 리턴됩니다.

stem.SQLD

SQLDA에 실제로 들어 있는 변수 요소의 수

stem.n.SQLTYPE

n번째 요소의 자료 유형(예를 들어 첫 번째 요소는 어간 stem.1.SQLTYPE에 있음)을 표시하는 정수

다음 자료 유형은 허용되지 않습니다.

400/401	널(null) 종료 그래픽 스트링
404/405	BLOB 호스트 변수
408/409	CLOB 호스트 변수
412/413	DBCLOB 호스트 변수
460/461	널(null) 종료 문자 스트링
476/477	PASCAL L-스트링
496/497	큰 정수(소수 자릿수가 0보다 클 때)
500/501	작은 정수(소수 자릿수가 0보다 클 때)
504/505	DISPLAY SIGN LEADING SEPARATE
904/905	ROWID
916/917	BLOB 파일 참조 변수
920/921	CLOB 파일 참조 변수
924/925	DBCLOB 파일 참조 변수
960/961	BLOB 위치 지정자
964/965	CLOB 위치 지정자
968/969	DBCLOB 위치 지정자

stem.n.SQLLEN

SQLTYPE이 DECIMAL 또는 NUMERIC 자료 유형을 표시하지 않을 경우 stem.n.SQLDATA에 들어 있는 자료의 최대 길이

stem.n.SQLLEN.SQLPRECISION

자료 유형이 DECIMAL 또는 NUMERIC일 경우 여기에는 숫자의 정밀도가 들어 있습니다.

stem.n.SQLLEN.SQLSCALE

유형이 DECIMAL 또는 NUMERIC일 경우 여기에는 숫자의 소수 자릿수가 들어 있습니다.

stem.n.SQLCCSID

자료의 n번째 열의 CCSID.

다음과 같은 변수들이 EXECUTE...USING DESCRIPTOR 또는 OPEN...USING DESCRIPTOR문 앞의 어플리케이션에 의해 제공되어야 하며 FETCH...USING DESCRIPTOR문 뒤의 어플리케이션으로 리턴됩니다. 이 변수들은 DESCRIBE, DESCRIBE TABLE 또는 PREPARE INTO문 뒤에 사용되지 않습니다.

stem.n.SQLDATA

어플리케이션에 의해 제공된 입력값 또는 SQL에 의해 페치된 출력값이 들어 있습니다.

이 값은 SQLTYPE, SQLLEN, SQLPRECISION 및 SQLSCALE에서 지정된 속성으로 변환됩니다.

stem.n.SQLIND

입력 또는 출력값이 널일 경우 이는 음수입니다.

REXX 어플리케이션에서 SQL문 삽입

SQL문은 REXX 명령이 위치할 수 있는 곳이면 어디에서나 위치할 수 있습니다.

REXX 프로시저의 각 SQL문은 반드시 EXECSQL(대소문자의 조합으로)로 시작하고 뒤에 다음 중 하나가 와야 합니다.

- 작은 따옴표 또는 큰 따옴표로 묶인 SQL문
- 명령문이 들어 있는 REXX 변수. SQL문이 들어 있을 때 콜론은 REXX에 선행해서는 안됩니다.

예를 들면 다음과 같습니다.

```
EXECSQL "COMMIT"
```

는 다음과 같습니다.

```
rexvar = "COMMIT"  
EXECSQL rexvar
```

명령은 일반적인 REXX 규칙을 따릅니다. 예를 들어 둘 이상의 REXX문이 들어갈 수 있는 하나의 행을 허용하려면 선택적으로 세미콜론(;)이 뒤에 옵니다. REXX는 또한 작은 따옴표 사이에 포함될 명령을 허용합니다. 예를 들면 다음과 같습니다.

```
'EXECSQL COMMIT'
```


SQL/REXX 인터페이스는 다음 SQL문을 지원합니다.

ALTER TABLE	EXECUTE IMMEDIATE
CALL ³	FETCH ²
CLOSE	GRANT
COMMENT ON	INSERT ^{2, 3}
COMMIT	LABEL ON
CREATE ALIAS	LOCK TABLE
CREATE DISTINCT TYPE	OPEN
CREATE FUNCTION	PREPARE
CREATE INDEX	RELEASE SAVEPOINT
CREATE PROCEDURE	RENAME
CREATE SCHEMA	REVOKE
CREATE TABLE	ROLLBACK
CREATE TRIGGER	SAVEPOINT
CREATE VIEW	SET OPTION ⁴
DECLARE CURSOR ³	SET PATH
DECLARE GLOBAL TEMPORARY TABLE	SET SCHEMA
DELETE ³	SET TRANSACTION
DESCRIBE	SET 변수 ³
DESCRIBE TABLE	UPDATE ³
DROP	VALUES INTO ³
EXECUTE	

다음 SQL문은 SQL/REXX 인터페이스에 의해 지원되지 않습니다.

BEGIN DECLARE SECTION	FREE LOCATOR
CONNECT	HOLD LOCATOR
DECLARE PROCEDURE	INCLUDE
DECLARE STATEMENT	RELEASE
DECLARE VARIABLE	SELECT INTO
DISCONNECT	SET CONNECTION
END DECLARE SECTION	SET RESULT SETS
	WHENEVER ⁵

세부사항은 다음 절을 참조하십시오.

- 140 페이지의 『SQL을 사용하는 REXX 어플리케이션에서 주식』
- 140 페이지의 『SQL을 사용하는 REXX 어플리케이션에서 SQL문의 연속』
- 140 페이지의 『SQL을 사용하는 REXX 어플리케이션에서 포함하는 코드』
- 140 페이지의 『SQL을 사용하는 REXX 어플리케이션에서 여백』
- 140 페이지의 『SQL을 사용하는 REXX 어플리케이션에서 이름』

2. 이 명령문의 블록화 양식은 지원되지 않습니다.

3. 이들 명령문은 호스트 변수가 들어 있는 경우 직접 실행될 수 없으며 PREPAREdml 오브젝트인 경우에만 EXECUTE입니다.

4. SET OPTION문은 SQL 문을 수행하는 데 사용된 옵션 처리의 일부를 변경하기 위해 REXX 프로시저에서 사용될 수 있습니다. 이 옵션에는 제약 제어 레벨 및 날짜 형식이 들어 있습니다. SET OPTION문에 대한 자세한 정보는 SQL 참조서를 참조하십시오.

5. 자세한 사항은 140 페이지의 『SQL을 사용하는 REXX 어플리케이션에서 오류 및 경고 처리』를 참조하십시오.

- 『SQL을 사용하는 REXX 어플리케이션에서 널(null)』
- 『SQL을 사용하는 REXX 어플리케이션에서 명령문 레이블』
- 『SQL을 사용하는 REXX 어플리케이션에서 오류 및 경고 처리』

SQL을 사용하는 REXX 어플리케이션에서 주석

SQL 주석(--) 또는 REXX 주석은 모두 SQL문을 표현하는 스트링에서는 허용되지 않습니다.

SQL을 사용하는 REXX 어플리케이션에서 SQL문의 연속

SQL문이 들어 있는 스트링은 여러 행에서 표준 REXX 사용에 따라 쉽표나 연결 연산자에 의해 분리된 여러 개의 스트링으로 분할될 수 있습니다.

SQL을 사용하는 REXX 어플리케이션에서 포함하는 코드

다른 호스트 언어와는 달리 외부에서 정의된 명령문을 포함하기 위한 지원은 제공되지 않습니다.

SQL을 사용하는 REXX 어플리케이션에서 여백

SQL/REXX 인터페이스를 위한 특별한 여백 규칙은 없습니다.

SQL을 사용하는 REXX 어플리케이션에서 이름

마침표(.)로 끝나는 유효한 REXX 이름은 호스트 변수에 사용될 수 있습니다. 이름은 64개 이하의 문자이어야 합니다.

변수명은 'SQL', 'RDI', 'DSN', 'RXSQL' 또는 'QRW' 문자로 시작되어서는 안 됩니다.

SQL을 사용하는 REXX 어플리케이션에서 널(null)

용어 널(null)이 REXX와 SQL 모두에서 사용되기는 하지만 이 용어는 두 언어에서 서로 다른 의미를 가집니다. REXX는 널 스트링(길이 0인 스트링)과 널 절(공백과 주석만으로 구성된 절)을 가집니다. SQL 널은 모든 비 널과 구별되는 특수값으로서 값(널이 아님)이 없음을 나타냅니다.

SQL을 사용하는 REXX 어플리케이션에서 명령문 레이블

REXX 명령문은 일반적인 경우에서와 같이 레이블 처리 수 있습니다.

SQL을 사용하는 REXX 어플리케이션에서 오류 및 경고 처리

WHENEVER문은 SQL/REXX 인터페이스에 의해 지원되지 않습니다. 그 대신, 다음 중 어느 것이나 사용할 수 있습니다.

- 데이터베이스 관리자에 의해 발행된(SQL/REXX 인터페이스에 의해 발행된 것에는 해당되지 않음) 오류 및 경고 조건을 검출하기 위한 각 SQL문 다음의 REXX SQLCODE 또는 SQLSTATE 변수 테스트.
- 오류 및 경고 조건을 검출하기 위한 각 SQL문 다음의 REXX RC 변수 테스트. 각 EXECSQL 명령의 사용으로 RC 변수가 다음과 같이 설정됩니다.

0 명령문이 완료됨
+10 SQL 경고 발생
-10 SQL 오류 발생
-100 SQL/REXX 인터페이스 오류 발생

이는 데이터베이스 관리자 또는 SQL/REXX 인터페이스에 의해 발행된 오류 및 경고를 검출하는 데 사용될 수 있습니다.

- SIGNAL ON ERROR 및 SIGNAL ON FAILURE 기능은 오류(음의 RC 값)를 검출하는 데 사용될 수 있으나 경고에는 사용될 수 없습니다.

SQL을 사용하는 REXX 어플리케이션에서 호스트 변수 사용

REXX는 변수 선언을 위해 제공되지 않습니다. LOB 및 ROWID 호스트 변수는 REXX에서 지원되지 않습니다. 새로운 변수는 지정문에서의 외양에 의해 인식됩니다. 따라서 선언절이 없으며 BEGIN DECLARE SECTION 및 END DECLARE SECTION문이 제공되지 않습니다.

SQL문 내의 모든 호스트 변수 앞에는 콜론(:)이 와야 합니다.

SQL/REXX 인터페이스는 명령문을 데이터베이스 관리자로 전달하기 전에 복합 변수에서 대체를 수행합니다. 예를 들면 다음과 같습니다.

```
a = 1
b = 2
EXECSQL 'OPEN c1 USING :x.a.b'
```

이것은 SQL로 전달되는 x.1.2의 내용을 만들어냅니다.

자세한 내용은 다음 절을 참조하십시오.

- 142 페이지의 『SQL을 사용하는 REXX 어플리케이션에서 입력 호스트 변수의 자료 유형 판별』
- 143 페이지의 『SQL을 사용하는 REXX 어플리케이션에서 출력 호스트 변수의 형식』
- 143 페이지의 『SQL을 사용하는 REXX 어플리케이션에서 REXX 변환 회피』

SQL을 사용하는 REXX 어플리케이션에서 입력 호스트 변수의 자료 유형 판별

REXX의 모든 자료는 스트링 양식으로 되어 있습니다. 입력 호스트 변수의 자료 유형 (즉, EXECUTE 또는 OPEN문의 'USING 호스트 변수'절에서 사용된 호스트 변수)은 표 11에 따라 변수 내용에서 수행시 데이터베이스 관리자에 의해 추정됩니다.

이 규칙은 숫자, 문자 또는 그래픽 값을 정의합니다. 숫자 값은 모든 유형의 숫자열에 대한 입력으로서 사용될 수 있습니다. 문자 값은 모든 유형의 문자열 또는 날짜, 시간이나 시간소인 열에 대한 입력으로서 사용될 수 있습니다. 그래픽 값은 모든 유형의 그래픽 열에 대한 입력으로서 사용될 수 있습니다.

표 11. REXX 내에 있는 호스트 변수의 자료 유형 판별

호스트 변수 내용	가정된 자료 유형	SQL 유형 코드	SQL 유형 설명
정의되지 않은 변수	값이 할당되지 않은 변수	없음	유효하지 않은 자료가 검출되었습니다.
두 개의 분리문자를 제거한 후 길이가 n인, 선행 및 후미 작은 따옴표(')나 큰 따옴표(")가 있는 스트링. 또는 선행 X 또는 작은 따옴표(')나 큰 따옴표(") 및 후미 작은 따옴표(')나 큰 따옴표(")가 붙는 x가 있는 스트링. 이 스트링은 X나 x 및 두 개의 분리문자를 제거한 후 길이 2n을 가집니다. 나머지 각 쌍의 문자는 한 문자의 16진 표시입니다. 또는 길이 n의 스트링은 이 표에서의 다른 규칙에 의해 문자, 숫자 또는 그래픽으로 인식될 수 없습니다.	가변 길이 문자 스트링	448/449	VARCHAR(n)
다음과 같은 스트링이 선행되는 작은 따옴표(')나 큰 따옴표(")로 묶인 스트링. ⁶ • G, g, N 또는 n으로 시작되는 스트링. 그 다음에는 작은 따옴표나 큰 따옴표 및 SO (x'0E')이 옵니다. 이는 각각 2개의 문자 길이를 갖는 n의 그래픽 문자가 뒤에 옵니다. 스트링은 한 글/한자 끝 제어 문자(X'0F') 및 작은 따옴표나 큰 따옴표(스트링이 시작될 때 붙는 모든 것)로 종료되어야 합니다. • 선행 GX, Gx, gX 또는 gx로 시작되고 작은 따옴표나 큰 따옴표 및 한글/한자 시작 제어 문자(x'0E')가 붙는 스트링. 이는 각각 2개의 문자 길이를 갖는 n의 그래픽 문자가 뒤에 옵니다. 스트링은 한글/한자 끝 제어 문자(X'0F') 및 작은 따옴표나 큰 따옴표(스트링이 시작될 때 붙는 모든 것)로 종료되어야 합니다. 이 스트링은 GX 및 분리문자를 제거한 후 4n의 길이를 가집니다. 각각의 나머지 4자의 그룹은 단일 그래픽 문자의 16진 표시입니다.	가변 길이 그래픽 스트링	464/465	VARGRAPHIC(n)
과학 또는 공학에서의 표기법으로 된 숫자(즉, 'E' 또는 'e', 선행 부동 소수점적인 덧셈이나 뺄셈 부호 및 일련의 숫자가 바로 뒤에 오는). 앞에 덧셈 또는 뺄셈 부호를 가질 수 있습니다.		480/481	FLOAT

표 11. REXX 내에 있는 호스트 변수의 자료 유형 판별 (계속)

호스트 변수 내용	가정된 자료 유형	SQL 유형 코드	SQL 유형 설명
소수점을 포함하고 있으나 지수는 없는 숫자 또는 소수점 또는 지수를 포함하고 있지 않으며 2147483647 이상이거나 -2147483647 이하인 숫자. 앞에 덧셈 또는 뺄셈 부호를 가질 수 있습니다. m은 숫자에서 총 자릿수입니다. n은 소수점의 왼쪽에 있는 자릿수입니다(있는 경우).	팩 십진수	484/485	DECIMAL(m, n)
소수점과 지수 모두가 없는 숫자. 앞에 덧셈 또는 뺄셈 부호를 가질 수 있습니다.	부호가 있는 정수	496/497	INTEGER

SQL을 사용하는 REXX 어플리케이션에서 출력 호스트 변수의 형식

출력 호스트 변수를 판별할 필요는 없습니다(즉, FETCH문의 'INTO 호스트 변수'절에
서 사용되는 호스트 변수). 출력 값은 다음과 같이 호스트 변수에 할당됩니다.

- 문자값은 선행 및 후미 작은 따옴표 없이 할당됩니다.
- 그래픽 값은 선행 G나 작은 따옴표 없이, 후미 작은 따옴표 없이 그리고 한글/한자
제어 문자(SO/SI) 없이 할당됩니다.
- 숫자값은 스트링으로 변환됩니다.
- 정수값은 앞에 어떤 0 값도 갖지 않습니다. 음의 값은 앞에 뺄셈 부호를 가집니다.
- 10진값은 정밀도와 소수 자릿수에 따라 앞뒤에 0를 갖습니다. 음의 값은 앞에 뺄셈
부호를 가집니다. 양의 값은 앞에 덧셈 부호를 갖지 않습니다.
- 부동 소수점 값은 소수점 왼쪽에 하나의 자릿수가 있는 과학 표기법입니다. 'E'는 대
문자로 되어 있습니다.

SQL을 사용하는 REXX 어플리케이션에서 REXX 변환 회피

스트링이 숫자로 변환되지 않도록 하거나 그래픽 유형으로 인식되도록 하려면 스트링을
"'"으로 묶어야 합니다. 단순히 스트링을 작은 따옴표로 묶는 것은 효과가 없습니다. 예
를 들면 다음과 같습니다.

```
stringvar = '100'
```

이는 REXX로 하여금 변수 *stringvar*를 문자 스트링 100(작은 따옴표 포함 안함)으로
변환하도록 합니다. 즉, SQL/REXX 인터페이스에 의해 숫자 100으로 평가되며 이 상
태에서 SQL로 전달됩니다.

한편,

```
stringvar = "'100'"
```

6. 선행 작은 따옴표 바로 다음에 오는 바이트는 X'0E' SO이고 후미 작은 따옴표 바로 앞에 오는 바이트는 X'0F' SI입니다.

이 경우에는 REXX로 하여금 변수 *stringvar*를 문자 스트링 '100'(작은 따옴표 포함)으로 설정하도록 합니다. 이는 SQL/REXX 인터페이스에 의해 스트링 100으로 평가되며 이 상태에서 SQL로 전달됩니다.

SQL을 사용하는 REXX 어플리케이션에서 인디케이터 변수 사용

인디케이터 변수는 정수입니다. 검색시, 인디케이터 변수는 연관된 호스트 변수가 널(null) 값으로 할당되었는지를 나타내는데 사용됩니다. 음의 인디케이터 변수는 열에 할당할 때 널이 할당되어야 함을 나타내기 위해 사용됩니다.

다른 언어와는 달리 연관된 인디케이터 변수에 음의 값이 들어 있을 경우 유효한 값은 호스트 변수에서 지정되어야 합니다.

자세한 정보는 SQL 참조서의 인디케이터 변수 주제를 참조하십시오.

제 8 장 SQL문을 사용하는 프로그램 준비 및 실행

이 장에서는 어플리케이션 프로그램을 준비하고 실행하기 위한 일부 TASK들을 설명합니다. 자세한 내용은 다음 절을 참조하십시오.

- 『SQL 사전컴파일러의 기본 처리』
- 153 페이지의 『BILE SQL 사전컴파일러 명령』
- 154 페이지의 『ILE SQL 사전컴파일러 명령』
- 156 페이지의 『SQL을 사용하는 어플리케이션에서 컴파일 오류 해석』
- 157 페이지의 『SQL을 사용하는 어플리케이션 바인딩』
- 159 페이지의 『SQL 사전컴파일러 옵션 표시』
- 159 페이지의 『삽입된 SQL을 사용하는 프로그램 실행』

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

SQL 사전컴파일러의 기본 처리

삽입된 SQL문을 수행하기 전에 그것이 들어 있는 어플리케이션 프로그램을 사전컴파일 및 컴파일해야 합니다.

주: REXX 프로시저 내의 SQL문은 사전컴파일 및 컴파일되지 않습니다.

그러한 프로그램의 사전컴파일링은 SQL 사전컴파일러에 의해 수행됩니다. SQL 사전컴파일러는 어플리케이션 프로그램 소스의 각 명령문을 검색하고 다음을 수행합니다.

- SQL문과 호스트 변수명의 정의를 찾습니다. 변수명과 정의는 SQL문을 검증하는 데 사용됩니다. SQL 사전컴파일러가 오류가 발생했는지를 알아보기 위한 처리를 완료한 후 사용자는 리스트를 체크할 수 있습니다.
- 각각의 SQL문이 유효하고 구문 오류가 없는지 확인합니다. 유효성 검사 프로시저는 발생하는 모든 오류를 정정할 때 도움을 주는 출력 리스트에 오류 메시지를 제공합니다.
- 데이터베이스의 설명을 사용하여 SQL문의 유효성을 확인합니다. 사전컴파일시 SQL문은 유효한 표명, 뷰명 및 열명에 대해 체크됩니다. 지정된 표나 뷰가 존재하지 않거나 사전컴파일이나 컴파일시 표나 뷰에 대해 권한이 없으면 유효성 검사는 수행시에 이루어집니다. 수행시에 표나 뷰가 존재하지 않으면 오류가 발생합니다.

주:

1. 대체는 외부 정의를 검색할 때 처리됩니다. 자세한 내용은 데이터베이스 프로그래밍 책 및 파일 관리 책을 참조하십시오.

2. 사용자가 SQL문의 유효성을 검사하기 위해서는 SQL문에서 참조된 표나 뷰에 대한 일부 권한(최소한 *OBJOPR)이 필요합니다. SQL문을 처리하는 데 필요한 실제 권한은 수행시 체크됩니다. SQL문에 대한 자세한 정보는 SQL 참조서를 참조하십시오.
 3. RDB 매개변수가 CRTSQLxxx 명령에 지정될 때 사전컴파일러는 표 및 뷰 설명을 구하기 위하여 지정된 관계형 데이터베이스에 액세스합니다.
- 호스트 언어로 컴파일할 SQL문을 준비합니다. 대부분의 SQL문에 대해 SQL 사전컴파일러는 주석 및 CALL문을 SQL 인터페이스 모듈 중 하나에 삽입합니다.
 - QSROUTE
 - QSQLOPEN
 - QSQCCLSE
 - QSQCMT

일부 SQL문(예를 들어 DECLARE문)의 경우 SQL 사전컴파일러는 주석을 제외하고는 호스트 언어를 생성하지 않습니다.

- 사전컴파일된 각각의 SQL문에 관한 정보를 작성합니다. 정보는 바인드 처리시 사용 가능한 임시 소스 파일 멤버에 내부적으로 저장됩니다.

사전컴파일시 완전한 진단 정보를 얻으려면 다음 중 하나를 지정하십시오.

- CRTSQLxxx에 대한 OPTION(*SOURCE *XREF)(xxx=CBL, PLI 또는 RPG)
- CRTSQLxxx에 대한 OPTION(*XREF) OUTPUT(*PRINT)(xxx=CI, CBLI 또는 RPGI)

자세한 내용은 다음 절을 참조하십시오.

- 『SQL 사전컴파일러에 대한 입력』
- 147 페이지의 『SQL 사전컴파일러에서 소스 파일 CCSID』
- 147 페이지의 『SQL 사전컴파일러로부터 출력』

SQL 사전컴파일러에 대한 입력

어플리케이션 프로그램 작성 명령문과 삽입된 SQL문이 SQL 사전컴파일러에 대한 1차 입력입니다. PL/I, C 및 C++ 프로그램에서 SQL문은 CRTSQLPLI, CRTSQLCI, CRTSQLCPPI 명령의 MARGINS 매개변수에 지정된 여백을 사용해야 합니다.

SQL 사전컴파일러는 호스트 언어 명령문이 구문적으로 올바르다고 가정합니다. 호스트 언어 명령문이 구문적으로 올바르지 않으면 사전컴파일러는 SQL문과 호스트 변수 선언 부분을 올바로 식별할 수 없는 경우도 있습니다. 사전컴파일러를 통해 전달될 수 있는 소스문의 형식에는 한계가 있습니다. 어플리케이션 언어 컴파일러에 의해 허용되지 않는 리터럴과 주석은 사전컴파일러 소스 검색 처리를 방해하고 오류를 발생시킵니다.

SQL INCLUDE문은 CRTSQLxxx⁷의 INCFILE 매개변수에 의해 지정된 파일의 2차 입력을 얻는 데 사용될 수 있습니다. SQL INCLUDE문을 사용하면 멤버의 끝에 도달할 때까지 지정된 멤버로부터 입력이 읽혀집니다. 포함된 멤버에는 다른 사전컴파일러 INCLUDE문이 들어 있지 않아도 되지만 어플리케이션 프로그램과 SQL문 모두 포함시킬 수 있습니다.

SQL 사전컴파일러 이전에 다른 사전처리가 소스문을 처리하는 수도 있습니다. 그러나 SQL 사전컴파일 이전에 수행된 모든 사전처리는 SQL문을 처리할 수 있어야 합니다.

혼합 DBCS 상수가 어플리케이션 프로그램 소스에서 지정되는 경우 소스 파일은 혼합 CCSID이어야 합니다.

SQL SET OPTION문을 사용하여 입력 소스 멤버에 있는 많은 사전컴파일러 옵션들을 지정할 수 있습니다. SET OPTION 구문에 대한 내용은 SQL 참조서를 참조하십시오.

SQL 사전컴파일러에서 소스 파일 CCSID

SQL 사전컴파일러는 소스 파일의 CCSID를 사용하여 소스 레코드를 읽습니다. SQL INCLUDE문 처리시 포함 소스는 필요한 경우 소스 원본 파일의 CCSID로 변환됩니다. 포함 소스가 원래 소스 파일의 CCSID로 변환될 수 없으면 오류가 발생합니다.

SQL 사전컴파일러는 소스 CCSID를 사용하여 SQL문을 처리합니다. 이는 대부분 가변 문자에 영향을 미칩니다. 예를 들어, not sign(¬)은 CCSID 500의 'BA'X에 위치합니다. 이는 소스 파일의 CCSID가 500인 경우 SQL은 not sign(¬)이 'BA'X에 위치될 것으로 예상한다는 것을 의미합니다.

소스 파일 CCSID가 65535이면 SQL은 가변 문자가 CCSID 37을 가지는 것처럼 처리합니다. 이는 SQL이 not sign(¬)을 '5F'X에서 찾음을 의미합니다.

SQL 사전컴파일러로부터 출력

이 섹션에서는 사전컴파일러에 의해 제공된 여러 종류의 출력을 설명합니다.

리스트

출력 리스트는 CRTSQLxxx 명령의 PRTRILE 매개변수에 의해 지정된 인쇄 파일로 보내집니다. 다음 항목들은 프린터 파일에 작성되는 항목입니다.

- 사전컴파일러 옵션

옵션은 CRTSQLxxx 명령에 지정됩니다.

7. 이 명령에서 xxx는 호스트 언어 인디케이터를 말합니다. iSeries용 COBOL 언어의 경우 CBL, iSeries용 ILE COBOL 언어의 경우 CBLL, iSeries용 PL/I 언어의 경우 PL/I, iSeries용 ILE C 언어의 경우 CI, iSeries용 RPG 언어의 경우 RPG, iSeries용 ILE RPG 언어의 경우 RPGL, ILE C++/400 언어의 경우 CPPI입니다.

- 사전컴파일러 소스
리스트 옵션이 수행될 경우 이 출력은 사전컴파일러에 의해 할당된 레코드 번호가 있는 사전컴파일러 소스문을 제공합니다.
- 사전컴파일러 상호 참조
*XREF가 OPTION 매개변수에 지정될 경우 이 출력은 상호 참조표를 제공합니다. 이 리스트는 호스트명과 열명에 참조되어 있는 SQL문의 사전컴파일러 레코드 번호를 보여줍니다.
- 사전컴파일러 진단
이 출력은 오류가 있는 명령문의 사전컴파일러 레코드 번호를 표시하는 진단 메시지를 제공합니다.
프린터 파일로의 출력은 CCSID 값 65535를 사용합니다. 자료가 프린터 파일에 대해 기록되는 경우에는 변환되지 않습니다.

SQL 사전컴파일러에 의해 작성된 임시 소스 파일 멤버

사전컴파일러가 처리한 소스문은 출력 소스 파일에 기록됩니다. 사전컴파일러 변경 소스 코드에서 SQL문은 주석으로 변환되며, SQL 실행시 호출합니다. SQL에서 처리되는 포함이 확장됩니다.

출력 소스 파일이 TOSRCFILE 매개변수에서 CRTSQLxxx 명령에 지정됩니다. C 및 C++가 아닌 언어의 경우 디폴트 파일은 QTEMP 라이브러리에서 QSQLTEMP(iSeries용 ILE RPG의 경우 QSQLTEMP1)입니다. C 및 C++의 경우 *CALC는 출력 소스 파일로 지정되면 QSQLTEMP는 소스 파일의 레코드 길이가 92 이하인 경우에 사용됩니다. 레코드 길이가 92보다 큰 C 또는 C++ 소스 파일의 경우 출력 소스 파일명은 QSQLTxxxxx로 생성됩니다. 여기서 xxxxx는 레코드 길이입니다. 출력 소스 파일 멤버의 이름은 CRTSQLxxx 명령의 PGM 또는 OBJ 매개변수에 지정된 이름과 같습니다. 이 멤버는 컴파일러에 대한 입력으로 사용되기 전에는 변경될 수 없습니다. SQL이 출력 소스 파일을 작성할 때 새로운 파일에 대한 CCSID 값으로 소스 파일의 CCSID 값을 사용합니다.

사전컴파일러가 QTEMP의 소스 파일에 출력을 생성하면 나중에 컴파일하려는 경우 파일은 영구 라이브러리로 이동될 수 있습니다. 사용자는 소스 멤버의 레코드를 변경할 수 없으며 그렇지 않으면 시도된 컴파일러 실패합니다.

사전컴파일러의 결과로 SQL에서 생성된 소스 멤버는 또 다른 사전컴파일 단계에 대한 입력 멤버로 편집되고 다시 작성되어서는 안됩니다. 처음 사전컴파일러 소스 멤버로 저장된 추가 SQL 정보로 인해 두 번째 사전컴파일러 실패가 발생합니다. 일단 이 정보가 소스 멤버에 접속되면 멤버가 작성될 때까지 멤버로 남아 있습니다.

SQL 사전컴파일러는 출력 소스 파일을 작성하기 위해 CRTSRCPF 명령을 사용합니다. 이 명령에 대한 디폴트가 변경될 경우 결과는 예측할 수 없습니다. SQL 사전컴파일러 대신 사용자가 소스 파일을 작성하는 경우에 역시 파일 속성이 달라질 수 있습니다.

다. 출력 소스 파일은 SQL에 의해 작성되도록 권장됩니다. 일단 SQL에 의해 파일이 작성되면 이것은 다음의 사전컴파일러에 다시 사용될 수 있습니다.

샘플 SQL 사전컴파일러 출력

사전컴파일러 출력은 프로그램 소스에 관한 정보를 제공할 수 있습니다. 리스트를 생성하려면 다음과 같이 하십시오.

- 비ILE 사전컴파일러의 경우 CRTSQLxxx 명령의 OPTION 매개변수에 *SOURCE(*SRC) 및 *XREF 옵션을 지정하십시오.
- ILE 사전컴파일러의 경우 CRTSQLxxx 명령에 OPTION(*XREF) 및 OUTPUT(*PRINT)을 지정하십시오.

사전컴파일러 출력 형식은 다음과 같습니다.

```
5722ST1 V5R2M0 020719          Create SQL COBOL Program          CBLTEST1          08/06/02 11:14:21  Page  1
Source type.....COBOL
Program name.....CORPDATA/CBLTEST1
Source file.....CORPDATA/SRC
Member.....CBLTEST1
To source file.....QTEMP/QSQLTEMP
1 Options.....*SRC      *XREF      *SQL
Target release.....V5R2M0
INCLUDE file.....*LIBL/*SRCFILE
Commit.....*CHG
Allow copy of data.....*YES
Close SQL cursor.....*ENDPGM
Allow blocking.....*READ
Delay PREPARE.....*NO
Generation level.....10
Printer file.....*LIBL/QSYSPRT
Date format.....*JOB
Date separator.....*JOB
Time format.....*HMS
Time separator.....*JOB
Replace.....*YES
Relational database.....*LOCAL
User.....*CURRENT
RDB connect method.....*DUW
Default Collection.....*NONE
Package name.....*PGLIB/*PGM
Path.....*NAMING
Created object type.....*PGM
User profile.....*NAMING
Dynamic User Profile.....*USER
Sort Sequence.....*JOB
Language ID.....*JOB
IBM SQL flagging.....*NOFLAG
ANS flagging.....*NONE
Text.....*SRCMBRTXT
Source file CCSID.....65535
Job CCSID.....65535
2 06/06/00 10:16:44에 변경된 소스 멤버
```

- 1 SQL 사전컴파일러가 호출되었을 때 지정한 옵션 리스트
- 2 소스 멤버가 마지막으로 변경된 날짜

그림 2. COBOL 사전컴파일러 출력 형식의 샘플 (1/4)

1 Record	*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8	2 SEQNBR	3 Last Change
1	IDENTIFICATION DIVISION.	100	
2	PROGRAM-ID. CBLTEST1.	200	
3	ENVIRONMENT DIVISION.	300	
4	CONFIGURATION SECTION.	400	
5	SOURCE-COMPUTER. IBM-AS400.	500	
6	OBJECT-COMPUTER. IBM-AS400.	600	
7	INPUT-OUTPUT SECTION.	700	
8	FILE-CONTROL.	800	
9	SELECT OUTFILE, ASSIGN TO PRINTER-QPRINT,	900	
10	FILE STATUS IS FSTAT.	1000	
11	DATA DIVISION.	1100	
12	FILE SECTION.	1200	
13	FD OUTFILE	1300	
14	DATA RECORD IS REC-1,	1400	
15	LABEL RECORDS ARE OMITTED.	1500	
16	01 REC-1.	1600	
17	05 CC PIC X.	1700	
18	05 DEPT-NO PIC X(3).	1800	
19	05 FILLER PIC X(5).	1900	
20	05 AVERAGE-EDUCATION-LEVEL PIC ZZZ.	2000	
21	05 FILLER PIC X(5).	2100	
22	05 AVERAGE-SALARY PIC ZZZZ9.99.	2200	
23	01 ERROR-RECORD.	2300	
24	05 CC PIC X.	2400	
25	05 ERROR-CODE PIC S9(5).	2500	
26	05 ERROR-MESSAGE PIC X(70).	2600	
27	WORKING-STORAGE SECTION.	2700	
28	EXEC SQL	2800	
29	INCLUDE SQLCA	2900	
30	END-EXEC.	3000	
31	77 FSTAT PIC XX.	3100	
32	01 AVG-RECORD.	3200	
33	05 WORKDEPT PIC X(3).	3300	
34	05 AVG-EDUC PIC S9(4) USAGE COMP-4.	3400	
35	05 AVG-SALARY PIC S9(6)V99 COMP-3.	3500	
36	PROCEDURE DIVISION.	3600	
37	*****	3700	
38	* This program will get the average education level and the *	3800	
39	* average salary by department. *	3900	
40	*****	4000	
41	A000-MAIN-PROCEDURE.	4100	
42	OPEN OUTPUT OUTFILE.	4200	
43	*****	4300	
44	* Set-up WHENEVER statement to handle SQL errors. *	4400	
45	*****	4500	
46	EXEC SQL	4600	
47	WHENEVER SQLERROR GO TO B000-SQL-ERROR	4700	
48	END-EXEC.	4800	
49	*****	4900	
50	* Declare cursor *	5000	
51	*****	5100	
52	EXEC SQL	5200	
53	DECLARE CURS CURSOR FOR	5300	
54	SELECT WORKDEPT, AVG(EDLEVEL), AVG(SALARY)	5400	
55	FROM CORPDATA.EMPLOYEE	5500	
56	GROUP BY WORKDEPT	5600	
57	END-EXEC.	5700	
58	*****	5800	
59	* Open cursor *	5900	
60	*****	6000	
61	EXEC SQL	6100	
62	OPEN CURS	6200	
63	END-EXEC.	6300	

- 1** 사전컴파일러가 소스 레코드를 읽을 때 할당하는 레코드 번호. 레코드 번호는 오류 메시지와 SQL 실행시(run-time) 처리의 소스 레코드를 식별하는 데 사용됩니다.
- 2** 소스 레코드로부터 가져온 순번. 순번은 사용자가 소스 입력 유틸리티(SEU)를 사용하여 소스 멤버를 편집할 때 표시되는 번호입니다.
- 3** 소스 레코드가 마지막으로 변경된 날짜. 마지막 변경이 공백인 경우 이는 레코드가 작성된 이후 변경되지 않았음을 나타냅니다.

그림 2. COBOL 사전컴파일러 출력 형식의 샘플 (2/4)

```

5722ST1 V5R2M0 020719          Create SQL COBOL Program          CBLTEST1          08/06/02 11:14:21 Page 3
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change
64 *****
65 * Fetch all result rows *
66 *****
67     PERFORM A010-FETCH-PROCEDURE THROUGH A010-FETCH-EXIT
68     UNTIL SQLCODE IS = 100.
69 *****
70 * Close cursor *
71 *****
72     EXEC SQL
73     CLOSE CURS
74     END-EXEC.
75     CLOSE OUTFILE.
76     STOP RUN.
77 *****
78 * Fetch a row and move the information to the output record. *
79 *****
80     A010-FETCH-PROCEDURE.
81     MOVE SPACES TO REC-1.
82     EXEC SQL
83     FETCH CURS INTO :AVG-RECORD
84     END-EXEC.
85     IF SQLCODE IS = 0
86     MOVE WORKDEPT TO DEPT-NO
87     MOVE AVG-SALARY TO AVERAGE-SALARY
88     MOVE AVG-EDUC TO AVERAGE-EDUCATION-LEVEL
89     WRITE REC-1 AFTER ADVANCING 1 LINE.
90     A010-FETCH-EXIT.
91     EXIT.
92 *****
93 * An SQL error occurred. Move the error number to the error *
94 * record and stop running. *
95 *****
96     B000-SQL-ERROR.
97     MOVE SPACES TO ERROR-RECORD.
98     MOVE SQLCODE TO ERROR-CODE.
99     MOVE "AN SQL ERROR HAS OCCURRED" TO ERROR-MESSAGE.
100    WRITE ERROR-RECORD AFTER ADVANCING 1 LINE.
101    CLOSE OUTFILE.
102    STOP RUN.
*****
***** E N D O F S O U R C E *****

```

그림 2. COBOL 사전컴파일러 출력 형식의 샘플 (3/4)

1	2	3
Data Names	Define	Reference
AVERAGE-EDUCATION-LEVEL	20	IN REC-1
AVERAGE-SALARY	22	IN REC-1
AVG-EDUC	34	SMALL INTEGER PRECISION(4,0) IN AVG-RECORD
AVG-RECORD	32	STRUCTURE 83
AVG-SALARY	35	DECIMAL(8,2) IN AVG-RECORD
BIRTHDATE	55	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
BONUS	55	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
B000-SQL-ERROR	****	LABEL 47
CC	17	CHARACTER(1) IN REC-1
CC	24	CHARACTER(1) IN ERROR-RECORD
COMM	55	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
CORPDATA	****	4 COLLECTION 5 55
CURS	53	CURSOR 62 73 83
DEPT-NO	18	CHARACTER(3) IN REC-1
EDLEVEL	****	COLUMN 54
EDLEVEL	55	6 SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMPLOYEE	****	TABLE IN CORPDATA 7 55
EMPNO	55	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
ERROR-CODE	25	NUMERIC(5,0) IN ERROR-RECORD
ERROR-MESSAGE	26	CHARACTER(70) IN ERROR-RECORD
ERROR-RECORD	23	STRUCTURE
FIRSTNME	55	VARCHAR(12) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
FSTAT	31	CHARACTER(2)
HIREDATE	55	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
JOB	55	CHARACTER(8) COLUMN IN CORPDATA.EMPLOYEE
LASTNAME	55	VARCHAR(15) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
MIDNIT	55	CHARACTER(1) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
PHONENO	55	CHARACTER(4) COLUMN IN CORPDATA.EMPLOYEE
REC-1	16	
SALARY	****	COLUMN 54
SALARY	55	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
SEX	55	CHARACTER(1) COLUMN IN CORPDATA.EMPLOYEE
WORKDEPT	33	CHARACTER(3) IN AVG-RECORD
WORKDEPT	****	COLUMN
54 56		
WORKDEPT	55	CHARACTER(3) COLUMN IN CORPDATA.EMPLOYEE

No errors found in source
102 Source records processed

***** END OF LISTING *****

1 자료명은 소스문에 사용되는 기호명입니다.

2 정의 열은 이름이 정의된 행 번호를 지정합니다. 행 번호는 SQL 사전컴파일러에 의해 생성됩니다. ****는 오브젝트가 정의되지 않았거나 사전컴파일러가 선언을 인식하지 않았음을 의미합니다.

3 참조 열에는 두 가지 정보 유형이 있습니다.

- **4** 기호 정의명
- **5** 기호 정의명이 나타난 행 번호

기호명이 유효한 호스트 변수를 참조하는 경우 data-type **6** 또는 data-structure **7** 도 표시됩니다.

그림 2. COBOL 사전컴파일러 출력 형식의 샘플 (4/4)

비ILE SQL 사전컴파일러 명령

DB2 UDB 조회 관리자 및 SQL 개발 킷에는 CRTSQLCBL(iSeries용 COBOL의 경우), CRTSQLPLI(iSeries PL/I의 경우) 및 CRTSQLRPG(iSeries용 RPG의 일부인 RPG III의 경우) 호스트 언어를 위한 비ILE 사전컴파일러 명령이 포함되어 있습니다. 일부 옵션은 특정 언어에만 적용됩니다. 예를 들어 옵션 *APOST와 *QUOTE는 COBOL에만 고유합니다. 그 옵션들은 다른 언어에 대한 명령에는 포함되지 않습니다. 자세한 사항은 199 페이지의 부록 B 『호스트 언어 사전컴파일러에 대한 iSeries용 DB2 UDB CL 명령 설명』을 참조하십시오.

자세한 내용은 『SQL을 사용하는 비ILE 어플리케이션 프로그램 컴파일링』을 참조하십시오.

SQL을 사용하는 비ILE 어플리케이션 프로그램 컴파일링

SQL 사전컴파일러는 *NOGEN이 지정되지 않을 경우 사전컴파일러가 성공적으로 완료된 후 자동으로 호스트 언어 컴파일러를 호출합니다. CRTxxxPGM 명령은 프로그램명, 소스 파일명, 사전컴파일러로 작성된 소스 멤버명, 텍스트 및 USRPRF를 지정하여 수행됩니다.

이와 같은 언어에서는 다음 매개변수가 전달됩니다.

- COBOL의 경우 *QUOTE 또는 *APOST가 CRTCBPLPGM 명령에 전달됩니다.
- RPG와 COBOL의 경우 SAAFLAG(*FLAG)가 CRTxxxPGM 명령에 전달됩니다.
- RPG와 COBOL에서 CRTSQLxxx 명령에서 나온 SRTSEQ와 LANGID 매개변수는 CRTSQLxxx 명령에 지정됩니다.
- RPG와 COBOL의 경우 CVTOPT(*DATETIME *VARCHAR)는 항상 CRTxxxPGM 명령에 지정됩니다.
- COBOL과 RPG의 경우 CRTSQLxxx 명령에서 나온 TGTRLS 매개변수 값은 CRTxxxPGM 명령에 지정됩니다. TGTRLS는 CRTPLIPGM 명령에 지정됩니다. 프로그램은 CRTSQLPLI 명령의 TGTRLS 매개변수에 지정된 레벨로 저장 또는 복원될 수 있습니다.
- PL/I의 경우 MARGINS는 임시 소스 파일에 설정됩니다.
- 모든 언어에 대해 CRTxxxPGM 명령의 REPLACE 매개변수는 CRTxxxPGM 명령에 지정됩니다.

패키지가 사전컴파일 처리의 일부로 작성될 경우 CRTSQLxxx 명령의 REPLACE 매개변수 값이 CRTSQLPKG 명령에서 지정됩니다.

- 모든 언어의 경우 USRPRF(*USER)가 지정되거나 USRPRF(*NAMING)와 함께 시스템명(*SYS)가 지정되면 USRPRF(*USER)가 CRTxxxPGM 명령에 지정됩니다. USRPRF(*OWNER)가 지정되거나 USRPRF(*NAMING)와 함께 SQL명(*SQL)이 지정되면 CRTxxxPGM 명령이 지정됩니다.

CRTxxxPGM 명령의 다른 매개변수에는 디폴트가 사용됩니다.

사전컴파일러 명령의 OPTION 매개변수에 *NOGEN을 지정함으로써 호스트 언어 컴파일러에 대한 호출을 인터럽트시킬 수 있습니다. *NOGEN은 호스트 언어 컴파일러가 호출되지 않음을 지정합니다. CRTSQLxxx 명령에서 멤버명으로 지정된 오브젝트명을 사용할 때 사전컴파일러는 CRTSQLxxx 명령의 TOSRCFILE 매개변수로 지정한 출력 소스 파일에서 소스 멤버를 작성했습니다. 명시적으로 호스트 언어 컴파일러 호출, 출력 소스 파일에 소스 멤버 지정 및 디폴트를 변경할 수 있습니다. 사전컴파일과 컴파일을 분리 단계로 수행하면 CRTSQLPKG 명령을 사용하여 분산 프로그램에 대한 SQL 패키지를 작성할 수 있습니다.

주: CRTxxxPGM 명령을 발행하기 전에 QTEMP/QSQLTEMP에 소스 멤버를 변경해서는 안됩니다. 소스 멤버를 변경하면 컴파일이 실패합니다.

ILE SQL 사전컴파일러 명령

DB2 UDB 조회 관리자 및 SQL 개발 킷에는 CRTSQLCI, CRTSQLCBLI, CRTSQLRPGI 및 CRTSQLCPPI와 같은 ILE 사전컴파일러 명령이 존재합니다. iSeries용 ILE C, iSeries용 ILE COBOL 및 iSeries용 ILE RPG 등의 호스트 언어마다 사전컴파일러 명령이 있습니다. 언어별로 개별 명령을 사용하여 필수 매개변수를 지정한 후, 나머지 매개변수에 대해 디폴트를 지정합니다. 디폴트는 사용중인 언어에만 적용될 수 있습니다. 예를 들어 옵션 *APOST와 *QUOTE는 COBOL에만 고유합니다. 그 옵션들은 다른 언어에 대한 명령에는 포함되지 않습니다. 199 페이지의 부록 B 『호스트 언어 사전컴파일러에 대한 iSeries용 DB2 UDB CL 명령 설명』에 자세한 정보가 나옵니다.

자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 ILE 어플리케이션 프로그램 컴파일링』

SQL을 사용하는 ILE 어플리케이션 프로그램 컴파일링

SQL 사전컴파일러는 *NOGEN이 지정되지 않을 경우 사전컴파일이 완전히 완료된 후 자동으로 호스트 언어 컴파일러를 호출합니다. *MODULE 옵션이 지정된 경우 SQL 사전컴파일러는 모듈을 작성하기 위해 CRTxxxMOD 명령을 발행합니다. *PGM 옵션이 지정되면 SQL 사전컴파일러는 프로그램 작성을 위해 CRTBNDxxx 명령을 발행합니다. *SRVPGM 옵션이 지정되면 SQL 사전컴파일러는 모듈 작성을 위한 CRTxxxMOD 명령을 발행하고 그 뒤에 서비스 프로그램을 작성하기 위한 서비스 프로그램 작성(CRTSRVPGM) 명령을 함께 발행합니다. CRTSQLCPPI 명령만이 *MODULE 오브젝트를 작성합니다.

이와 같은 언어에서는 다음 매개변수가 전달됩니다.

- DBGVIEW(*SOURCE)가 CRTSQLxxx 명령에 지정될 경우 DBGVIEW(*ALL)가 CRTxxxMOD 및 CRTBNDxxx 명령 모두에 지정됩니다.
- OUTPUT(*PRINT)이 CRTSQLxxx 명령에 지정될 경우 CRTxxxMOD 및 CRTBNDxxx 명령 모두에 전달됩니다.
OUTPUT(*NONE)이 CRTSQLxxx 명령에서 지정될 경우 CRTxxxMOD 명령 또는 CRTBNDxxx 명령 중 하나에는 지정되지 않습니다.
- CRTSQLxxx 명령의 TGTRLS 매개변수 값이 CRTxxxMOD, CRTBNDxxx 및 CRTSRVPGM(서비스 프로그램 작성) 명령에 지정됩니다.
- CRTSQLxxx 명령의 REPLACE 매개변수 값이 CRTxxxMOD, CRTBNDxxx 및 CRTSRVPGM 명령에 지정됩니다.
패키지가 사전컴파일 처리의 일부로 작성될 경우 CRTSQLxxx 명령의 REPLACE 매개변수 값이 CRTSQLPKG 명령에서 지정됩니다.
- OBJTYPE이 *PGM이나 *SRVPGM 그리고 USRPRF(*USER)가 지정되거나 USRPRF(*NAMING)과 함께 시스템명(*SYS)가 지정되면 CRTBNDxxx 또는 CRTSRVPGM 명령에 대해 USRPRF(*USER)가 지정됩니다.
OBJTYPE이 *PGM이나 *SRVPGM 그리고 USRPRF(*OWNER)가 지정되거나 USRPRF(*NAMING)과 함께 SQL명(*SQL)이 지정되면 CRTBNDxxx 또는 CRTSRVPGM 명령에 대해 USRPRF(*OWNER)가 지정됩니다.
- C와 C++의 경우 MARGINS는 임시 소스 파일에 설정됩니다.
LOB 호스트 변수의 총 길이를 약 15M으로 사전컴파일러에서 연산한 경우 TERASPACE(*YES *TSIFC) 옵션이 CRTCMOD, CRTBNDc 또는 CRTCPPMOD 명령에서 지정됩니다.
- COBOL의 경우 *QUOTE 또는 *APOST는 CRTBNDcBL 또는 CRTcBLMOD 명령에 전달됩니다.
- RPG 및 COBOL의 경우 CRTSQLxxx 명령의 SRTSEQ 및 LANGID 매개변수는 CRTxxxMOD 및 CRTBNDxxx 명령에 지정됩니다.
- COBOL의 경우 CVTOPT(*VARCHAR *DATETIME *PICGGRAPHIC *FLOAT)는 항상 CRTcBLMOD 및 CRTBNDcBL 명령에서 지정됩니다. OPTION(*NOCVTDT)이 지정되면(제공된 명령 디폴트), 추가 옵션 *DATE *TIME *TIMESTAMP도 CVTOPT에 지정됩니다.
- RPG의 경우 OPTION(*CVTDT)가 지정되었으면 CVTOPT(*DATETIME)가 CRTRPGMOD와 CRTBNDRPG 명령에 지정됩니다.

사전컴파일러 명령의 OPTION 매개변수에 *NOGEN을 지정함으로써 호스트 언어 컴파일러에 대한 호출을 인터럽트시킬 수 있습니다. *NOGEN은 호스트 언어 컴파일러가 호출되지 않았음을 지정합니다. CRTSQLxxx 명령에서 지정된 프로그램명을 멤버명으로 사용하면 사전컴파일러는 출력 소스 파일(TOSRCFILE 매개변수)에서 소스 멤버를 작성합니다. 이제 명시적으로 호스트 언어 컴파일러 호출, 출력 소스 파일에 소스 멤버

지정 및 디폴트를 변경할 수 있습니다. 사전컴파일과 컴파일을 분리 단계로 수행하면 CRTSQLPKG 명령을 사용하여 분산 프로그램에 대한 SQL 패키지를 작성할 수 있습니다.

프로그램 또는 서비스 프로그램이 후에 작성된다면 USRPRF 매개변수는 CRTBNDC(바인드 C 작성), CRTPGM(프로그램 작성) 또는 CRTSRVPGM(서비스 프로그램 작성) 명령에 올바르게 설정될 수 없습니다. SQL 프로그램은 USRPRF 매개변수가 바르게 된 후에만 예측한 대로 실행됩니다. 시스템명이 사용된다면 USRPRF 매개변수는 *USER에 설정되어야 합니다. SQL명이 사용된다면 USRPRF 매개변수는 *OWNER에 설정되어야 합니다.

SQL을 사용하는 어플리케이션에서 컴파일 오류 해석

경고: 사용자가 사전컴파일과 컴파일 단계를 분리하고 소스 프로그램이 외부적으로 설명된 파일을 참조한다면 참조된 파일이 사전컴파일과 컴파일 사이에서 변경되어서는 안 됩니다. 그렇지 않으면 필드 정의에서의 변경이 임시 소스 멤버에서 변경되지 않기 때문에 예측할 수 없는 결과가 발생합니다.

외부 서술 파일의 예는 다음과 같습니다.

- COBOL에서는 COPY DDS
- PL/I에서 %INCLUDE
- C 또는 C++에서 #pragma mapinc 및 #include
- RPG에서는 자료 구조

SQL 사전컴파일러가 호스트 변수를 인식하지 않을 때 소스를 컴파일하십시오. 컴파일러는 EXEC SQL문을 인식하지 않고 이러한 오류를 무시합니다. 컴파일러가 언어에 대한 SQL 사전컴파일러에 의해 정의된 대로 호스트 변수 선언 부분을 해석하는지 확인하십시오.

자세한 내용은 『SQL을 사용하는 어플리케이션 프로그램의 컴파일시 오류 및 경고 메시지』를 참조하십시오.

SQL을 사용하는 어플리케이션 프로그램의 컴파일시 오류 및 경고 메시지

다음은 컴파일 처리시 오류 또는 경고 메시지를 발생시킬 수도 있는 조건에 대한 설명입니다.

PL/I, C 또는 C++ 컴파일시 오류 및 경고 메시지

EXEC SQL(MARGINS 매개변수, 디폴트로 지정된 대로)을 왼쪽 여백 전에 시작하면 SQL 사전컴파일러는 명령문을 SQL문으로 인식하지 않습니다. 결과적으로 그것은 컴파일러에 대한 것처럼 전달됩니다.

COBOL 컴파일시 오류 및 경고 메시지

EXEC SQL이 12열 앞에서 시작하면 SQL 사전컴파일러는 명령문을 SQL문으로 인식하지 않습니다. 결과적으로 그것은 컴파일러에 대한 것처럼 전달됩니다.

RPG 컴파일시 오류 및 경고 메시지

EXEC SQL이 8-16열 사이에 코딩되지 않고 7열에 '/' 문자가 없으면 SQL 사전컴파일러는 명령문을 SQL문으로 인식하지 않습니다. 결과적으로 그것은 컴파일러에 대한 것처럼 전달됩니다.

자세한 사항은 제 2 장 『C 및 C++ 어플리케이션에서 SQL문 코딩』과 제 7 장 『REXX 어플리케이션에서의 SQL문 코딩』에서 상세한 프로그래밍 예를 참조하십시오.

SQL을 사용하는 어플리케이션 바인딩

어플리케이션 프로그램을 실행하기 전에 프로그램과 지정된 표 및 뷰 사이의 관계가 설정되어야 합니다. 이 프로세스를 바인딩이라고 합니다. 바인딩의 결과는 액세스 계획입니다.

액세스 계획은 각 SQL 요구를 만족시키기 위해 필요한 조치를 설명하는 제어 구조입니다. 액세스 계획에는 프로그램과 프로그램이 사용하려는 자료에 관한 정보가 들어 있습니다.

비분산 SQL 프로그램의 경우 액세스 계획은 프로그램에 저장됩니다. 분산 SQL 프로그램(RDB 매개변수가 CRTSQLxxx 명령에 지정되어 있는)의 경우 액세스 계획은 지정된 관계형 데이터베이스의 SQL 패키지에 저장됩니다.

SQL은 자동으로 프로그램 오브젝트 작성시 액세스 계획을 바인드하고 작성하려 합니다. 비ILE 컴파일의 경우 이는 성공적인 CRTxxxPGM의 결과로서 발생합니다. ILE 컴파일의 경우 이는 성공적인 CRTBNDxxx, CRTPGM 또는 CRTSRVPGM 명령의 결과로서 발생합니다. iSeries용 DB2 UDB에서 수행시 액세스 계획이 유효하지 않음을 발견하게 되거나 (예를 들어 참조된 표가 다른 라이브러리에 들어 있음) 성능을 향상시킬 수도 있는 데이터베이스에 대한 변경이 발생하는 경우(예를 들어 색인 추가), 새로운 액세스 계획이 자동으로 작성됩니다. 바인딩은 세 가지를 수행합니다.

1. 데이터베이스의 설명을 사용하여 SQL문의 유효성을 확인합니다. 바인드 처리시 SQL 문은 유효한 표, 뷰 및 열명에 대해 체크됩니다. 지정된 표나 뷰가 사전컴파일이나 컴파일시 존재하지 않으면 수행시에 유효성 검사가 수행됩니다. 표나 뷰가 수행시 존재하지 않으면 다음의 SQLCODE가 리턴됩니다.
2. 프로그램이 처리하려는 자료에 액세스하는 데 필요한 색인을 선택합니다. 색인, 표 크기 및 다른 인자 선택시 액세스 계획이 빌드됩니다. 그것은 자료에 액세스하는 데 사용할 수 있는 모든 색인을 고려하고 자료에 대한 경로를 선택할 때 사용할 것(있는 경우)을 결정합니다.

3. 액세스 계획을 빌드하려 합니다. 모든 SQL문이 유효한 경우 바인드 처리는 프로그램에 액세스 계획을 빌드하고 저장합니다.

프로그램이 액세스하는 표나 뷰의 특성이 변경되면 액세스 계획은 더 이상 유효하지 않습니다. 유효하지 않은 액세스 계획이 들어 있는 프로그램을 수행하려 할 때 시스템은 자동으로 액세스 계획을 다시 빌드합니다. 액세스 계획이 재구성되지 않으면 다음의 SQLCODE가 리턴됩니다. 이 경우 프로그램의 SQL문을 변경해야 할 수 있으며, CRTSQLxxx 명령을 다시 발행하여 상황을 정정해야 할 수 있습니다.

예를 들어 TABLEA의 COLUMNNA를 참조하는 SQL문이 들어 있는 프로그램에서 사용자가 TABLEA를 삭제하고 재작성하여 COLUMNNA가 더 이상 존재하지 않는 경우 사용자가 프로그램을 호출하면 COLUMNNA가 더 이상 존재하지 않으므로 자동 리바인드가 제대로 수행되지 않습니다. 이 경우 프로그램 소스를 변경해야 하고 CRTSQLxxx 명령을 재발행해야 합니다.

자세한 내용은 『SQL을 사용하는 어플리케이션에서 프로그램 참조』를 참조하십시오.

SQL을 사용하는 어플리케이션에서 프로그램 참조

SQL 프로그램 내의 SQL문에서 참조된 모든 스키마, 표, 뷰, SQL 패키지 및 색인은 프로그램이 작성될 때 라이브러리의 오브젝트 정보 저장소(OIR)에 있습니다.

CL 명령인 DSPPGMREF(프로그램 참조 표시)를 사용하여 프로그램 내의 모든 오브젝트 참조를 표시할 수 있습니다. SQL 명명 규칙이 사용되면 라이브러리명은 다음의 세 가지 방법 중 하나로 OIR에 저장됩니다.

1. SQL명이 완전히 규정되면 콜렉션명은 이름 규정자로 저장됩니다.
2. SQL명이 완전히 규정되지 않고 DFTRDBCOL 매개변수가 지정되지 않으면 명령문의 권한부여 ID가 이름 규정자로 저장됩니다.
3. SQL명이 완전히 규정되지 않고 DFTRDBCOL 매개변수가 지정되면 DFTRDBCOL 매개변수에 지정된 스키마명이 이름 규정자로 저장됩니다.

시스템 명명 규칙이 사용되면 라이브러리명은 다음 세 가지 방법 중 하나로 OIR에 저장됩니다.

1. 오브젝트명이 완전히 규정되면 라이브러리명은 이름 규정자로 저장됩니다.
2. 오브젝트명이 완전히 규정되지 않고 DFTRDBCOL 매개변수가 지정되지 않으면 *LIBL이 저장됩니다.
3. SQL명이 완전히 규정되지 않고 DFTRDBCOL 매개변수가 지정되면 DFTRDBCOL 매개변수에 지정된 스키마명이 이름 규정자로 저장됩니다.

SQL 사전컴파일러 옵션 표시

SQL 어플리케이션 프로그램이 컴파일될 때 SQL 사전컴파일러에서 지정된 옵션의 일부를 결정하기 위해서는 DSPMOD(모듈 표시), DSPPGM(프로그램 표시) 또는 DSPSRVPGM(서비스 프로그램 표시) 명령이 사용될 수 있습니다. 이 정보는 프로그램의 소스가 변경되어야 할 때 필요할 수도 있습니다. 이같은 SQL 사전컴파일러 옵션은 프로그램이 다시 컴파일될 때 CRTSQLxxx 명령에서 지정될 수 있습니다.

PRTSQLINF(SQL 정보 인쇄) 명령은 SQL 사전컴파일러에서 지정된 옵션의 일부를 결정하는 데 사용될 수 있습니다.

삽입된 SQL을 사용하는 프로그램 실행

사전컴파일과 컴파일이 제대로 수행된 후 삽입된 SQL문으로 호스트 언어 프로그램을 실행하는 것은 호스트 프로그램의 실행과 동일합니다. 시스템 명령 행에 다음을 입력하십시오.

```
CALL pgm-name
```

프로그램 실행에 대한 자세한 정보는 CL Programming  을 참조하십시오.

주: 새로운 릴리스를 설치하고 나면, 사용자가 프로그램에 대한 *CHANGE 권한을 가지고 있지 않을 경우 SQL 프로그램을 사용하는 QHST에서 CPF2218 메시지가 발생할 수 있습니다. *CHANGE 권한을 가지고 있는 사용자가 그 프로그램을 호출하면 액세스 계획이 갱신되고 메시지가 제공됩니다.

자세한 내용은 다음 절을 참조하십시오.

- 『삽입된 SQL을 사용하는 프로그램 실행: OS/400 DDM 고려사항』
- 『삽입된 SQL을 사용하는 프로그램 실행: 대체 고려사항』
- 160 페이지의 『삽입된 SQL을 사용하는 프로그램 실행: SQL 리턴 코드』

삽입된 SQL을 사용하는 프로그램 실행: OS/400 DDM 고려사항

SQL은 분산 자료 관리(DDM) 파일을 통해 리모트 파일 액세스를 지원하지 않습니다. SQL은 분산 관계형 데이터베이스 구조™(DRDA®)를 통해 리모트 액세스를 지원합니다.

삽입된 SQL을 사용하는 프로그램 실행: 대체 고려사항

대체(OVRDBF 명령으로 지정)를 사용하면 다른 표나 뷰에 대한 참조를 지정하거나 프로그램이나 SQL 패키지의 조작 특성을 변경할 수 있습니다. 다음 매개변수들은 대체가 지정되는 경우에 처리되는 것입니다.

TOFILE

MBR
SEQONLY
INHWRT
WAITRCD

다른 모든 대체 매개변수들은 무시됩니다. SQL 패키지에서의 명령문 대체는 다음 두 가지 방법을 통해 이루어집니다.

1. OVRDBF 명령에 OVRSCOPE(*JOB) 매개변수를 지정합니다.
2. SBMRMTCMD(제출 리모트 명령)을 사용하여 명령을 어플리케이션 서버에 보냅니다.

긴 이름으로 작성된 표와 뷰를 대체하려면 이 표와 뷰에 연관된 시스템명을 사용하여 대체를 작성할 수 있습니다. SQL문에 긴 이름이 지정되는 경우 해당 시스템명을 사용하면 대체를 찾을 수 있습니다.

별명은 실제로 DDM 파일로 작성됩니다. 별명(DDM 파일)으로 대체를 작성할 수 있습니다. 이런 경우 대체를 실제로 보유하는 파일의 SQL문은 별명과 연관된 파일을 사용합니다.

대체에 대한 자세한 정보는 데이터베이스 프로그래밍 및 파일 관리를 참조하십시오.

삽입된 SQL을 사용하는 프로그램 실행: SQL 리턴 코드

SQL 리턴 코드 리스트는 iSeries Information Center의 SQL 메시지 및 코드에 나와 있습니다.

부록 A. iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램

이 부록에는 iSeries용 DB2 UDB 시스템에서 지원되는 각각의 언어에 SQL문을 코딩하는 방법을 표시하는 샘플 어플리케이션이 들어 있습니다.

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

SQL문을 사용하는 프로그램의 예

호스트 언어를 사용하여 SQL문을 코딩하는 방법의 예를 지원하는 프로그램이 다음 프로그래밍 언어에 대해 제공됩니다.

- ILE C 및 C++
- COBOL 및 ILE COBOL
- PL/I
- iSeries용 RPG
- iSeries용 ILE RPG
- REXX

이 샘플 어플리케이션은 수당에 근거한 임금 인상을 보여줍니다.

각 샘플 프로그램은 동일한 보고서를 작성하며 이 보고서는 부록의 맨 뒤에 제공됩니다. 보고서의 첫 번째 부분에서는 프로젝트별로 임금 인상을 적용받는 프로젝트에 대해 작업하는 모든 사원을 보여줍니다. 보고서의 두 번째 부분에서는 각 프로젝트에 대한 새로운 급여 정보를 보여줍니다.

샘플 프로그램에 대한 주:

다음 주가 모든 샘플 프로그램에 적용됩니다.

SQL문은 대문자나 소문자로 입력할 수 있습니다.

- 1** 이 호스트 언어 명령문은 SQL 표 PROJECT의 외부 정의를 검색합니다. 이러한 정의는 호스트 변수나 호스트 구조로 사용될 수 있습니다.

주:

1. iSeries용 RPG에서는 6자 이상의 외부 서술 구조의 필드명이 변경되어야 합니다.
2. REXX는 외부 정의 검색을 지원하지 않습니다.

- 2** SQL INCLUDE SQLCA문은 PL/I, C 및 COBOL 프로그램에 SQLCA를 포함시키는데 사용됩니다. RPG 프로그램의 경우 SQL 사전컴파일러는 자동으로

SQLCA 자료 구조를 입력 스펙 섹션 끝에 있는 소스에 둡니다. REXX의 경우 SQLCA 필드는 SQLCA에 의해 맵핑되는 연속의 자료영역이 아니라 개별 변수에서 유지보수됩니다.

3 이 SQL WHENEVER문은 SQL문에서 SQLERROR(SQLCODE < 0)가 발생하는 경우에 제어가 전달되는 호스트 언어 레이블을 정의합니다. WHENEVER SQLERROR문은 다음 WHENEVER SQLERROR문이 발생될 때까지 모든 후속 SQL문에 적용됩니다. REXX는 WHENEVER문을 지원하지 않습니다. 대신에 SIGNAL ON ERROR 기능을 사용합니다.

4 이 SQL UPDATE문이 SALARY열을 갱신하며, 그 열에는 호스트 변수 PERCENTAGE(RPG의 경우 PERCNT)의 비율별로 사원 급여가 들어 있습니다. 갱신된 행은 사원 수당이 2000보다 큰 행입니다. REXX의 경우 호스트 변수가 있으면 UPDATE를 직접 실행할 수 없으므로 PREPARE나 EXECUTE 문을 사용합니다.

5 이 SQL COMMIT문은 SQL UPDATE 문에 의해 수행된 변경 내용을 확약합니다. 변경된 모든 행에 대한 레코드 잠금이 해제됩니다.

주: COMMIT(*CHG)를 사용하여 프로그램이 사전컴파일되었습니다.(REXX의 경우 *CHG가 디폴트입니다.)

6 이 SQL DECLARE CURSOR문은 두 개의 표 EMPLOYEE 및 EMPPROJECT를 결합시키는 커서 C1을 정의하며, 임금이 인상된(수당 > 2000) 사원에 대한 행을 리턴합니다. 행은 프로젝트 번호와 사원 번호(PROJNO 및 EMPNO)에 의해 오름차순으로 리턴됩니다. REXX의 경우 호스트 변수가 있으면 명령문 스트링 내에서 직접 DECLARE CURSOR문을 실행할 수 없으므로 PREPARE와 DECLARE CURSOR를 사용합니다.

7 이 SQL OPEN문은 행이 폐치될 수 있도록 커서 C1을 엽니다.

8 이 SQL WHENEVER문은 모든 행이 폐치될 때(SQLCODE = 100) 제어가 전달되는 호스트 언어 레이블을 정의합니다. REXX의 경우 SQLCODE는 명시적으로 체크되어야 합니다.

9 이 SQL FETCH문은 커서 C1에 대한 모든 행을 리턴시키고 리턴값을 호스트 구조의 해당 요소에 위치시킵니다.

10 모든 행이 폐치된 후 제어가 이 레이블로 전달됩니다. SQL CLOSE문은 커서 C1을 닫습니다.

11 이 SQL DECLARE CURSOR문은 세 개의 표 EMPPROJECT, PROJECT 및 EMPLOYEE를 결합시키는 커서 C2를 정의합니다. 결과는 열 PROJNO와 PROJNAME에 의해 그룹화됩니다. COUNT 함수는 각 그룹의 행 수를 계산합니다. SUM 함수는 각 프로젝트에 대한 새로운 급여 경비를 계산합니다. ORDER BY 절은 행이 마지막 결과 열(EMPPROJECT.PROJNO)의 내용

에 따라 검색되도록 지정합니다. REXX의 경우 호스트 변수가 있으면 명령문 스트링 내에서 직접 DECLARE CURSOR문을 실행할 수 없으므로 PREPARE와 DECLARE CURSOR를 사용합니다.

- 12** 이 SQL FETCH문은 커서 C2에 대한 결과 열을 리턴시키고, 리턴된 값을 프로그램에 의해 설명된 호스트 구조의 해당 요소에 위치시킵니다.
- 13** CONTINUE 옵션이 지정된 이 SQL WHENEVER문은 SQL문에 오류가 발생하는 것에 관계없이 다음 명령문으로 처리가 계속되도록 합니다. SQL ROLLBACK문에서 오류가 발생하지 않기를 기대하지만, 이렇게 하면 실제 오류가 발생하더라도 프로그램이 다시 루프 안에 들어가는 것을 막습니다. 다음 WHENEVER SQLERROR문을 만날 때까지 SQL문이 계속됩니다. REXX는 WHENEVER문을 지원하지 않습니다. REXX는 그 대신에 SIGNAL OFF ERROR 기능을 사용합니다.
- 14** 이 SQL ROLLBACK문은 갱신중에 오류가 발생하는 경우에 표를 원래 상태로 복원시킵니다.

예: ILE C 및 C++ 프로그램에서 SQL문

이 샘플 프로그램은 C 프로그래밍 언어로 작성되어 있습니다. 동일한 프로그램이 아래와 같은 조건이 충족되는 경우에 작동될 것입니다.

- SQL BEGIN DECLARE SECTION 명령문이 18행 앞에서 추가되었습니다.
- SQL END DECLARE SECTION 명령문이 42행 앞에서 추가되었습니다.

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

```

5722ST1 V5R2M0 020719          Create SQL ILE C Object          CEX          08/06/02 15:52:26 Page 1
Source type.....C
Object name.....CORPDATA/CEX
Source file.....CORPDATA/SRC
Member.....CEX
To source file.....QTEMP/QSQLTEMP
Options.....*XREF
Listing option.....*PRINT
Target release.....V5R2M0
INCLUDE file.....*LIBL/*SRCFILE
Commit.....*CHG
Allow copy of data.....*YES
Close SQL cursor.....*ENDACTGRP
Allow blocking.....*READ
Delay PREPARE.....*NO
Generation level.....10
Margins.....*SRCFILE
Printer file.....*LIBL/QSYSPRT
Date format.....*JOB
Date separator.....*JOB
Time format.....*HMS
Time separator.....*JOB
Replace.....*YES
Relational database.....*LOCAL
User.....*CURRENT
RDB connect method.....*DUW
Default collection.....*NONE
Dynamic default
  collection.....*NO
Package name.....*OBJLIB/*OBJ
Path.....*NAMING
Created object type.....*PGM
Debugging view.....*NONE
User profile.....*NAMING
Dynamic user profile.....*USER
Sort Sequence.....*JOB
Language ID.....*JOB
IBM SQL flagging.....*NOFLAG
ANS flagging.....*NONE
Text.....*SRCMBRTXT
Source file CCSID.....65535
Job CCSID.....65535
Source member changed on 06/06/00 17:15:17

```

Record	*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8	SEQNBR	Last change
1	#include "string.h"	100	
2	#include "stdlib.h"	200	
3	#include "stdio.h"	300	
4		400	
5	main()	500	
6	{	600	
7	/* A sample program which updates the salaries for those employees */	700	
8	/* whose current commission total is greater than or equal to the */	800	
9	/* value of 'commission'. The salaries of those who qualify are */	900	
10	/* increased by the value of 'percentage' retroactive to 'raise_date'*/	1000	
11	/* A report is generated showing the projects which these employees */	1100	
12	/* have contributed to ordered by project number and employee ID. */	1200	
13	/* A second report shows each project having an end date occurring */	1300	
14	/* after 'raise_date' (is potentially affected by the retroactive */	1400	
15	/* raises) with its total salary expenses and a count of employees */	1500	
16	/* who contributed to the project. */	1600	
17		1700	

그림 3. SQL문을 사용하는 C 프로그램의 샘플 (1/5)

```

5722ST1 V5R2M0 020719          Create SQL ILE C Object          CEX          08/06/02 15:52:26 Page 2
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change
18 short work_days = 253; /* work days during in one year */ 1800
19 float commission = 2000.00; /* cutoff to qualify for raise */ 1900
20 float percentage = 1.04; /* raised salary as percentage */ 2000
21 char raise_date??(12??) = "1982-06-01"; /* effective raise date */ 2100
22 2200
23 /* File declaration for qprint */ 2300
24 FILE *qprint; 2400
25 2500
26 /* Structure for report 1 */ 2600
27 1 #pragma mapinc ("project", "CORPDATA/PROJECT(PROJECT)", "both", "p z") 2700
28 #include "project" 2800
29 struct { 2900
30     CORPDATA_PROJECT_PROJECT_both_t Proj_struct; 3000
31     char empno??(7??); 3100
32     char name??(30??); 3200
33     float salary; 3300
34     } rpt1; 3400
35 3500
36 /* Structure for report 2 */ 3600
37 struct { 3700
38     char projno??(7??); 3800
39     char project_name??(37??); 3900
40     short employee_count; 4000
41     double total_proj_cost; 4100
42     } rpt2; 4200
43 4300
44 2 exec sql include SQLCA; 4400
45 4500
46 qprint=fopen("QPRINT", "w"); 4600
47 4700
48 /* Update the selected projects by the new percentage. If an error */ 4800
49 /* occurs during the update, ROLLBACK the changes. */ 4900
50 3 EXEC SQL WHENEVER SQLERROR GO TO update_error; 5000
51 4 EXEC SQL 5100
52     UPDATE CORPDATA/EMPLOYEE 5200
53     SET SALARY = SALARY * :percentage 5300
54     WHERE COMM >= :commission ; 5400
55 5500
56 /* Commit changes */ 5600
57 5 EXEC SQL 5700
58     COMMIT; 5800
59 EXEC SQL WHENEVER SQLERROR GO TO report_error; 5900
60 6000
61 /* Report the updated statistics for each employee assigned to the */ 6100
62 /* selected projects. */ 6200
63 6300
64 /* Write out the header for Report 1 */ 6400
65 fprintf(qprint, " REPORT OF PROJECTS AFFECTED \ 6500
66 BY RAISES"); 6600
67 fprintf(qprint, "\n\nPROJECT EMPID EMPLOYEE NAME "); 6700
68 fprintf(qprint, " SALARY\n"); 6800
69 6900
70 6 exec sql 7000
71     declare c1 cursor for 7100
72     select distinct projno, empproject.empno, 7200
73     lastname||', '||firstname, salary 7300
74     from corpdata/empproject, corpdata/employee 7400
75     where empproject.empno = employee.empno and comm >= :commission 7500
76     order by projno, empno; 7600
77 7 EXEC SQL 7700
78     OPEN C1; 7800
79 7900
80 /* Fetch and write the rows to QPRINT */ 8000
81 8 EXEC SQL WHENEVER NOT FOUND GO TO done1; 8100
82 8200
83 do { 8300
84 10 EXEC SQL 8400
85     FETCH C1 INTO :Proj_struct.PROJNO, :rpt1.empno, 8500
86     :rpt1.name, :rpt1.salary; 8600
87     fprintf(qprint, "\n%6s %6s %-30s %8.2f", 8700
88     rpt1.Proj_struct.PROJNO, rpt1.empno, 8800
89     rpt1.name, rpt1.salary); 8900
90     } 9000
91 while (SQLCODE=0); 9100
92 9200
93 done1: 9300
94 EXEC SQL 9400
95     CLOSE C1; 9500

```

그림 3. SQL문을 사용하는 C 프로그램의 샘플 (2/5)

```

5722ST1 V5R2M0 020719          Create SQL ILE C Object          CEX          08/06/02 15:52:26 Page 3
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change
 96                                     9600
 97 /* For all projects ending at a date later than the 'raise_date' */ 9700
 98 /* (i.e. those projects potentially affected by the salary raises) */ 9800
 99 /* generate a report containing the project number, project name */ 9900
100 /* the count of employees participating in the project and the */ 10000
101 /* total salary cost of the project. */ 10100
102                                     10200
103 /* Write out the header for Report 2 */ 10300
104 fprintf(qprint,"\n\n\n          ACCUMULATED STATISTICS\
105 BY PROJECT"); 10500
106 fprintf(qprint, "\n\nPROJECT          \ 10600
107 NUMBER OF          TOTAL"); 10700
108 fprintf(qprint, "\nNUMBER PROJECT NAME          \ 10800
109 EMPLOYEES          COST\n"); 10900
110                                     11000
111 11 EXEC SQL 11100
112 DECLARE C2 CURSOR FOR 11200
113 SELECT EMPPROJECT.PROJNO, PROJNAME, COUNT(*), 11300
114 SUM ( ( DAYS(EMENDATE) - DAYS(EMSTDATE) ) * EMPTIME * 11400
115 (DECIMAL( SALARY / :work_days ,8,2))) 11500
116 FROM CORPDATA/EMPPROJECT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE 11600
117 WHERE EMPPROJECT.PROJNO=PROJECT.PROJNO AND 11700
118 EMPPROJECT.EMPNO =EMPLOYEE.EMPNO AND 11800
119 PRENDATE > :raise_date 11900
120 GROUP BY EMPPROJECT.PROJNO, PROJNAME 12000
121 ORDER BY 1; 12100
122 EXEC SQL 12200
123 OPEN C2; 12300
124                                     12400
125 /* Fetch and write the rows to QPRINT */ 12500
126 EXEC SQL WHENEVER NOT FOUND GO TO done2; 12600
127                                     12700
128 do { 12800
129 12 EXEC SQL 12900
130 FETCH C2 INTO :rpt2; 13000
131 fprintf(qprint,"\n%6s %-36s %6d %9.2f", 13100
132 rpt2.projno,rpt2.project_name,rpt2.employee_count, 13200
133 rpt2.total_proj_cost); 13300
134 } 13400
135 while (SQLCODE==0); 13500
136                                     13600
137 done2: 13700
138 EXEC SQL 13800
139 CLOSE C2; 13900
140 goto finished; 14000
141                                     14100
142 /* Error occured while updating table. Inform user and rollback */ 14200
143 /* changes. */ 14300
144 update_error: 14400
145 13 EXEC SQL WHENEVER SQLERROR CONTINUE; 14500
146 fprintf(qprint,"*** ERROR Occurred while updating table. SQLCODE=" 14600
147 "%5d\n",SQLCODE); 14700
148 14 EXEC SQL 14800
149 ROLLBACK; 14900
150 goto finished; 15000
151                                     15100
152 /* Error occured while generating reports. Inform user and exit. */ 15200
153 report_error: 15300
154 fprintf(qprint,"*** ERROR Occurred while generating reports. " 15400
155 "SQLCODE=%5d\n",SQLCODE); 15500
156 goto finished; 15600
157                                     15700
158 /* All done */ 15800
159 finished: 15900
160 fclose(qprint); 16000
161 exit(0); 16100
162                                     16200
163 } 16300
          * * * * * E N D O F S O U R C E * * * * *

```

그림 3. SQL문을 사용하는 C 프로그램의 샘플 (3/5)

CROSS REFERENCE

Data Names	Define	Reference
commission	19	FLOAT(24) 54 75
done1	****	LABEL 81
done2	****	LABEL 126
employee_count	40	SMALL INTEGER PRECISION(4,0) IN rpt2
empno	31	VARCHAR(7) IN rpt1 85
name	32	VARCHAR(30) IN rpt1
percentage	20	FLOAT(24) 53
project_name	39	VARCHAR(37) IN rpt2
projno	38	VARCHAR(7) IN rpt2
raise_date	21	VARCHAR(12) 119
report_error	****	LABEL 59
rpt1	34	
rpt2	42	STRUCTURE 130
salary	33	FLOAT(24) IN rpt1
total_proj_cost	41	FLOAT(53) IN rpt2
update_error	****	LABEL
work_days	18	SMALL INTEGER PRECISION(4,0) 115
ACTNO	74	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
BIRTHDATE	74	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
BONUS	74	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMM	****	COLUMN 54 75
COMM	74	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
CORPDATA	****	COLLECTION 52 74 74 116 116 116
C1	71	CURSOR 78 85 95
C2	112	CURSOR 123 130 139
DEPTNO	27	VARCHAR(3) IN Proj_struct
DEPTNO	116	CHARACTER(3) COLUMN (NOT NULL) IN CORPDATA.PROJECT
EDLEVEL	74	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMENDATE	74	DATE(10) COLUMN IN CORPDATA.EMPPROJECT
EMENDATE	****	COLUMN 114
EMPLOYEE	****	TABLE IN CORPDATA 52 74 116
EMPLOYEE	****	TABLE 75 118
EMPNO	****	COLUMN IN EMPPROJECT 72 75 76 118
EMPNO	****	COLUMN IN EMPLOYEE 75 118
EMPNO	74	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
EMPNO	74	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMPPROJECT	****	TABLE 72 75 113 117 118 120
EMPPROJECT	****	TABLE IN CORPDATA 74 116
EMPTIME	74	DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJECT
EMPTIME	****	COLUMN 114
EMSTDATE	74	DATE(10) COLUMN IN CORPDATA.EMPPROJECT
EMSTDATE	****	COLUMN 114
FIRSTNME	****	COLUMN 73
FIRSTNME	74	VARCHAR(12) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
HIREDATE	74	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
JOB	74	CHARACTER(8) COLUMN IN CORPDATA.EMPLOYEE
LASTNAME	****	COLUMN 73
LASTNAME	74	VARCHAR(15) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE

그림 3. SQL문을 사용하는 C 프로그램의 샘플 (4/5)

```

MAJPROJ          27      VARCHAR(6) IN Proj_struct
MAJPROJ          116     CHARACTER(6) COLUMN IN CORPDATA.PROJECT
MIDINIT          74      CHARACTER(1) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
Proj_struct      30      STRUCTURE IN rpt1
PHONENO          74      CHARACTER(4) COLUMN IN CORPDATA.EMPLOYEE
PRENDATE         27      DATE(10) IN Proj_struct
PRENDATE         ****      COLUMN
                                     119
PRENDATE         116     DATE(10) COLUMN IN CORPDATA.PROJECT
PROJECT          ****      TABLE IN CORPDATA
                                     116
PROJECT          ****      TABLE
                                     117
PROJNAME         27      VARCHAR(24) IN Proj_struct
PROJNAME         ****      COLUMN
                                     113 120
PROJNAME         116     VARCHAR(24) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PROJNO           27      VARCHAR(6) IN Proj_struct
                                     85
PROJNO           ****      COLUMN
                                     72 76
PROJNO           74      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
PROJNO           ****      COLUMN IN EMPPROJECT
                                     113 117 120
PROJNO           ****      COLUMN IN PROJECT
                                     117
PROJNO           116     CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PRSTAFF          27      DECIMAL(5,2) IN Proj_struct
PRSTAFF          116     DECIMAL(5,2) COLUMN IN CORPDATA.PROJECT
PRSTDATE         27      DATE(10) IN Proj_struct
PRSTDATE         116     DATE(10) COLUMN IN CORPDATA.PROJECT
RESPEMP          27      VARCHAR(6) IN Proj_struct
RESPEMP          116     CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
SALARY           ****      COLUMN
                                     53 53 73 115
SALARY           74      DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
SEX              74      CHARACTER(1) COLUMN IN CORPDATA.EMPLOYEE
WORKDEPT         74      CHARACTER(3) COLUMN IN CORPDATA.EMPLOYEE
No errors found in source
163 Source records processed
* * * * * E N D O F L I S T I N G * * * * *

```

그림 3. SQL문을 사용하는 C 프로그램의 샘플 (5/5)

예: COBOL과 ILE COBOL 프로그램에서 SQL문

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

```

5722ST1 V5R2M0 020719          Create SQL COBOL Program          CBLEX          08/06/02 11:09:13 Page 1
Source type.....COBOL
Program name.....CORPDATA/CBLEX
Source file.....CORPDATA/SRC
Member.....CBLEX
To source file.....QTEMP/QSQLTEMP
Options.....*SRC          *XREF
Target release.....V5R2M0
INCLUDE file.....*LIBL/*SRCFILE
Commit.....*CHG
Allow copy of data.....*YES
Close SQL cursor.....*ENDPGM
Allow blocking.....*READ
Delay PREPARE.....*NO
Generation level.....10
Printer file.....*LIBL/QSYSPRT
Date format.....*JOB
Date separator.....*JOB
Time format.....*HMS
Time separator.....*JOB
Replace.....*YES
Relational database.....*LOCAL
User.....*CURRENT
RDB connect method.....*DUW
Default collection.....*NONE
Dynamic default
  collection.....*NO
Package name.....*PGLIB/*PGM
Path.....*NAMING
Created object type.....*PGM
User profile.....*NAMING
Dynamic user profile.....*USER
Sort Sequence.....*JOB
Language ID.....*JOB
IBM SQL flagging.....*NOFLAG
ANS flagging.....*NONE
Text.....*SRCMBRTXT
Source file CCSID.....65535
Job CCSID.....65535
Source member changed on 07/01/96 09:44:58
1
2      *****
3      * A sample program which updates the salaries for those *
4      * employees whose current commission total is greater than or *
5      * equal to the value of COMMISSION. The salaries of those who *
6      * qualify are increased by the value of PERCENTAGE retroactive *
7      * to RAISE-DATE. A report is generated showing the projects *
8      * which these employees have contributed to ordered by the *
9      * project number and employee ID. A second report shows each *
10     * project having an end date occurring after RAISE-DATE *
11     * (i.e. potentially affected by the retroactive raises ) with *
12     * its total salary expenses and a count of employees who *
13     * contributed to the project. *
14     *****
15
16
17     IDENTIFICATION DIVISION.
18
19     PROGRAM-ID. CBLEX.
20     ENVIRONMENT DIVISION.
21     CONFIGURATION SECTION.
22     SOURCE-COMPUTER. IBM-AS400.
23     OBJECT-COMPUTER. IBM-AS400.
24     INPUT-OUTPUT SECTION.
25
26     FILE-CONTROL.
27         SELECT PRINTFILE ASSIGN TO PRINTER-QPRINT
28         ORGANIZATION IS SEQUENTIAL.
29
30     DATA DIVISION.
31

```

그림 4. SQL문을 사용하는 COBOL 프로그램의 샘플 (1/7)

```

5722ST1 V5R2M0 020719          Create SQL COBOL Program          CBLEX          08/06/02 11:09:13          Page 2
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8  SEQNBR Last change
32      FILE SECTION.
33
34      FD PRINTFILE
35         BLOCK CONTAINS 1 RECORDS
36         LABEL RECORDS ARE OMITTED.
37      01 PRINT-RECORD PIC X(132).
38
39      WORKING-STORAGE SECTION.
40      77 WORK-DAYS PIC S9(4) BINARY VALUE 253.
41      77 RAISE-DATE PIC X(11) VALUE "1982-06-01".
42      77 PERCENTAGE PIC S999V99 PACKED-DECIMAL.
43      77 COMMISSION PIC S99999V99 PACKED-DECIMAL VALUE 2000.00.
44
45      *****
46      * Structure for report 1.          *
47      *****
48
49      1 01 RPT1.
50         COPY DDS-PROJECT OF CORPDATA-PROJECT.
51         05 EMPNO    PIC X(6).
52         05 NAME     PIC X(30).
53         05 SALARY   PIC S9(6)V99 PACKED-DECIMAL.
54
55
56      *****
57      * Structure for report 2.          *
58      *****
59
60      01 RPT2.
61         15 PROJNO PIC X(6).
62         15 PROJECT-NAME PIC X(36).
63         15 EMPLOYEE-COUNT PIC S9(4) BINARY.
64         15 TOTAL-PROJ-COST PIC S9(10)V99 PACKED-DECIMAL.
65
66      2 EXEC SQL
67         INCLUDE SQLCA
68         END-EXEC.
69      77 CODE-EDIT PIC ---99.
70
71      *****
72      * Headers for reports.          *
73      *****
74
75      01 RPT1-HEADERS.
76         05 RPT1-HEADER1.
77            10 FILLER PIC X(21) VALUE SPACES.
78            10 FILLER PIC X(111)
79               VALUE "REPORT OF PROJECTS AFFECTED BY RAISES".
80         05 RPT1-HEADER2.
81            10 FILLER PIC X(9) VALUE "PROJECT".
82            10 FILLER PIC X(10) VALUE "EMPID".
83            10 FILLER PIC X(35) VALUE "EMPLOYEE NAME".
84            10 FILLER PIC X(40) VALUE "SALARY".
85      01 RPT2-HEADERS.
86         05 RPT2-HEADER1.
87            10 FILLER PIC X(21) VALUE SPACES.
88            10 FILLER PIC X(111)
89               VALUE "ACCUMULATED STATISTICS BY PROJECT".
90         05 RPT2-HEADER2.
91            10 FILLER PIC X(9) VALUE "PROJECT".
92            10 FILLER PIC X(38) VALUE SPACES.
93            10 FILLER PIC X(16) VALUE "NUMBER OF".
94            10 FILLER PIC X(10) VALUE "TOTAL".
95         05 RPT2-HEADER3.
96            10 FILLER PIC X(9) VALUE "NUMBER".
97            10 FILLER PIC X(38) VALUE "PROJECT NAME".
98            10 FILLER PIC X(16) VALUE "EMPLOYEES".
99            10 FILLER PIC X(65) VALUE "COST".
100
101      01 RPT1-DATA.
102         05 PROJNO    PIC X(6).
103         05 FILLER    PIC XXX VALUE SPACES.
104         05 EMPNO     PIC X(6).
105         05 FILLER    PIC X(4) VALUE SPACES.
106         05 NAME      PIC X(30).
107         05 FILLER    PIC X(3) VALUE SPACES.
108         05 SALARY    PIC ZZZZ9.99.
109         05 FILLER    PIC X(96) VALUE SPACES.

```

그림 4. SQL문을 사용하는 COBOL 프로그램의 샘플 (2/7)


```

5722ST1 V5R2M0 020719          Create SQL COBOL Program          CBLEX          08/06/02 11:09:13 Page 3
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change
109      01 RPT2-DATA.
110          05 PROJNO PIC X(6).
111          05 FILLER PIC XXX VALUE SPACES.
112          05 PROJECT-NAME PIC X(36).
113          05 FILLER PIC X(4) VALUE SPACES.
114          05 EMPLOYEE-COUNT PIC ZZ9.
115          05 FILLER PIC X(5) VALUE SPACES.
116          05 TOTAL-PROJ-COST PIC ZZZZZZZ9.99.
117          05 FILLER PIC X(56) VALUE SPACES.
              118
119      PROCEDURE DIVISION.
120
121      A000-MAIN.
122          MOVE 1.04 TO PERCENTAGE.
123          OPEN OUTPUT PRINTFILE.
124
125      *****
126      * Update the selected employees by the new percentage. If an *
127      * error occurs during the update, ROLLBACK the changes, *
128      *****
129
130      3 EXEC SQL
131          WHENEVER SQLERROR GO TO E010-UPDATE-ERROR
132      END-EXEC.
133      4 EXEC SQL
134          UPDATE CORPDATA/EMPLOYEE
135             SET SALARY = SALARY * :PERCENTAGE
136             WHERE COMM >= :COMMISSION
137      END-EXEC.
138
139      *****
140      * Commit changes. *
141      *****
142
143      5 EXEC SQL
144          COMMIT
145      END-EXEC.
146
147      EXEC SQL
148          WHENEVER SQLERROR GO TO E020-REPORT-ERROR
149      END-EXEC.
150
151      *****
152      * Report the updated statistics for each employee receiving *
153      * a raise and the projects that s/he participates in *
154      *****
155
156      *****
157      * Write out the header for Report 1. *
158      *****
159
160          write print-record from rpt1-header1
161             before advancing 2 lines.
162          write print-record from rpt1-header2
163             before advancing 1 line.
164      6 exec sql
165          declare c1 cursor for
166             SELECT DISTINCT projno, empproject.empno,
167                lastname||", "||firstnme ,salary
168             from corpdata/empproject, corpdata/employee
169             where empproject.empno =employee.empno and
170                comm >= :commission
171             order by projno, empno
172      end-exec.
173      7 EXEC SQL
174          OPEN C1
175      END-EXEC.
176
177          PERFORM B000-GENERATE-REPORT1 THRU B010-GENERATE-REPORT1-EXIT
178          UNTIL SQLCODE NOT EQUAL TO ZERO.
179

```

주: **8** 과 **9** 는 이 그림의 파트 5에 있습니다.

그림 4. SQL문을 사용하는 COBOL 프로그램의 샘플 (3/7)

Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change

```

180 10 A100-DONE1.
181 EXEC SQL
182 CLOSE C1
183 END-EXEC.
184
185 *****
186 * For all projects ending at a date later than the RAISE- *
187 * DATE ( i.e. those projects potentially affected by the *
188 * salary raises generate a report containing the project *
189 * project number, project name, the count of employees *
190 * participating in the project and the total salary cost *
191 * for the project *
192 *****
193
194
195 *****
196 * Write out the header for Report 2. *
197 *****
198
199 MOVE SPACES TO PRINT-RECORD.
200 WRITE PRINT-RECORD BEFORE ADVANCING 2 LINES.
201 WRITE PRINT-RECORD FROM RPT2-HEADER1
202 BEFORE ADVANCING 2 LINES.
203 WRITE PRINT-RECORD FROM RPT2-HEADER2
204 BEFORE ADVANCING 1 LINE.
205 WRITE PRINT-RECORD FROM RPT2-HEADER3
206 BEFORE ADVANCING 2 LINES.
207
208 EXEC SQL
209 11 DECLARE C2 CURSOR FOR
210 SELECT EMPPROJECT.PROJNO, PROJNAME, COUNT(*),
211 SUM ( (DAYS(EMENDATE)-DAYS(EMSTDATE)) *
212 EMPTIME * DECIMAL((SALARY / :WORK-DAYS),8,2))
213 FROM CORPDATA/EMPPROJECT, CORPDATA/PROJECT,
214 CORPDATA/EMPLOYEE
215 WHERE EMPPROJECT.PROJNO=PROJECT.PROJNO AND
216 EMPPROJECT.EMPNO =EMPLOYEE.EMPNO AND
217 PRENDATE > :RAISE-DATE
218 GROUP BY EMPPROJECT.PROJNO, PROJNAME
219 ORDER BY 1
220 END-EXEC.
221 EXEC SQL
222 OPEN C2
223 END-EXEC.
224
225 PERFORM C000-GENERATE-REPORT2 THRU C010-GENERATE-REPORT2-EXIT
226 UNTIL SQLCODE NOT EQUAL TO ZERO.
227
228 A200-DONE2.
229 EXEC SQL
230 CLOSE C2
231 END-EXEC
232
233 *****
234 * All done. *
235 *****
236
237 A900-MAIN-EXIT.
238 CLOSE PRINTFILE.
239 STOP RUN.
240

```

그림 4. SQL문을 사용하는 COBOL 프로그램의 샘플 (4/7)

```

5722ST1 V5R2M0 020719          Create SQL COBOL Program          CBLEX          08/06/02 11:09:13 Page 5
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change
241 *****
242 * Fetch and write the rows to PRINTFILE. *
243 *****
244
245 B000-GENERATE-REPORT1.
246 8 EXEC SQL
247     WHENEVER NOT FOUND GO TO A100-DONE1
248 END-EXEC.
249 9 EXEC SQL
250     FETCH C1 INTO :PROJECT.PROJNO, :RPT1.EMPNO,
251                   :RPT1.NAME, :RPT1.SALARY
252 END-EXEC.
253     MOVE CORRESPONDING RPT1 TO RPT1-DATA.
254     MOVE PROJNO OF RPT1 TO PROJNO OF RPT1-DATA.
255     WRITE PRINT-RECORD FROM RPT1-DATA
256     BEFORE ADVANCING 1 LINE.
257
258 B010-GENERATE-REPORT1-EXIT.
259     EXIT.
260
261 *****
262 * Fetch and write the rows to PRINTFILE. *
263 *****
264
265 C000-GENERATE-REPORT2.
266     EXEC SQL
267     WHENEVER NOT FOUND GO TO A200-DONE2
268 END-EXEC.
269 12 EXEC SQL
270     FETCH C2 INTO :RPT2
271 END-EXEC.
272     MOVE CORRESPONDING RPT2 TO RPT2-DATA.
273     WRITE PRINT-RECORD FROM RPT2-DATA
274     BEFORE ADVANCING 1 LINE.
275
276 C010-GENERATE-REPORT2-EXIT.
277     EXIT.
278
279 *****
280 * Error occured while updating table. Inform user and *
281 * rollback changes. *
282 *****
283
284 E010-UPDATE-ERROR.
285 13 EXEC SQL
286     WHENEVER SQLERROR CONTINUE
287 END-EXEC.
288     MOVE SQLCODE TO CODE-EDIT.
289     STRING "*** ERROR Occurred while updating table. SQLCODE="
290           CODE-EDIT DELIMITED BY SIZE INTO PRINT-RECORD.
291     WRITE PRINT-RECORD.
292 14 EXEC SQL
293     ROLLBACK
294 END-EXEC.
295     STOP RUN.
296
297 *****
298 * Error occured while generating reports. Inform user and *
299 * exit. *
300 *****
301
302 E020-REPORT-ERROR.
303     MOVE SQLCODE TO CODE-EDIT.
304     STRING "*** ERROR Occurred while generating reports. SQLCODE
305 -      =" CODE-EDIT DELIMITED BY SIZE INTO PRINT-RECORD.
306     WRITE PRINT-RECORD.
307     STOP RUN.
308
309 * * * * * E N D O F S O U R C E * * * * *

```

그림 4. SQL문을 사용하는 COBOL 프로그램의 샘플 (5/7)

Data Names	Define	Reference
ACTNO	168	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
A100-DONE1	****	LABEL 247
A200-DONE2	****	LABEL
267		
BIRTHDATE	134	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
BONUS	134	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
CODE-EDIT	69	
COMM	****	COLUMN 136 170
COMM	134	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMMISSION	43	DECIMAL(7,2) 136 170
CORPDATA	****	COLLECTION 134 168 168 213 213 214
C1	165	CURSOR 174 182 250
C2	209	CURSOR 222 230 270
DEPTNO	50	CHARACTER(3) IN PROJECT
DEPTNO	213	CHARACTER(3) COLUMN (NOT NULL) IN CORPDATA.PROJECT
EDLEVEL	134	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMENDATE	168	DATE(10) COLUMN IN CORPDATA.EMPPROJECT
EMENDATE	****	COLUMN 211
EMPLOYEE	****	TABLE IN CORPDATA 134 168 214
EMPLOYEE	****	TABLE 169 216
EMPLOYEE-COUNT	63	SMALL INTEGER PRECISION(4,0) IN RPT2
EMPLOYEE-COUNT	114	IN RPT2-DATA
EMPNO	51	CHARACTER(6) IN RPT1 250
EMPNO	103	CHARACTER(6) IN RPT1-DATA
EMPNO	134	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMPNO	****	COLUMN IN EMPPROJECT 166 169 171 216
EMPNO	****	COLUMN IN EMPLOYEE 169 216
EMPNO	168	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
EMPPROJECT	****	TABLE 166 169 210 215 216 218
EMPPROJECT	****	TABLE IN CORPDATA 168 213
EMPTIME	168	DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJECT
EMPTIME	****	COLUMN 212
EMSTDATE	168	DATE(10) COLUMN IN CORPDATA.EMPPROJECT
EMSTDATE	****	COLUMN 211
E010-UPDATE-ERROR	****	LABEL 131
E020-REPORT-ERROR	****	LABEL 148
FIRSTNME	134	VARCHAR(12) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
FIRSTNME	****	COLUMN 167
HIREDATE	134	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
JOB	134	CHARACTER(8) COLUMN IN CORPDATA.EMPLOYEE
LASTNAME	134	VARCHAR(15) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
LASTNAME	****	COLUMN 167
MAJPROJ	50	CHARACTER(6) IN PROJECT
MAJPROJ	213	CHARACTER(6) COLUMN IN CORPDATA.PROJECT
MIDNIT	134	CHARACTER(1) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
NAME	52	CHARACTER(30) IN RPT1 251
NAME	105	CHARACTER(30) IN RPT1-DATA

그림 4. SQL문을 사용하는 COBOL 프로그램의 샘플 (6/7)

```

CROSS REFERENCE
PERCENTAGE          42      DECIMAL(5,2)
                        135
PHONENO             134      CHARACTER(4) COLUMN IN CORPDATA.EMPLOYEE
PRENDATE           50       DATE(10) IN PROJECT
PRENDATE           ****    COLUMN
                        217
PRENDATE           213      DATE(10) COLUMN IN CORPDATA.PROJECT
PRINT-RECORD       37       CHARACTER(132)
PROJECT            50       STRUCTURE IN RPT1
PROJECT            ****    TABLE IN CORPDATA
                        213
PROJECT            ****    TABLE
                        215
PROJECT-NAME        62       CHARACTER(36) IN RPT2
PROJECT-NAME       112      CHARACTER(36) IN RPT2-DATA
PROJNAME           50       VARCHAR(24) IN PROJECT
PROJNAME           ****    COLUMN
                        210 218
PROJNAME           213      VARCHAR(24) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PROJNO             50       CHARACTER(6) IN PROJECT
                        250
PROJNO             61       CHARACTER(6) IN RPT2
PROJNO             101      CHARACTER(6) IN RPT1-DATA
PROJNO             110      CHARACTER(6) IN RPT2-DATA
PROJNO             ****    COLUMN
                        166 171
PROJNO             168      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
PROJNO             ****    COLUMN IN EMPPROJECT
                        210 215 218
PROJNO             ****    COLUMN IN PROJECT
                        215
PROJNO             213      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PRSTAFF           50       DECIMAL(5,2) IN PROJECT
PRSTAFF           213      DECIMAL(5,2) COLUMN IN CORPDATA.PROJECT
PRSTDATE          50       DATE(10) IN PROJECT
PRSTDATE          213      DATE(10) COLUMN IN CORPDATA.PROJECT
RAISE-DATE        41       CHARACTER(11)
                        217
RESPEMP           50       CHARACTER(6) IN PROJECT
RESPEMP           213      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
RPT1              49
RPT1-DATA         100
RPT1-HEADERS      75
RPT1-HEADER1      76       IN RPT1-HEADERS
RPT1-HEADER2      80       IN RPT1-HEADERS
RPT2              60       STRUCTURE
                        270
RPT2-DATA         109
SS REFERENCE
RPT2-HEADERS      85
RPT2-HEADER1      86       IN RPT2-HEADERS
RPT2-HEADER2      90       IN RPT2-HEADERS
RPT2-HEADER3      95       IN RPT2-HEADERS
SALARY            53       DECIMAL(8,2) IN RPT1
                        251
SALARY            107      IN RPT1-DATA
SALARY            ****    COLUMN
                        135 135 167 212
SALARY            134      DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
SEX               134      CHARACTER(1) COLUMN IN CORPDATA.EMPLOYEE
TOTAL-PROJ-COST   64       DECIMAL(12,2) IN RPT2
TOTAL-PROJ-COST   116      IN RPT2-DATA
WORK-DAYS         40       SMALL INTEGER PRECISION(4,0)
                        212
WORKDEPT          134      CHARACTER(3) COLUMN IN CORPDATA.EMPLOYEE

```

No errors found in source

307 Source records processed

***** END OF LISTING *****

그림 4. SQL문을 사용하는 COBOL 프로그램의 샘플 (7/7)

예: PL/I에서 SQL문

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

```
5722ST1 V5R2M0 020719          Create SQL PL/I Program          PLIEX          08/06/02 12:53:36 Page 1
Source type.....PLI
Program name.....CORPDATA/PLIEX
Source file.....CORPDATA/SRC
Member.....PLIEX
To source file.....QTEMP/QSQLTEMP
Options.....*SRC *XREF
Target release.....V5R2M0
INCLUDE file.....*LIBL/*SRCFILE
Commit.....*CHG
Allow copy of data.....*YES
Close SQL cursor.....*ENDPGM
Allow blocking.....*READ
Delay PREPARE.....*NO
Generation level.....10
Margins.....*SRCFILE
Printer file.....*LIBL/QSYSPRT
Date format.....*JOB
Date separator.....*JOB
Time format.....*HMS
Time separator.....*JOB
Replace.....*YES
Relational database.....*LOCAL
User.....*CURRENT
RDB connect method.....*DUW
Default collection.....*NONE
Dynamic default
  collection.....*NO
Package name.....*PGMLIB/*PGM
Path.....*NAMING
User profile.....*NAMING
Dynamic user profile.....*USER
Sort sequence.....*JOB
Language ID.....*JOB
IBM SQL flagging.....*NOFLAG
ANS flagging.....*NONE
Text.....*SRCMBRTXT
Source file CCSID.....65535
Job CCSID.....65535
Source member changed on 07/01/96 12:53:08
```

```
1  /* A sample program which updates the salaries for those employees */          100
2  /* whose current commission total is greater than or equal to the */          200
3  /* value of COMMISSION. The salaries of those who qualify are */          300
4  /* increased by the value of PERCENTAGE, retroactive to RAISE_DATE. */          400
5  /* A report is generated showing the projects which these employees */          500
6  /* have contributed to, ordered by project number and employee ID. */          600
7  /* A second report shows each project having an end date occurring */          700
8  /* after RAISE_DATE (i.e. is potentially affected by the retroactive */          800
9  /* raises) with its total salary expenses and a count of employees */          900
10 /* who contributed to the project. */          1000
11 /****** */          1100
12 */          1200
```

그림 5. SQL문을 사용하는 PL/I 프로그램의 샘플 (1/6)

Record	5722ST1 V5R2M0 020719	Create SQL PL/I Program	PLIEX	08/06/02 12:53:36	Page 2					
	*...+... 1	+...+... 2	+...+... 3	+...+... 4	+...+... 5	+...+... 6	+...+... 7	+...+... 8	SEQNBR	Last change
13									1300	
14		PLIEX: PROC;							1400	
15									1500	
16		DCL RAISE_DATE CHAR(10);							1600	
17		DCL WORK_DAYS FIXED BIN(15);							1700	
18		DCL COMMISSION FIXED DECIMAL(8,2);							1800	
19		DCL PERCENTAGE FIXED DECIMAL(5,2);							1900	
20									2000	
21		/* File declaration for sysprint */							2100	
22		DCL SYSPRINT FILE EXTERNAL OUTPUT STREAM PRINT;							2200	
23									2300	
24		/* Structure for report 1 */							2400	
25		DCL 1 RPT1,							2500	
26	1	%INCLUDE PROJECT (PROJECT, RECORD,,COMMA);							2600	
27		15 EMPNO CHAR(6),							2700	
28		15 NAME CHAR(30),							2800	
29		15 SALARY FIXED DECIMAL(8,2);							2900	
30									3000	
31		/* Structure for report 2 */							3100	
32		DCL 1 RPT2,							3200	
33		15 PROJNO CHAR(6),							3300	
34		15 PROJECT_NAME CHAR(36),							3400	
35		15 EMPLOYEE_COUNT FIXED BIN(15),							3500	
36		15 TOTL_PROJ_COST FIXED DECIMAL(10,2);							3600	
37									3700	
38	2	EXEC SQL INCLUDE SQLCA;							3800	
39									3900	
40		COMMISSION = 2000.00;							4000	
41		PERCENTAGE = 1.04;							4100	
42		RAISE_DATE = '1982-06-01';							4200	
43		WORK_DAYS = 253;							4300	
44		OPEN FILE(SYSPRINT);							4400	
45									4500	
46		/* Update the selected employee's salaries by the new percentage. */							4600	
47		/* If an error occurs during the update, ROLLBACK the changes. */							4700	
48	3	EXEC SQL WHENEVER SQLERROR GO TO UPDATE_ERROR;							4800	
49	4	EXEC SQL							4900	
50		UPDATE CORPDATA/EMPLOYEE							5000	
51		SET SALARY = SALARY * :PERCENTAGE							5100	
52		WHERE COMM >= :COMMISSION ;							5200	
53									5300	
54		/* Commit changes */							5400	
55	5	EXEC SQL							5500	
56		COMMIT;							5600	
57		EXEC SQL WHENEVER SQLERROR GO TO REPORT_ERROR;							5700	
58									5800	
59		/* Report the updated statistics for each project supported by one */							5900	
60		/* of the selected employees. */							6000	
61									6100	
62		/* Write out the header for Report 1 */							6200	
63		put file(sysprint)							6300	
64		edit('REPORT OF PROJECTS AFFECTED BY EMPLOYEE RAISES')							6400	
65		(col(22),a);							6500	
66		put file(sysprint)							6600	
67		edit('PROJECT','EMPID','EMPLOYEE NAME','SALARY')							6700	
68		(skip(2),col(1),a,col(10),a,col(20),a,col(55),a);							6800	
69									6900	
70	6	exec sql							7000	
71		declare c1 cursor for							7100	
72		select DISTINCT projno, EMPPROJECT.empno,							7200	
73		lastname ', ' firstnme, salary							7300	
74		from CORPDATA/EMPPROJECT, CORPDATA/EMPLOYEE							7400	
75		where EMPPROJECT.empno = EMPLOYEE.empno and							7500	
76		comm >= :COMMISSION							7600	
77		order by projno, empno;							7700	
78	7	EXEC SQL							7800	
79		OPEN C1;							7900	
80									8000	

그림 5. SQL문을 사용하는 PL/I 프로그램의 샘플 (2/6)


```

5722ST1 V5R2M0 020719          Create SQL PL/I Program          PLIEX          08/06/02 12:53:36  Page    4
150  14 EXEC SQL                  15000
151      ROLLBACK;                15100
152      GO TO FINISHED;          15200
153                                15300
154      /* Error occured while generating reports. Inform user and exit. */ 15400
155      REPORT ERROR:             15500
156      PUT FILE(SYSPRINT) EDIT('*** ERROR Occurred while generating '||
157      'reports.  SQLCODE=',SQLCODE)(A,F(5)); 15600
158      GO TO FINISHED;          15700
159                                15800
160      /* All done */           15900
161      FINISHED:                 16000
162      CLOSE FILE(SYSPRINT);     16100
163      RETURN;                   16200
164                                16300
165      END PLIEX;                16400
                                   16500
                                   * * * * * E N D O F S O U R C E * * * * *

```

그림 5. SQL문을 사용하는 PL/I 프로그램의 샘플 (4/6)

CROSS REFERENCE

Data Names	Define	Reference
ACTNO	74	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
BIRTHDATE	74	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
BONUS	74	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMM	****	COLUMN 52 76
COMM	74	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMMISSION	18	DECIMAL(8,2) 52 76
CORPDATA	****	COLLECTION 50 74 74 118 118 118
C1	71	CURSOR 79 86 95
C2	114	CURSOR 125 132 141
DEPTNO	26	CHARACTER(3) IN RPT1
DEPTNO	118	CHARACTER(3) COLUMN (NOT NULL) IN CORPDATA.PROJECT
DONE1	****	LABEL 82
DONE2	****	LABEL 128
EDLEVEL	74	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMENDATE	74	DATE(10) COLUMN IN CORPDATA.EMPPROJECT
EMENDATE	****	COLUMN 116
EMPLOYEE	****	TABLE IN CORPDATA 50 74 118
EMPLOYEE	****	TABLE 75 120
EMPLOYEE_COUNT	35	SMALL INTEGER PRECISION(4,0) IN RPT2
EMPNO	27	CHARACTER(6) IN RPT1
86		
EMPNO	****	COLUMN IN EMPPROJECT 72 75 77 120
EMPNO	****	COLUMN IN EMPLOYEE 75 120
EMPNO	74	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
EMPNO	74	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMPPROJECT	****	TABLE 72 75 115 119 120 122
EMPPROJECT	****	TABLE IN CORPDATA 74 118
EMPTIME	74	DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJECT
EMPTIME	****	COLUMN 116
EMSTDATE	74	DATE(10) COLUMN IN CORPDATA.EMPPROJECT
EMSTDATE	****	COLUMN 116
FIRSTNME	****	COLUMN 73
FIRSTNME	74	VARCHAR(12) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
HIREDATE	74	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
JOB	74	CHARACTER(8) COLUMN IN CORPDATA.EMPLOYEE
LASTNAME	****	COLUMN 73
LASTNAME	74	VARCHAR(15) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
MAJPROJ	26	CHARACTER(6) IN RPT1
MAJPROJ	118	CHARACTER(6) COLUMN IN CORPDATA.PROJECT
MIDINIT	74	CHARACTER(1) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
NAME	28	CHARACTER(30) IN RPT1
86		
PERCENTAGE	19	DECIMAL(5,2)
51		
PHONENO	74	CHARACTER(4) COLUMN IN CORPDATA.EMPLOYEE

그림 5. SQL문을 사용하는 PL/I 프로그램의 샘플 (5/6)

```

CROSS REFERENCE
PRENDATE          26      DATE(10) IN RPT1
PRENDATE          ****   COLUMN
121
PRENDATE          118     DATE(10) COLUMN IN CORPDATA.PROJECT
PROJECT           ****   TABLE IN CORPDATA
118
PROJECT           ****   TABLE
119
PROJECT_NAME      34      CHARACTER(36) IN RPT2
PROJNAME          26      VARCHAR(24) IN RPT1
PROJNAME          ****   COLUMN
115 122
PROJNAME          118     VARCHAR(24) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PROJNO            26      CHARACTER(6) IN RPT1
86
PROJNO            33      CHARACTER(6) IN RPT2
PROJNO            ****   COLUMN
72 77
PROJNO            74      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
PROJNO            ****   COLUMN IN EMPPROJECT
115 119 122
PROJNO            ****   COLUMN IN PROJECT
119
PROJNO            118     CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PRSTAFF          26      DECIMAL(5,2) IN RPT1
PRSTAFF          118     DECIMAL(5,2) COLUMN IN CORPDATA.PROJECT
PRSTDATE         26      DATE(10) IN RPT1
PRSTDATE         118     DATE(10) COLUMN IN CORPDATA.PROJECT
RAISE_DATE       16      CHARACTER(10)
121
REPORT_ERROR     ****   LABEL
57
RESPEMP          26      CHARACTER(6) IN RPT1
RESPEMP          118     CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
RPT1             25      STRUCTURE
RPT2             32      STRUCTURE
132
SALARY           29      DECIMAL(8,2) IN RPT1
87
SALARY           ****   COLUMN
51 51 73 117
SALARY           74      DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
SEX              74      CHARACTER(1) COLUMN IN CORPDATA.EMPLOYEE
SYSPRINT         22
TOTL_PROJ_COST   36      DECIMAL(10,2) IN RPT2
UPDATE_ERROR     ****   LABEL
48
WORK_DAYS        17      SMALL INTEGER PRECISION(4,0)
117
WORKDEPT         74      CHARACTER(3) COLUMN IN CORPDATA.EMPLOYEE
No errors found in source
165 Source records processed
***** END OF LISTING *****

```

그림 5. SQL문을 사용하는 PL/I 프로그램의 샘플 (6/6)

예: iSeries용 RPG 프로그램에서 SQL문

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

```

Source type.....RPG
Program name.....CORPDATA/RPGEX
Source file.....CORPDATA/SRC
Member.....RPGEX
To source file.....QTEMP/QSQLTEMP
Options.....*SRC *XREF
Target release.....V5R2M0
INCLUDE file.....*LIBL/*SRCFILE
Commit.....*CHG
Allow copy of data.....*YES
Close SQL cursor.....*ENDPGM
Allow blocking.....*READ
Delay PREPARE.....*NO
Generation level.....10
Printer file.....*LIBL/QSYSPRT
Date format.....*JOB
Date separator.....*JOB
Time format.....*HMS
Time separator.....*JOB
Replace.....*YES
Relational database.....*LOCAL
User.....*CURRENT
RDB connect method.....*DUW
Default collection.....*NONE
Dynamic default
  collection.....*NO
Package name.....*PGMLIB/*PGM
Path.....*NAMING
User profile.....*NAMING
Dynamic user profile.....*USER
Sort sequence.....*JOB
Language ID.....*JOB
IBM SQL flagging.....*NOFLAG
ANS flagging.....*NONE
Text.....*SRCMBRTXT
Source file CCSID.....65535
Job CCSID.....65535
Source member changed on 07/01/96 17:06:17

```

```

1      H                                100
2      F* File declaration for QPRINT    200
3      F*                                300
4      FQPRINT  0  F 132      PRINTER    400
5      I*                                500
6      I* Structure for report 1.        600
7      I*                                700
8      1 IRPT1      E DSPROJECT          800
9      I          PROJNAME              PROJNM  900
10     I          RESPEMP              RESEM   1000
11     I          PRSTAFF              STAFF   1100
12     I          PRSTDATE             PRSTD   1200
13     I          PRENDATE             PREND   1300
14     I          MAJPROJ              MAJPRJ  1400
15     I*                                1500
16     I          DS                    1600
17     I          1  6 EMPNO            1700
18     I          7 36 NAME             1800
19     I          P 37 412SALARY        1900
20     I*                                2000
21     I* Structure for report 2.        2100
22     I*                                2200
23     IRPT2      DS                    2300
24     I          1  6 PRJNUM           2400
25     I          7 42 PNAME            2500
26     I          B 43 440EMPCNT        2600
27     I          P 45 492PRCOST        2700
28     I*                                2800
29     I          DS                    2900
30     I          B 1 20WRKDAY           3000
31     I          P 3 62COMMI           3100
32     I          7 16 RDATE            3200
33     I          P 17 202PERCNT        3300

```

그림 6. SQL문을 사용하는 샘플 iSeries용 RPG 프로그램 (1/6)

```

5722ST1 V5R2M0 020719          Create SQL RPG Program          RPGEX          08/06/02 12:55:22  Page  2
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8  SEQNBR Last change
34      2 C*                               3400
35      C          Z-ADD253          WRKDAY          3500
36      C          Z-ADD2000.00      COMMI          3600
37      C          Z-ADD1.04         PERCNT          3700
38      C          MOVE1'1982-06-'RDATE          3800
39      C          MOVE '01'         RDATE          3900
40      C          SETON                               LR          3901
41      C*                               4000
42      C* Update the selected projects by the new percentage. If an          4100
43      C* error occurs during the update, ROLLBACK the changes.          4200
44      C*                               4300
45      3 C/EXEC SQL WHENEVER SQLERROR GOTO UPDERR          4400
46      C/END-EXEC                               4500
47      C*                               4600
48      4 C/EXEC SQL                               4700
49      C+ UPDATE CORPDATA/EMPLOYEE          4800
50      C+   SET SALARY = SALARY * :PERCNT          4900
51      C+   WHERE COMM >= :COMMI          5000
52      C/END-EXEC                               5100
53      C*                               5200
54      C* Commit changes.          5300
55      C*                               5400
56      5 C/EXEC SQL COMMIT                               5500
57      C/END-EXEC                               5600
58      C*                               5700
59      C/EXEC SQL WHENEVER SQLERROR GO TO RPERR          5800
60      C/END-EXEC                               5900
61      C*                               6000
62      C* Report the updated statistics for each employee assigned to          6100
63      C* selected projects.          6200
64      C*                               6300
65      C* Write out the header for report 1.          6400
66      C*                               6500
67      C          EXCPTRECA                               6600
68      6 C/EXEC SQL DECLARE C1 CURSOR FOR          6700
69      C+   SELECT DISTINCT PROJNO, EMPPROJECT.EMPNO,          6800
70      C+   LASTNAME||', '||FIRSTNAME, SALARY          6900
71      C+   FROM CORPDATA/EMPPROJECT, CORPDATA/EMPLOYEE          7000
72      C+   WHERE EMPPROJECT.EMPNO = EMPLOYEE.EMPNO AND          7100
73      C+   COMM >= :COMMI          7200
74      C+   ORDER BY PROJNO, EMPNO          7300
75      C/END-EXEC                               7400
76      C*                               7500
77      7 C/EXEC SQL                               7600
78      C+ OPEN C1          7700
79      C/END-EXEC                               7800
80      C*                               7900
81      C* Fetch and write the rows to QPRINT.          8000
82      C*                               8100
83      8 C/EXEC SQL WHENEVER NOT FOUND GO TO DONE1          8200
84      C/END-EXEC                               8300
85      C          SQLCOD          DOUNEO          8400
86      C/EXEC SQL                               8500
87      9 C+ FETCH C1 INTO :PROJNO, :EMPNO, :NAME, :SALARY          8600
88      C/END-EXEC                               8700
89      C          EXCPTRECB          8800
90      C          END          8900
91      C          DONE1          TAG          9000
92      C/EXEC SQL                               9100
93      10 C+ CLOSE C1          9200
94      C/END-EXEC                               9300
95      C*                               9400
96      C* For all project ending at a date later than the raise date          9500
97      C* (i.e. those projects potentially affected by the salary raises)          9600
98      C* generate a report containing the project number, project name,          9700
99      C* the count of employees participating in the project and the          9800
100     C* total salary cost of the project.          9900
101     C*                               10000
102     C* Write out the header for report 2.          10100
103     C*                               10200
104     C          EXCPTRECC          10300

```

그림 6. SQL문을 사용하는 샘플 iSeries용 RPG 프로그램 (2/6)

Record	Code	Text	SEQNBR
5722ST1	V5R2M0	020719	
		Create SQL RPG Program	RPGEX
			08/06/02 12:55:22
			Page 3
Record	*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8		SEQNBR
			Last change
105	11	C/EXEC SQL	10400
106	C+	DECLARE C2 CURSOR FOR	10500
107	C+	SELECT EMPPROJACT.PROJNO, PROJNAME, COUNT(*),	10600
108	C+	SUM((DAYS(EMENDATE) - DAYS(EMSTDATE)) * EMPTIME * DECIMAL((SALARY/:WRKDAY),8,2))	10700
109	C+	FROM CORPDATA/EMPPROJACT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE	10800
110	C+	WHERE EMPPROJACT.PROJNO = PROJECT.PROJNO AND	10900
111	C+	EMPPROJACT.EMPNO = EMPLOYEE.EMPNO AND	11000
112	C+	PRENDATE > :RDATE	11100
113	C+	GROUP BY EMPPROJACT.PROJNO, PROJNAME	11200
114	C+	ORDER BY 1	11300
115	C+	C/END-EXEC	11400
116	C*		11500
117	C*		11600
118	C/EXEC SQL	OPEN C2	11700
119	C/END-EXEC		11800
120	C*		11900
121	C*	Fetch and write the rows to QPRINT.	12000
122	C*		12100
123	C/EXEC SQL	WHENEVER NOT FOUND GO TO DONE2	12200
124	C/END-EXEC		12300
125	C	SQLCOD DOUNE0	12400
126	C/EXEC SQL		12500
127	12	C+ FETCH C2 INTO :RPT2	12600
128	C/END-EXEC		12700
129	C	EXCPTRECD	12800
130	C	END	12900
131	C	DONE2 TAG	13000
132	C/EXEC SQL	CLOSE C2	13100
133	C/END-EXEC		13200
134	C	RETRN	13300
135	C*		13400
136	C*	Error occured while updating table. Inform user and rollback	13500
137	C*	changes.	13600
138	C*		13700
139	C	UPDERR TAG	13800
140	C	EXCPTRECE	13900
141	13	C/EXEC SQL WHENEVER SQLERROR CONTINUE	14000
142	C/END-EXEC		14100
143	C*		14200
144	14	C/EXEC SQL	14300
145	C+	ROLLBACK	14400
14400			
146	C/END-EXEC		14500
147	C	RETRN	14600
148	C*		14700
149	C*	Error occured while generating reports. Inform user and exit.	14800
150	C*		14900
151	C	RPTERR TAG	15000
152	C	EXCPTRECF	15100
153	C*		15200
154	C*	All done.	15300
155	C*		15400
156	C	FINISH TAG	15500
157	OQPRINT	E 0201 RECA	15700
158	O		45 'REPORT OF PROJECTS AFFEC'
159	O		64 'TED BY EMPLOYEE RAISES'
160	O	E 01 RECA	16000
161	O		7 'PROJECT'
162	O		17 'EMPLOYEE'
163	O		32 'EMPLOYEE NAME'
164	O		60 'SALARY'
165	O	E 01 RECB	16500
166	O	PROJNO	6
167	O	EMPNO	15
168	O	NAME	50
169	O	SALARYL	61
170	O	E 22 RECC	17000
171	O		42 'ACCUMULATED STATISTIC'
172	O		54 'S BY PROJECT'
173	O	E 01 RECC	17300
174	O		7 'PROJECT'
175	O		56 'NUMBER OF'
176	O		67 'TOTAL'
177	O	E 02 RECC	17700
178	O		6 'NUMBER'
179	O		21 'PROJECT NAME'
180	O		56 'EMPLOYEES'
181	O		66 'COST'

그림 6. SQL문을 사용하는 샘플 iSeries용 RPG 프로그램 (3/6)

```

5722ST1 V5R2M0 020719          Create SQL RPG Program          RPGEX          08/06/02 12:55:22 Page 4
182      0      E 01          RECD          18200
195      0          57 'CODE='          19500
183      0          PRJNUM      6          18300
184      0          PNAME      45          18400
185      0          EMPCNTL    54          18500
186      0          PRCOSTL    70          18600
187      0      E 01          RECE          18700
188      0          28 '*** ERROR Occurred while!'          18800
189      0          52 ' updating table. SQLCODE'          18900
190      0          53 '='          19000
191      0          SQLCODL    62          19100
192      0      E 01          RECF          19200
193      0          28 '*** ERROR Occurred while!'          19300
194      0          52 ' generating reports. SQL'          19400
196      0          SQLCODL    67          19600
          * * * * * E N D   O F   S O U R C E   * * * * *

```

그림 6. SQL문을 사용하는 샘플 iSeries용 RPG 프로그램 (4/6)

CROSS REFERENCE

Data Names	Define	Reference
ACTNO	68	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
BIRTHDATE	48	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
BONUS	48	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMM	****	COLUMN 48 68
COMM	48	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMMI	31	DECIMAL(7,2) 48 68
CORPDATA	****	COLLECTION 48 68 68 105 105
C1	68	CURSOR 77 86 92
C2	105	CURSOR 118 126 132
DEPTNO	8	CHARACTER(3) IN RPT1
DEPTNO	105	CHARACTER(3) COLUMN (NOT NULL) IN CORPDATA.PROJECT
DONE1	91	LABEL 83
DONE2	131	LABEL 123
EDLEVEL	48	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMENDATE	68	DATE(10) COLUMN IN CORPDATA.EMPPROJECT
EMENDATE	****	COLUMN 105
EMPCNT	26	SMALL INTEGER PRECISION(4,0) IN RPT2
EMPLOYEE	****	TABLE IN CORPDATA 48 68 105
EMPLOYEE	****	TABLE 68 105
EMPNO	17	CHARACTER(6)
86		
EMPNO	48	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMPNO	****	COLUMN IN EMPPROJECT 68 68 68 105
EMPNO	****	COLUMN IN EMPLOYEE 68 105
EMPNO	68	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
EMPPROJECT	****	TABLE 68 68 105 105 105
EMPPROJECT	****	TABLE IN CORPDATA 68 105
EMPTIME	68	DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJECT
EMPTIME	****	COLUMN 105
EMSTDATE	68	DATE(10) COLUMN IN CORPDATA.EMPPROJECT
EMSTDATE	****	COLUMN 105
FINISH	156	LABEL
FIRSTNME	48	VARCHAR(12) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
FIRSTNME	****	COLUMN 68
HIREDATE	48	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
JOB	48	CHARACTER(8) COLUMN IN CORPDATA.EMPLOYEE
LASTNAME	48	VARCHAR(15) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
LASTNAME	****	COLUMN 68
MAJPRJ	8	CHARACTER(6) IN RPT1
MAJPROJ	105	CHARACTER(6) COLUMN IN CORPDATA.PROJECT
MIDINIT	48	CHARACTER(1) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
NAME	18	CHARACTER(30)
86		
PERCNT	33	DECIMAL(7,2) 48
PHONENO	48	CHARACTER(4) COLUMN IN CORPDATA.EMPLOYEE
PNAME	25	CHARACTER(36) IN RPT2
PRCOST	27	DECIMAL(9,2) IN RPT2
PREND	8	DATE(10) IN RPT1
PRENDATE	****	COLUMN 105
PRENDATE	105	DATE(10) COLUMN IN CORPDATA.PROJECT
PRJNUM	24	CHARACTER(6) IN RPT2

그림 6. SQL문을 사용하는 샘플 iSeries용 RPG 프로그램 (5/6)


```

CROSS REFERENCE
PROJECT          ****  TABLE IN CORPDATA
                  105
PROJECT          ****  TABLE
                  105
PROJNAME         ****  COLUMN
                  105 105
PROJNAME         105  VARCHAR(24) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PROJNM           8    VARCHAR(24) IN RPT1
PROJNO          8    CHARACTER(6) IN RPT1
86
PROJNO          ****  COLUMN
                  68 68
PROJNO          68   CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
PROJNO          ****  COLUMN IN EMPPROJECT
                  105 105 105
PROJNO          ****  COLUMN IN PROJECT
                  105
PROJNO          105  CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PRSTAFF         105  DECIMAL(5,2) COLUMN IN CORPDATA.PROJECT
PRSTD           8    DATE(10) IN RPT1
PRSTDATE        105  DATE(10) COLUMN IN CORPDATA.PROJECT
RDATE          32   CHARACTER(10)
                  105
RESEM           8    CHARACTER(6) IN RPT1
RESPEMP        105  CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
RPTERR         151  LABEL
                  59
RPT1            8    STRUCTURE
RPT2           23   STRUCTURE
                  126
SALARY          19   DECIMAL(9,2)
86
SALARY          ****  COLUMN
                  48 48 68 105
SALARY          48   DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
SEX            48   CHARACTER(1) COLUMN IN CORPDATA.EMPLOYEE
STAFF          8    DECIMAL(5,2) IN RPT1
UPDERR         139  LABEL
                  45
WORKDEPT        48   CHARACTER(3) COLUMN IN CORPDATA.EMPLOYEE
WRKDAY         30   SMALL INTEGER PRECISION(4,0)
                  105

```

No errors found in source

196 Source records processed

* * * * * E N D O F L I S T I N G * * * * *

그림 6. SQL문을 사용하는 샘플 iSeries용 RPG 프로그램 (6/6)

예: iSeries용 ILE RPG 프로그램에서 SQL문

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

```
Source type.....RPG
Object name.....CORPDATA/RPGLEEX
Source file.....CORPDATA/SRC
Member.....*OBJ
To source file.....QTEMP/QSQLTEMP1
Options.....*XREF
Listing option.....*PRINT
Target release.....V5R2M0
INCLUDE file.....*LIBL/*SRCFILE
Commit.....*CHG
Allow copy of data.....*YES
Close SQL cursor.....*ENDMOD
Allow blocking.....*READ
Delay PREPARE.....*NO
Generation level.....10
Printer file.....*LIBL/QSYSPRT
Date format.....*JOB
Date separator.....*JOB
Time format.....*HMS
Time separator.....*JOB
Replace.....*YES
Relational database.....*LOCAL
User.....*CURRENT
RDB connect method.....*DUW
Default collection.....*NONE
Dynamic default
  collection.....*NO
Package name.....*OBJLIB/*OBJ
Path.....*NAMING
Created object type.....*PGM
Debugging view.....*NONE
User profile.....*NAMING
Dynamic user profile.....*USER
Sort sequence.....*JOB
Language ID.....*JOB
IBM SQL flagging.....*NOFLAG
ANS flagging.....*NONE
Text.....*SRCMBRTXT
Source file CCSID.....65535
Job CCSID.....65535
Source member changed on 07/01/96 15:55:32
```

```
1      H                                100
2      F* File declaration for QPRINT    200
3      F*                                300
4      FQPRINT    0    F 132          PRINTER    400
5      D*                                500
6      D* Structure for report 1.        600
7      D*                                700
8      1 DRPT1          E DS          EXTNAME(PROJECT)    800
9      D*                                900
10     D              DS              1000
11     D EMPNO          1      6      1100
12     D NAME          7      36      1200
13     D SALARY        37     41P 2    1300
14     D*                                1400
15     D* Structure for report 2.        1500
16     D*                                1600
17     DRPT2          DS              1700
18     D PRJNUM        1      6      1800
19     D PNAME         7      42      1900
20     D EMPCNT        43     44B 0    2000
21     D PRCOST        45     49P 2    2100
22     D*                                2200
23     D              DS              2300
24     D WRKDAY        1      2B 0     2400
25     D COMMI         3      6P 2     2500
26     D RDATE         7      16      2600
27     D PERCNT        17     20P 2    2700
28     *                                2800
```

그림 7. SQL문을 사용하는 샘플 iSeries용 ILE RPG 프로그램 (1/6)

```

5722ST1 V5R2M0 020719          Create SQL ILE RPG Object          RPGLEEX          08/06/02 16:03:02 Page 2
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change      Comments
29      2      C              Z-ADD      253          WRKDAY          2900
30      C              Z-ADD      2000.00      COMMI           3000
31      C              Z-ADD      1.04          PERCNT          3100
32      C              MOVE      '1982-06-'      RDATE           3200
33      C              MOVE      '01'          RDATE           3300
34      C              SETON                    LR             3400
35      C*                                3500
36      C* Update the selected projects by the new percentage. If an
37      C* error occurs during the update, ROLLBACK the changes.      3700
38      C*                                3800
39      3      C/EXEC SQL WHENEVER SQLERROR GOTO UPDERR          3900
40      C/END-EXEC                                4000
41      C*                                4100
42      C/EXEC SQL                                4200
43      4      C+ UPDATE CORPDATA/EMPLOYEE          4300
44      C+   SET SALARY = SALARY * :PERCNT          4400
45      C+   WHERE COMM >= :COMMI                 4500
46      C/END-EXEC                                4600
47      C*                                4700
48      C* Commit changes.                        4800
49      C*                                4900
50      5      C/EXEC SQL COMMIT                    5000
51      C/END-EXEC                                5100
52      C*                                5200
53      C/EXEC SQL WHENEVER SQLERROR GO TO RPTERR  5300
54      C/END-EXEC                                5400
55      C*                                5500
56      C* Report the updated statistics for each employee assigned to
57      C* selected projects.                      5700
58      C*                                5800
59      C* Write out the header for report 1.      5900
60      C*                                6000
61      C              EXCEPT RECA              6100
62      6      C/EXEC SQL DECLARE C1 CURSOR FOR          6200
63      C+   SELECT DISTINCT PROJNO, EMPPROJACT.EMPNO,
64      C+   LASTNAME||', '||FIRSTNME, SALARY
65      C+   FROM CORPDATA/EMPPROJACT, CORPDATA/EMPLOYEE
66      C+   WHERE EMPPROJACT.EMPNO = EMPLOYEE.EMPNO AND
67      C+   COMM >= :COMMI
68      C+   ORDER BY PROJNO, EMPNO              6800
69      C/END-EXEC                                6900
70      C*                                7000
71      7      C/EXEC SQL                                7100
72      C+ OPEN C1                                7200
73      C/END-EXEC                                7300
74      C*                                7400
75      C* Fetch and write the rows to QPRINT.    7500
76      C*                                7600
77      8      C/EXEC SQL WHENEVER NOT FOUND GO TO DONE1  7700
78      C/END-EXEC                                7800
79      C   SQLCOD      DOUNE      0              7900
80      C/EXEC SQL                                8000
81      9      C+ FETCH C1 INTO :PROJNO, :EMPNO, :NAME, :SALARY  8100
82      C/END-EXEC                                8200
83      C              EXCEPT RECB              8300
84      C              END                        8400
85      C   DONE1      TAG                        8500
86      C/EXEC SQL                                8600
87      10     C+ CLOSE C1                          8700
88      C/END-EXEC                                8800
89      C*                                8900
90      C* For all project ending at a date later than the raise date
91      C* (i.e. those projects potentially affected by the salary raises)
92      C* generate a report containing the project number, project name,
93      C* the count of employees participating in the project and the
94      C* total salary cost of the project.        9300
95      C*                                9400
96      C* Write out the header for report 2.      9500
97      C*                                9600
98      C              EXCEPT RECC              9700
99      C/EXEC SQL                                9800

```

12000

그림 7. SQL문을 사용하는 샘플 iSeries용 ILE RPG 프로그램 (2/6)

```

5722ST1 V5R2M0 020719          Create SQL ILE RPG Object          RPGLEEX          08/06/02 16:03:02 Page 3
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change Comments
100 11 C+ DECLARE C2 CURSOR FOR 10000
101 C+ SELECT EMPPROJECT.PROJNO, PROJNAME, COUNT(*), 10100
102 C+ SUM((DAYS(EMENDATE) - DAYS(EMSTDATE)) * EMPTIME * 10200
103 C+ DECIMAL((SALARY/:WRKDAY),8,2)) 10300
104 C+ FROM CORPDATA/EMPPROJECT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE 10400
105 C+ WHERE EMPPROJECT.PROJNO = PROJECT.PROJNO AND 10500
106 C+ EMPPROJECT.EMPNO = EMPLOYEE.EMPNO AND 10600
107 C+ PRENDATE > :RDATE 10700
108 C+ GROUP BY EMPPROJECT.PROJNO, PROJNAME 10800
109 C+ ORDER BY 1 10900
110 C/END-EXEC 11000
111 C* 11100
112 C/EXEC SQL OPEN C2 11200
113 C/END-EXEC 11300
114 C* 11400
115 C* Fetch and write the rows to QPRINT. 11500
116 C* 11600
117 C/EXEC SQL WHENEVER NOT FOUND GO TO DONE2 11700
118 C/END-EXEC 11800
119 C SQLCOD DOUNE 0 11900
120 C/EXEC SQL
121 12 C+ FETCH C2 INTO :RPT2 12100
122 C/END-EXEC 12200
123 C EXCEPT RECD 12300
124 C END 12400
125 C DONE2 TAG 12500
126 C/EXEC SQL CLOSE C2 12600
127 C/END-EXEC 12700
128 C RETURN 12800
129 C* 12900
130 C* Error ocured while updating table. Inform user and rollback 13000
131 C* changes. 13100
132 C* 13200
133 C UPDERR TAG 13300
134 C EXCEPT RECE 13400
135 13 C/EXEC SQL WHENEVER SQLERROR CONTINUE 13500
136 C/END-EXEC 13600
137 C* 13700
138 14 C/EXEC SQL 13800
139 C+ ROLLBACK 13900
140 C/END-EXEC 14000
141 C RETURN 14100
142 C* 14200
143 C* Error ocured while generating reports. Inform user and exit. 14300
144 C* 14400
145 C RPTERR TAG 14500
146 C EXCEPT RECF 14600
147 C* 14700
148 C* All done. 14800
149 C* 14900
150 C FINISH TAG 15000
151 QQPRINT E RECA 0 2 01 15100
152 0 42 'REPORT OF PROJECTS AFFEC' 15200
153 0 64 'TED BY EMPLOYEE RAISES' 15300
154 0 E RECA 0 1 15400
155 0 7 'PROJECT' 15500
156 0 17 'EMPLOYEE' 15600
157 0 32 'EMPLOYEE NAME' 15700
158 0 60 'SALARY' 15800
159 0 E RECB 0 1 15900
160 0 PROJNO 6 16000
161 0 EMPNO 15 16100
162 0 NAME 50 16200
163 0 SALARY L 61 16300
164 0 E RECC 2 2 16400
165 0 42 'ACCUMULATED STATISTIC' 16500
166 0 54 'S BY PROJECT' 16600

```

그림 7. SQL문을 사용하는 샘플 iSeries용 ILE RPG 프로그램 (3/6)

```

5722ST1 V5R2M0 020719      Create SQL ILE RPG Object      RPGLEEX      08/06/02 16:03:02 Page 4
167      0      E      RECC      0 1      16700
168      0      7 'PROJECT'      16800
169      0      56 'NUMBER OF'      16900
170      0      67 'TOTAL'      17000
171      0      E      RECC      0 2      17100
172      0      6 'NUMBER'      17200
173      0      21 'PROJECT NAME'      17300
174      0      56 'EMPLOYEES'      17400
175      0      66 'COST'      17500
176      0      E      RECD      0 1      17600
177      0      PRJNUM      6      17700
178      0      PNAME      45      17800
179      0      EMPCNT      L 54      17900
180      0      PRCOST      L 70      18000
181      0      E      RECE      0 1      18100
182      0      28 '*** ERROR Occurred while'      18200
183      0      52 ' updating table. SQLCODE'      18300
184      0      53 '='      18400
185      0      SQLCOD      L 62      18500
186      0      E      RECF      0 1      18600
187      0      28 '*** ERROR Occurred while'      18700
188      0      52 ' generating reports. SQL'      18800
189      0      57 'CODE='      18900
190      0      SQLCOD      L 67      19000
      * * * * * E N D O F S O U R C E * * * * *

```

그림 7. SQL문을 사용하는 샘플 iSeries용 ILE RPG 프로그램 (4/6)

CROSS REFERENCE

Data Names

	Define	Reference
ACTNO	62	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
BIRTHDATE	42	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
BONUS	42	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMM	****	COLUMN
		42 62
COMM	42	DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMMI	25	DECIMAL(7,2)
		42 62
CORPDATA	****	COLLECTION
		42 62 62 99 99 99
C1	62	CURSOR
		71 80 86
C2	99	CURSOR
		112 120 126
DEPTNO	8	CHARACTER(3) IN RPT1
DEPTNO	99	CHARACTER(3) COLUMN (NOT NULL) IN CORPDATA.PROJECT
DONE1	85	
DONE1	****	LABEL
77		
DONE2	125	
DONE2	****	LABEL
		117
EDLEVEL	42	SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMENDATE	62	DATE(10) COLUMN IN CORPDATA.EMPPROJACT
EMENDATE	****	COLUMN
		99
EMPCNT	20	SMALL INTEGER PRECISION(4,0) IN RPT2
EMPLOYEE	****	TABLE IN CORPDATA
		42 62 99
EMPLOYEE	****	TABLE
		62 99
EMPNO	11	CHARACTER(6) DBCS-open
80		
EMPNO	42	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMPNO	****	COLUMN IN EMPPROJACT
		62 62 62 99
EMPNO	****	COLUMN IN EMPLOYEE
		62 99
EMPNO	62	CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
EMPPROJACT	****	TABLE
		62 62 99 99 99 99
EMPPROJACT	****	TABLE IN CORPDATA
		62 99
EMPTIME	62	DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJACT
EMPTIME	****	COLUMN
		99
EMSTDATE	62	DATE(10) COLUMN IN CORPDATA.EMPPROJACT
EMSTDATE	****	COLUMN
		99
FINISH	150	
FIRSTNME	42	VARCHAR(12) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
FIRSTNME	****	COLUMN
		62
HIREDATE	42	DATE(10) COLUMN IN CORPDATA.EMPLOYEE
JOB	42	CHARACTER(8) COLUMN IN CORPDATA.EMPLOYEE
LASTNAME	42	VARCHAR(15) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
LASTNAME	****	COLUMN
		62
MAJPROJ	8	CHARACTER(6) IN RPT1
MAJPROJ	99	CHARACTER(6) COLUMN IN CORPDATA.PROJECT
MIDINIT	42	CHARACTER(1) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
NAME	12	CHARACTER(30) DBCS-open
80		
PERCNT	27	DECIMAL(7,2)
42		
PHONENO	42	CHARACTER(4) COLUMN IN CORPDATA.EMPLOYEE
PNAME	19	CHARACTER(36) DBCS-open IN RPT2
PRCOST	21	DECIMAL(9,2) IN RPT2
PRENDATE	8	DATE(8) IN RPT1
PRENDATE	****	COLUMN
		99
PRENDATE	99	DATE(10) COLUMN IN CORPDATA.PROJECT
PRJNUM	18	CHARACTER(6) DBCS-open IN RPT2

그림 7. SQL문을 사용하는 샘플 iSeries용 ILE RPG 프로그램 (5/6)

```

CROSS REFERENCE
PROJECT          ****  TABLE IN CORPDATA
                  99
PROJECT          ****  TABLE
                  99
PROJNAME         8     VARCHAR(24) IN RPT1
PROJNAME         ****  COLUMN
                  99 99
PROJNAME         99    VARCHAR(24) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PROJNO           8     CHARACTER(6) IN RPT1
80
PROJNO           ****  COLUMN
                  62 62
PROJNO           62    CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJECT
PROJNO           ****  COLUMN IN EMPPROJECT
                  99 99 99
PROJNO           ****  COLUMN IN PROJECT
                  99
PROJNO           99    CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PRSTAFF          8     DECIMAL(5,2) IN RPT1
PRSTAFF          99    DECIMAL(5,2) COLUMN IN CORPDATA.PROJECT
PRSTDATE         8     DATE(8) IN RPT1
PRSTDATE         99    DATE(10) COLUMN IN CORPDATA.PROJECT
RDATE            26    CHARACTER(10) DBCS-open
                  99
RESPEMP          8     CHARACTER(6) IN RPT1
RESPEMP          99    CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
RPTERR           145
RPTERR           ****  LABEL
                  53
RPT1             8     STRUCTURE
RPT2             17    STRUCTURE
120
SALARY           13    DECIMAL(9,2)
80
SALARY           ****  COLUMN
                  42 42 62 99
SALARY           42    DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
SEX              42    CHARACTER(1) COLUMN IN CORPDATA.EMPLOYEE
UPDERR           133
UPDERR           ****  LABEL
                  39
WORKDEPT         42    CHARACTER(3) COLUMN IN CORPDATA.EMPLOYEE
WRKDAY           24    SMALL INTEGER PRECISION(4,0)
                  99

```

No errors found in source

190 Source records processed

***** END OF LISTING *****

그림 7. SQL문을 사용하는 샘플 iSeries용 ILE RPG 프로그램 (6/6)

예: REXX 프로그램에서 SQL문

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

```

Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
1  /*****
2  /* A sample program which updates the salaries for those employees */
3  /* whose current commission total is greater than or equal to the */
4  /* value of COMMISSION. The salaries of those who qualify are */
5  /* increased by the value of PERCENTAGE, retroactive to RAISE_DATE. */
6  /* A report is generated and dumped to the display which shows the */
7  /* projects which these employees have contributed to, ordered by */
8  /* project number and employee ID. A second report shows each */
9  /* project having an end date occurring after RAISE DATE (i.e. is */
10 /* potentially affected by the retroactive raises) with its total */
11 /* salary expenses and a count of employees who contributed to the */
12 /* project.
13  *****/
14
15
16 /* Initialize RC variable */
17 RC = 0
18
19 /* Initialize HV for program usage */
20 COMMISSION = 2000.00;
21 PERCENTAGE = 1.04;
22 RAISE_DATE = '1982-06-01';
23 WORK_DAYS = 253;
24
25 /* Create the output file to dump the 2 reports. Perform an OVRDBF */
26 /* to allow us to use the SAY REXX command to write to the output */
27 /* file.
28  ADDRESS '*COMMAND',
29  'DLTF FILE(CORPDATA/REPORTFILE)'
30  ADDRESS '*COMMAND',
31  'CRTPF FILE(CORPDATA/REPORTFILE) RCDLEN(80)'
32  ADDRESS '*COMMAND',
33  'OVRDBF FILE(STDOUT) TOFILE(CORPDATA/REPORTFILE) MBR(REPORTFILE)'
34
35 /* Update the selected employee's salaries by the new percentage. */
36 /* If an error occurs during the update, ROLLBACK the changes. */
37 3 SIGNAL ON ERROR
38  ERRLOC = 'UPDATE_ERROR'
39  UPDATE_STMT = 'UPDATE CORPDATA/EMPLOYEE ',
40  'SET SALARY = SALARY * ? ',
41  'WHERE COMM >= ? '
42  EXECSQL,
43  'PREPARE S1 FROM :UPDATE_STMT'
44  4 EXECSQL,
45  'EXECUTE S1 USING :PERCENTAGE,',
46  ':COMMISSION '
47 /* Commit changes */
48 5 EXECSQL,
49  'COMMIT'
50  ERRLOC = 'REPORT_ERROR'
51
52 /* Report the updated statistics for each project supported by one */
53 /* of the selected employees.
54
55 /* Write out the header for Report 1 */
56 SAY ' '
57 SAY ' '
58 SAY ' '
59 SAY ' REPORT OF PROJECTS AFFECTED BY EMPLOYEE RAISES'
60 SAY ' '
61 SAY 'PROJECT EMPID EMPLOYEE NAME SALARY'
62 SAY '-----'
63 SAY ' '
64
65 SELECT_STMT = 'SELECT DISTINCT PROJNO, EMPPROJACT.EMPNO, ',
66 ' LASTNAME||', '||FIRSTNAME, SALARY ',
67 'FROM CORPDATA/EMPPROJACT, CORPDATA/EMPLOYEE ',
68 'WHERE EMPPROJACT.EMPNO = EMPLOYEE.EMPNO AND ',
69 ' COMM >= ? ',
70 'ORDER BY PROJNO, EMPNO '
71 EXECSQL,
72 'PREPARE S2 FROM :SELECT_STMT'
73 6 EXECSQL,
74 'DECLARE C1 CURSOR FOR S2'

```

그림 8. SQL문을 사용하는 REXX 프로시저의 샘플 (1/3)


```

Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
75 7 EXECESQL,
76 'OPEN C1 USING :COMMISSION'
77
78 /* Handle the FETCH errors and warnings inline */
79 SIGNAL OFF ERROR
80
81 /* Fetch all of the rows */
82 DO UNTIL (SQLCODE <> 0)
83 9 EXECESQL,
84 'FETCH C1 INTO :RPT1.PROJNO, :RPT1.EMPNO,',
85 ' :RPT1.NAME, :RPT1.SALARY '
86
87 /* Process any errors that may have occurred. Continue so that */
88 /* we close the cursor for any warnings. */
89 IF SQLCODE < 0 THEN
90 SIGNAL ERROR
91
92 /* Stop the loop when we hit the EOF. Don't try to print out the */
93 /* fetched values. */
94 8 IF SQLCODE = 100 THEN
95 LEAVE
96
97 /* Print out the fetched row */
98 SAY RPT1.PROJNO ' ' RPT1.EMPNO ' ' RPT1.NAME ' ' RPT1.SALARY
99 END;
100
101 10 EXECESQL,
102 'CLOSE C1'
103
...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
104 /* For all projects ending at a date later than 'raise_date' */
105 /* (i.e. those projects potentially affected by the salary raises) */
106 /* generate a report containing the project number, project name */
107 /* the count of employees participating in the project and the */
108 /* total salary cost of the project. */
109
110 /* Write out the header for Report 2 */
111 SAY ' '
112 SAY ' '
113 SAY ' '
114 SAY ' ACCUMULATED STATISTICS BY PROJECT'
115 SAY ' '
116 SAY 'PROJECT PROJECT NAME NUMBER OF TOTAL'
117 SAY 'NUMBER EMPLOYEES COST'
118 SAY '-----'
119 SAY ' '
120
121
122 /* Go to the common error handler */
123 SIGNAL ON ERROR
124
125 SELECT_STMT = 'SELECT EMPPROJACT.PROJNO, PROJNAME, COUNT(*), ',
126 ' SUM( (DAYS(EMENDATE) - DAYS(EMSTDATE)) * EMPTIME * ',
127 ' DECIMAL(( SALARY / ? ),8,2) ) ',
128 'FROM CORPDATA/EMPPROJACT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE',
129 'WHERE EMPPROJACT.PROJNO = PROJECT.PROJNO AND ',
130 ' EMPPROJACT.EMPNO = EMPLOYEE.EMPNO AND ',
131 ' PRENDATE > ? ',
132 'GROUP BY EMPPROJACT.PROJNO, PROJNAME ',
133 'ORDER BY 1 '
134
135 EXECESQL,
136 'PREPARE S3 FROM :SELECT_STMT'
137 11 EXECESQL,
138 'DECLARE C2 CURSOR FOR S3'
139 EXECESQL,
140 'OPEN C2 USING :WORK_DAYS, :RAISE_DATE'
141
142 /* Handle the FETCH errors and warnings inline */
143 SIGNAL OFF ERROR
144
145 /* Fetch all of the rows */
146 DO UNTIL (SQLCODE <> 0)

```

그림 8. SQL문을 사용하는 REXX 프로시저어의 샘플 (2/3)

```

Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
146      12 EXECSQL,
147          'FETCH C2 INTO :RPT2.PROJNO, :RPT2.PROJNAME, ',
148          ':RPT2.EMPCOUNT, :RPT2.TOTAL_COST '
149
150      /* Process any errors that may have occurred. Continue so that */
151      /* we close the cursor for any warnings. */
152      IF SQLCODE < 0 THEN
153          SIGNAL ERROR
154
155      /* Stop the loop when we hit the EOF. Don't try to print out the */
156      /* fetched values. */
157      IF SQLCODE = 100 THEN
158          LEAVE
159
160      /* Print out the fetched row */
161      SAY RPT2.PROJNO ' ' RPT2.PROJNAME ' ',
162          RPT2.EMPCOUNT ' ' RPT2.TOTAL_COST
163  END;
164
165  EXECSQL,
166      'CLOSE C2'
167
168      /* Delete the OVRDBF so that we will continue writing to the output */
169      /* display. */
170      ADDRESS '*COMMAND',
171          'DLTOVR FILE(STDOUT)'
172
173      /* Leave procedure with a successful or warning RC */
174      EXIT RC
175
176
177      /* Error occurred while updating the table or generating the */
178      /* reports. If the error occurred on the UPDATE, rollback all of */
179      /* the changes. If it occurred on the report generation, display the */
180      /* REXX RC variable and the SQLCODE and exit the procedure. */
181      ERROR:
182
183      13 SIGNAL OFF ERROR
184
185      /* Determine the error location */
186      SELECT
187          /* When the error occurred on the UPDATE statement */
188          WHEN ERRLOC = 'UPDATE_ERROR' THEN
189              DO
190                  SAY '*** ERROR Occurred while updating table.',
191                      'SQLCODE = ' SQLCODE
192                  14 EXECSQL,
193                      'ROLLBACK'
194              END
195
196          /* When the error occurred during the report generation */
197          WHEN ERRLOC = 'REPORT_ERROR' THEN
198              SAY '*** ERROR Occurred while generating reports. ',
199                  'SQLCODE = ' SQLCODE
200          OTHERWISE
201              SAY '*** Application procedure logic error occurred '
202          END
203
204      /* Delete the OVRDBF so that we will continue writing to the */
205      /* output display. */
206      ADDRESS '*COMMAND',
207          'DLTOVR FILE(STDOUT)'
208
209      /* Return the error RC received from SQL. */
210      EXIT RC
211
212          * * * * * E N D   O F   S O U R C E * * * * *

```

SQL을 사용하는 샘플 프로그램에 의해 작성된 보고서

다음 보고서는 각 샘플 프로그램에 의해 작성된 보고서입니다.

REPORT OF PROJECTS AFFECTED BY RAISES

PROJECT	EMPID	EMPLOYEE NAME	SALARY
AD3100	000010	HAAS, CHRISTINE	54860.00
AD3110	000070	PULASKI, EVA	37616.80
AD3111	000240	MARINO, SALVATORE	29910.40
AD3113	000270	PEREZ, MARIA	28475.20
IF1000	000030	KWAN, SALLY	39780.00
IF1000	000140	NICHOLLS, HEATHER	29556.80
IF2000	000030	KWAN, SALLY	39780.00
IF2000	000140	NICHOLLS, HEATHER	29556.80
MA2100	000010	HAAS, CHRISTINE	54860.00
MA2100	000110	LUCCHESSI, VICENZO	48360.00
MA2110	000010	HAAS, CHRISTINE	54860.00
MA2111	000200	BROWN, DAVID	28849.60
MA2111	000220	LUTZ, JENNIFER	31033.60
MA2112	000150	ADAMSON, BRUCE	26291.20
OP1000	000050	GEYER, JOHN	41782.00
OP1010	000090	HENDERSON, EILEEN	30940.00
OP1010	000280	SCHNEIDER, ETHEL	27300.00
OP2010	000050	GEYER, JOHN	41782.00
OP2010	000100	SPENSER, THEODORE	27196.00
OP2012	000330	LEE, WING	26384.80
PL2100	000020	THOMPSON, MICHAEL	42900.00

ACCUMULATED STATISTICS BY PROJECT

PROJECT NUMBER	PROJECT NAME	NUMBER OF EMPLOYEES	TOTAL COST
AD3100	ADMIN SERVICES	1	19623.11
AD3110	GENERAL ADMIN SYSTEMS	1	58877.28
AD3111	PAYROLL PROGRAMMING	7	66407.56
AD3112	PERSONNEL PROGRAMMING	9	28845.70
AD3113	ACCOUNT PROGRAMMING	14	72114.52
IF1000	QUERY SERVICES	4	35178.99
IF2000	USER EDUCATION	5	55212.61
MA2100	WELD LINE AUTOMATION	2	114001.52
MA2110	W L PROGRAMMING	1	85864.68
MA2111	W L PROGRAM DESIGN	3	93729.24
MA2112	W L ROBOT DESIGN	6	166945.84
MA2113	W L PROD CONT PROGS	5	71509.11
OP1000	OPERATION SUPPORT	1	16348.86
OP1010	OPERATION	5	167828.76
OP2010	SYSTEMS SUPPORT	2	91612.62
OP2011	SCP SYSTEMS SUPPORT	2	31224.60
OP2012	APPLICATIONS SUPPORT	2	41294.88
OP2013	DB/DC SUPPORT	2	37311.12
PL2100	WELD LINE PLANNING	1	43576.92

부록 B. 호스트 언어 사전컴파일러에 대한 iSeries용 DB2 UDB CL 명령 설명

이 부록에는 본 안내서 및 SQL 참조서에 나오는 구문 다이어그램이 들어 있습니다.

자세한 내용은 『SQL 사전컴파일러 명령』을 참조하십시오.

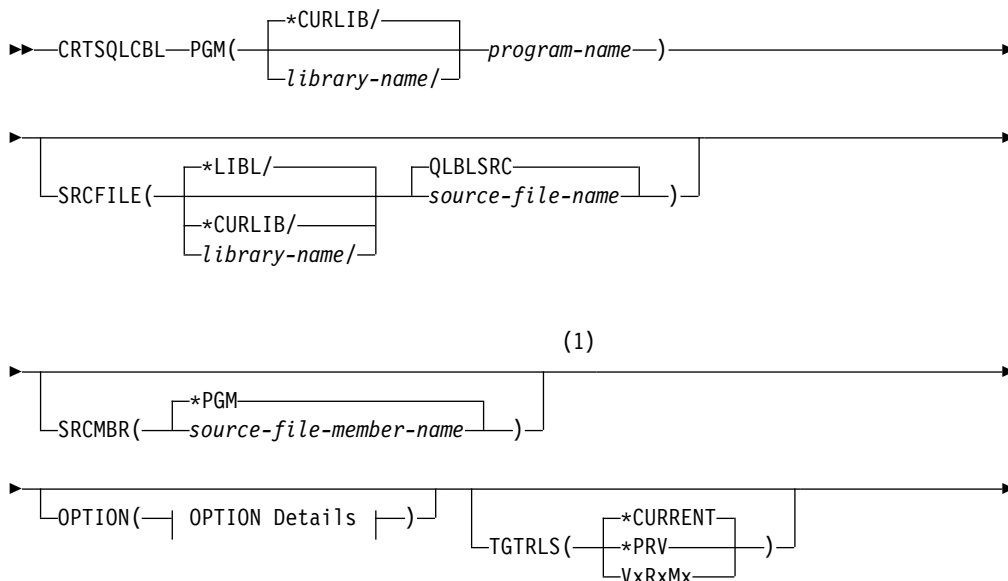
SQL 사전컴파일러 명령

iSeries용 DB2 UDB는 다음 프로그래밍 언어에서 코딩된 사전컴파일러 프로그램에 대한 명령을 제공합니다.

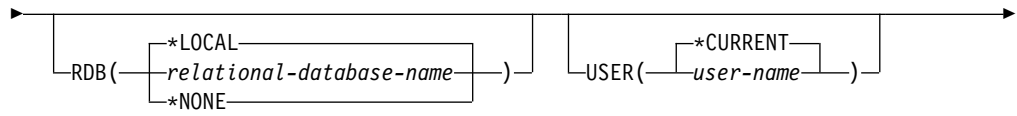
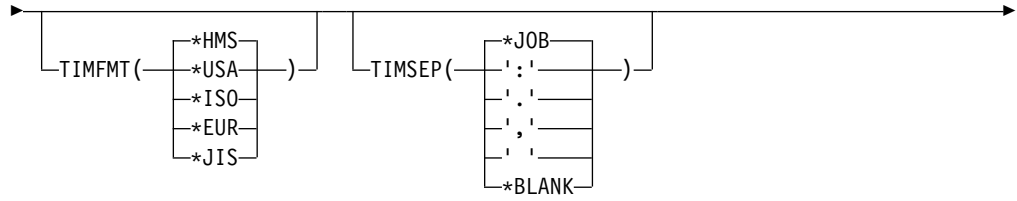
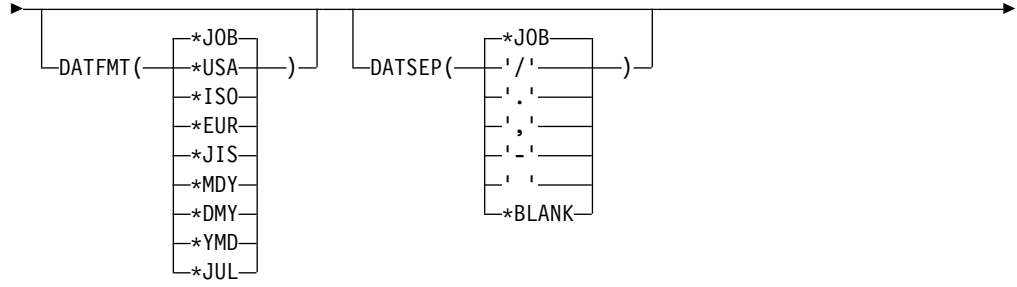
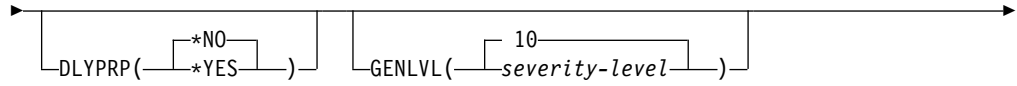
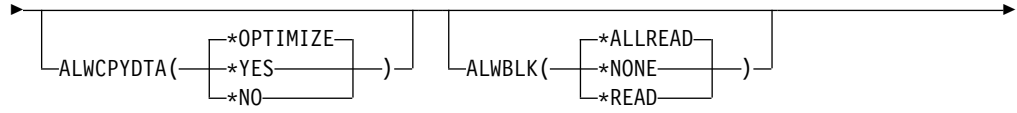
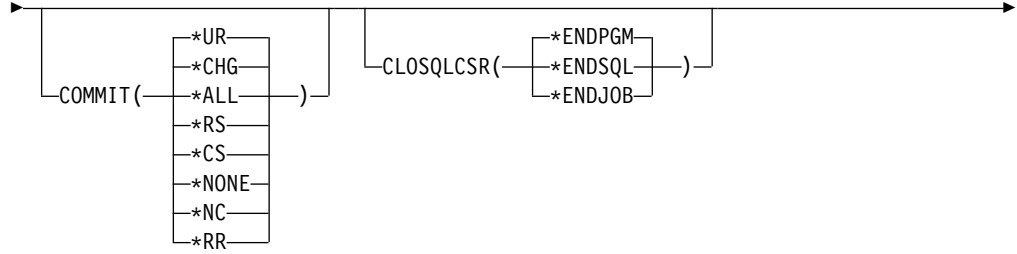
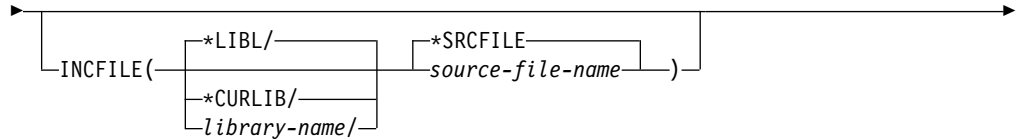
- COBOL
- ILE COBOL
- ILE C
- C++
- PL/I
- RPG
- ILE RPG

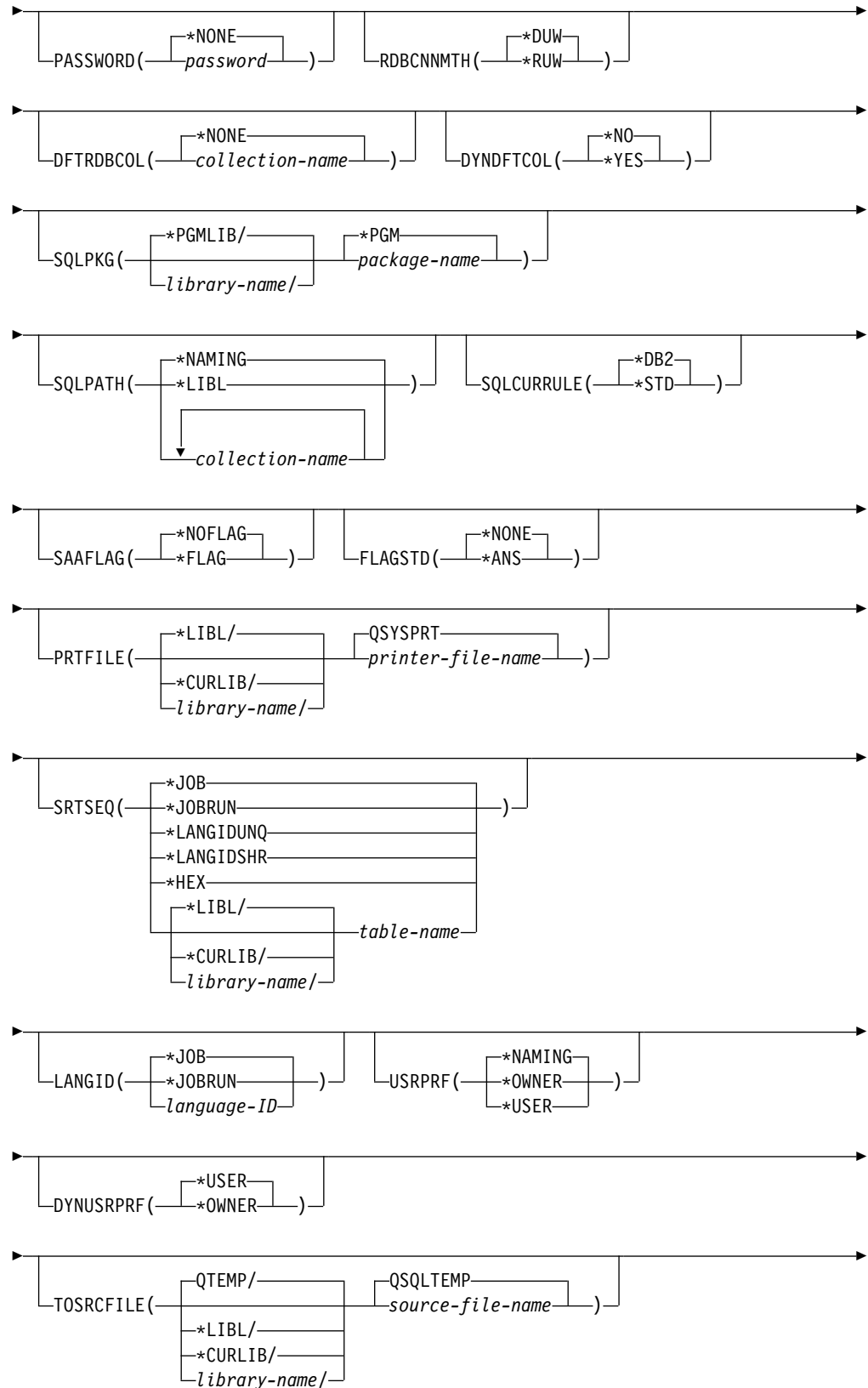
CRTSQLCBL(구조화 조회 언어 COBOL 작성) 명령

Job: B,I Pgm: B,I REXX: B,I Exec

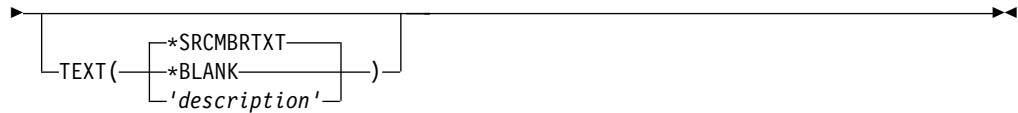


CRTSQLCBL

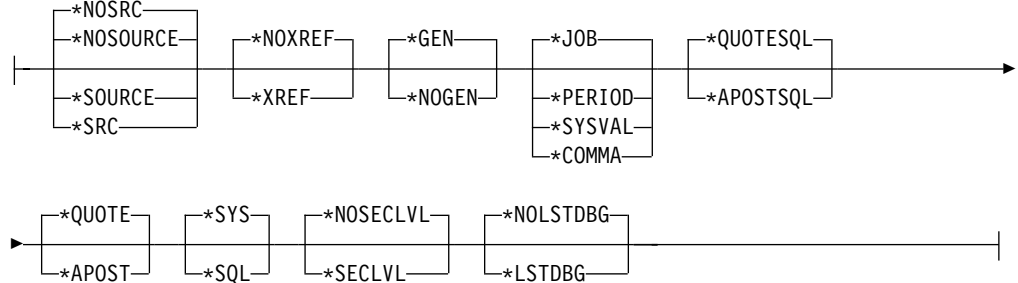




CRTSQLCBL



OPTION 세부사항:



주:

- 1 이 앞에 오는 모든 매개변수는 앞뒤 관계에 따라 지정할 수 있습니다.

목적:

CRTSQLCBL(구조화 조회 언어 COBOL 작성) 명령은 SQL문이 들어 있는 COBOL 소스문을 사전컴파일하는 구조화 조회 언어(SQL) 사전컴파일러를 호출하여 임시 소스 멤버를 작성한 후, 선택적으로 프로그램을 호출하기 위해 COBOL 컴파일러를 호출합니다.

매개변수:

PGM

컴파일된 프로그램의 규정화명을 지정합니다.

컴파일된 COBOL 프로그램의 이름은 다음 라이브러리 값 중 하나로 규정될 수 있습니다.

***CURLIB** 컴파일된 COBOL 프로그램이 해당 작업에 대한 현재 라이브러리에 작성됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library name: 컴파일된 COBOL 프로그램이 작성된 라이브러리의 이름을 지정하십시오.

program-name: 컴파일된 COBOL 프로그램의 이름을 지정하십시오.

SRCFILE

SQL문과 함께 COBOL 소스문이 들어 있는 소스 파일의 규정된 이름을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QLBLSRC: COBOL 소스 파일명이 지정되지 않은 경우 IBM 제공 소스 파일 QLBLSRC에 COBOL 소스가 포함됩니다.

source-file-name: COBOL 소스가 들어 있는 소스 파일의 이름을 지정하십시오. 이 소스 파일의 레코드 길이는 92바이트입니다. 소스 파일은 데이터베이스 파일, 장치 파일 또는 인라인 자료 파일이 될 수 있습니다.

SRCMBR

COBOL 소스가 들어 있는 소스 파일 멤버명을 지정합니다. 이 매개변수는 SRCFILE 매개변수의 소스 파일명이 데이터베이스 파일일 경우에만 지정됩니다. 이 매개변수가 지정되지 않으면 PGM 매개변수에 지정된 PGM명이 사용됩니다.

***PGM:** COBOL 소스가 PGM 매개변수에 지정된 것과 동일한 이름을 가진 소스 파일 멤버에 있음을 지정합니다.

source-file-member-name: COBOL 소스를 포함하는 멤버의 이름을 지정하십시오.

OPTION

COBOL 소스문이 사전컴파일될 때 다음 옵션 중 하나 또는 여러 개가 사용되는지의 여부를 지정합니다. 옵션이 두 번 이상 지정되면 마지막에 지정된 옵션이 사용됩니다.

요소 1: 소스 리스팅 옵션

***NOSOURCE** 또는 ***NOSRC**: 사전컴파일러나 패키지 작성 중에 오류가 검출되지 않는 한, 소스 인쇄 출력이 사전컴파일러에 의해 작성되지 않습니다.

***SOURCE** 또는 ***SRC**: 사전컴파일러가 COBOL 소스 입력으로 구성된 소스 인쇄 출력을 작성합니다.

요소 2: 상호 참조 옵션

***NOXREF**: 사전컴파일러가 상호 참조명을 지정하지 않습니다.

***XREF**: 사전컴파일러는 그러한 항목을 참조하는 프로그램 내의 명령문 번호 항목을 상호 참조합니다.

요소 3: 프로그램 작성 옵션

***GEN**: 컴파일러는 프로그램 컴파일 후 수행될 수 있는 프로그램을 작성합니다. SQL 패키지 오브젝트는 데이터베이스명이 RDB 매개변수에 지정된 경우에 작성됩니다.

***NOGEN**: 사전컴파일러가 COBOL 컴파일러를 호출하지 않고, 프로그램과 SQL 패키지가 작성되지 않습니다.

요소 4: 소수점 옵션

***JOB**: SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 사전컴파일러 작업에 대해 지정된 소수점 표현입니다.

***SYSVAL**: SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 QDECFMT 시스템 값입니다.

주: QDECFMT가 소수점으로 사용되는 값을 쉼표로 지정하면 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)를 뒤에 공백이 오는 쉼표해야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 소수점이 마침표인 VALUES(1.1,2.23,4.1)와 동일합니다.

***PERIOD**: SQL문에 있는 숫자 상수의 소수점으로 사용되는 값이 마침표입니다.

***COMMA**: SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 쉼표입니다.

주: 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)를 뒤에 공백이 오는 쉼표로 분리해야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 마침표가 소수점으로 지정된 VALUES(1.1,2.23,4.1)와 동일합니다.

요소 5: 스트링 분리문자 옵션

***QUOTESQL**: SQL문에서 큰 따옴표(")가 스트링 분리문자입니다.

***APOSTSQL:** SQL문에서 작은 따옴표(')가 스트링 분리문자입니다.

요소 6: 리터럴 옵션

***QUOTE:** 큰 따옴표(")가 COBOL문에서 비슷자 리터럴과 부울 리터럴에 사용됩니다.

***APOST:** 작은 따옴표(')는 COBOL문에서 비슷자 리터럴과 부울 리터럴에 사용됩니다.

요소 7: 명명 규칙 옵션

***SYS:** 시스템 명명 규칙(library-name/file-name)이 사용됩니다.

***SQL:** SQL 명명 규칙(schema-name.table-name)이 사용됩니다. iSeries 시스템 이외의 리모트 데이터베이스에서 프로그램을 작성할 때 *SQL은 명명 규칙을 따라 지정해야 합니다.

요소 8: 2차 레벨 메세지 텍스트 옵션

***NOSECLVL:** 2차 레벨 텍스트 설명이 인쇄 출력에 추가되지 않습니다.

***SECLVL:** 대체 자료를 가진 2차 레벨 텍스트가 리스팅의 모든 메세지에 대해 추가됩니다.

요소 9: 디버그 리스팅 뷰

***NOLSTDBG:** 오류 및 디버그 정보가 생성되지 않습니다.

***LSTDBG:** SQL 사전컴파일러는 리스트 뷰, 오류 및 이 뷰에 필요한 디버그 정보를 생성합니다. 프로그램을 컴파일하기 위해 CODE/400 제품을 사용중인 경우에만 *LSTDBG를 사용할 수 있습니다.

TGTRLS

작성될 오브젝트를 사용하려고 하는 오퍼레이팅 시스템의 릴리스 레벨을 지정합니다.

*CURRENT와 *PRV 값이 주어질 때와 릴리스 레벨 값을 지정할 때에는 VxRxMx의 형식으로 릴리스를 지정하게 되며, 여기서 Vx는 버전, Rx는 릴리스, Mx는 수정 레벨을 나타냅니다. 예를 들어, V2R3M0은 버전 2, 릴리스 3, 수정 레벨 0을 나타냅니다.

***CURRENT:** 오브젝트는 사용자 시스템에서 현재 실행 중인 오퍼레이팅 시스템의 릴리스에 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *CURRENT는 V2R3M5가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

주: 시스템에서 V2R3M5가 실행되고 있으며, V2R3M0가 설치된 시스템에서 오브젝트를 사용하려는 경우에는 TGTRLS(*CURRENT)가 아니고 TGTRLS(V2R3M0)를 지정해야 합니다.

***PRV:** 오브젝트는 오퍼레이팅 시스템의 이전 릴리스, 수정 레벨 0에서 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *PRV는 V2R2M0가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

release-level: VxRxMx 형식의 릴리스를 지정하십시오. 지정된 릴리스의 시스템이나 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스의 시스템에서도 오브젝트를 사용할 수 있습니다.

유효한 값들은 현재 버전, 릴리스, 수정 레벨에 기초하고 있으며, 새로운 릴리스로 변경될 수 있습니다. 이 명령이 지원하는 가장 이전의 릴리스 레벨보다 더 앞의 릴리스 레벨을 지정하면 지원되는 릴리스 레벨 가운데 가장 이전의 릴리스가 표시된 오류 메시지를 받게 됩니다.

INCFILE

SQL INCLUDE문을 사용하여 프로그램에 포함시킨 멤버에 들어 있는 소스 파일의 규정화명을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리명을 지정하십시오.

***SRCFILE:** SRCFILE 매개변수에 지정된 규정된 소스 파일은 SQL INCLUDE문에 지정된 소스 파일 멤버를 가지고 있습니다.

source-file-name: SQL INCLUDE문에 지정된 소스 파일 멤버를 포함하는 소스 파일의 이름을 지정하십시오. 여기에서 지정된 소스 파일의 레코드 길이는 SRCFILE 매개변수에 지정된 소스 파일의 레코드 길이보다 짧아야 합니다.

COMMIT

컴파일된 프로그램의 SQL문이 확약 제어하에서 실행되는지 여부를 지정합니다. 호스트 언어 소스문에 참조된 파일은 이 옵션의 영향을 받지 않습니다. SQL문에 참조된 SQL 표, SQL 뷰 및 SQL 패키지만이 영향을 받습니다.

주: COBOL 소스문에서 참조된 파일은 이 옵션의 영향을 받지 않습니다.

***CHG 또는 *UR:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON 및 REVOKE문에서 참조된 오브젝트들을 지정하며, 갱신,

삭제 및 삽입된 행들은 작업 단위(트랜잭션)가 종료될 때까지 잠겨 있습니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 있습니다.

***ALL 또는 *RS:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입된 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 없습니다.

***CS:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 선택은 되었지만 갱신되지 않은 행은 다음 행이 선택될 때까지 잠깁니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 없습니다.

***NONE 또는 *NC:** 예약 제어가 사용되지 않음을 지정합니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 있습니다. SQL DROP SCHEMA문이 프로그램에 포함되는 경우에는 *NONE 또는 *NC가 반드시 사용되어야 합니다. RDB 매개변수에 관계형 데이터베이스가 지정되어 있고 관계형 데이터베이스가 iSeries가 아닌 시스템에 있으면, *NONE 또는 *NC를 지정할 수 없습니다.

***RR:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 없습니다. SELECT, UPDATE, DELETE 및 INSERT문에 참조된 모든 표는 작업 단위(트랜잭션)가 종료될 때까지 배타적으로 잠겨 있습니다.

CLOSQLCSR

SQL 커서가 내재적으로 닫히고 SQL 준비 명령문이 내재적으로 취소되고 LOCK TABLE 잠금이 해제되는 시기를 지정합니다. SQL 커서는 CLOSE, COMMIT 또는 ROLLBACK(HOLD 없이) SQL문을 발행할 때 명시적으로 닫힙니다.

***ENDPGM:** 프로그램 종료시 SQL 커서는 닫히고 SQL 준비 명령문은 취소됩니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 LOCK TABLE 잠금이 해제됩니다.

***ENDSQL:** SQL 커서가 호출 사이에 열린 상태로 남아 있고 다른 SQL OPEN을 실행하지 않고 폐지될 수 있습니다. 호출 스택상의 보다 높은 프로그램 중 하나가 최소한 하나의 SQL문을 실행해야 합니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 SQL 커서가 닫히고 준비된 SQL문이 취소되며, LOCK TABLE 잠금이 해제됩니다. *ENDSQL이 호출된 첫 번째 SQL 프로그램(호출 스택의 첫 번째 SQL 프로그램)인 프로그램에 대해 지정되는 경우 프로그램은 *ENDPGM이 지정된 것처럼 처리됩니다.

***ENDJOB:** SQL 커서가 호출 사이에 열린 상태로 남아 있고 다른 SQL OPEN 을 실행하지 않고 폐지될 수 있습니다. 호출 스택에 나중에 들어간 프로그램은 SQL 문을 실행할 필요가 없습니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 SQL 커서는 열린 상태로 남아있고, SQL 준비 명령문이 보존되며, LOCK TABLE 잠금이 보유됩니다. SQL 커서가 닫히고, SQL 준비 명령문은 취소되며 작업 종료시 LOCK TABLE 잠금이 해제됩니다.

ALWCOPYDTA

자료의 사본이 SELECT문에 사용될 수 있는지 여부를 지정합니다.

***OPTIMIZE:** 시스템은 데이터베이스에서 직접 검색된 자료를 사용할지 또는 자료의 사본을 사용할지의 여부를 판별합니다. 결정은 최상의 성능을 제공할 수 있는 방법에 근거하여 이루어집니다. COMMIT가 *CHG 또는 *CS이면 ALWBLK가 *ALLREAD가 아니거나, COMMIT가 *ALL 또는 *RR이면 조회 수행에 필요한 경우에만 자료의 사본이 사용됩니다.

***YES:** 필요할 때만 자료의 사본이 사용됩니다.

***NO:** 자료의 사본은 허용되지 않습니다. 조회를 수행하는 데 자료의 임시 사본이 필요하면 오류 메시지가 표시됩니다.

ALWBLK

데이터베이스 관리자가 레코드 블록화를 사용할 수 있는지 여부와 블록화가 읽기 전용 커서에 사용될 수 있는 확장 범위를 지정합니다.

***ALLREAD:** *NONE 또는 *CHG가 COMMIT 매개변수에 지정된 경우 읽기 전용 커서에 대해 행이 블록화됩니다. 명시적으로 갱신될 수 없는 프로그램 내의 모든 커서들은 EXECUTE 또는 EXECUTE IMMEDIATE문이 프로그램 내에 있을 지라도 읽기 전용으로 열립니다.

***ALLREAD 지정:**

- *READ에 대해 허용된 블록화 이외의 제약 레벨 *CHG하에서 레코드의 블록화를 허용합니다.
- 프로그램에서 거의 모든 읽기 전용 커서의 성능이 향상될 수 있으나 다음과 같은 방법으로 조회를 제한합니다.
 - 롤백(ROLLBACK) 명령, 호스트 언어의 ROLLBACK문 또는 ROLLBACK HOLD SQL문은 *ALLREAD가 지정된 경우 읽기 전용 커서의 위치를 재 지정하지 않습니다.
 - 커서에 대한 DECLARE문에 FOR UPDATE절이 없는 한, 지정된 UPDATE 또는 DELETE문의 동적 수행(예를 들면, EXECUTE IMMEDIATE 사용)은 커서의 행을 갱신하는 데 사용될 수 없습니다.

***NONE:** 커서에 대한 자료의 검색을 위해 행이 블록화되지 않습니다.

***NONE을 지정하는 경우:**

- 검색된 자료가 현재의 자료임을 보장합니다.
- 조회시 자료의 첫 번째 행을 검색하는 데 필요한 시간을 줄일 수 있습니다.
- 조회가 닫히기 전에 조회의 처음 몇 행만이 검색될 때 프로그램에 의해 사용되지 않은 자료 행 블록의 검색으로부터 데이터베이스 관리자를 중단시킵니다.
- 많은 행을 검색하는 조회의 전반적인 성능이 저하될 수 있습니다.

***READ:** 레코드는 다음의 경우 커서에 대한 자료의 읽기 전용 검색에 대해 블록화됩니다.

- *NONE이 COMMIT 매개변수에 지정되어 확약 제어가 사용되지 않음을 표시하는 경우
- FOR READ ONLY절로 커서가 선언되었거나 커서에 대해 지정된 UPDATE 또는 DELETE문을 수행할 수 있는 동적 명령문이 없는 경우

*READ를 지정하면 위의 조건을 만족하는 조회의 전반적인 성능이 향상되고 많은 수의 레코드를 검색할 수 있습니다.

DLYPRP

PREPARE문에 대한 동적 명령문 유효성 검사가 OPEN, EXECUTE 또는 DESCRIBE문 실행시까지 지연되는지 여부를 지정합니다. 유효성 검사가 지연되면 불필요한 확인이 제거되므로 성능이 향상됩니다.

***NO:** 동적 명령문 유효성 검사가 지연되지 않습니다. 동적 명령문이 준비될 때 액세스 계획의 유효성이 확인됩니다. 동적 명령문이 OPEN 또는 EXECUTE문에 사용될 때 액세스 계획의 유효성이 재확인됩니다. 동적 명령문에 의해 참조되는 오브젝트의 존재 또는 권한으로 인해, 동적 명령문이 아직 유효한지 확인하기 위해 OPEN 또는 EXECUTE문을 발행하기 전에 SQLCODE 또는 SQLSTATE를 체크해야 합니다.

***YES:** 동적 명령문 유효성 검사는 동적 명령문이 OPEN, EXECUTE 또는 DESCRIBE SQL문에 사용될 때까지 지연됩니다. 동적 명령문이 사용될 때 유효성 검사가 완료되고 액세스 계획이 수립됩니다. 이 매개변수에 *YES를 지정하면 OPEN, EXECUTE 또는 DESCRIBE문을 실행한 후에 SQLCODE와 SQLSTATE를 체크하여 동적 명령문이 유효한지 확인해야 합니다.

주: *YES를 지정하면 PREPARE에 INTO절이 사용되거나, 동적 명령문에 대한 OPEN문이 발행되기 전에 DESCRIBE문에서 그 동적 명령문을 사용하는 경우 성능은 향상되지 않습니다.

GENLVL

작성 조작이 실패하는 심각도 레벨을 지정합니다. 이 값 이상의 심각도 레벨을 가지는 오류가 발생하면 조작이 종료됩니다.

10: 디폴트 심각도 레벨은 10입니다.

심각도 레벨: 0-40 범위 내의 값을 지정하십시오.

DATFMT

결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 날짜 필드가 지정된 형식으로 리턴됩니다. 입력 날짜 스트링의 경우 지정된 값은 유효한 형식으로 날짜가 지정되었는지 여부를 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 iSeries 시스템이 아닌 시스템에 있으면, *USA, *ISO, *EUR 또는 *JIS가 지정되어야 합니다.

***JOB:** 작업에 지정된 형식이 사용됩니다. 작업 표시(DSPJOB) 명령을 사용하여 작업에 대한 현재 날짜 형식을 결정하십시오.

***USA:** 미국식 날짜 형식(mm/dd/yyyy)이 사용됩니다.

***ISO:** 국제 표준화 기구(ISO) 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***EUR:** 유럽식 날짜 형식(dd.mm.yyyy)이 사용됩니다.

***JIS:** 일본 산업 표준 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***MDY:** 날짜 형식(mm/dd/yy)이 사용됩니다.

***DMY:** 날짜 형식(dd/mm/yy)이 사용됩니다.

***YMD:** 날짜 형식(yy/mm/dd)이 사용됩니다.

***JUL:** 율리우스력 날짜 형식(yy/ddd)이 사용됩니다.

DATSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *JOB, *MDY, *DMY, *YMD 또는 *JUL이 DATFMT 매개변수에 지정되어 있을 때만 적용됩니다.

***JOB:** 사전검파일시 작업에 지정된 날짜 분리자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘/’: 슬래시(/)가 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘-’: 붙임표(-)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

TIMFMT

결과 열에 액세스하기 위해 사용되는 형식을 지정합니다. 입력 시간 스트링의 경우 지정된 값은 시간이 유효한 형식으로 지정되었는지 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 또 다른 iSeries 시스템이 아닌 시스템에 있으면, 시간 형식이 콜론이나 마침표의 시간 분리문자가 있는 *USA, *ISO, *EUR, *JIS 또는 *HMS이어야 합니다.

***HMS:** (hh:mm:ss) 형식이 사용됩니다.

***USA:** 미국식 시간 형식(hh:mm xx)이 사용되는데, 여기서 xx는 AM(오전) 또는 PM(오후)입니다.

***ISO:** 국제 표준화 기구(ISO) 시간 형식(hh.mm.ss)이 사용됩니다.

***EUR:** 유럽식 시간 형식(hh.mm.ss)이 사용됩니다.

***JIS:** 일본 산업 표준 시간 형식(hh:mm:ss)이 사용됩니다.

TIMSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *HMS가 TIMFMT 매개변수에 지정되어 있을 때에만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 시간 분리 문자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘:’: 콜론(:)이 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

REPLACE

동일한 이름의 프로그램 또는 SQL 패키지가 동일한 라이브러리에 있을 때 새로운 프로그램 또는 SQL 패키지가 작성되는지를 지정합니다. 이 매개변수의 값은 CRTCLPGM 명령으로 전달됩니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 REPLACE 매개변수에 나와 있습니다.

***YES:** 새로운 패키지나 SQL 패키지가 작성되고, 지정된 라이브러리 내에서 같은 이름 및 유형을 가지는 기존의 프로그램 또는 SQL 패키지는 QRPLOBJ로 이동됩니다.

***NO:** 새로운 프로그램 또는 SQL 패키지는 같은 이름과 유형의 오브젝트가 이미 지정된 라이브러리에 존재하는 경우 작성되지 않습니다.

RDB

SQL 패키지 오브젝트가 작성되는 관계형 데이터베이스의 이름을 지정합니다.

***LOCAL:** 프로그램은 분산 SQL 프로그램으로서 작성됩니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다. SQL 패키지 오브젝트는 사전컴파일 프로세스의 일부분으로서 작성되지 않습니다. 구조화 조회 언어 패키지 작성(CRTSQLPKG) 명령이 사용될 수 있습니다.

relational-database-name: 새로운 SQL 패키지 오브젝트가 작성될 때 관계형 데이터베이스의 이름을 지정하십시오. 로컬 관계형 데이터베이스명이 지정되면 작성된 프로그램은 계속 분산 SQL 프로그램입니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다.

***NONE:** SQL 패키지 오브젝트가 작성되지 않습니다. 프로그램 오브젝트는 분산 프로그램이 아니며 CRTSQLPKG(구조화 조회 언어 패키지 작성) 명령은 사용될 수 없습니다.

USER

변환 시작시 리모트 시스템으로 송신되는 사용자명을 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***CURRENT:** 현재 작업이 실행 중인 사용자 프로파일이 사용됩니다.

user-name: 어플리케이션 서버 작업에 사용될 사용자 이름을 지정하십시오.

PASSWORD

리모트 시스템에 사용될 암호를 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***NONE:** 암호가 송신되지 않습니다. 이 값이 지정되면 USER(*CURRENT) 또한 지정되어야 합니다.

password: USER 매개변수에 지정된 사용자명의 암호를 지정하십시오.

RDBCNNMTH

CONNECT문에 사용되는 의미를 지정합니다. 자세한 정보는 *SQL 참조서*에서 CONNECT(유형 1) 및 CONNECT(유형 2)를 참조하십시오.

***DUW:** CONNECT(유형 2) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 추가 관계형 데이터베이스에 대한 연속된 CONNECT문이 이전의 연결을 단절시키지는 않습니다.

***RUW:** CONNECT(유형 1) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 연속된 CONNECT문은 새로운 연결이 설정되기 전에 이전의 연결을 단절시킵니다.

DFTRDBCOL

표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용되는 스키마명을 지정합니다. 이 매개변수는 정적 SQL문에만 적용됩니다.

***NONE:** OPTION 매개변수에 정의된 명명 규칙이 사용됩니다.

schema-name: 스키마 ID의 이름을 지정하십시오. 이 값은 OPTION 매개변수에 지정된 명명 규칙 대신 사용됩니다.

DYNDFTCOL

DFTRDBCOL 매개변수에 대해 지정된 디폴트 스키마명도 동적 명령문에 사용되는지를 지정합니다.

***NO:** 표, 뷰, 색인 및 동적 SQL문에 대한 SQL 패키지의 규정되지 않은 이름으로 DFTRDBCOL 매개변수에 지정된 값을 사용하지 마십시오. OPTION 매개변수에 지정된 명명 규칙이 사용됩니다.

***YES:** DFTRDBCOL 매개변수에 지정된 콜렉션명은 표, 뷰, 색인 및 동적 SQL문의 규정되지 않은 이름으로 사용됩니다.

SQLPKG

이 명령의 RDB 매개변수에 지정된 관계형 데이터베이스에 작성된 SQL 패키지의 규정된 이름을 지정합니다.

라이브러리 값은 다음과 같습니다.

***PGMLIB:** 패키지가 프로그램을 포함하는 라이브러리와 같은 이름의 라이브러리에 작성됩니다.

library-name: 패키지가 작성된 라이브러리의 이름을 지정하십시오.

***PGM:** 패키지의 이름이 프로그램명과 같습니다.

package-name: RDB 매개변수에 지정된 리모트 데이터베이스에 작성된 패키지의 이름을 지정하십시오.

SQLPATH

정적 SQL문의 사용자 정의 유형, 프로시저 및 함수 등을 찾기 위해 사용될 경로를 지정합니다.

***NAMING:** 사용되는 경로는 OPTION 매개변수에 지정된 명명 규칙에 따릅니다.

***SYS** 명명의 경우 사용되는 경로는 *LIBL로, 실행시 현재 라이브러리 리스트입니다.

***SQL** 명명의 경우 사용되는 경로는 "QSYS", "QSYS2", "userid"입니다. 여기서 "userid"는 USER 특수 레지스터의 값입니다. DFTRDBCOL 매개변수에 schema-name을 지정한 경우 schema-name이 사용자 ID 대신 사용됩니다.

***LIBL:** 사용되는 경로는 실행시 라이브러리 리스트입니다.

schema-name: 하나 이상의 스키마명 리스트를 지정하십시오. 최대 268개의 개별 스키마를 지정할 수 있습니다.

SQLCURRULE

SQL문에 사용되는 의미를 지정합니다.

***DB2:** 모든 SQL문의 의미에 대한 디폴트는 DB2에 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 문자 자료로 처리됩니다.

***STD:** 모든 SQL문의 의미에 대한 디폴트는 ISO 및 ANSI SQL 표준에서 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 2진 자료로 처리됩니다.

SAAFLAG

IBM SQL 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 IBM SQL 구문을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다. IBM SQL 구문을 생성하는 IBM 데이터베이스에 대한 정보는 *DRDA IBM SQL Reference, SC26-3255-00* 책을 참조하십시오.

***NOFLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사하지 않습니다.

***FLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사합니다.

FLAGSTD

미국 표준 협회(ANSI) 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 다음과 같은 표준을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다.

ANSI X3.135-1992 항목

ISO 9075-1992 항목

FIPS 127.2 항목

***NONE:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크하지 않습니다.

***ANS:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크합니다.

PRTFILE

리스트가 전달되는 프린터 파일의 규정화명을 지정합니다. 파일의 최소 레코드 길이는 132바이트여야 하며, 그렇지 않으면 정보가 유실됩니다.

프린터 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

QSYSPRT: 파일명이 지정되지 않으면 사전컴파일러 인쇄 출력이 IBM 제공 프린터 파일 QSYSPRT로 전달됩니다.

printer-file-name: 사전컴파일러 인쇄 출력이 전달될 프린터 파일의 이름을 지정하십시오.

SRTSEQ

SQL문의 스트링 비교에 사용될 정렬 순서표를 지정합니다.

주: *HEX는 어플리케이션 서버가 iSeries 시스템에 없거나 릴리스 레벨이 V2R3M0 이전인 분산 어플리케이션의 매개변수에 대해서만 지정되어야 합니다

***JOB:** 작업에 대한 SRTSEQ 값이 사전컴파일 중에 검색됩니다.

***JOB RUN:** 작업에 대한 SRTSEQ 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 SRTSEQ(*JOB RUN)는 LANGID(*JOB RUN)가 지정될 때만 유효합니다.

***LANGIDUNQ:** LANGID 매개변수에 지정된 고유 가중치 정렬표가 사용됩니다.

***LANGIDSHR:** LANGID 매개변수에 지정된 공유 가중치 정렬표가 사용됩니다.

***HEX:** 정렬 순서표가 사용되지 않습니다. 문자의 16진수 값이 정렬 순서를 결정하기 위해 사용됩니다.

정렬 순서표의 이름은 다음 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

table-name: 사용될 정렬 순서표의 이름을 지정하십시오.

LANGID

SRTSEQ(*LANGIDUNQ) 또는 SRTSEQ(*LANGIDSHR) 지정시 사용될 언어 ID를 지정합니다.

***JOB**: 작업에 대한 LANGID 값이 사전컴파일 중에 검색됩니다.

***JOB RUN**: 작업에 대한 LANGID 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 LANGID(*JOB RUN)는 SRTSEQ(*JOB RUN)가 지정될 때만 유효합니다.

language-id: 프로그램에 사용될 언어 ID를 지정하십시오.

USRPRF

정적 SQL문에서 각 오브젝트에 대해 프로그램 오브젝트가 가지는 권한을 포함하여 컴파일된 프로그램 오브젝트를 실행할 때 사용되는 사용자 프로파일을 지정합니다. 프로그램 소유자 또는 프로그램 사용자의 프로파일이 프로그램 오브젝트가 사용할 수 있는 오브젝트를 제어하는 데 사용됩니다.

***NAMING**: 사용자 프로파일은 명명 규칙에 의해 판별됩니다. 명명 규칙이 *SQL이면 USRPRF(*OWNER)가 사용됩니다. 명명 규칙이 *SYS이면 USRPRF(*USER)가 사용됩니다.

***USER**: 프로그램 오브젝트를 실행 중인 사용자의 프로파일이 사용됩니다.

***OWNER**: 프로그램 소유자와 프로그램 사용자 모두의 사용자 프로파일이 프로그램 실행시 사용됩니다.

DYNUSRPRF

SQL문에 사용되는 사용자 프로파일을 지정합니다.

***USER**: 로컬 동적 SQL문은 프로그램 사용자의 파일하에서 실행됩니다. 분산 동적 SQL문은 어플리케이션 서버 작업의 사용자 프로파일하에서 실행됩니다.

***OWNER**: 로컬 동적 SQL문은 프로그램 소유자의 사용자 프로파일하에서 실행됩니다. 분산 동적 SQL문은 SQL 패키지 소유자의 사용자 프로파일하에서 실행됩니다.

TOSRCFILE

SQL 사전컴파일러에 의해 처리된 출력 소스 멤버가 있게 될 소스 파일의 규정화명을 지정합니다. 지정된 소스 파일을 찾을 수 없는 경우에는 이를 작성합니다. 출력 멤버는 SRCMBR 매개변수용으로 지정한 이름과 동일한 이름을 갖게 됩니다.

가능한 라이브러리 값은 다음과 같습니다.

QTEMP: 라이브러리 QTEMP가 사용될 것입니다.

CRTSQLCBL

***LIBL:** 작업 라이브러리 리스트가 지정 파일에 대해 탐색됩니다. 만일 라이브러리 리스트의 어떠한 라이브러리에서도 이 파일이 없는 경우에 파일은 현재 라이브러리에서 작성됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 사용될 것입니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없는 경우 QGPL 라이브러리가 사용될 것입니다.

library-name: 출력 소스 파일이 작성될 라이브러리명을 지정하십시오.

QSQLTEMP: 소스 파일 QSQLTEMP가 사용될 것입니다.

source-file-name: 출력 소스 멤버가 들어 있는 소스 파일의 이름을 지정하십시오.

TEXT

프로그램과 그 기능을 간단하게 설명하는 텍스트를 지정합니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 TEXT 매개변수에 나와 있습니다.

***SRCMBRTXT:** COBOL 프로그램을 작성하는 데 사용되는 소스 파일 멤버로부터 텍스트를 취합니다. 데이터베이스 소스 멤버에 대한 텍스트는 소스 입력 유틸리티(SEU) 소스 입력 유틸리티 시작(STRSEU) 명령을 사용하거나 실제 파일 멤버 추가(ADDPFM) 또는 실제 파일 멤버 변경(CHGPFM) 명령을 사용함으로써 추가되거나 변경될 수 있습니다. 소스 파일이 인라인 파일이나 장치 파일일 경우 텍스트는 공백입니다.

***BLANK:** 텍스트는 지정되지 않습니다.

'description': 50자 미만의 텍스트를 어포스트로피로 묶어 지정하십시오.

예:

```
CRTSQLCBL PGM(ACCTS/STATS) SRCFILE(ACCTS/ACTIVE)
TEXT('Statistical analysis program for
active accounts')
```

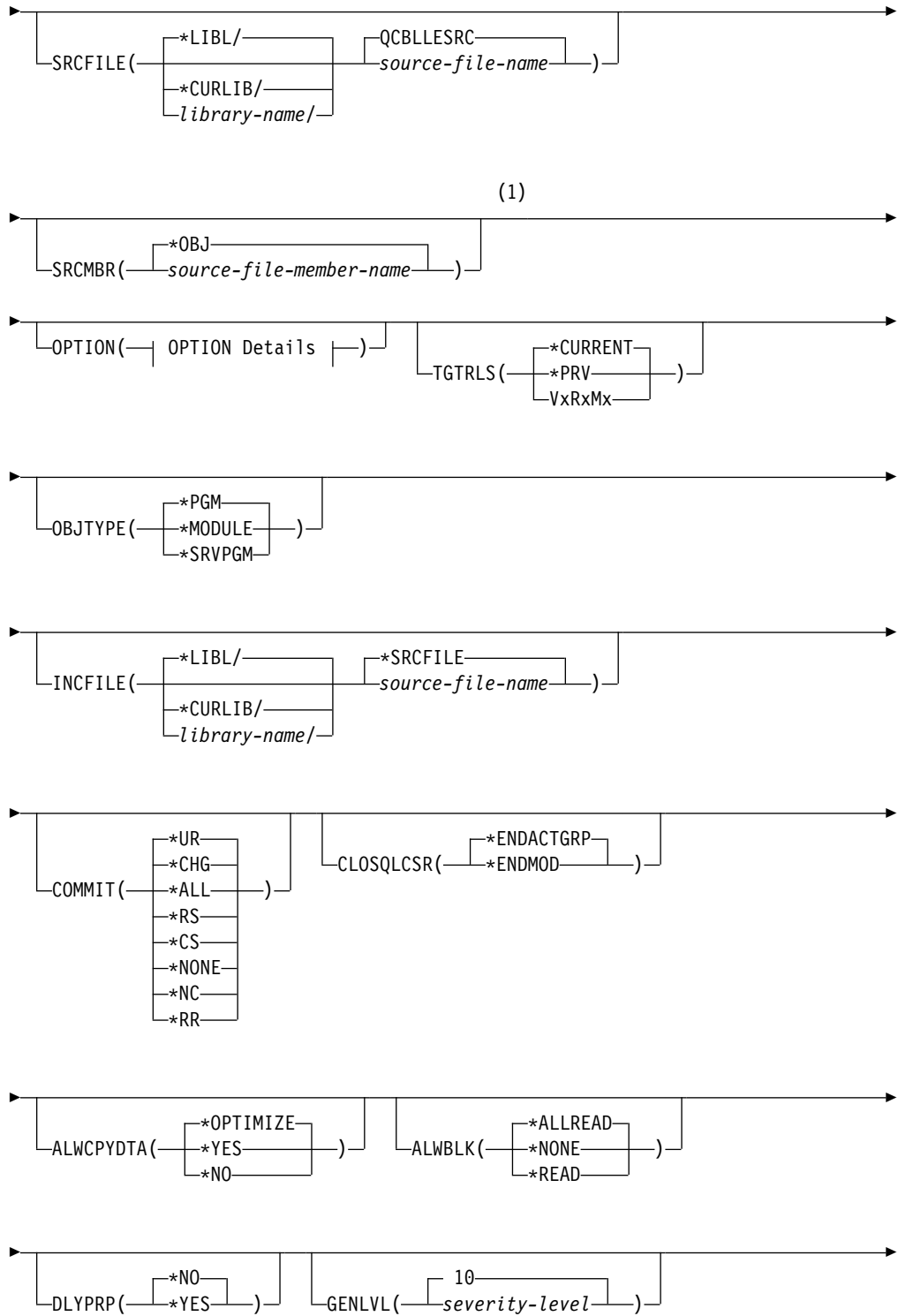
이 명령은 소스문을 컴파일하는 SQL 컴파일러를 실행하여 변경된 소스문을 라이브러리 QTEMP에 있는 파일 QSQLTEMP의 멤버 STATS에 저장합니다. SQL 사전컴파일러에 의해 작성된 소스 멤버를 사용하여 라이브러리 ACCTS에 프로그램 STATS를 작성하기 위해서 COBOL 컴파일러가 호출됩니다.

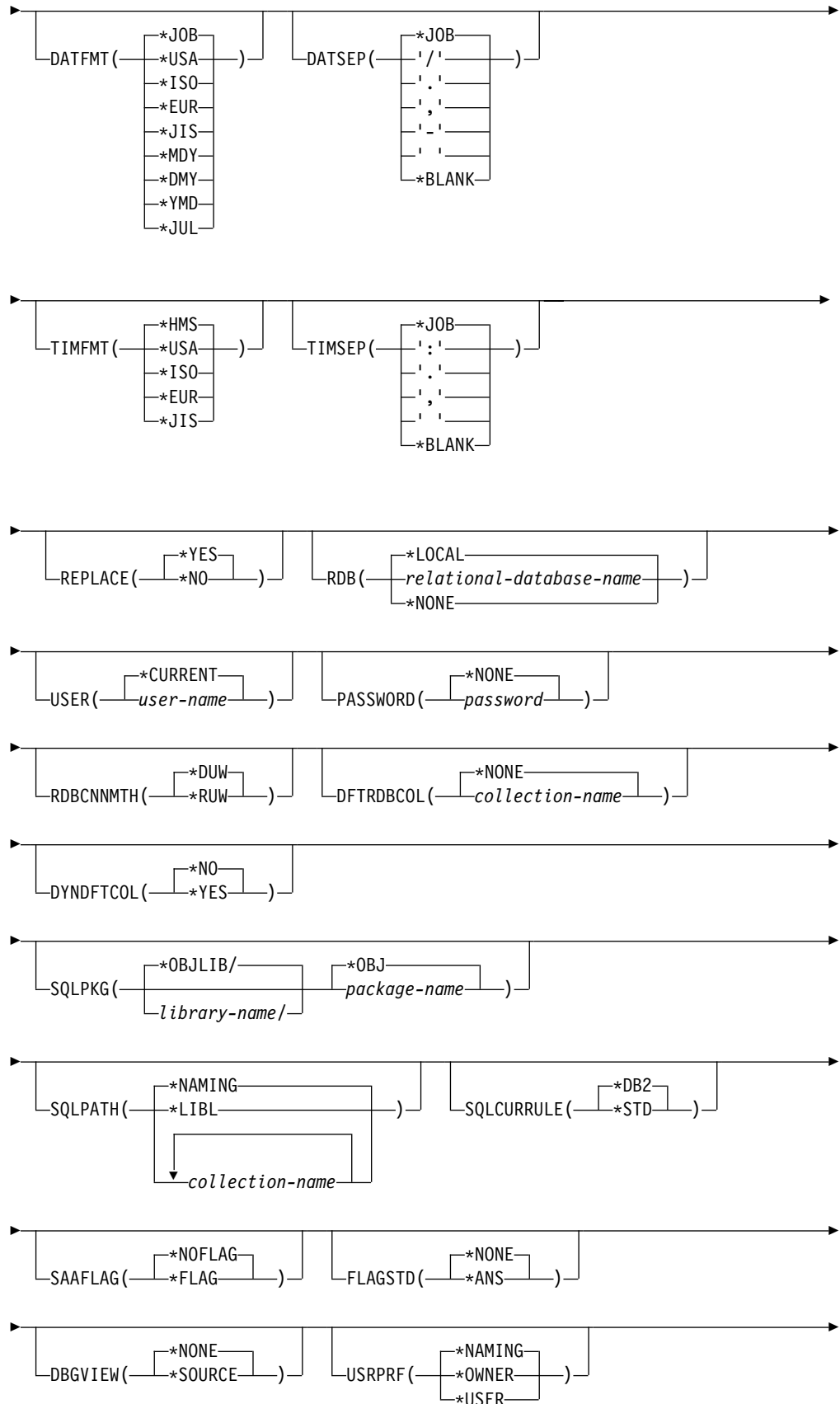
CRTSQLCBLI(SQL ILE COBOL 오브젝트 작성) 명령

Job: B,I Pgm: B,I REXX: B,I Exec

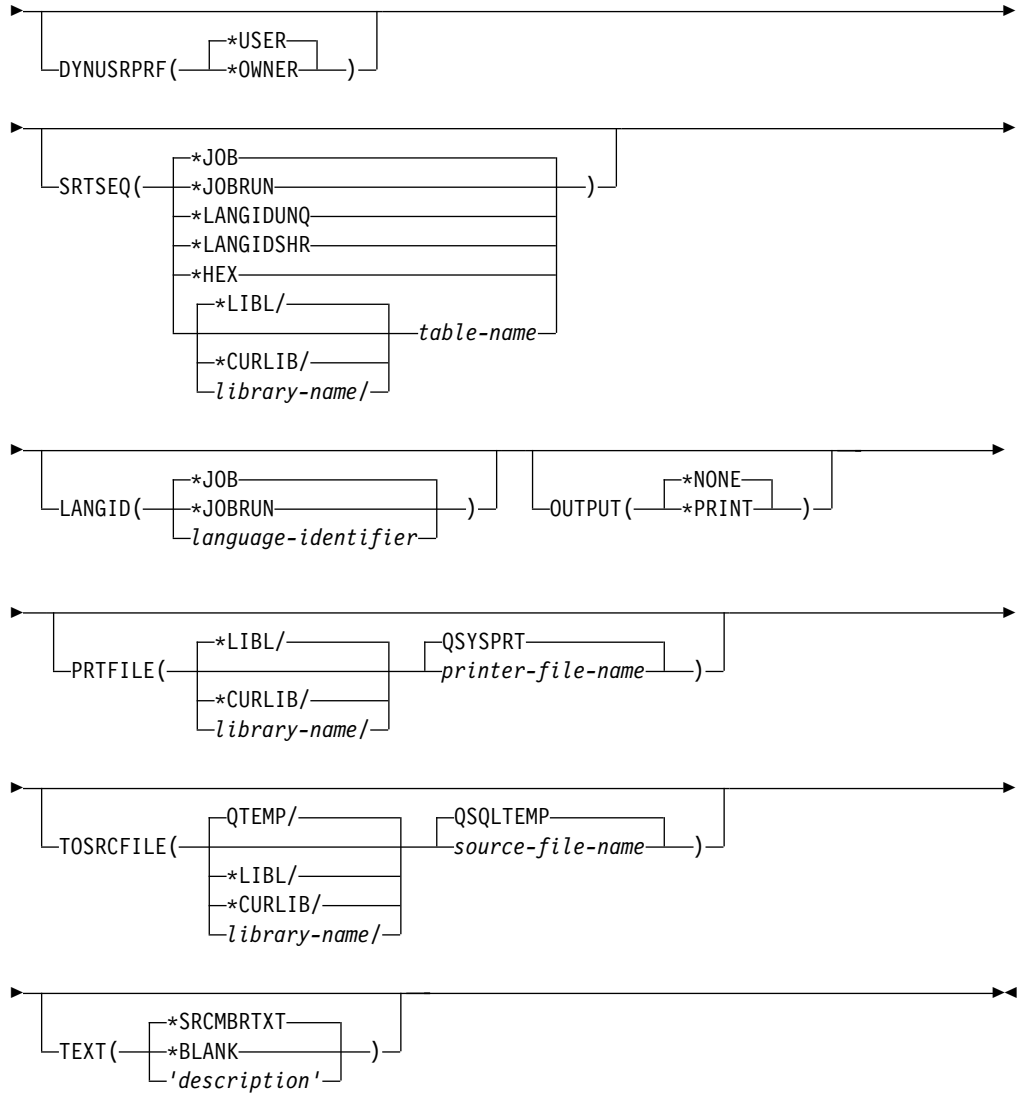
▶▶ CRTSQLCBLI—OBJ(*CURLIB/
 library-name/)—object-name—▶▶

CRTSQLCBLI

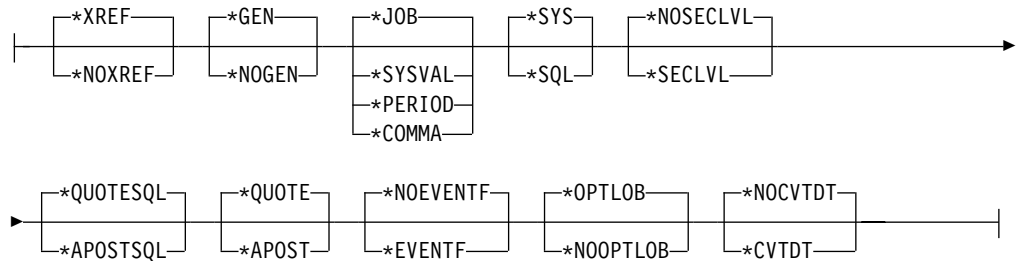




CRTSQLCBLI



OPTION 세부사항:



주:

1 이 앞에 오는 모든 매개변수는 앞뒤 관계에 따라 지정할 수 있습니다.

목적:

CRTSQLCBLI(구조화 조회 언어 ILE COBOL 오브젝트 작성) 명령은 SQL문이 들어 있는 COBOL 소스문을 사전컴파일하는 구조화 조회 언어(SQL) 사전컴파일러를 호

출하고, 임시 소스 멤버를 작성하며, 모듈, 프로그램 또는 서비스 프로그램을 작성하기 위하여 ILE COBOL 컴파일러를 선택적으로 호출합니다.

매개변수:

OBJ

작성된 오브젝트의 규정화명을 지정합니다.

***CURLIB:** 새로운 오브젝트가 해당 작업에 대한 현재 라이브러리에 작성됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 오브젝트가 작성된 라이브러리의 이름을 지정하십시오.

object-name: 작성 중인 오브젝트의 이름을 지정하십시오.

SRCFILE

SQL문과 함께 COBOL 소스문이 들어 있는 소스 파일의 규정된 이름을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QCBLLSRC: 소스 파일명이 지정되지 않은 경우 소스 파일 QCBLLSRC에는 COBOL 소스가 포함됩니다.

source-file-name: COBOL 소스가 들어 있는 소스 파일의 이름을 지정하십시오.

SRCMBR

COBOL 소스가 들어 있는 소스 파일 멤버명을 지정합니다. 이 매개변수는 SRCFILE 매개변수의 소스 파일명이 데이터베이스 파일일 경우에만 지정됩니다. 이 매개변수가 지정되지 않으면 OBJ 매개변수에 지정된 OBJ명이 사용됩니다.

***OBJ:** COBOL 소스가 OBJ 매개변수에 지정된 것과 같은 이름의 소스 파일 멤버에 있음을 지정합니다.

source-file-member-name: COBOL 소스를 포함하는 멤버의 이름을 지정하십시오.

OPTION

COBOL 소스문이 사전컴파일될 때 다음 옵션 중 하나 또는 여러 개가 사용되는지의 여부를 지정합니다. 옵션이 두 번 이상 지정되면 마지막에 지정된 옵션이 사용됩니다.

요소 1: 상호 참조 옵션

***XREF:** 사전컴파일러는 그러한 항목을 참조하는 프로그램 내의 명령문 번호 항목을 상호 참조합니다.

***NOXREF:** 사전컴파일러는 이름을 상호 참조하지 않습니다.

요소 2: 프로그램 작성 옵션

***GEN:** 사전컴파일러가 OBJTYPE 매개변수에 의해 지정된 오브젝트를 작성합니다.

***NOGEN:** 사전컴파일러가 ILE COBOL 컴파일러를 호출하지 않으며 모듈, 프로그램, 서비스 프로그램 또는 SQL 패키지가 작성되지 않습니다.

요소 3: 소수점 옵션

***JOB:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 사전컴파일시 작업에 대해 지정된 소수점 표현입니다.

***SYSVAL:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 QDECFMT 시스템 값입니다.

***PERIOD:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 마침표(.)입니다.

주: QDECFMT가 소수점으로 사용될 값을 쉼표로 지정하면 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)가 쉼표로 분리되고 그 뒤에 공백이 와야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 소수점이 마침표인 VALUES(1.1,2.23,4.1)와 동일합니다.

***COMMA:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 쉼표(.)입니다.

주: 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)는 쉼표로 분리되고 그 뒤에 공백이 와야 합니다. 예를 들어, VALUES(1,1, 2,23, 4,1)는 마침표가 소수점으로 지정된 VALUES(1.1,2.23,4.1)와 동일합니다.

요소 4: 명명 규칙 옵션

***SYS:** 시스템 명명 규칙(library-name/file-name)이 사용됩니다.

***SQL:** SQL 명명 규칙(schema-name.table-name)이 사용됩니다.

iSeries 시스템 이외의 리모트 데이터베이스에서 프로그램을 작성할 때 *SQL은 명명 규칙을 따라 지정해야 합니다.

요소 5: 1차 레벨 메시지 텍스트 옵션

***NOSECLVL:** 2차 레벨 텍스트 설명이 인쇄 출력에 추가되지 않습니다.

***SECLVL:** 대체 자료를 가진 2차 레벨 텍스트가 리스팅의 모든 메시지에 대해 추가됩니다.

요소 6: 스트링 분리문자 옵션

***QUOTESQL:** SQL문에서 큰 따옴표(")가 스트링 분리문자입니다.

***APOSTSQL:** SQL문에서 작은 따옴표(')가 스트링 분리문자입니다.

요소 7: 리터럴 옵션

***QUOTE:** 큰 따옴표(")가 COBOL문에서 비슷자 리터럴과 부울 리터럴이 아닌 리터럴에 사용됩니다.

***APOST:** 작은 따옴표(')는 COBOL문에서 비슷자 리터럴과 부울 리터럴이 아닌 리터럴에 사용됩니다.

요소 8: 이벤트 파일 작성

***NOEVENTF:** 컴파일러가 CoOperative Development Environment/400(CODE/400)에서 사용할 이벤트 파일을 작성하지 않습니다.

***EVENTF:** 컴파일러가 CoOperative Development Environment/400(CODE/400)에서 사용할 이벤트 파일을 작성합니다. 이벤트 파일은 사용자 소스 라이브러리에 있는 파일 EVFEVENT의 멤버로 생성됩니다. CODE/400은 이 파일을 이용하여 CODE/400 에디터와 함께 통합된 오류 피드백을 제공합니다. 이 옵션은 일반적으로 사용자를 대신해서 CODE/400에 의해 지정됩니다.

요소 9: DRDA에 대한 대형 오브젝트 최적화

***OPTLOB:** 커서의 첫 번째 FETCH는 모든 후속 FETCH에서 커서가 LOB(대형 오브젝트)에 대해 사용될 방법을 결정합니다. 이 옵션은 커서가 닫힐 때까지 적용됩니다.

첫 번째 FETCH가 LOB 위치 지정자를 사용하여 LOB 열에 액세스할 경우 그 커서에 대한 어떤 후속 FETCH도 그 LOB 열을 LOB 호스트 변수에 페치할 수 없습니다.

첫 번째 FETCH가 LOB 열을 LOB 호스트 변수에 배치할 경우 그 커서에 대한 어떤 후속 FETCH도 그 열에 대해 LOB 위치 지정자를 사용할 수 없습니다.

***NOOPTLOB:** 열이 LOB 위치 지정자나 LOB 호스트 변수에 검색되는지에 대해서는 제한이 없습니다. 이 옵션을 사용하면 성능이 떨어질 수도 있습니다.

요소 10: 날짜 변환

***NOCVTD:** 외부적으로 서술된 데이터베이스 파일로부터 검색되는 날짜, 시간 및 시간소인 자료 유형이 날짜, 시간 및 시간소인 날짜 유형을 사용하여 처리됩니다.

***CVTD:** 외부적으로 서술된 데이터베이스 파일로부터 검색되는 날짜, 시간 및 시간소인 자료 유형이 고정 길이 문자 필드로서 처리됩니다.

TGTRLS

작성될 오브젝트를 사용하려고 하는 오퍼레이팅 시스템의 릴리스 레벨을 지정합니다.

***CURRENT**와 ***PRV** 값이 주어질 때와 릴리스 레벨 값을 지정할 때에는 **VxRxMx**의 형식으로 릴리스를 지정하게 되며, 여기서 **Vx**는 버전, **Rx**는 릴리스, **Mx**는 수정 레벨을 나타냅니다. 예를 들어, **V2R3M0**은 버전 2, 릴리스 3, 수정 레벨 0을 나타냅니다.

***CURRENT:** 오브젝트는 사용자 시스템에서 현재 실행 중인 오퍼레이팅 시스템의 릴리스에 사용됩니다. 예를 들어, 시스템에서 **V2R3M5**가 실행되고 있으면, ***CURRENT**는 **V2R3M5**가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

주: 시스템에서 **V2R3M5**가 실행되고 있으며, **V2R3M0**가 설치된 시스템에서 오브젝트를 사용하려는 경우에는 **TGTRLS(*CURRENT)**가 아니고 **TGTRLS(V2R3M0)**를 지정해야 합니다.

***PRV:** 오브젝트는 오퍼레이팅 시스템의 이전 릴리스, 수정 레벨 0에서 사용됩니다. 예를 들어, 시스템에서 **V2R3M5**가 실행되고 있으면, ***PRV**는 **V2R2M0**가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

release-level: **VxRxMx** 형식의 릴리스를 지정하십시오. 지정된 릴리스의 시스템이나 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스의 시스템에서도 오브젝트를 사용할 수 있습니다.

유효한 값들은 현재 버전, 릴리스, 수정 레벨에 기초하고 있으며, 새로운 릴리스로 변경될 수 있습니다. 이 명령이 지원하는 가장 이전의 릴리스 레벨보다 더 앞의 릴리스 레벨을 지정하면 지원되는 릴리스 레벨 가운데 가장 이전의 릴리스가 표시된 오류 메시지를 받게 됩니다.

OBJTYPE

작성될 오브젝트 유형을 지정합니다.

***PGM:** SQL 사전컴파일러가 **CRTBNDCBL** 명령을 발행하여 바인드된 프로그램을 작성합니다.

***MODULE:** SQL 사전컴파일러가 CRTCBMOD 명령을 사용하여 모듈을 작성합니다.

***SRVPGM:** SQL 사전컴파일러가 CRTCBMOD와 CRTSRVPGM 명령을 사용하여 서비스 프로그램을 작성합니다.

주:

1. OBJTYPE(*PGM), OBJTYPE(*SRVPGM) 또는 RDB 매개변수 지정시 CRTSQLPKG 명령은 프로그램이 작성된 후 SQL 사전컴파일러에 의해 발행됩니다. OBJTYPE(*MODULE) 지정시, SQL 패키지는 작성되지 않으며 CRTPGM 또는 CRTSRVPGM 명령으로 프로그램을 작성한 후 CRTSQLPKG 명령을 발행해야 합니다.
2. *NOGEN이 지정되는 경우 SQL 임시 소스 멤버만이 생성되며 모듈, 프로그램, 서비스 프로그램 또는 SQL 패키지는 작성되지 않습니다.

INCFILE

SQL INCLUDE문을 사용하여 프로그램에 포함시킨 멤버에 들어 있는 소스 파일의 규정화명을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리명을 지정하십시오.

***SRCFILE:** SRCFILE 매개변수에 지정된 규정된 소스 파일에는 모든 SQL INCLUDE문에 지정된 소스 파일 멤버가 포함됩니다.

source-file-name: 모든 SQL INCLUDE문에 지정된 소스 파일명을 포함하는 소스 파일의 이름을 지정하십시오. 여기에 지정된 소스 파일의 레코드 길이는 SRCFILE 매개변수에 지정된 소스 파일의 길이보다 짧아야 합니다.

COMMIT

컴파일된 장치에서 SQL문이 확약 제어하에 실행되는지를 지정합니다. 호스트 언어 소스문에 참조된 파일은 이 옵션의 영향을 받지 않습니다. SQL문에 참조된 SQL 표, SQL 뷰 및 SQL 패키지만이 영향을 받습니다.

***CHG 또는 *UR:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON 및 REVOKE문에서 참조된 오브젝트들을 지정하며, 갱신, 삭제 및 삽입된 행들은 작업 단위(트랜잭션)가 종료될 때까지 잠겨 있습니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다.

CRTSQLCBLI

***ALL** 또는 ***RS**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입된 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 없습니다.

***CS**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 선택은 되었지만 갱신되지 않은 행은 다음 행이 선택될 때까지 잠깁니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 없습니다.

***NONE** 또는 ***NC**: 예약 제어가 사용되지 않음을 지정합니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 있습니다. SQL DROP SCHEMA문이 프로그램에 포함되는 경우에는 ***NONE** 또는 ***NC**가 반드시 사용되어야 합니다. RDB 매개변수에 관계형 데이터베이스가 지정되어 있고 관계형 데이터베이스가 iSeries가 아닌 시스템에 있으면, ***NONE** 또는 ***NC**를 지정할 수 없습니다.

***RR**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 없습니다. SELECT, UPDATE, DELETE 및 INSERT문에 참조된 모든 표는 작업 단위(트랜잭션)가 종료될 때까지 배타적으로 잠겨 있습니다.

CLOSQLCSR

SQL 커서가 내재적으로 닫히고 SQL 준비 명령문이 내재적으로 취소되고 LOCK TABLE 잠금이 해제되는 시기를 지정합니다. SQL 커서는 CLOSE, COMMIT 또는 ROLLBACK(HOLD 없이) SQL문을 발행할 때 명시적으로 닫힙니다.

***ENDACTGRP**: 활성 그룹 종료시, SQL 커서는 닫히고 SQL 준비 명령문은 내재적으로 취소되며 LOCK TABLE 잠금은 해제됩니다.

***ENDMOD**: 모듈 종료시 SQL 커서는 닫히고 SQL 준비 명령문은 내재적으로 취소됩니다. LOCK TABLE 잠금은 활성 그룹 종료시 해제됩니다.

ALWCOPYDTA

자료의 사본이 SELECT문에 사용될 수 있는지 여부를 지정합니다.

***OPTIMIZE**: 시스템은 데이터베이스에서 직접 검색된 자료를 사용할지 또는 자료의 사본을 사용할지의 여부를 판별합니다. 결정은 최상의 성능을 제공할 수 있는 방법에 근거하여 이루어집니다. COMMIT가 ***CHG** 또는 ***CS**이면서 ALWBLK가 ***ALLREAD**가 아니거나, COMMIT가 ***ALL** 또는 ***RR**이면 조회 수행에 필요한 경우에만 자료의 사본이 사용됩니다.

***YES**: 필요할 때만 자료의 사본이 사용됩니다.

***NO:** 자료의 사본은 허용되지 않습니다. 조회를 수행하는 데 자료의 임시 사본이 필요하면 오류 메시지가 표시됩니다.

ALWBLK

데이터베이스 관리자가 레코드 블록화를 사용할 수 있는지 여부와 블록화가 읽기 전용 커서에 사용될 수 있는 확장 범위를 지정합니다.

***ALLREAD:** *NONE 또는 *CHG가 COMMIT 매개변수에 지정된 경우 읽기 전용 커서에 대해 행이 블록화됩니다. 명시적으로 갱신될 수 없는 프로그램 내의 모든 커서들은 EXECUTE 또는 EXECUTE IMMEDIATE문이 프로그램 내에 있을 지라도 읽기 전용으로 열립니다.

***ALLREAD 지정:**

- *READ에 대해 허용된 블록화 이외의 제약 제어 레벨 *CHG하에서 레코드의 블록화를 허용합니다.
- 프로그램에서 거의 모든 읽기 전용 커서의 성능이 향상될 수 있으나 다음과 같은 방법으로 조회를 제한합니다.
 - 롤백(ROLLBACK) 명령, 호스트 언어의 ROLLBACK문 또는 ROLLBACK HOLD SQL문은 *ALLREAD가 지정된 경우 읽기 전용 커서의 위치를 재 지정하지 않습니다.
 - 커서에 대한 DECLARE문에 FOR UPDATE절이 없는 한, 지정된 UPDATE 또는 DELETE문의 동적 수행(예를 들면, EXECUTE IMMEDIATE 사용)은 커서의 행을 갱신하는 데 사용될 수 없습니다.

***NONE:** 커서에 대한 자료의 검색을 위해 행이 블록화되지 않습니다.

***NONE을 지정하는 경우:**

- 검색된 자료가 현재의 자료임을 보장합니다.
- 조회시 자료의 첫 번째 행을 검색하는 데 필요한 시간을 줄일 수 있습니다.
- 조회가 닫히기 전에 조회의 처음 몇 행만이 검색될 때 프로그램에 의해 사용되지 않은 자료 행 블록의 검색으로부터 데이터베이스 관리자를 중단시킵니다.
- 많은 행을 검색하는 조회의 전반적인 성능이 저하될 수 있습니다.

***READ:** 레코드는 다음의 경우 커서에 대한 자료의 읽기 전용 검색에 대해 블록화됩니다.

- *NONE이 COMMIT 매개변수에 지정되어 제약 제어가 사용되지 않음을 표시하는 경우
- FOR READ ONLY절로 커서가 선언되었거나 커서에 대해 지정된 UPDATE 또는 DELETE문을 수행할 수 있는 동적 명령문이 없는 경우

*READ를 지정하면 위의 조건을 만족하는 조회의 전반적인 성능이 향상되고 많은 수의 레코드를 검색할 수 있습니다.

DLYPRP

PREPARE문에 대한 동적 명령문 유효성 검사가 OPEN, EXECUTE 또는 DESCRIBE문 실행시까지 지연되는지 여부를 지정합니다. 유효성 검사가 지연되면 불필요한 확인이 제거되므로 성능이 향상됩니다.

***NO:** 동적 명령문 유효성 검사가 지연되지 않습니다. 동적 명령문이 준비될 때 액세스 계획의 유효성이 확인됩니다. 동적 명령문이 OPEN 또는 EXECUTE문에 사용될 때 액세스 계획의 유효성이 재확인됩니다. 동적 명령문에 의해 참조되는 오브젝트의 존재 또는 권한으로 인해, 동적 명령문이 아직 유효한지 확인하기 위해 OPEN 또는 EXECUTE문을 발행하기 전에 SQLCODE 또는 SQLSTATE를 검사해야 합니다.

***YES:** 동적 명령문 유효성 검사는 동적 명령문이 OPEN, EXECUTE 또는 DESCRIBE SQL문에 사용될 때까지 지연됩니다. 동적 명령문이 사용될 때 유효성 검사가 완료되고 액세스 계획이 수립됩니다. 이 매개변수에 *YES를 지정하면 OPEN, EXECUTE 또는 DESCRIBE문을 실행한 후에 SQLCODE와 SQLSTATE를 체크하여 동적 명령문이 유효한지 확인해야 합니다.

주: *YES를 지정하면 PREPARE에 INTO절이 사용되거나, 동적 명령문에 대한 OPEN문이 발행되기 전에 DESCRIBE문에서 그 동적 명령문을 사용하는 경우 성능은 향상되지 않습니다.

GENLVL

작성 조작이 실패하는 심각도 레벨을 지정합니다. 이 값 이상의 심각도를 가진 오류가 발생하면 조작이 종료됩니다.

10: 디폴트 심각도 레벨은 10입니다.

심각도 레벨: 0-40 범위 내의 값을 지정하십시오.

DATFMT

결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 날짜 필드가 지정된 형식으로 리턴됩니다. 입력 날짜 스트링의 경우 지정된 값은 유효한 형식으로 날짜가 지정되었는지 여부를 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 iSeries 시스템이 아닌 시스템에 있으면, *USA, *ISO, *EUR 또는 *JIS가 지정되어야 합니다.

***JOB:** 작업에 지정된 형식이 사용됩니다. 작업 표시(DSPJOB) 명령을 사용하여 작업에 대한 현재 날짜 형식을 판별하십시오.

***USA:** 미국식 날짜 형식(mm/dd/yyyy)이 사용됩니다.

***ISO:** 국제 표준화 기구(ISO) 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***EUR:** 유럽식 날짜 형식(dd.mm.yyyy)이 사용됩니다.

***JIS:** 일본 산업 표준 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***MDY:** 날짜 형식(mm/dd/yy)이 사용됩니다.

***DMY:** 날짜 형식(dd/mm/yy)이 사용됩니다.

***YMD:** 날짜 형식(yy/mm/dd)이 사용됩니다.

***JUL:** 율리우스력 날짜 형식(yy/ddd)이 사용됩니다.

DATSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *JOB, *MDY, *DMY, *YMD 또는 *JUL이 DATFMT 매개변수에 지정되어 있을 때만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 날짜 분리자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘/’: 슬래시(/)가 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘-’: 붙임표(-)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

TIMFMT

결과 열에 액세스하기 위해 사용되는 형식을 지정합니다. 입력 시간 스트링의 경우 지정된 값은 시간이 유효한 형식으로 지정되었는지 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 또 다른 iSeries 시스템이 아닌 시스템에 있으면, 시간 형식이 콜론이나 마침표의 시간 분리문자가 있는 *USA, *ISO, *EUR, *JIS 또는 *HMS이어야 합니다.

***HMS:** hh:mm:ss 형식이 사용됩니다.

CRTSQLCBLI

***USA:** 미국식 시간 형식 **hh:mm xx**가 사용되는데, 여기서 **xx**는 AM(오전) 또는 PM(오후)입니다

***ISO:** 국제 표준화 기구(ISO) 시간 형식 **hh.mm.ss**가 사용됩니다.

***EUR:** 유럽식 시간 형식 **hh.mm.ss**가 사용됩니다.

***JIS:** 일본 산업 표준 시간 형식인 **hh:mm:ss**가 사용됩니다.

TIMSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 ***HMS**가 TIMFMT 매개변수에 지정되어 있을 때에만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 시간 분리 문자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘:’: 콜론(:)이 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

REPLACE

동일한 이름의 기존 SQL 모듈, 프로그램, 서비스 프로그램 또는 패키지가 있고 동일한 라이브러리에 입력되었을 때 SQL 모듈, 프로그램, 서비스 프로그램 또는 패키지가 작성되는지를 지정합니다. 이 매개변수의 값은 CRTMODC, CRTBNDC, CRTSRVPGM 및 CRTSQLPKG 명령에 전달됩니다.

***YES:** 새로운 SQL 모듈, 서비스 프로그램 또는 패키지가 작성되고, 지정된 라이브러리 내에서 같은 이름과 유형을 가지는 기존의 모든 SQL 오브젝트는 QRPLOBJ로 이동됩니다.

***NO:** 새로운 SQL 모듈, 프로그램, 서비스 프로그램 또는 패키지가 지정된 라이브러리에 동일한 이름과 유형의 SQL 오브젝트가 이미 있는 경우에는 작성되지 않습니다.

RDB

SQL 패키지 오브젝트가 작성되는 관계형 데이터베이스의 이름을 지정합니다.

***LOCAL:** 프로그램은 분산 SQL 프로그램으로서 작성됩니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다. SQL 패키지 오브젝트는 사전컴파일 프로세스의 일부분으로서 작성되지 않습니다. 구조화 조회 언어 패키지 작성(CRTSQLPKG) 명령이 사용될 수 있습니다.

relational-database-name: 새로운 SQL 패키지 오브젝트가 작성될 때 관계형 데이터베이스의 이름을 지정하십시오. 로컬 관계형 데이터베이스명이 지정되면 작성된 프로그램은 계속 분산 SQL 프로그램입니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다.

***NONE:** SQL 패키지 오브젝트가 작성되지 않습니다. 프로그램 오브젝트는 분산 프로그램이 아니며 CRTSQLPKG(구조화 조회 언어 패키지 작성) 명령은 사용될 수 없습니다.

USER

변환 시작시 리모트 시스템으로 송신되는 사용자명을 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***CURRENT:** 현재 작업이 실행 중인 사용자 프로파일이 사용됩니다.

user-name: 어플리케이션 서버 작업에 사용되는 사용자명을 지정하십시오.

PASSWORD

리모트 시스템에 사용될 암호를 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***NONE:** 암호가 송신되지 않습니다. 이 값이 지정되면 USER(*CURRENT) 또한 지정되어야 합니다.

password: USER 매개변수에 지정된 사용자명의 암호를 지정하십시오.

RDBCNNMTH

CONNECT문에 사용되는 의미를 지정합니다. 자세한 정보는 SQL 참조서에서 CONNECT(유형 1) 및 CONNECT(유형 2)를 참조하십시오.

***DUW:** CONNECT(유형 2) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 추가 관계형 데이터베이스에 대한 연속된 CONNECT문이 이전의 연결을 단절시키지는 않습니다.

***RUW:** CONNECT(유형 1) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 연속된 CONNECT문은 새로운 연결이 설정되기 전에 이전의 연결을 단절시킵니다.

DFTRDBCOL

표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용되는 스키마명을 지정합니다. 이 매개변수는 정적 SQL문에만 적용됩니다.

***NONE:** OPTION 매개변수에 정의된 명명 규칙이 사용됩니다.

CRTSQLCBLI

schema-name: 스키마 ID의 이름을 지정하십시오. 이 값은 OPTION 매개변수에 지정된 명명 규칙 대신 사용됩니다.

DYNDFTCOL

DFTRDBCOL 매개변수에 대해 지정된 디폴트 스키마명도 동적 명령문에 사용되는지를 지정합니다.

***NO**: 표, 뷰, 색인 및 동적 SQL문에 대한 SQL 패키지의 규정되지 않은 이름으로 DFTRDBCOL 매개변수에 지정된 값을 사용하지 마십시오. OPTION 매개변수에 지정된 명명 규칙이 사용됩니다.

***YES**: DFTRDBCOL 매개변수에 지정된 컬렉션명은 표, 뷰, 색인 및 동적 SQL문의 규정되지 않은 이름으로 사용됩니다.

SQLPKG

이 명령의 RDB 매개변수에 지정된 관계형 데이터베이스에 작성된 SQL 패키지의 규정된 이름을 지정합니다.

가능한 라이브러리 값은 다음과 같습니다.

***OBJLIB**: 패키지가 OBJ 매개변수에 지정된 라이브러리와 같은 이름의 라이브러리에 작성됩니다.

library-name: 패키지가 작성된 라이브러리의 이름을 지정하십시오.

***OBJ**: SQL 패키지의 이름은 OBJ 매개변수에 지정된 오브젝트명과 같습니다.

package-name: SQL 패키지의 이름을 지정하십시오. 리모트 시스템이 iSeries 시스템이 아니면, 9자 이상을 지정할 수 없습니다.

SQLPATH

정적 SQL문의 사용자 정의 유형, 프로시저 및 함수 등을 찾기 위해 사용될 경로를 지정합니다.

***NAMING**: 사용되는 경로는 OPTION 매개변수에 지정된 명명 규칙에 따릅니다.

***SYS** 명명의 경우 사용되는 경로는 *LIBL로, 실행시 현재 라이브러리 리스트입니다.

***SQL** 명명의 경우 사용되는 경로는 "QSYS", "QSYS2", "userid"입니다. 여기서 "userid"는 USER 특수 레지스터의 값입니다. DFTRDBCOL 매개변수에 *schema-name*을 지정한 경우 *schema-name*이 사용자 ID 대신 사용됩니다.

***LIBL**: 사용되는 경로는 실행시 라이브러리 리스트입니다.

schema-name: 하나 이상의 스키마명 리스트를 지정하십시오. 최대 268개의 개별 스키마를 지정할 수 있습니다.

SQLCURRULE

SQL문에 사용되는 의미를 지정합니다.

***DB2:** 모든 SQL문의 의미에 대한 디폴트는 DB2에 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 문자 자료로 처리됩니다.

***STD:** 모든 SQL문의 의미에 대한 디폴트는 ISO 및 ANSI SQL 표준에서 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 2진 자료로 처리됩니다.

SAAFLAG

IBM SQL 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 IBM SQL 구문을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다. IBM SQL 구문을 생성하는 IBM 데이터베이스에 대한 정보는 *DRDA IBM SQL Reference, SC26-3255-00* 책을 참조하십시오.

***NOFLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사하지 않습니다.

***FLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사합니다.

FLAGSTD

미국 표준 협회(ANSI) 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 다음과 같은 표준을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다.

ANSI X3.135-1992 항목
ISO 9075-1992 항목
FIPS 127.2 항목

***NONE:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크하지 않습니다.

***ANS:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크합니다.

DBGVIEW

처리되는 테이프 자료 파일의 레코드 유형과 블록화 속성을 지정합니다.

***NONE:** 소스 뷰가 생성되지 않습니다.

***SOURCE:** SQL 사전컴파일러는 루트와 SQL INCLUDE문(필요한 경우)에 대한 소스 뷰를 제공합니다. 사전컴파일러에 의해 생성된 명령문을 포함하는 뷰가 제공됩니다.

USRPRF

정적 SQL문에서 각 오브젝트에 대해 프로그램 오브젝트가 가지는 권한을 포함하여 컴파일된 프로그램 오브젝트를 실행할 때 사용되는 사용자 프로파일을 지정합니다. 프로그램 소유자 또는 프로그램 사용자의 프로파일이 프로그램 오브젝트가 사용할 수 있는 오브젝트를 제어하는 데 사용됩니다.

***NAMING:** 사용자 프로파일은 명명 규칙에 의해 판별됩니다. 명명 규칙이 *SQL 이면 USRPRF(*OWNER)가 사용됩니다. 명명 규칙이 *SYS이면 USRPRF(*USER)가 사용됩니다.

***USER:** 프로그램 오브젝트를 실행 중인 사용자의 프로파일이 사용됩니다.

***OWNER:** 프로그램 소유자와 프로그램 사용자 모두의 사용자 프로파일이 프로그램 실행시 사용됩니다.

DYNUSRPRF

동적 SQL문에 대해 사용될 사용자 프로파일을 지정합니다.

***USER:** 로컬 프로그램의 경우 동적 SQL문이 프로그램 사용자의 프로파일하에서 실행됩니다. 분산 프로그램의 경우 동적 SQL문은 SQL 패키지 사용자의 프로파일하에서 실행됩니다.

***OWNER:** 로컬 프로그램의 경우 동적 SQL문은 프로그램 소유자의 프로파일하에서 실행됩니다. 분산 프로그램의 경우 동적 SQL문이 SQL 패키지 소유자의 프로파일하에서 실행됩니다.

SRTSEQ

SQL문의 스트링 비교에 사용될 정렬 순서표를 지정합니다.

주: *HEX는 어플리케이션 서버가 iSeries 시스템에 없거나 릴리스 레벨이 V2R3M0 이전에 있는 분산 어플리케이션의 이 매개변수에 지정되어야 합니다.

***JOB:** 작업에 대한 SRTSEQ 값이 사전검파일 중에 검색됩니다.

***JOB RUN:** 작업에 대한 SRTSEQ 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 SRTSEQ(*JOB RUN)는 LANGID(*JOB RUN)가 지정될 때만 유효합니다.

***LANGID UNQ:** LANGID 매개변수에 지정된 고유 가중치 정렬표가 사용됩니다. 표의 이름은 다음 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

***LANGID SHR:** 정렬 순서표는 여러 문자에 대해 동일한 가중치를 사용하며, LANGID 매개변수에 지정된 언어와 연관된 공유 가중치 정렬 순서표입니다.

***HEX:** 정렬 순서가 사용되지 않습니다. 문자의 16진수 값이 정렬 순서를 결정하기 위해 사용됩니다.

table-name: 사용될 정렬 순서표의 이름을 지정하십시오.

LANGID

SRTSEQ(*LANGIDUNQ) 또는 SRTSEQ(*LANGIDSHR) 지정시 사용될 언어 ID를 지정합니다.

***JOB:** 작업에 대한 LANGID 값이 사전컴파일 중에 검색됩니다.

***JOB RUN:** 작업에 대한 LANGID 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 LANGID(*JOB RUN)는 SRTSEQ(*JOB RUN)가 지정될 때만 유효합니다.

language-identifier: 언어 ID를 지정하십시오.

OUTPUT

사전컴파일러 리스트가 생성되는지를 지정합니다.

***NONE:** 사전컴파일러 리스트가 생성되지 않습니다.

***PRINT:** 사전컴파일러 리스트가 생성됩니다.

PRTFILE

사전컴파일러 인쇄 출력을 지정한 프린터 파일의 규정화명을 지정합니다. 파일의 길이는 최소 132바이트이어야 합니다. 132바이트보다 적은 레코드 길이로 파일이 지정되는 경우 정보는 유실됩니다.

프린터 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리명을 지정하십시오.

QSYSPRT: 파일명이 지정되지 않으면 사전컴파일러 인쇄 출력이 IBM 제공 프린터 파일 QSYSPRT로 전달됩니다.

printer-file-name: 사전컴파일러 인쇄 출력이 전달될 프린터 파일의 이름을 지정하십시오.

TOSRCFILE

SQL 사전컴파일러에 의해 처리된 출력 소스 멤버가 있게 될 소스 파일의 규정화명을 지정합니다. 지정된 소스 파일을 찾을 수 없는 경우에는 이를 작성합니다. 출력 멤버는 SRCMBR 매개변수용으로 지정한 이름과 동일한 이름을 갖게 됩니다.

가능한 라이브러리 값은 다음과 같습니다.

QTEMP: 라이브러리 QTEMP가 사용될 것입니다.

***LIBL:** 작업 라이브러리 리스트가 지정 파일에 대해 탐색됩니다. 만일 라이브러리 리스트의 어떠한 라이브러리에서도 이 파일이 없는 경우에 파일은 현재 라이브러리에서 작성됩니다.

CRTSQLCBLI

***CURLIB:** 작업에 대한 현재 라이브러리가 사용될 것입니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없는 경우 QGPL 라이브러리가 사용될 것입니다.

library-name: 출력 소스 파일이 작성될 라이브러리명을 지정하십시오.

QSQLTEMP: 소스 파일 QSQLTEMP가 사용될 것입니다.

source-file-name: 출력 소스 멤버가 들어 있는 소스 파일의 이름을 지정하십시오.

TEXT

프린터 파일을 간단하게 설명하는 텍스트를 지정합니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 TEXT 매개변수에 나와 있습니다.

***SRCMBRTXT:** COBOL 프로그램을 작성하는 데 사용되는 소스 파일 멤버로부터 텍스트를 취합니다. STRSEU(소스 입력 유틸리티 시작) 명령을 사용하거나 ADDPFM(실제 파일 멤버 추가) 또는 CHGPFM(실제 파일 멤버 변경) 명령을 사용하여 데이터베이스 소스 멤버에 텍스트를 추가하거나 데이터베이스 소스 멤버의 텍스트를 변경할 수 있습니다. 소스 파일이 인라인 파일이나 장치 파일일 경우 텍스트는 공백입니다.

***BLANK:** 텍스트가 지정되지 않습니다.

'description': 50자 미만의 텍스트를 어포스트로피로 묶어 지정하십시오.

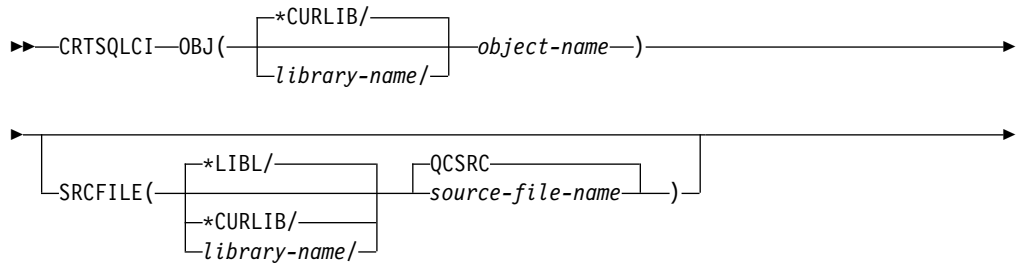
예:

```
CRTSQLCBLI PAYROLL OBJTYPE(*MODULE) TEXT('Payroll Program')
```

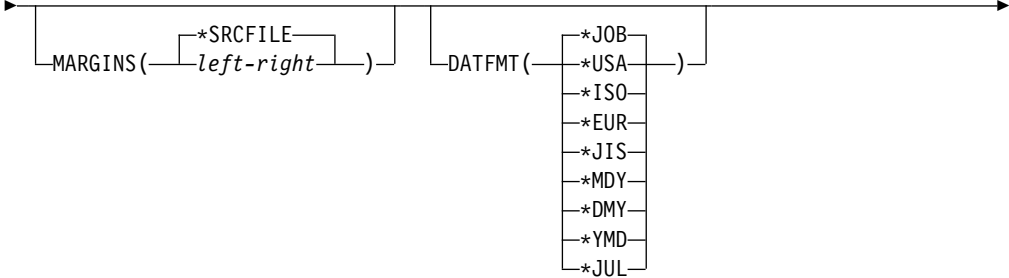
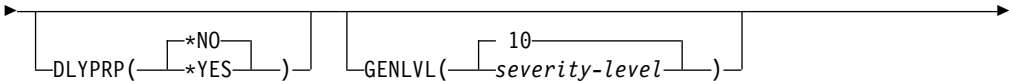
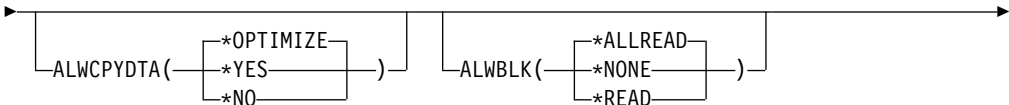
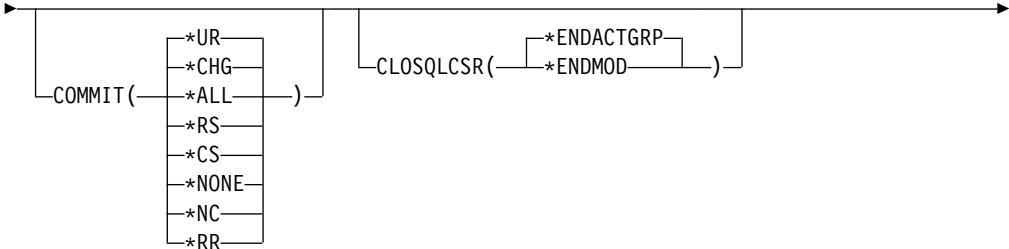
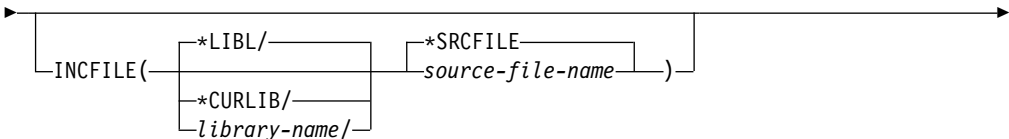
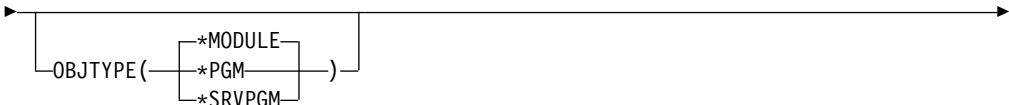
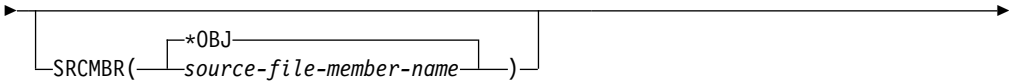
이 명령은 소스문을 사전컴파일하는 SQL 사전컴파일러를 실행하여 변경된 소스문을 라이브러리 QTEMP에 있는 파일 QSQLTEMP의 멤버 PAYROLL에 저장합니다. SQL 사전컴파일러에 의해 작성된 소스 멤버를 사용하여 현재 라이브러리에 모듈 PAYROLL을 작성하기 위해 ILE COBOL 컴파일러가 호출됩니다.

CRTSQLCI(구조화 조회 언어 ILE C 오브젝트 작성) 명령

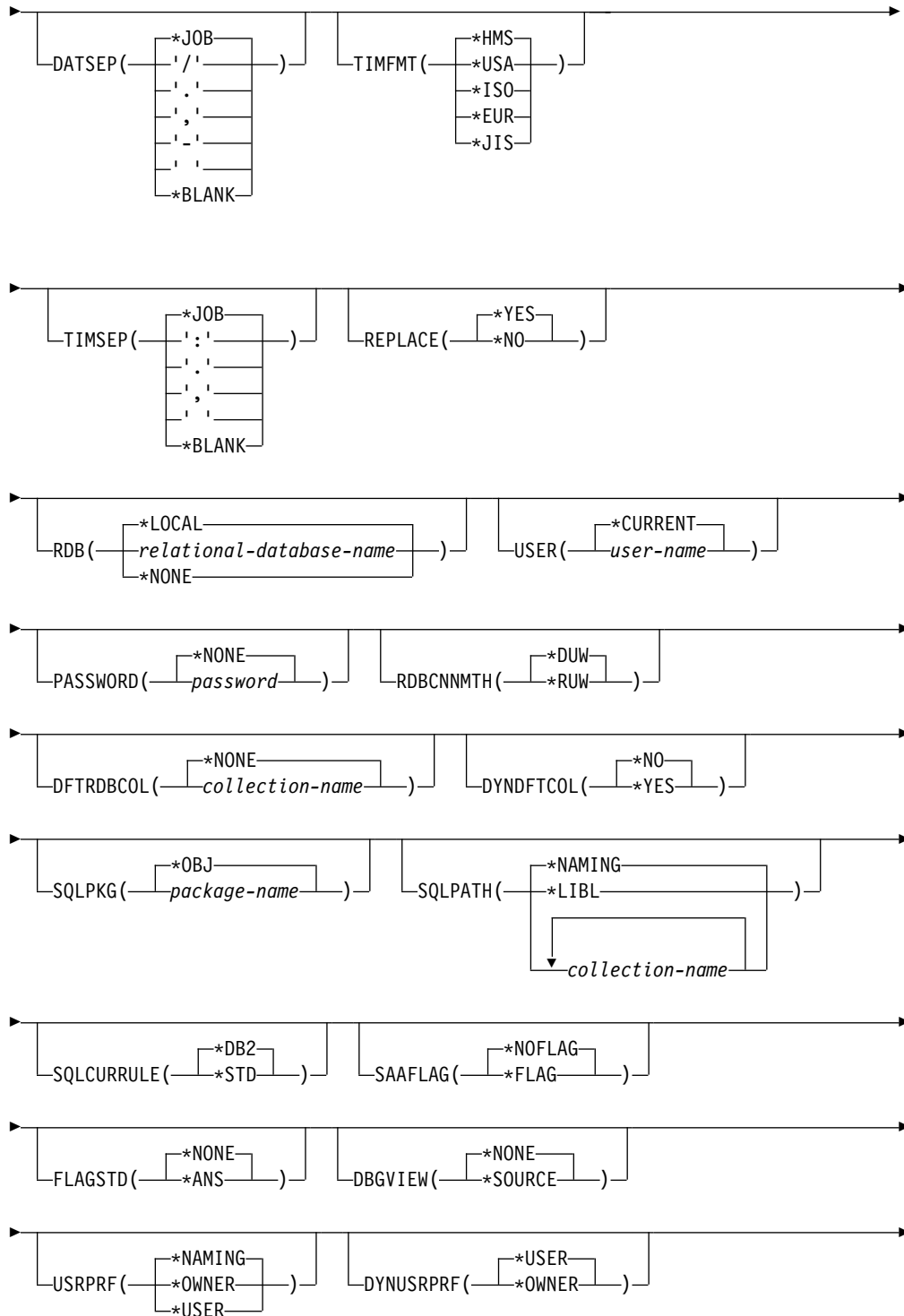
Job: B,I Pgm: B,I REXX: B,I Exec

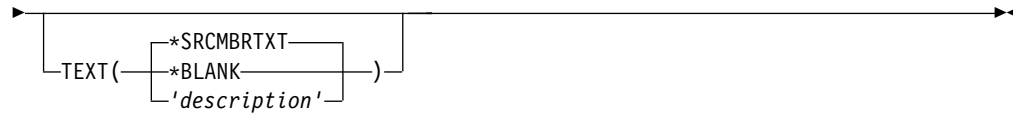
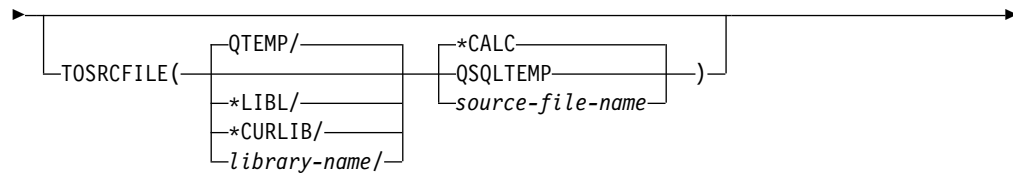
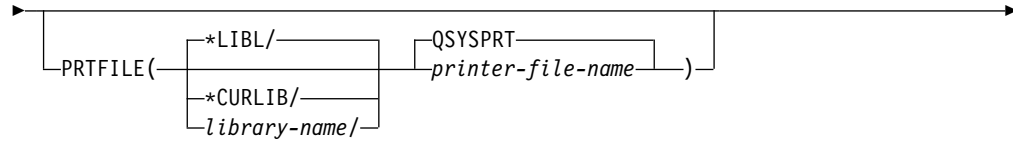
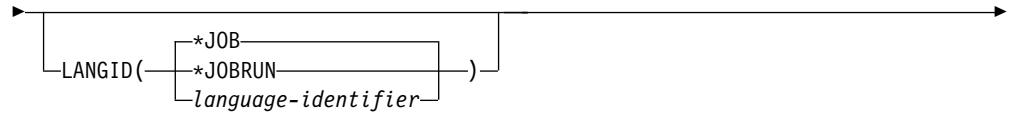
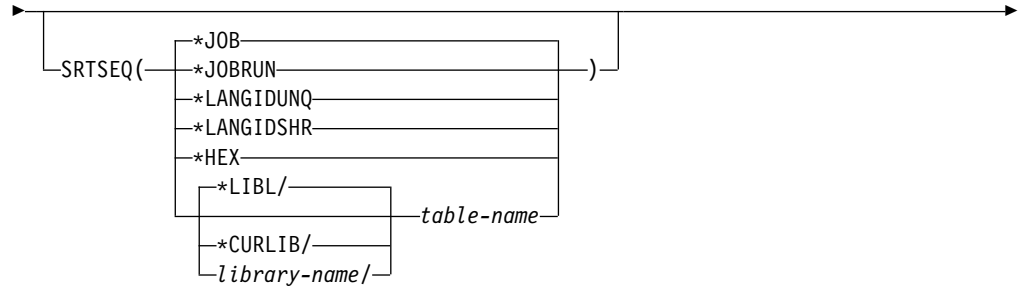


(1)

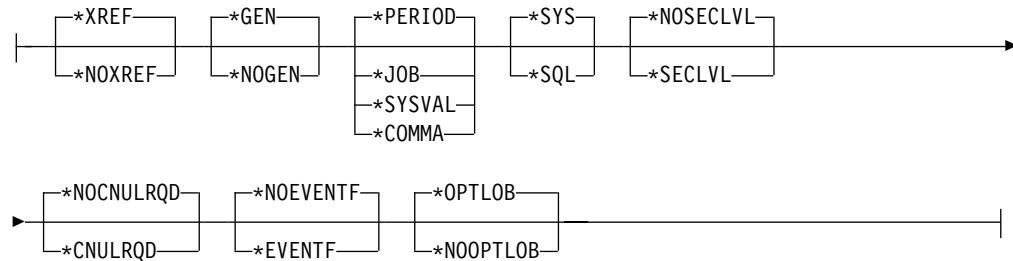


CRTSQLCI





OPTION 세부사항:



주:

1 이 앞에 오는 모든 매개변수는 앞뒤 관계에 따라 지정할 수 있습니다.

목적:

CRTSQLCI

구조화 조회 언어 ILE C 오브젝트 작성(CRTSQLCI) 명령은 SQL문에 있는 구조화 조회 언어(SQL) 사전컴파일러를 호출하고, 임시 소스 멤버를 작성하며 모듈, 프로그램 또는 서비스 프로그램을 작성하기 위하여 ILE C 컴파일러를 선택적으로 호출합니다.

매개변수:

OBJ

작성된 오브젝트의 규정화명을 지정합니다.

오브젝트의 이름은 다음 값 중 하나로 규정될 수 있습니다.

***CURLIB:** 오브젝트가 해당 작업에 대한 현재 라이브러리에 작성됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 오브젝트가 작성된 라이브러리의 이름을 지정하십시오.

object-name: 작성 중인 오브젝트의 이름을 지정하십시오.

SRCFILE

SQL문과 함께 C 소스문이 들어 있는 소스 파일의 규정된 이름을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QCSRC: 소스 파일명이 지정되지 않으면 IBM 제공 소스 파일 QCSRC에 C 소스가 포함됩니다.

source-file-name: C 소스가 들어 있는 소스 파일의 이름을 지정하십시오.

SRCMBR

C 소스문이 들어 있는 소스 파일 멤버명을 지정합니다. 이 매개변수는 SRCFILE 매개변수의 이름이 데이터베이스인 경우에만 지정됩니다. 이 매개변수가 지정되지 않으면 OBJ 매개변수에 지정된 OBJ명이 사용됩니다.

***OBJ:** C 소스가 OBJ 매개변수에 지정된 것과 같은 이름의 소스 파일 멤버에 있음을 지정합니다.

source-file-member-name: C 소스가 들어 있는 멤버의 이름을 지정하십시오.

OPTION

C 소스문이 사전컴파일될 때 다음 옵션 중 하나 또는 여러 개가 사용되는지의 여부를 지정합니다. 옵션이 두 번 이상 지정되면 마지막에 지정된 옵션이 사용됩니다.

요소 1: 상호 참조 옵션

***XREF:** 사전컴파일러는 그러한 항목을 참조하는 프로그램 내의 명령문 번호 항목을 상호 참조합니다.

***NOXREF:** 사전컴파일러는 이름을 상호 참조하지 않습니다.

요소 2: 프로그램 작성 옵션

***GEN:** 사전컴파일러가 OBJTYPE 매개변수에 의해 지정된 오브젝트를 작성합니다.

***NOGEN:** 사전컴파일러는 C 컴파일러를 호출하지 않으며, 모듈, 프로그램, 서비스 프로그램 또는 SQL 패키지가 작성되지 않습니다.

요소 3: 소수점 옵션

***PERIOD:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 마침표입니다.

***JOB:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 사전컴파일시 작업에 대해 지정된 소수점 표현입니다.

***SYSVAL:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 QDECFMT 시스템 값입니다.

주: QDECFMT가 소수점으로 사용되는 값을 쉼표로 지정하면 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)를 뒤에 공백이 오는 쉼표로 분리해야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 소수점이 마침표인 VALUES(1.1,2.23,4.1)와 동일합니다.

***COMMA:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 쉼표입니다.

주: 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)는 뒤에 공백이 오는 쉼표로 분리하십시오. 예를 들어 VALUES(1,1, 2,23, 4,1)는 마침표가 소수점으로 지정된 VALUES(1.1,2.23,4.1)와 동일합니다.

요소 4: 명명 규칙 옵션

***SYS:** 시스템 명명 규칙(library-name/file-name)이 사용됩니다.

***SQL:** SQL 명명 규칙(schema-name.table-name)이 사용됩니다. iSeries 시스템 이외의 리모트 데이터베이스에 패키지를 작성할 때 *SQL은 명명 규칙을 따라야 합니다.

요소 5: 1차 레벨 메시지 텍스트 옵션

***NOSECLVL:** 2차 레벨 텍스트 설명이 인쇄 출력에 추가되지 않습니다.

***SECLVL:** 대체 자료를 가진 2차 레벨 텍스트가 리스팅의 모든 메시지에 대해 추가됩니다.

요소 6: 필수 NUL 옵션

***NOCNULRQD:** 출력 문자 및 그래픽 호스트 변수의 경우 호스트 변수가 정확하게 자료와 같은 길이일 때는 NUL 종료자가 리턴되지 않습니다. 입력 문자와 그래픽 호스트 변수는 NUL 종료 프로그램을 필요로 하지 않습니다.

***CNULRQD:** 출력 문자와 그래픽 호스트 변수에 항상 NUL 종료자가 포함됩니다. NUL 종료자에 대한 충분한 공백이 없는 경우 자료가 절단되며 NUL 종료자가 종료됩니다. 입력 문자 및 그래픽 호스트 변수는 NUL 종료자를 요구합니다.

요소 7: 이벤트 파일 작성

***NOEVENTF:** 컴파일러가 CoOperative Development Environment/400(CODE/400)에서 사용할 이벤트 파일을 작성하지 않습니다.

***EVENTF:** 컴파일러가 CoOperative Development Environment/400(CODE/400)에서 사용할 이벤트 파일을 작성합니다. 이벤트 파일은 사용자 소스 라이브러리에 있는 파일 EVFEVENT의 멤버로 생성됩니다. CODE/400은 이 파일을 이용하여 CODE/400 에디터와 함께 통합된 오류 피드백을 제공합니다. 이 옵션은 일반적으로 사용자를 대신해서 CODE/400에 의해 지정됩니다.

요소 8: DRDA에 대한 대형 오브젝트 최적화

***OPTLOB:** 커서의 첫 번째 FETCH는 모든 후속 FETCH에서 커서가 LOB(대형 오브젝트)에 대해 사용될 방법을 결정합니다. 이 옵션은 커서가 닫힐 때까지 적용됩니다.

첫 번째 FETCH가 LOB 위치 지정자를 사용하여 LOB 열에 액세스할 경우 그 커서에 대한 어떤 후속 FETCH도 그 LOB 열을 LOB 호스트 변수에 페치할 수 없습니다.

첫 번째 FETCH가 LOB 열을 LOB 호스트 변수에 배치할 경우 그 커서에 대한 어떤 후속 FETCH도 그 열에 대해 LOB 위치 지정자를 사용할 수 없습니다.

***NOOPTLOB:** 열이 LOB 위치 지정자나 LOB 호스트 변수에 검색되는지에 대해서는 제한이 없습니다. 이 옵션을 사용하면 성능이 떨어질 수도 있습니다.

TGTRLS

작성될 오브젝트를 사용하려고 하는 오퍼레이팅 시스템의 릴리스 레벨을 지정합니다.

*CURRENT와 *PRV 값이 주어질 때와 릴리스 레벨 값을 지정할 때에는 VxRxMx의 형식으로 릴리스를 지정하게 되며, 여기서 Vx는 버전, Rx는 릴리스, Mx는 수정 레벨을 나타냅니다. 예를 들어, V2R3M0은 버전 2, 릴리스 3, 수정 레벨 0을 나타냅니다.

***CURRENT:** 오브젝트는 사용자 시스템에서 현재 실행 중인 오퍼레이팅 시스템의 릴리스에 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *CURRENT는 V2R3M5가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

주: 시스템에서 V2R3M5가 실행되고 있으며, V2R3M0가 설치된 시스템에서 오브젝트를 사용하려는 경우에는 TGTRLS(*CURRENT)가 아니고 TGTRLS(V2R3M0)를 지정해야 합니다.

***PRV:** 오브젝트는 오퍼레이팅 시스템의 이전 릴리스, 수정 레벨 0에서 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *PRV는 V2R2M0가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

release-level: VxRxMx 형식의 릴리스를 지정하십시오. 지정된 릴리스의 시스템이나 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스의 시스템에서도 오브젝트를 사용할 수 있습니다.

유효한 값들은 현재 버전, 릴리스, 수정 레벨에 기초하고 있으며, 새로운 릴리스로 변경될 수 있습니다. 이 명령이 지원하는 가장 이전의 릴리스 레벨보다 더 앞의 릴리스 레벨을 지정하면 지원되는 릴리스 레벨 가운데 가장 이전의 릴리스가 표시된 오류 메시지를 받게 됩니다.

OBJTYPE

작성될 오브젝트 유형을 지정합니다.

***MODULE:** SQL 사전컴파일러가 CRTCMOD 명령을 발행하여 모듈을 작성합니다.

***PGM:** SQL 사전컴파일러가 CRTBNDC 명령을 발행하여 바인드된 프로그램을 작성합니다.

***SRVPGM:** SQL 사전컴파일러가 CRTCMOD와 CRTSRVPGM 명령을 발행하여 서비스 프로그램을 작성합니다.

사용자는 OBJ 매개변수에 지정된 이름과 동일한 이름이 있는 QSRVSRV의 소스 멤버를 작성해야 합니다. 소스 멤버에는 모듈에 대한 내보내기 정보가 있어야 합니다. 내보낸 파일에 대한 자세한 정보는 Integrated Language Environment**C/400* Programmers Guide에 나와 있습니다.

주:

1. OBJTYPE(*PGM), OBJTYPE(*SRVPGM) 또는 RDB 매개변수 지정시 CRTSQLPKG 명령은 프로그램이 작성된 후 SQL 사전컴파일러에 의해 발행됩니다. OBJTYPE(*MODULE) 지정시, SQL 패키지는 작성되지 않으며 사용자가 CRTPGM 또는 CRTSRVPGM 명령에 의해 프로그램을 작성한 후 CRTSQLPKG 명령을 발행해야 합니다.
2. *NOGEN이 지정되는 경우 SQL 임시 소스 멤버만이 생성되며 모듈, 프로그램, 서비스 프로그램 또는 SQL 패키지는 작성되지 않습니다.

INCFILE

SQL INCLUDE문을 사용하여 프로그램에 포함시킨 멤버에 들어 있는 소스 파일의 규정화명을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리명을 지정하십시오.

***SRCFILE:** SRCFILE 매개변수에 지정된 규정된 소스 파일에는 모든 SQL INCLUDE문에 지정된 소스 파일 멤버가 포함됩니다.

source-file-name: 모든 SQL INCLUDE문에 지정된 소스 파일명을 포함하는 소스 파일의 이름을 지정하십시오. 여기에 지정된 소스 파일의 레코드 길이는 SRCFILE 매개변수에 지정된 소스 파일의 길이보다 짧아야 합니다.

COMMIT

컴파일된 장치에서 SQL문이 확약 제어하에 실행되는지를 지정합니다. 호스트 언어 소스문에 참조된 파일은 이 옵션의 영향을 받지 않습니다. SQL문에 참조된 SQL 표, SQL 뷰 및 SQL 패키지만이 영향을 받습니다.

***CHG 또는 *UR:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON 및 REVOKE문에서 참조된 오브젝트들을 지정하며, 갱신, 삭제 및 삽입된 행들은 작업 단위(트랜잭션)가 종료될 때까지 잠겨 있습니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다.

***ALL 또는 *RS:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입된 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***CS:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 갱

신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 선택은 되었지만 갱신되지 않은 행은 다음 행이 선택될 때까지 잠깁니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 없습니다.

***NONE** 또는 ***NC**: 예약 제어가 사용되지 않음을 지정합니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 있습니다. SQL DROP SCHEMA문이 프로그램에 포함되는 경우에는 *NONE 또는 *NC가 반드시 사용되어야 합니다. RDB 매개변수에 관계형 데이터베이스가 지정되어 있고 관계형 데이터베이스가 iSeries가 아닌 시스템에 있으면, *NONE 또는 *NC를 지정할 수 없습니다.

***RR**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 없습니다. SELECT, UPDATE, DELETE 및 INSERT문에 참조된 모든 표는 작업 단위(트랜잭션)가 종료될 때까지 배타적으로 잠겨 있습니다.

CLOSQLCSR

SQL 커서가 내재적으로 닫히고 SQL 준비 명령문이 내재적으로 취소되고 LOCK TABLE 잠금이 해제되는 시기를 지정합니다. SQL 커서는 CLOSE, COMMIT 또는 ROLLBACK(HOLD 없이) SQL문을 발행할 때 명시적으로 닫힙니다.

***ENDACTGRP**: 활성 그룹 종료시, SQL 커서는 닫히고 SQL 준비 명령문은 내재적으로 취소되며 LOCK TABLE 잠금은 해제됩니다.

***ENDMOD**: 모듈 종료시 SQL 커서는 닫히고 SQL 준비 명령문은 내재적으로 취소됩니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 LOCK TABLE 잠금이 해제됩니다.

ALWCPYDTA

자료의 사본이 SELECT문에 사용될 수 있는지 여부를 지정합니다.

***OPTIMIZE**: 시스템은 데이터베이스에서 직접 검색된 자료를 사용할지 또는 자료의 사본을 사용할지의 여부를 판별합니다. 결정은 최상의 성능을 제공할 수 있는 방법에 근거하여 이루어집니다. COMMIT가 *CHG 또는 *CS이면서 ALWBLK가 *ALLREAD가 아니거나, COMMIT가 *ALL 또는 *RR이면 조회 수행에 필요한 경우에만 자료의 사본이 사용됩니다.

***YES**: 필요할 때만 자료의 사본이 사용됩니다.

***NO**: 자료의 사본은 허용되지 않습니다. 조회를 수행하는 데 자료의 임시 사본이 필요하면 오류 메시지가 표시됩니다.

ALWBLK

데이터베이스 관리자가 레코드 블록화를 사용할 수 있는지 여부와 블록화가 읽기 전용 커서에 사용될 수 있는 확장 범위를 지정합니다.

***ALLREAD:** *NONE 또는 *CHG가 COMMIT 매개변수에 지정된 경우 읽기 전용 커서에 대해 행이 블록화됩니다. 명시적으로 갱신될 수 없는 프로그램 내의 모든 커서들은 EXECUTE 또는 EXECUTE IMMEDIATE문이 프로그램 내에 있을 지라도 읽기 전용으로 열립니다.

*ALLREAD 지정:

- *READ에 대해 허용된 블록화 이외의 제약 제어 레벨 *CHG하에서 레코드의 블록화를 허용합니다.
- 프로그램에서 거의 모든 읽기 전용 커서의 성능이 향상될 수 있으나 다음과 같은 방법으로 조회를 제한합니다.
 - 롤백(ROLLBACK) 명령, 호스트 언어의 ROLLBACK문 또는 ROLLBACK HOLD SQL문은 *ALLREAD가 지정된 경우 읽기 전용 커서의 위치를 재 지정하지 않습니다.
 - 커서에 대한 DECLARE문에 FOR UPDATE절이 없는 한, 지정된 UPDATE 또는 DELETE문의 동적 수행(예를 들면, EXECUTE IMMEDIATE 사용)은 커서의 행을 갱신하는 데 사용될 수 없습니다.

***NONE:** 커서에 대한 자료의 검색을 위해 행이 블록화되지 않습니다.

*NONE을 지정하는 경우:

- 검색된 자료가 현재의 자료임을 보장합니다.
- 조회시 자료의 첫 번째 행을 검색하는 데 필요한 시간을 줄일 수 있습니다.
- 조회가 닫히기 전에 조회의 처음 몇 행만이 검색될 때 프로그램에 의해 사용되지 않은 자료 행 블록의 검색으로부터 데이터베이스 관리자를 중단시킵니다.
- 많은 행을 검색하는 조회의 전반적인 성능이 저하될 수 있습니다.

***READ:** 레코드는 다음의 경우 커서에 대한 자료의 읽기 전용 검색에 대해 블록화됩니다.

- *NONE이 COMMIT 매개변수에 지정되어 제약 제어가 사용되지 않음을 표시하는 경우
- FOR READ ONLY절로 커서가 선언되었거나 커서에 대해 지정된 UPDATE 또는 DELETE문을 수행할 수 있는 동적 명령문이 없는 경우

*READ를 지정하면 위의 조건을 만족하는 조회의 전반적인 성능이 향상되고 많은 수의 레코드를 검색할 수 있습니다.

DLYPRP

PREPARE문에 대한 동적 명령문 유효성 검사가 OPEN, EXECUTE 또는 DESCRIBE문 실행시까지 지연되는지 여부를 지정합니다. 유효성 검사가 지연되면 불필요한 확인이 제거되므로 성능이 향상됩니다.

CRTSQLCI

***NO:** 동적 명령문 유효성 검사가 지연되지 않습니다. 동적 명령문이 준비될 때 액세스 계획의 유효성이 확인됩니다. 동적 명령문이 OPEN 또는 EXECUTE문에 사용될 때 액세스 계획의 유효성이 재확인됩니다. 동적 명령문에 의해 참조되는 오브젝트의 존재 또는 권한으로 인해, 동적 명령문이 아직 유효한지 확인하기 위해 OPEN 또는 EXECUTE문을 발행하기 전에 SQLCODE 또는 SQLSTATE를 체크해야 합니다.

***YES:** 동적 명령문 유효성 검사는 동적 명령문이 OPEN, EXECUTE 또는 DESCRIBE SQL문에 사용될 때까지 지연됩니다. 동적 명령문이 사용될 때 유효성 검사가 완료되고 액세스 계획이 수립됩니다. 이 매개변수에 *YES를 지정하면 OPEN, EXECUTE 또는 DESCRIBE문을 실행한 후에 SQLCODE와 SQLSTATE를 체크하여 동적 명령문이 유효한지 확인해야 합니다.

주: *YES를 지정하면 PREPARE에 INTO절이 사용되거나, 동적 명령문에 대한 OPEN문이 발행되기 전에 DESCRIBE문에서 그 동적 명령문을 사용하는 경우 성능은 향상되지 않습니다.

GENLVL

작성 조작이 실패하는 심각도 레벨을 지정합니다. 이 값 이상의 심각도를 가진 오류가 발생하면 조작이 종료됩니다.

10: 디폴트 심각도 레벨은 10입니다.

심각도 레벨: 0-40 범위 내의 값을 지정하십시오.

MARGINS

소스 텍스트가 들어 있는 사전컴파일러 입력 레코드의 부분을 지정합니다.

***SRCFILE:** 사전컴파일러는 사용자가 SRCMBR 매개변수에 지정한 멤버의 여백 값을 사용합니다.

요소 1: 좌측 여백

왼쪽: 명령문의 시작 위치를 지정합니다. 유효한 값의 범위는 1 - 32754입니다.

요소 2: 우측 여백

오른쪽: 명령문의 끝 위치를 지정합니다. 유효한 값의 범위는 1 - 32754입니다.

DATFMT

결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 날짜 필드가 지정된 형식으로 리턴됩니다. 입력 날짜 스트링의 경우 지정된 값은 유효한 형식으로 날짜가 지정되었는지 여부를 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

CRTSQLCI

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 iSeries 시스템이 아닌 시스템에 있으면, *USA, *ISO, *EUR 또는 *JIS가 지정되어야 합니다.

***JOB:** 작업에 지정된 형식이 사용됩니다. 작업 표시(DSPJOB) 명령을 사용하여 작업에 대한 현재 날짜 형식을 판별하십시오.

***USA:** 미국식 날짜 형식(mm/dd/yyyy)이 사용됩니다.

***ISO:** 국제 표준화 기구(ISO) 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***EUR:** 유럽식 날짜 형식(dd.mm.yyyy)이 사용됩니다.

***JIS:** 일본 산업 표준 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***MDY:** 날짜 형식(mm/dd/yy)이 사용됩니다.

***DMY:** 날짜 형식(dd/mm/yy)이 사용됩니다.

***YMD:** 날짜 형식(yy/mm/dd)이 사용됩니다.

***JUL:** 율리우스력 날짜 형식(yy/ddd)이 사용됩니다.

DATSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *JOB, *MDY, *DMY, *YMD 또는 *JUL이 DATFMT 매개변수에 지정되어 있을 때만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 날짜 분리자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘/’: 슬래시(/)가 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘-’: 붙임표(-)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

TIMFMT

결과 열에 액세스하기 위해 사용되는 형식을 지정합니다. 입력 시간 스트링의 경우 지정된 값은 시간이 유효한 형식으로 지정되었는지 판별하는 데 사용됩니다.

주: *USA, *ISO, *EUR 또는 *JIS 형식을 사용하는 입력 시간 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 또 다른 iSeries 시스템이 아닌 시스템에 있으면, 시간 형식이 콜론이나 마침표의 시간 분리문자가 있는 *USA, *ISO, *EUR, *JIS 또는 *HMS이어야 합니다.

***HMS:** hh:mm:ss 형식이 사용됩니다.

***USA:** 미국식 시간 형식 hh:mm xx가 사용되는데, 여기서 xx는 AM(오전) 또는 PM(오후)입니다.

***ISO:** 국제 표준화 기구(ISO) 시간 형식 hh.mm.ss가 사용됩니다.

***EUR:** 유럽식 시간 형식 hh.mm.ss가 사용됩니다.

***JIS:** 일본 산업 표준 시간 형식인 hh:mm:ss가 사용됩니다.

TIMSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *HMS가 TIMFMT 매개변수에 지정되어 있을 때에만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 시간 분리 문자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘:’: 콜론(:)이 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

REPLACE

동일한 이름의 기존 SQL 모듈, 프로그램, 서비스 프로그램 또는 패키지가 있고 동일한 라이브러리에 입력되었을 때 SQL 모듈, 프로그램, 서비스 프로그램 또는 패키지가 작성되는지를 지정합니다. 이 매개변수의 값은 CRTMODC, CRTBNDC, CRTSRVPGM 및 CRTSQLPKG 명령에 전달됩니다.

***YES:** 새로운 SQL 모듈, 서비스 프로그램 또는 패키지가 작성되고, 지정된 라이브러리에서 같은 이름과 유형을 가지는 기존의 모든 오브젝트는 QRPLOBJ로 이동됩니다.

CRTSQLCI

***NO:** 새로운 SQL 모듈, 프로그램, 서비스 프로그램 또는 패키지는 지정된 라이브러리에 동일한 이름과 유형의 오브젝트가 이미 존재하면 작성되지 않습니다.

RDB

SQL 패키지 오브젝트가 작성되는 관계형 데이터베이스의 이름을 지정합니다.

***LOCAL:** 프로그램은 분산 SQL 프로그램으로서 작성됩니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다. SQL 패키지 오브젝트는 사전컴파일 프로세스의 일부분으로서 작성되지 않습니다. 구조화 조회 언어 패키지 작성(CRTSQLPKG) 명령이 사용될 수 있습니다.

relational-database-name: 새로운 SQL 패키지 오브젝트가 작성될 때 관계형 데이터베이스의 이름을 지정하십시오. 로컬 관계형 데이터베이스명이 지정되면 작성된 프로그램은 계속 분산 SQL 프로그램입니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다.

***NONE:** SQL 패키지 오브젝트가 작성되지 않습니다. 프로그램 오브젝트는 분산 프로그램이 아니며 CRTSQLPKG(구조화 조회 언어 패키지 작성) 명령은 사용될 수 없습니다.

USER

변환 시작시 리모트 시스템으로 송신되는 사용자명을 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***CURRENT:** 현재 작업이 실행 중인 사용자 프로파일이 사용됩니다.

user-name: 어플리케이션 서버 작업에 사용되는 사용자명을 지정하십시오.

PASSWORD

리모트 시스템에 사용될 암호를 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***NONE:** 암호가 송신되지 않습니다. 이 값이 지정되면 USER(*CURRENT) 또한 지정되어야 합니다.

password: USER 매개변수에 지정된 사용자명의 암호를 지정하십시오.

RDBCNNMTH

CONNECT문에 사용되는 의미를 지정합니다. 자세한 정보는 SQL 참조서를 참조하십시오.

***DUW:** CONNECT(유형 2) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 추가 관계형 데이터베이스에 대한 연속된 CONNECT문이 이전의 연결을 단절시키지는 않습니다.

***RUW:** CONNECT(유형 1) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 연속된 CONNECT문은 새로운 연결이 설정되기 전에 이전의 연결을 단절시킵니다.

DFTRDBCOL

표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용되는 스키마명을 지정합니다. 이 매개변수는 정적 SQL문에만 적용됩니다.

***NONE:** OPTION 매개변수에 정의된 명명 규칙이 사용됩니다.

schema-name: 스키마 ID의 이름을 지정하십시오. 이 값은 OPTION 매개변수에 지정된 명명 규칙 대신 사용됩니다.

DYNDFTCOL

DFTRDBCOL 매개변수에 대해 지정된 디폴트 스키마명도 동적 명령문에 사용되는지를 지정합니다.

***NO:** 표, 뷰, 색인 및 동적 SQL문에 대한 SQL 패키지의 규정되지 않은 이름으로 DFTRDBCOL 매개변수에 지정된 값을 사용하지 마십시오. OPTION 매개변수에 지정된 명명 규칙이 사용됩니다.

***YES:** DFTRDBCOL 매개변수에 지정된 컬렉션명은 표, 뷰, 색인 및 동적 SQL문의 규정되지 않은 이름으로 사용됩니다.

SQLPKG

이 명령의 RDB 매개변수에 지정된 관계형 데이터베이스에 작성된 SQL 패키지의 규정된 이름을 지정합니다.

가능한 라이브러리 값은 다음과 같습니다.

***OBJLIB:** 패키지가 OBJ 매개변수에 지정된 라이브러리와 같은 이름의 라이브러리에 작성됩니다.

library-name: 패키지가 작성된 라이브러리의 이름을 지정하십시오.

***OBJ:** SQL 패키지의 이름은 OBJ 매개변수에 지정된 오브젝트명과 같습니다.

package-name: SQL 패키지의 이름을 지정하십시오. 리모트 시스템이 iSeries 시스템이 아니면, 9자 이상을 지정할 수 없습니다.

SQLPATH

정적 SQL문의 사용자 정의 유형, 프로시저 및 함수 등을 찾기 위해 사용될 경로를 지정합니다.

***NAMING:** 사용되는 경로는 OPTION 매개변수에 지정된 명명 규칙에 따릅니다.

***SYS** 명명의 경우 사용되는 경로는 *LIBL로, 실행시 현재 라이브러리 리스트입니다.

***SQL** 명명의 경우 사용되는 경로는 "QSYS", "QSYS2", "userid"입니다. 여기서 "userid"는 USER 특수 레지스터의 값입니다. DFTRDBCOL 매개변수에 schema-name을 지정한 경우 schema-name이 사용자 ID 대신 사용됩니다.

***LIBL:** 사용되는 경로는 실행시 라이브러리 리스트입니다.

CRTSQLCI

schema-name: 하나 이상의 스키마명 리스트를 지정하십시오. 최대 268개의 개별 스키마를 지정할 수 있습니다.

SQLCURRULE

SQL문에 사용되는 의미를 지정합니다.

***DB2:** 모든 SQL문의 의미에 대한 디폴트는 DB2에 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 문자 자료로 처리됩니다.

***STD:** 모든 SQL문의 의미에 대한 디폴트는 ISO 및 ANSI SQL 표준에서 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 2진 자료로 처리됩니다.

SA AFLAG

IBM SQL 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 IBM SQL 구문을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다. IBM SQL 구문을 생성하는 IBM 데이터베이스에 대한 정보는 *DRDA IBM SQL Reference*, SC26-3255-00 책을 참조하십시오.

***NOFLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사하지 않습니다.

***FLAG:** 사전컴파일러는 SQL문이 IBM SQL 구문에 따르는지 알아보기 위해 체크합니다.

FLAGSTD

미국 표준 협회(ANSI) 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 다음과 같은 표준을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다.

ANSI X3.135-1992 항목
ISO 9075-1992 항목
FIPS 127.2 항목

***NONE:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크하지 않습니다.

***ANS:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크합니다.

DBGVIEW

이 매개변수는 SQL 사전컴파일러에 의해 제공된 소스 수정 정보의 유형을 지정합니다.

***NONE:** 소스 뷰가 생성되지 않습니다.

***SOURCE:** SQL 사전컴파일러는 루트와 SQL INCLUDE문(필요한 경우)에 대한 소스 뷰를 제공합니다. 사전컴파일러에 의해 생성된 명령문이 있는 뷰가 제공됩니다.

USRPRF

정적 SQL문에서 각 오브젝트에 대해 프로그램 오브젝트가 가지는 권한을 포함하여 컴파일된 프로그램 오브젝트를 실행할 때 사용되는 사용자 프로파일을 지정합니다. 프로그램 소유자 또는 프로그램 사용자의 프로파일이 프로그램 오브젝트가 사용할 수 있는 오브젝트를 제어하는 데 사용됩니다.

***NAMING:** 사용자 프로파일은 명명 규칙에 의해 판별됩니다. 명명 규칙이 *SQL이면 USRPRF(*OWNER)가 사용됩니다. 명명 규칙이 *SYS이면 USRPRF(*USER)가 사용됩니다.

***USER:** 프로그램 오브젝트를 실행 중인 사용자의 프로파일이 사용됩니다.

***OWNER:** 프로그램 소유자와 프로그램 사용자 모두의 사용자 프로파일이 프로그램 실행시 사용됩니다.

DYNUSRPRF

동적 SQL문에 대해 사용될 사용자 프로파일을 지정합니다.

***USER:** 로컬 동적 SQL문은 프로그램 사용자의 파일하에서 실행됩니다. 분산 동적 SQL문은 SQL패키지의 사용자 프로파일하에서 실행됩니다.

***OWNER:** 로컬 동적 SQL문은 프로그램 소유자의 프로파일하에서 실행됩니다. 분산 동적 SQL문은 SQL 패키지 소유자의 프로파일하에서 실행됩니다.

SRTSEQ

SQL문의 스트링 비교에 사용될 정렬 순서표를 지정합니다.

주: *HEX는 어플리케이션 서버가 iSeries 시스템에 없거나 릴리스 레벨이 V2R3M0 이전인 분산 어플리케이션의 매개변수에 대해서만 지정되어야 합니다

***JOB:** 작업에 대한 SRTSEQ 값이 사전컴파일 중에 검색됩니다.

***JOBRUN:** 작업에 대한 LANGID 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 LANGID(*JOBRUN)는 SRTSEQ(*JOBRUN)가 지정될 때만 유효합니다.

***HEX:** 정렬 순서표가 사용되지 않습니다. 문자의 16진수 값이 정렬 순서를 결정하기 위해 사용됩니다.

***LANGIDSHR:** 정렬 순서표는 여러 문자에 대해 동일한 가중치를 사용하며, LANGID 매개변수에 지정된 언어와 연관된 공유 가중치 정렬 순서표입니다.

***LANGIDUNQ:** LANGID 매개변수에 지정된 고유 가중치 정렬표가 사용됩니다.

CRTSQLCI

표의 이름은 다음 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

table-name: 사용될 정렬 순서표의 이름을 지정하십시오.

LANGID

SRTSEQ(*LANGIDUNQ) 또는 SRTSEQ(*LANGIDSHR) 지정시 사용될 언어 ID를 지정합니다.

***JOB:** 작업에 대한 LANGID 값이 사전컴파일 중에 검색됩니다.

***JOBRUN:** 작업에 대한 LANGID 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 LANGID(*JOBRUN)는 SRTSEQ(*JOBRUN)가 지정될 때만 유효합니다.

language-identifier: 언어 ID를 지정하십시오.

OUTPUT

사전컴파일러 리스트가 생성되는지를 지정합니다.

***NONE:** 사전컴파일러 리스트가 생성되지 않습니다.

***PRINT:** 사전컴파일러 리스트가 생성됩니다.

PRTFILE

사전컴파일러 인쇄 출력을 지정한 프린터 파일의 규정화명을 지정합니다. 파일의 길이는 최소 132바이트이어야 합니다. 132바이트보다 적은 레코드 길이로 파일이 지정되는 경우 정보는 유실됩니다.

프린터 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QSYSPRT: 파일명이 지정되지 않으면 사전컴파일러 인쇄 출력이 IBM 제공 프린터 파일 QSYSPRT로 전달됩니다.

printer-file-name: 사전컴파일러 인쇄 출력이 전달될 프린터 파일의 이름을 지정하십시오.

TOSRCFILE

SQL 사전컴파일러가 처리한 출력 소스가 들어 소스 파일의 규정된 이름을 지정합니다. 사전컴파일러가 지정 소스 파일을 찾을 수 없는 경우에 이것은 소스 파일을 작성합니다. 출력 멤버는 SRCMBR 매개변수용으로 지정한 이름과 동일한 이름을 갖게 됩니다.

가능한 라이브러리 값은 다음과 같습니다.

QTEMP: 라이브러리 QTEMP가 사용될 것입니다.

***LIBL:** 작업 라이브러리 리스트가 지정 파일에 대해 탐색됩니다. 만일 라이브러리 리스트의 어떠한 라이브러리에서도 이 파일이 없는 경우에 파일은 현재 라이브러리에서 작성됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 사용될 것입니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없는 경우 QGPL 라이브러리가 사용될 것입니다.

library-name: 출력 소스 파일이 작성될 라이브러리명을 지정하십시오.

***CALC:** 출력 소스 파일명은 소스 파일의 여백에 따라 생성됩니다. 이름은 QSQLTxxxxx이며, 여기서 xxxxx는 소스 파일의 폭입니다. 소스 파일 레코드 길이가 92보다 작거나 같으면, 이름은 QSQLTEMP입니다.

QSQLTEMP: 소스 파일 QSQLTEMP가 사용될 것입니다.

source-file-name: 출력 소스 멤버가 들어 있는 소스 파일의 이름을 지정하십시오.

TEXT

프로그램과 그 기능을 간단하게 설명하는 텍스트를 지정합니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 TEXT 매개변수에 나와 있습니다.

***SRCMBRTXT:** C 프로그램을 작성하는 데 사용되는 소스 파일 멤버로부터 텍스트를 취합니다. 텍스트는 소스 입력 유틸리티 시작(STRSEU) 명령을 사용하거나 실제 파일 멤버 추가(ADDPFM) 명령이나 실제 파일 멤버 변경(CHGPFM) 명령을 사용하여 데이터베이스 소스 멤버에 대해 추가되거나 변경될 수 있습니다. 소스 파일이 인라인 파일이나 장치 파일일 경우 텍스트는 공백입니다.

***BLANK:** 텍스트가 지정되지 않습니다.

'description': 50자 미만의 텍스트를 어포스트로피로 묶어 지정하십시오.

예:

```
CRTSQLCI PAYROLL OBJTYPE(*MODULE)
TEXT('Payroll Program')
```

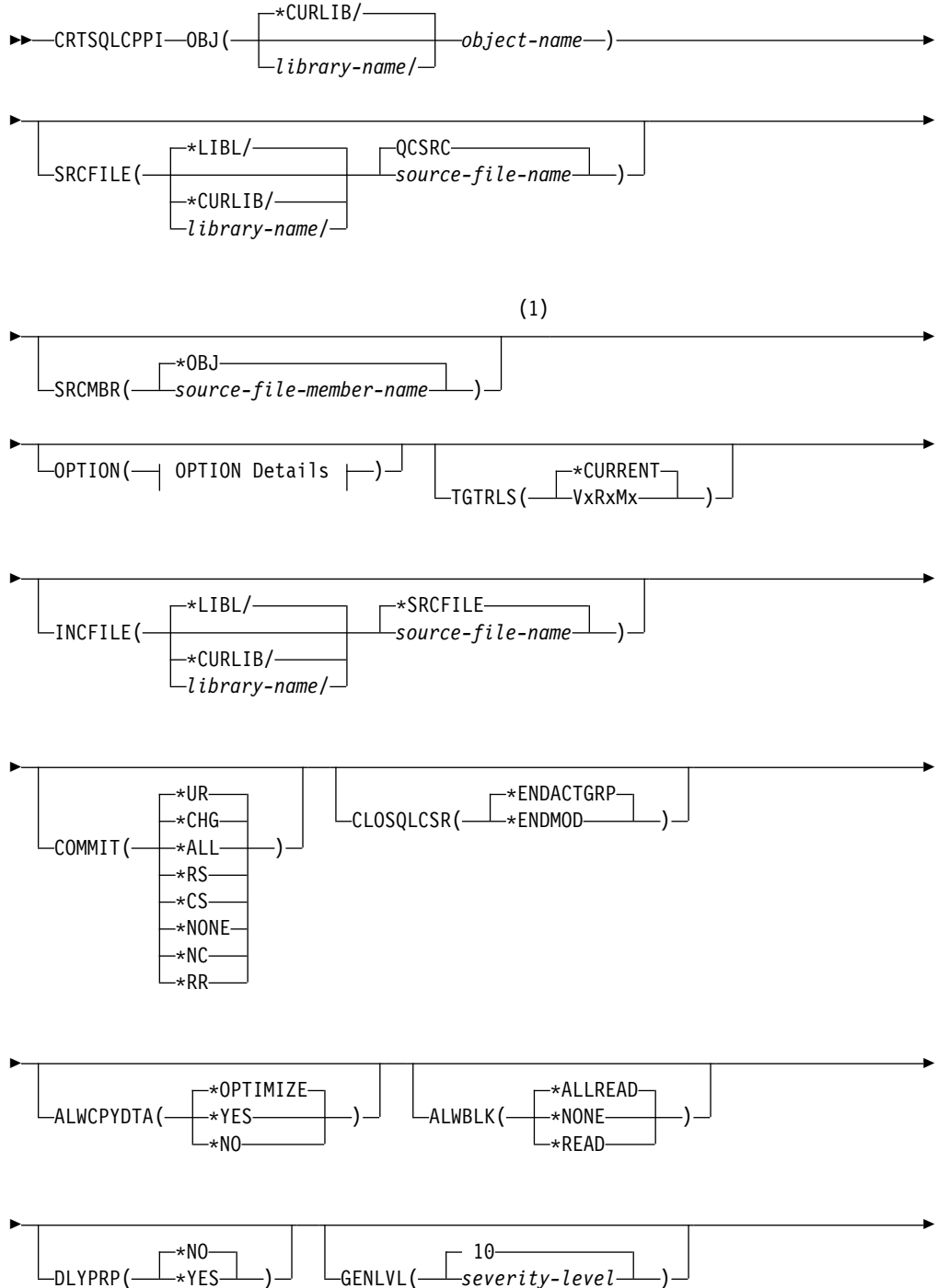
이 명령은 소스문을 사전컴파일하는 SQL 사전컴파일러를 실행하여 변경된 소스문을 라이브러리 QTEMP에 있는 파일 QSQLTEMP의 멤버 PAYROLL에 저장합니다.

CRTSQLCI

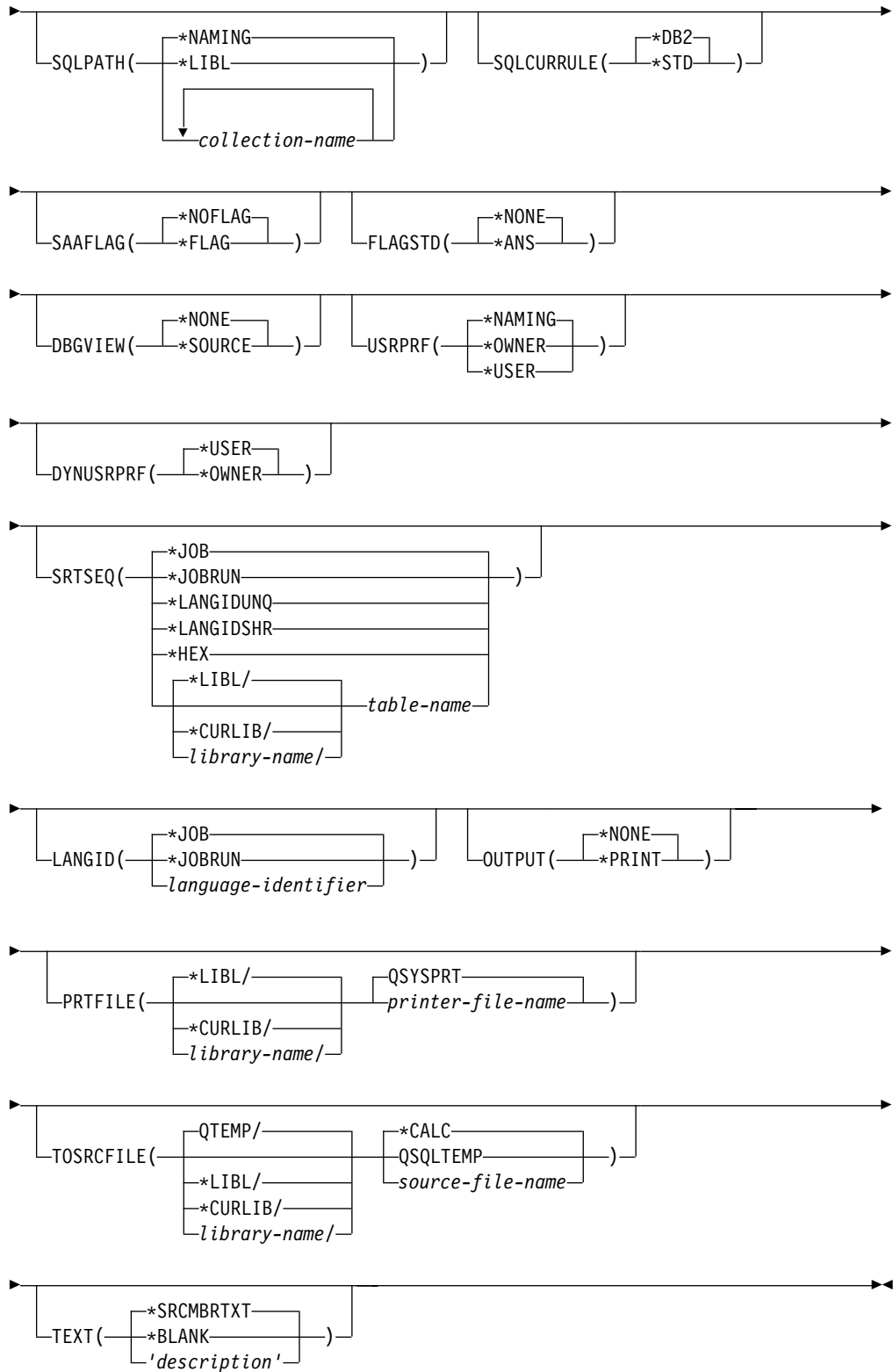
iSeries용 ILE C 컴파일러는 SQL 사전컴파일러에서 작성된 소스 멤버를 사용하여 현재 라이브러리에 모듈 PAYROLL을 작성하기 위해 호출됩니다.

CRTSQLCPPI(구조화 조회 언어 C++ 오브젝트 작성) 명령

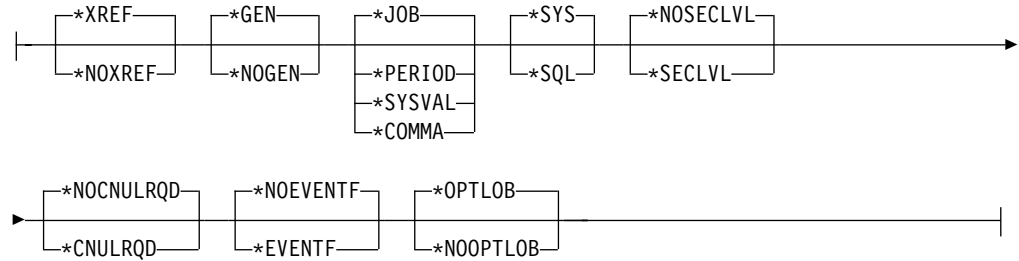
Job: B,I Pgm: B,I REXX: B,I Exec



CRTSQLCPPI



OPTION 세부사항:



주:

1 이 앞에 오는 모든 매개변수는 앞뒤 관계에 따라 지정할 수 있습니다.

목적:

CRTSQLCPPI(구조화 조회 언어 C++ 오브젝트 작성) 명령은 SQL 사전컴파일러를 호출합니다. SQL 사전컴파일러는 SQL문이 들어 있는 C++ 소스를 사전컴파일하고 임시 소스 멤버를 만든 후 선택적으로 C++ 컴파일러를 호출하여 모듈을 작성합니다.

매개변수:

OBJ

사전컴파일러가 작성한 오브젝트의 규정된 이름을 지정합니다.

오브젝트 이름은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***CURLIB** 오브젝트는 해당 작업에 대한 현재 라이브러리에 작성됩니다. 작업에 대한 현재 지정된 라이브러리가 없는 경우 사전컴파일러는 QGPL 라이브러리를 사용합니다.

library-name: 오브젝트가 작성된 라이브러리의 이름을 지정하십시오.

object-name: 사전컴파일러가 작성하는 오브젝트명을 지정합니다.

SRCFILE

SQL문과 함께 C++ 소스가 들어 있는 소스 파일의 규정된 이름을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 사전컴파일러는 처음으로 부합되는 부분이 발견될 때까지 작업 라이브러리 리스트에 있는 모든 라이브러리를 탐색합니다.

***CURLIB:** 사전컴파일러가 작업에 대한 현재 라이브러리를 탐색합니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없는 경우 QGPL 라이브러리가 사용됩니다.

library-name: 사전컴파일러가 탐색하는 라이브러리명을 지정합니다.

QCSRC: 소스 파일명이 지정되지 않으면 IBM 제공 소스 파일 QCSRC에 C++ 소스가 포함됩니다.

source-file-name: C++ 소스가 들어 있는 소스 파일의 이름을 지정하십시오.

SRCMBR

C++ 소스가 들어 있는 소스 파일 멤버명을 지정합니다. SRCFILE 매개변수의 소스 파일명이 데이터베이스 파일일 경우에만 이 매개변수를 지정하십시오. 이 매개변수를 지정하지 않는다면, 사전컴파일러는 OBJ 매개변수에 지정된 OBJ 이름을 사용합니다.

***OBJ:** C++ 소스가 OBJ 매개변수에 지정된 것과 같은 이름의 소스 파일 멤버에 있음을 지정합니다.

source-file-member-name: C++ 소스를 포함하는 멤버의 이름을 지정하십시오.

OPTION

C++ 소스문이 사전컴파일될 때 다음 옵션 중 하나 또는 여러 개가 사용되는지의 여부를 지정합니다. 옵션이 두 번 이상 지정되면 마지막에 지정된 옵션이 사용됩니다.

요소 1: 상호 참조 옵션

***XREF:** 사전컴파일러는 그러한 항목을 참조하는 프로그램 내의 명령문 번호 항목을 상호 참조합니다.

***NOXREF:** 사전컴파일러는 이름을 상호 참조하지 않습니다.

요소 2: 프로그램 작성 옵션

***GEN:** 사전컴파일러가 모듈 오브젝트를 작성합니다.

***NOGEN:** 사전컴파일러가 C++ 컴파일러를 호출하지 않고, 모듈이 작성되지 않습니다.

요소 3: 소수점 옵션

***JOB:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값은 사전컴파일시 작업에 대해 지정된 소수점 표현입니다.

주: 이 작업이 소수점으로 사용하는 값을 쉼표로 지정하면 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)를 뒤에 공백이 오는 쉼표로 분리해야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 소수점이 마침표인 VALUES(1.1,2.23,4.1)와 동일합니다.

***PERIOD:** SQL문에 있는 숫자 상수의 소수점으로 사용되는 값이 마침표입니다.

***COMMA:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 쉼표입니다.

주: 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)는 뒤에 공백이 오는 쉼표로 분리해야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 마침표가 소수점으로 지정된 VALUES(1.1,2.23,4.1)와 동일합니다.

요소 4: 명명 규칙 옵션

***SYS:** 시스템 명명 규칙(library-name/file-name)이 사용됩니다.

***SQL:** SQL 명명 규칙(schema-name.table-name)이 사용됩니다. iSeries 시스템 이외의 리모트 데이터베이스에서 패키지를 작성할 때 *SQL은 명명 규칙을 따라 지정되어야 합니다.

요소 5: 1차 레벨 메시지 텍스트 옵션

***NOSECLVL:** 2차 레벨 텍스트 설명이 인쇄 출력에 추가되지 않습니다.

***SECLVL:** 대체 자료를 가진 2차 레벨 텍스트가 리스팅의 모든 메시지에 대해 추가됩니다.

요소 6: 필수 NUL 옵션

***NOCNULRQD:** 출력 문자 및 그래픽 호스트 변수의 경우 호스트 변수가 정확하게 자료와 같은 길이일 때는 NUL 종료자가 리턴되지 않습니다. 입력 문자와 그래픽 호스트 변수는 NUL 종료 프로그램을 필요로 하지 않습니다.

***CNULRQD:** 출력 문자와 그래픽 호스트 변수에 항상 NUL 종료자가 포함됩니다. NUL 종료자에 대한 충분한 공백이 없는 경우 자료가 절단되며 NUL 종료자가 추가됩니다. 입력 문자 및 그래픽 호스트 변수는 NUL 종료자를 요구합니다.

요소 7: 이벤트 파일 작성

***NOEVENTF:** 컴파일러가 CoOperative Development Environment/400(CODE/400)에서 사용할 이벤트 파일을 작성하지 않습니다.

***EVENTF:** 컴파일러가 CoOperative Development Environment/400(CODE/400)에서 사용할 이벤트 파일을 작성합니다. 이벤트 파일은 소스 라이브러리 내의 파일 EVFEVENT의 멤버로서 작성됩니다. CODE/400은 이 파일을 이용하여 CODE/400 에디터와 함께 통합된 오류 피드백을 제공합니다. CODE/400은 사용자 편의를 위해 이러한 옵션을 정규적으로 지정합니다.

요소 8: DRDA에 대한 대형 오브젝트 최적화

***OPTLOB:** 커서의 첫 번째 FETCH는 모든 후속 FETCH에서 커서가 LOB(대형 오브젝트)에 대해 사용될 방법을 결정합니다. 이 옵션은 커서가 닫힐 때까지 적용됩니다.

첫 번째 FETCH가 LOB 위치 지정자를 사용하여 LOB 열에 액세스할 경우 그 커서에 대한 어떤 후속 FETCH도 그 LOB 열을 LOB 호스트 변수에 페치할 수 없습니다.

첫 번째 FETCH가 LOB 열을 LOB 호스트 변수에 배치할 경우 그 커서에 대한 어떤 후속 FETCH도 그 열에 대해 LOB 위치 지정자를 사용할 수 없습니다.

***NOOPTLOB:** 열이 LOB 위치 지정자나 LOB 호스트 변수에 검색되는지에 대해서는 제한이 없습니다. 이 옵션을 사용하면 성능이 떨어질 수도 있습니다.

TGTRLS

작성될 오브젝트를 사용하려고 하는 오퍼레이팅 시스템의 릴리스를 지정합니다.

릴리스 레벨 값 이외에 *CURRENT 값도 릴리스를 VxRxMx형식으로 지정합니다. 여기서, Vx는 버전 번호, Rx는 릴리스 번호 그리고 Mx는 수정 레벨입니다. 예를 들어, V2R3M0은 버전 2, 릴리스 3, 수정 레벨 0을 나타냅니다.

***CURRENT:** 오브젝트는 사용자 시스템에서 현재 실행 중인 오퍼레이팅 시스템의 릴리스에 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *CURRENT는 V2R3M5가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

주: 시스템에서 V2R3M5가 실행되고 있으며, V2R3M0이 설치된 시스템에서 오브젝트를 사용하려는 경우에는 TGTRLS(*CURRENT)가 아니고 TGTRLS(V2R3M0)를 지정해야 합니다.

release-level: VxRxMx 형식의 릴리스를 지정하십시오. 지정된 릴리스의 시스템이 나 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스의 시스템에서도 오브젝트를 사용할 수 있습니다.

유효한 값들은 현재 버전, 릴리스, 수정 레벨에 기초하고 있으며, 새로운 릴리스로 변경될 수 있습니다. 이 명령이 지원하는 가장 이전의 릴리스 레벨보다 더 앞의 릴리스 레벨을 지정하면 지원되는 릴리스 레벨 가운데 가장 이전의 릴리스가 표시된 오류 메시지가 전송됩니다.

INCFILE

SQL INCLUDE문을 사용하여 프로그램에 포함시킨 멤버에 있는 소스 파일의 규정된 이름을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리명을 지정하십시오.

***SRCFILE:** SRCFILE 매개변수에 지정된 규정된 소스 파일에는 모든 SQL INCLUDE문에 지정된 소스 파일 멤버가 포함됩니다.

source-file-name: 모든 SQL INCLUDE문에 지정된 소스 파일 멤버를 포함하는 소스 파일의 이름을 지정하십시오. 여기에 지정된 소스 파일의 레코드 길이는 SRCFILE 매개변수에 지정된 소스 파일의 길이보다 짧아야 합니다.

COMMIT

컴파일된 장치에서 SQL문이 확약 제어하에 실행되는지를 지정합니다. 호스트 언어 소스문에 참조된 파일은 이 옵션의 영향을 받지 않습니다. SQL문에 참조된 SQL 표, SQL 뷰 및 SQL 패키지만이 영향을 받습니다.

***CHG** 또는 ***UR**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON 및 REVOKE문에서 참조된 오브젝트들을 지정하며, 갱신, 삭제 및 삽입된 행들은 작업 단위(트랜잭션)가 종료될 때까지 잠겨 있습니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다.

***ALL** 또는 ***RS**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입된 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***CS**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)의 종료될 때까지 잠깁니다. 선택은 되었지만 갱신되지 않은 행은 다음 행이 선택될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***NONE** 또는 ***NC**: 확약 제어가 사용되지 않음을 지정합니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다. SQL DROP SCHEMA문이 프로그램에 포함되는 경우에는 *NONE 또는 *NC가 반드시 사용되어야 합니다. RDB 매개변수에 관계형 데이터베이스가 지정되어 있고 관계형 데이터베이스가 iSeries가 아닌 시스템에 있으면, *NONE 또는 *NC를 지정할 수 없습니다.

***RR**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다. SELECT, UPDATE, DELETE 및 INSERT문에 참조된 모든 표는 작업 단위(트랜잭션)가 종료될 때까지 배타적으로 잠겨 있습니다.

CLOSQLCSR

SQL 커서가 내재적으로 닫히고 SQL 준비 명령문이 내재적으로 취소되고 LOCK TABLE 잠금이 해제되는 시기를 지정합니다. SQL 커서는 CLOSE, COMMIT 또는 ROLLBACK(HOLD 없이) SQL문을 발행할 때 명시적으로 닫힙니다.

***ENDACTGRP**: 활성 그룹 종료시, SQL 커서는 닫히고 SQL 준비 명령문은 내재적으로 취소되며 LOCK TABLE 잠금은 해제됩니다.

***ENDMOD:** 모듈 종료시 SQL 커서는 닫히고 SQL 준비 명령문은 내재적으로 취소됩니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 LOCK TABLE 잠금이 해제됩니다.

ALWCPYDTA

자료의 사본이 SELECT문에 사용될 수 있는지 여부를 지정합니다.

***OPTIMIZE:** 시스템은 데이터베이스에서 직접 검색된 자료를 사용할지 또는 자료의 사본을 사용할지의 여부를 판별합니다. 결정은 최상의 성능을 제공할 수 있는 방법에 근거하여 이루어집니다. COMMIT가 *CHG 또는 *CS이면 ALWBLK가 *ALLREAD가 아니거나, COMMIT가 *ALL 또는 *RR이면 조회 수행에 필요한 경우에만 자료의 사본이 사용됩니다.

***YES:** 필요할 때만 자료의 사본이 사용됩니다.

***NO:** 자료의 사본은 허용되지 않습니다. 조회를 수행하는 데 자료의 임시 사본이 필요하면 오류 메시지가 표시됩니다.

ALWBLK

데이터베이스 관리자가 레코드 블록화를 사용할 수 있는지 여부와 블록화가 읽기 전용 커서에 사용될 수 있는 확장 범위를 지정합니다.

***ALLREAD:** *NONE 또는 *CHG가 COMMIT 매개변수에 지정된 경우 읽기 전용 커서에 대해 행이 블록화됩니다. 명시적으로 갱신될 수 없는 프로그램 내의 모든 커서들은 EXECUTE 또는 EXECUTE IMMEDIATE문이 프로그램 내에 있을 지라도 읽기 전용으로 열립니다.

***ALLREAD 지정:**

- *READ에 대해 허용된 블록화 이외의 확약 제어 레벨 *CHG하에서 레코드의 블록화를 허용합니다.
- 프로그램에서 거의 모든 읽기 전용 커서의 성능이 향상될 수 있으나 다음과 같은 방법으로 조회를 제한합니다.
 - 롤백(ROLLBACK) 명령, 호스트 언어의 ROLLBACK문 또는 ROLLBACK HOLD SQL문은 *ALLREAD가 지정된 경우 읽기 전용 커서의 위치를 재 지정하지 않습니다.
 - 커서에 대한 DECLARE문에 FOR UPDATE절이 없는 한, 지정된 UPDATE 또는 DELETE문의 동적 수행(예를 들면, EXECUTE IMMEDIATE 사용)은 커서의 행을 갱신하는 데 사용될 수 없습니다.

***NONE:** 커서에 대한 자료의 검색을 위해 행이 블록화되지 않습니다.

***NONE을 지정하는 경우:**

- 검색된 자료가 현재의 자료임을 보장합니다.
- 조회시 자료의 첫 번째 행을 검색하는 데 필요한 시간을 줄일 수 있습니다.

- 조회가 닫히기 전에 조회의 처음 몇 행만이 검색될 때 프로그램에 의해 사용되지 않은 자료 행 블록의 검색으로부터 데이터베이스 관리자를 중단시킵니다.
- 많은 행을 검색하는 조회의 전반적인 성능이 저하될 수 있습니다.

***READ:** 레코드는 다음의 경우 커서에 대한 자료의 읽기 전용 검색에 대해 블록화됩니다.

- *NONE이 COMMIT 매개변수에 지정되어 확약 제어가 사용되지 않음을 표시하는 경우
- FOR READ ONLY절로 커서가 선언되었거나 커서에 대해 지정된 UPDATE 또는 DELETE문을 수행할 수 있는 동적 명령문이 없는 경우

*READ를 지정하면 위의 조건을 만족하는 조회의 전반적인 성능이 향상되고 많은 수의 레코드를 검색할 수 있습니다.

DLYPRP

PREPARE문에 대한 동적 명령문 유효성 검사가 OPEN, EXECUTE 또는 DESCRIBE문 실행시까지 지연되는지 여부를 지정합니다. 유효성 검사가 지연되면 불필요한 확인이 제거되므로 성능이 향상됩니다.

***NO:** 동적 명령문 유효성 검사가 지연되지 않습니다. 동적 명령문이 준비될 때 액세스 계획의 유효성이 확인됩니다. 동적 명령문이 OPEN 또는 EXECUTE문에 사용될 때 액세스 계획의 유효성이 재확인됩니다. 동적 명령문에 의해 참조되는 오브젝트의 존재 또는 권한으로 인해, 동적 명령문이 아직 유효한지 확인하기 위해 OPEN 또는 EXECUTE문을 발행하기 전에 SQLCODE 또는 SQLSTATE를 체크해야 합니다.

***YES:** 동적 명령문 유효성 검사는 동적 명령문이 OPEN, EXECUTE 또는 DESCRIBE SQL문에 사용될 때까지 지연됩니다. 동적 명령문이 사용될 때 유효성 검사가 완료되고 액세스 계획이 수립됩니다. 이 매개변수에 *YES를 지정하면 OPEN, EXECUTE 또는 DESCRIBE문을 실행한 후에 SQLCODE와 SQLSTATE를 체크하여 동적 명령문이 유효한지 확인해야 합니다.

주: *YES를 지정하면 PREPARE에 INTO절이 사용되거나, 동적 명령문에 대한 OPEN문이 발행되기 전에 DESCRIBE문에서 그 동적 명령문을 사용하는 경우 성능은 향상되지 않습니다.

GENLVL

작성 조작이 실패하는 심각도 레벨을 지정합니다. 이 값 이상의 심각도를 가진 오류가 발생하면 조작이 종료됩니다.

10: 디폴트 심각도 레벨은 10입니다.

심각도 레벨: 0-40 범위 내의 값을 지정하십시오.

MARGINS

소스 텍스트가 들어 있는 사전컴파일러 입력 레코드의 부분을 지정합니다.

***SRCFILE:** 사용자가 SRCMBR 매개변수에 지정한 파일 멤버의 여백 값이 사용 됩니다.

요소 1: 좌측 여백

왼쪽: 명령문의 시작 위치를 지정합니다. 유효한 값의 범위는 1 - 32754입니다.

요소 2: 우측 여백

오른쪽: 명령문의 끝 위치를 지정합니다. 유효한 값의 범위는 1 - 32754입니다.

DATFMT

결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 날짜 필드가 지정 된 형식으로 리턴됩니다. 입력 날짜 스트링의 경우 지정된 값은 유효한 형식으로 날짜가 지정되었는지 여부를 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 iSeries 시스템이 아닌 시스템에 있으면, *USA, *ISO, *EUR 또는 *JIS가 지정되어야 합니다.

***JOB:** 작업에 지정된 형식이 사용됩니다. 작업 표시(DSPJOB) 명령을 사용하여 작업에 대한 현재 날짜 형식을 판별하십시오.

***USA:** 미국식 날짜 형식(mm/dd/yyyy)이 사용됩니다.

***ISO:** 국제 표준화 기구(ISO) 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***EUR:** 유럽식 날짜 형식(dd.mm.yyyy)이 사용됩니다.

***JIS:** 일본 산업 표준 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***MDY:** 날짜 형식(mm/dd/yy)이 사용됩니다.

***DMY:** 날짜 형식(dd/mm/yy)이 사용됩니다.

***YMD:** 날짜 형식(yy/mm/dd)이 사용됩니다.

***JUL:** 율리우스력 날짜 형식(yy/ddd)이 사용됩니다.

DATSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *JOB, *MDY, *DMY, *YMD 또는 *JUL이 DATFMT 매개변수에 지정되어 있을 때만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 날짜 분리자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘/’: 슬래시(/)가 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘-’: 붙임표(-)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

TIMFMT

결과 열에 액세스하기 위해 사용되는 형식을 지정합니다. 입력 시간 스트링의 경우 지정된 값은 시간이 유효한 형식으로 지정되었는지 판별하는 데 사용됩니다.

주: *USA, *ISO, *EUR 또는 *JIS 형식을 사용하는 입력 시간 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 또 다른 iSeries 시스템이 아닌 시스템에 있으면, 시간 형식이 콜론이나 마침표의 시간 분리문자가 있는 *USA, *ISO, *EUR, *JIS 또는 *HMS이어야 합니다.

***HMS:** hh:mm:ss 형식이 사용됩니다.

***USA:** 미국식 시간 형식 hh:mm xx가 사용되는데, 여기서 xx는 AM(오전) 또는 PM(오후)입니다.

***ISO:** 국제 표준화 기구(ISO) 시간 형식 hh.mm.ss가 사용됩니다.

***EUR:** 유럽식 시간 형식 hh.mm.ss가 사용됩니다.

***JIS:** 일본 산업 표준 시간 형식인 hh:mm:ss가 사용됩니다.

TIMSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *HMS가 TIMFMT 매개변수에 지정되어 있을 때에만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 시간 분리 문자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘:’: 콜론(:)이 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

REPLACE

동일한 라이브러리에 동일한 이름을 가진 SQL 모듈이 존재할 때 새로운 SQL 모듈이 작성되는지의 여부를 지정합니다. 이 매개변수 값은 CRTCPMOD 명령으로 전달됩니다.

***YES:** 새로운 SQL 모듈이 작성되고, 지정된 라이브러리 내에서 같은 이름을 가지는 기존의 오브젝트는 모두 QRPLOBJ로 이동됩니다.

***NO:** 새로운 SQL 모듈은 같은 이름의 오브젝트가 이미 지정된 라이브러리에 존재하는 경우 작성되지 않습니다.

RDB

SQL 패키지 오브젝트가 작성되는 관계형 데이터베이스의 이름을 지정합니다.

***LOCAL:** 프로그램은 분산 SQL 프로그램으로서 작성됩니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다. SQL 패키지 오브젝트는 사전컴파일 프로세스의 일부분으로서 작성되지 않습니다. 구조화 조회 언어 패키지 작성(CRTSQLPKG) 명령이 사용될 수 있습니다.

relational-database-name: 새로운 SQL 패키지 오브젝트가 작성될 때 관계형 데이터베이스의 이름을 지정하십시오. 로컬 관계형 데이터베이스명이 지정되면 작성된 프로그램은 계속 분산 SQL 프로그램입니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다.

***NONE:** SQL 패키지 오브젝트가 작성되지 않습니다. 프로그램 오브젝트는 분산 프로그램이 아니며 CRTSQLPKG(구조화 조회 언어 패키지 작성) 명령은 사용될 수 없습니다.

USER

변환 시작시 리모트 시스템으로 송신되는 사용자명을 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***CURRENT:** 현재 작업이 실행 중인 사용자 프로파일이 사용됩니다.

user-name: 어플리케이션 서버 작업에 사용되는 사용자명을 지정하십시오.

PASSWORD

리모트 시스템에 사용될 암호를 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***NONE:** 암호가 송신되지 않습니다. 이 값이 지정되면 USER(*CURRENT) 또한 지정되어야 합니다.

password: USER 매개변수에 지정된 사용자명의 암호를 지정하십시오.

RDBCNNMTH

SQL 패키지 오브젝트가 작성되는 관계형 데이터베이스의 이름을 지정합니다.

***DUW:** CONNECT(유형 2) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 추가 관계형 데이터베이스에 대한 연속된 CONNECT문이 이전의 연결을 단절시키지는 않습니다.

***RUW:** CONNECT(유형 1) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 연속된 CONNECT문은 새로운 연결이 설정되기 전에 이전의 연결을 단절시킵니다.

DFTRDBCOL

표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용되는 스키마명을 지정합니다. 이 매개변수는 정적 SQL문에만 적용됩니다.

***NONE:** OPTION 매개변수에 정의된 명명 규칙이 사용됩니다.

schema-name: 스키마 ID의 이름을 지정하십시오. 이 값은 OPTION 매개변수에 지정된 명명 규칙 대신 사용됩니다.

DYNDFTCOL

DFTRDBCOL 매개변수에 대해 지정된 디폴트 스키마명도 동적 명령문에 사용되 는지를 지정합니다.

***NO:** 표, 뷰, 색인 및 동적 SQL문에 대한 SQL 패키지의 규정되지 않은 이름으로 DFTRDBCOL 매개변수에 지정된 값을 사용하지 마십시오. OPTION 매개변수에 지정된 명명 규칙이 사용됩니다.

***YES:** DFTRDBCOL 매개변수에 지정된 컬렉션명은 표, 뷰, 색인 및 동적 SQL문의 규정되지 않은 이름으로 사용됩니다.

SQLPKG

이 명령의 RDB 매개변수에 지정된 관계형 데이터베이스에 작성된 SQL 패키지의 규정된 이름을 지정합니다.

가능한 라이브러리 값은 다음과 같습니다.

***OBJLIB:** 패키지가 OBJ 매개변수에 지정된 라이브러리와 같은 이름의 라이브러리에 작성됩니다.

library-name: 패키지가 작성된 라이브러리의 이름을 지정하십시오.

***OBJ:** SQL 패키지의 이름은 OBJ 매개변수에 지정된 오브젝트명과 같습니다.

package-name: SQL 패키지의 이름을 지정하십시오. 리모트 시스템이 iSeries 시스템이 아니면, 9자 이상을 지정할 수 없습니다.

SQLPATH

정적 SQL문의 사용자 정의 유형, 프로시저 및 함수 등을 찾기위해 사용될 경로를 지정합니다.

***NAMING:** 사용되는 경로는 OPTION 매개변수에 지정된 명명 규칙에 따릅니다.

***SYS** 명명의 경우 사용되는 경로는 *LIBL로, 실행시 현재 라이브러리 리스트입니다.

***SQL** 명명의 경우 사용되는 경로는 "QSYS", "QSYS2", "userid"입니다. 여기서 "userid"는 USER 특수 레지스터의 값입니다. DFTRDBCOL 매개변수에 schema-name을 지정한 경우 schema-name이 사용자 ID 대신 사용됩니다.

***LIBL:** 사용되는 경로는 실행시 라이브러리 리스트입니다.

schema-name: 하나 이상의 스키마명 리스트를 지정하십시오. 최대 268개의 개별 스키마를 지정할 수 있습니다.

SQLCURRULE

SQL문에 사용되는 의미를 지정합니다.

***DB2:** 모든 SQL문의 의미에 대한 디폴트는 DB2에 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 문자 자료로 처리됩니다.

***STD:** 모든 SQL문의 의미에 대한 디폴트는 ISO 및 ANSI SQL 표준에서 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 2진 자료로 처리됩니다.

SAAFLAG

IBM SQL 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 IBM SQL 구문을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다. IBM SQL 구문을 생성하는 IBM 데이터베이스에 대한 정보는 *DRDA IBM SQL Reference, SC26-3255-00* 책을 참조하십시오.

***NOFLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사하지 않습니다.

***FLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사합니다.

FLAGSTD

미국 표준 협회(ANSI) 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 다음과 같은 표준을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다.

- ANSI X3.135-1992 항목
- ISO 9075-1992 항목
- FIPS 127.2 항목

***NONE:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크하지 않습니다.

***ANS:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크합니다.

DBGVIEW

이 매개변수는 SQL 사전컴파일러에 의해 제공된 소스 수정 정보의 유형을 지정합니다.

***NONE:** 소스 뷰가 생성되지 않습니다.

***SOURCE:** SQL 사전컴파일러는 루트와 SQL INCLUDE문(필요한 경우)에 대한 소스 뷰를 제공합니다. 사전컴파일러에 의해 생성된 명령문이 있는 뷰가 제공됩니다.

USRPRF

정적 SQL문에서 각 오브젝트에 대해 프로그램 오브젝트가 가지는 권한을 포함하여 컴파일된 프로그램 오브젝트를 실행할 때 사용되는 사용자 프로파일을 지정합니다. 프로그램 소유자 또는 프로그램 사용자의 프로파일이 프로그램 오브젝트가 사용할 수 있는 오브젝트를 제어하는 데 사용됩니다.

***NAMING:** 사용자 프로파일은 명명 규칙에 의해 판별됩니다. 명명 규칙이 *SQL이면 USRPRF(*OWNER)가 사용됩니다. 명명 규칙이 *SYS이면 USRPRF(*USER)가 사용됩니다.

***USER:** 프로그램 오브젝트를 실행 중인 사용자의 프로파일이 사용됩니다.

***OWNER:** 프로그램 소유자와 프로그램 사용자 모두의 사용자 프로파일이 프로그램 실행시 사용됩니다.

DYNUSRPRF

동적 SQL문에 대해 사용될 사용자 프로파일을 지정합니다.

***USER:** 로컬 동적 SQL문은 프로그램 사용자의 파일하에서 실행됩니다. 분산 동적 SQL문은 SQL패키지의 사용자 프로파일하에서 실행됩니다.

***OWNER:** 로컬 동적 SQL문은 프로그램 소유자의 프로파일하에서 실행됩니다. 분산 동적 SQL문은 SQL 패키지 소유자의 프로파일하에서 실행됩니다.

SRTSEQ

SQL문의 스트링 비교에 사용될 정렬 순서표를 지정합니다.

주: *HEX는 어플리케이션 서버가 iSeries 시스템에 없거나 릴리스 레벨이 V2R3M0 이전인 분산 어플리케이션의 매개변수에 대해서만 지정되어야 합니다

***JOB:** 작업에 대한 SRTSEQ 값이 사전컴파일 중에 검색됩니다.

***JOB RUN:** 작업에 대한 SRTSEQ 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 SRTSEQ(*JOB RUN)는 LANGID(*JOB RUN)가 지정될 때만 유효합니다.

***HEX:** 정렬 순서표가 사용되지 않습니다. 문자의 16진수 값이 정렬 순서를 결정하기 위해 사용됩니다.

***LANGID SHR:** 정렬 순서표는 각 문자에 대해 고유한 가중치를 사용하며 LANGID 매개변수에 지정된 언어에 대한 고유한 가중치 정렬표입니다.

***LANGID UNQ:** LANGID 매개변수에 지정된 언어에 대한 고유 가중치 정렬표가 사용됩니다.

표의 이름은 다음 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

table-name: 표명을 지정하십시오.

LANGID

SRTSEQ(*LANGID UNQ) 또는 SRTSEQ(*LANGID SHR) 지정시 사용될 언어 ID를 지정합니다.

***JOB:** 작업에 대한 LANGID 값이 사전컴파일 중에 검색됩니다.

***JOB RUN:** 작업에 대한 LANGID 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 LANGID(*JOB RUN)는 SRTSEQ(*JOB RUN)가 지정될 때만 유효합니다.

language-identifier: 언어 ID를 지정하십시오.

OUTPUT

사전컴파일러 리스트가 생성되는지를 지정합니다.

***NONE:** 사전컴파일러 리스트가 생성되지 않습니다.

***PRINT:** 사전컴파일러 리스트가 생성됩니다.

PRTFILE

사전컴파일러 인쇄 출력을 지정한 프린터 파일의 규정화명을 지정합니다. 파일의 길이는 최소 132바이트이어야 합니다. 132바이트보다 적은 레코드 길이로 파일이 지정되는 경우 정보는 유실됩니다.

프린터 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QSYSPRT: 파일명이 지정되지 않으면 사전컴파일러 인쇄 출력이 IBM 제공 프린터 파일 QSYSPRT로 전달됩니다.

printer-file-name: 사전컴파일러 인쇄 출력이 전달될 프린터 파일의 이름을 지정하십시오.

TOSRCFILE

SQL 사전컴파일러에 의해 처리된 출력 소스 멤버가 있게 될 소스 파일의 규정화명을 지정합니다. 지정된 소스 파일을 찾을 수 없는 경우에는 이를 작성합니다. 출력 멤버는 SRCMBR 매개변수용으로 지정한 이름과 동일한 이름을 갖게 됩니다.

가능한 라이브러리 값은 다음과 같습니다.

QTEMP: 라이브러리 QTEMP가 사용될 것입니다.

***LIBL:** 작업 라이브러리 리스트가 지정 파일에 대해 탐색됩니다. 만일 라이브러리 리스트의 어떠한 라이브러리에서도 이 파일이 없는 경우에 파일은 현재 라이브러리에서 작성됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 사용될 것입니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없는 경우 QGPL 라이브러리가 사용될 것입니다.

library-name: 출력 소스 파일이 작성될 라이브러리명을 지정하십시오.

***CALC:** 출력 소스 파일명은 소스 파일의 여백에 따라 생성됩니다. 이름은 QSQLTxxxxx이며, 여기서 xxxxxx는 소스 파일의 폭입니다. 소스 파일 레코드 길이가 92보다 작거나 같으면, 이름은 QSQLTEMP입니다.

QSQLTEMP: 소스 파일 QSQLTEMP가 사용될 것입니다.

source-file-name: 출력 소스 멤버가 들어 있는 소스 파일의 이름을 지정하십시오.

TEXT

프로그램과 그 기능을 간단하게 설명하는 텍스트를 지정합니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 TEXT 매개변수에 나와 있습니다.

***SRCMBRTXT:** C++ 프로그램을 작성하는 데 사용되는 소스 파일 멤버로부터 텍스트를 취합니다. STRSEU(소스 입력 유틸리티 시작) 명령을 사용하여 데이터베이스 소스 멤버에 대한 텍스트를 추가하거나 변경할 수 있습니다. 또한 ADDPFM(실

CRTSQLCPPI

제 파일 멤버 추가) 명령 또는 CHGPFM(실제 파일 멤버 변경) 명령으로도 가능합니다. 소스 파일이 인라인 파일이나 장치 파일일 경우 텍스트는 공백입니다.

***BLANK:** 텍스트가 지정되지 않습니다.

'description': 50자 미만의 텍스트를 어포스트로피로 묶어 지정하십시오.

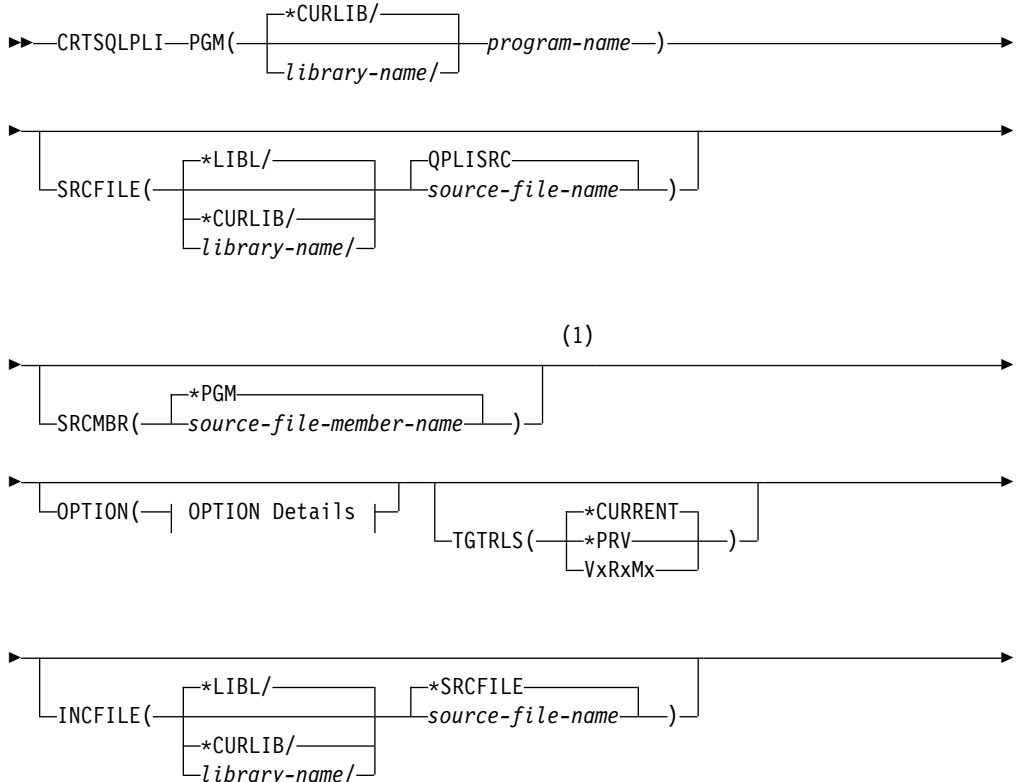
예:

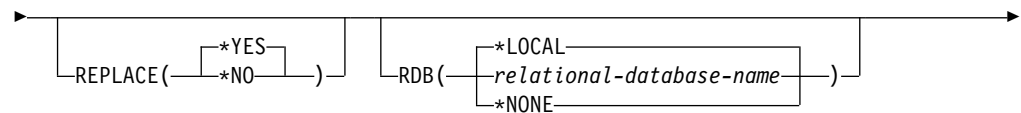
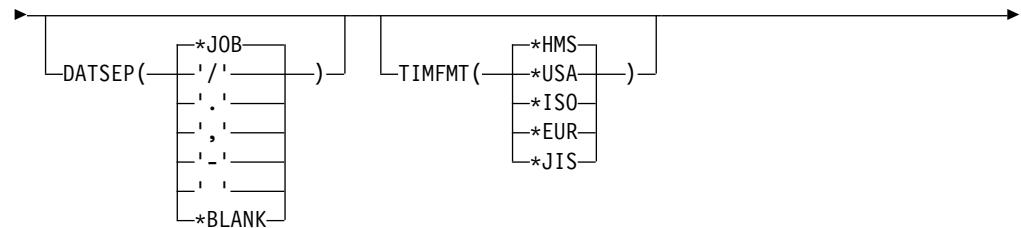
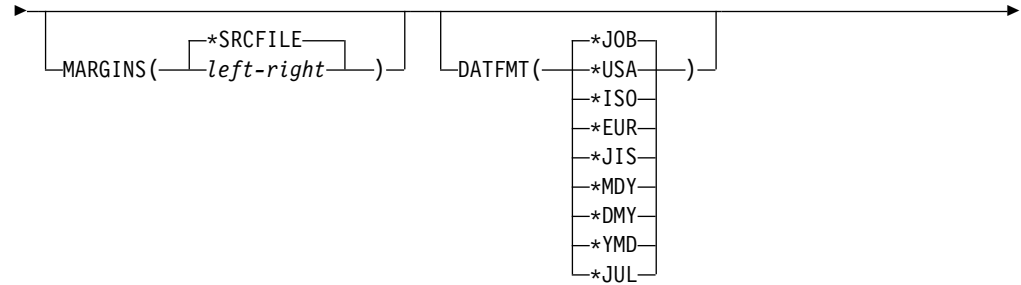
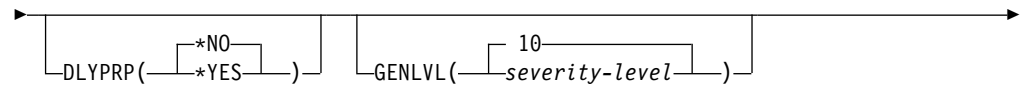
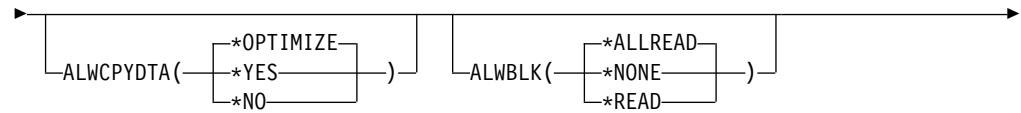
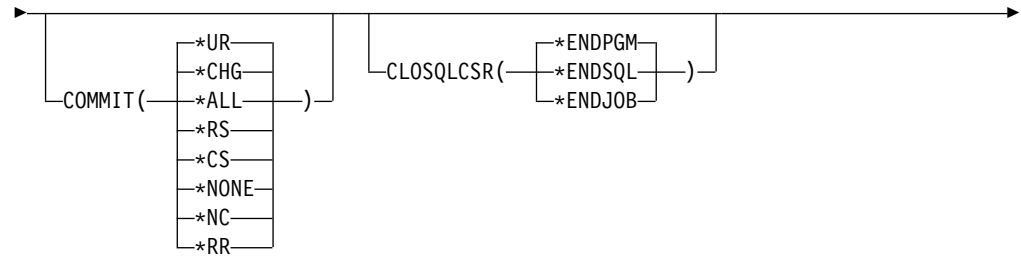
```
CRTSQLCPPI PAYROLL OBJTYPE(*MODULE)
TEXT('Payroll Program')
```

이 명령은 소스문을 사전컴파일하는 SQL 사전컴파일러를 실행하여 변경된 소스문을 라이브러리 QTEMP에 있는 파일 QSQLTEMP의 멤버 PAYROLL에 저장합니다. 이 명령은 SQL 사전컴파일러에 의해 작성된 소스 멤버를 사용하여 현재 라이브러리에 있는 모듈 PAYROLL을 작성하기 위해서 ILE C++ 컴파일러를 호출합니다.

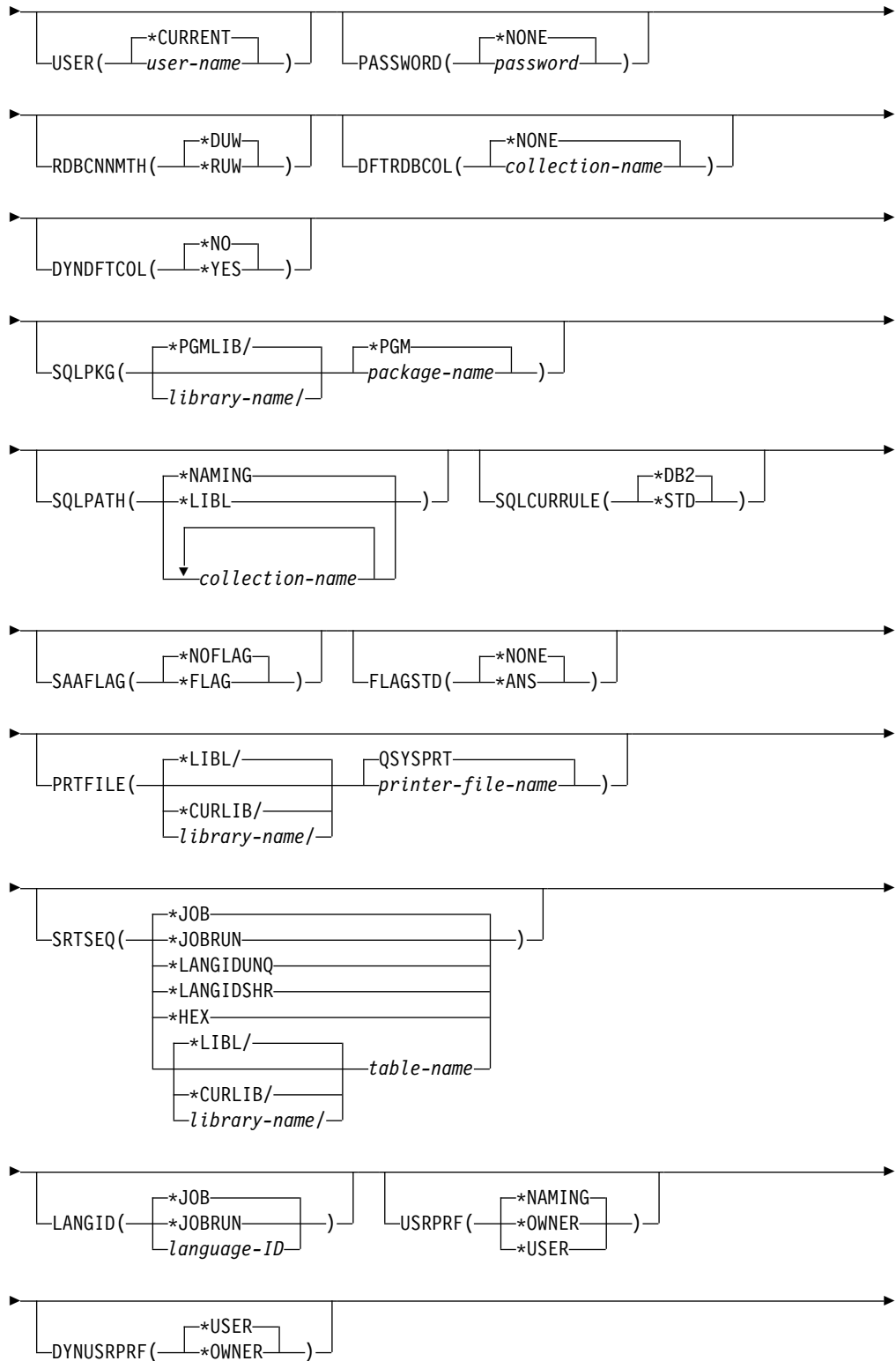
CRTSQLPLI(구조화 조회 언어 PL/I 작성) 명령

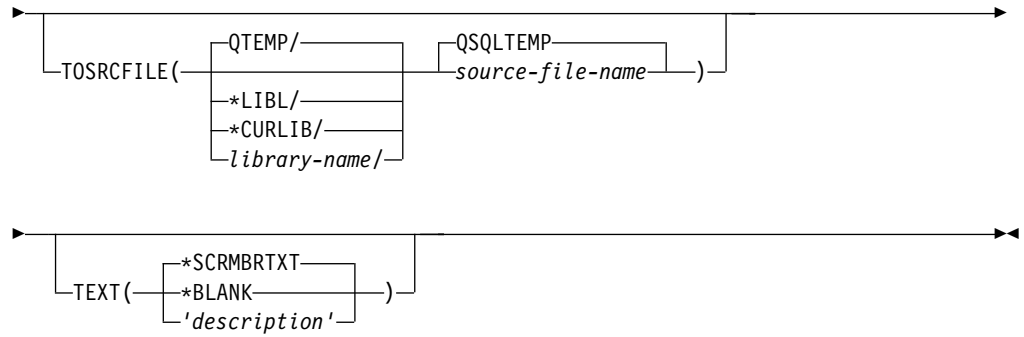
Job: B,I Pgm: B,I REXX: B,I Exec



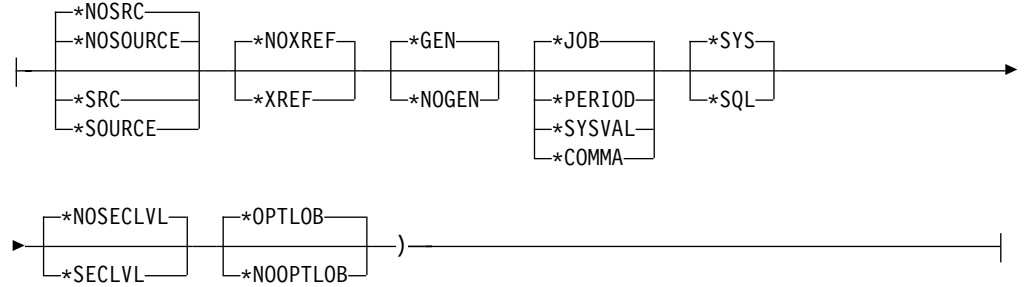


CRTSQLPLI





OPTION 세부사항:



주:

1 이 앞에 오는 모든 매개변수는 앞뒤 관계에 따라 지정할 수 있습니다.

목적:

CRTSQLPLI(구조화 조회 언어 PL/I 작성) 명령은 SQL문이 들어 있는 PL/I 소스문을 사전컴파일하는 구조화 조회 언어(SQL) 사전컴파일러를 호출하여 임시 소스 멤버를 작성한 후 선택적으로 프로그램을 컴파일하기 위해 PL/I 컴파일러를 호출합니다.

매개변수:

PGM

컴파일된 프로그램의 규정화명을 지정합니다.

컴파일된 PL/I 프로그램의 이름은 다음 값 중 하나로 규정될 수 있습니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 컴파일된 PL/I 프로그램이 작성된 라이브러리의 이름을 지정하십시오.

program-name: 컴파일된 프로그램의 이름을 지정하십시오.

SRCFILE

SQL문과 함께 PL/I 소스가 들어 있는 소스 파일의 규정된 이름을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

CRTSQLPLI

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QPLISRC: 소스 파일명이 지정되지 않은 경우 IBM 제공 소스 파일 QPLISRC에 PL/I 소스가 포함됩니다.

source-file-name: PL/I 소스가 들어 있는 소스 파일의 이름을 지정하십시오.

SRCMBR

PL/I 소스가 들어 있는 소스 파일 멤버명을 지정합니다. 이 매개변수는 SRCFILE 매개변수의 소스 파일명이 데이터베이스 파일일 경우에만 지정됩니다. 이 매개변수가 지정되지 않으면 PGM 매개변수에 지정된 PGM명이 사용됩니다.

***PGM:** PL/I 소스가 PGM 매개변수에 지정된 것과 같은 이름의 소스 파일 멤버에 있음을 지정합니다.

source-file-member-name: PL/I 소스가 들어 있는 멤버의 이름을 지정하십시오.

OPTION

PL/I 소스가 사전컴파일될 때 다음 옵션 중 하나 또는 여러 개가 사용되는지의 여부를 지정합니다. 옵션이 두 번 이상 지정되면 마지막에 지정된 옵션이 사용됩니다.

요소 1: 소스 리스팅 옵션

***NOSOURCE** 또는 ***NOSRC:** 사전컴파일러나 패키지 작성 중에 오류가 검출되지 않는 한, 소스 인쇄 출력이 사전컴파일러에 의해 작성되지 않습니다.

***SOURCE** 또는 ***SRC:** 사전컴파일러가 PL/I 소스 입력으로 구성된 소스 인쇄 출력을 작성합니다.

요소 2: 상호 참조 옵션

***NOXREF:** 사전컴파일러가 상호 참조명을 지정하지 않습니다.

***XREF:** 사전컴파일러는 그러한 항목을 참조하는 프로그램 내의 명령문 번호 항목을 상호 참조합니다.

요소 3: 프로그램 작성 옵션

***GEN:** 컴파일러는 프로그램 컴파일 후 수행될 수 있는 프로그램을 작성합니다. SQL 패키지 오브젝트는 데이터베이스명이 RDB 매개변수에 지정된 경우에 작성됩니다.

***NOGEN:** 사전컴파일러가 C 컴파일러를 호출하지 않고, 프로그램과 SQL 패키지가 작성되지 않습니다.

요소 4: 소수점 옵션

***JOB:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 사전컴파일시 작업에 대해 지정된 소수점 표현입니다.

***PERIOD:** SQL문에 사용된 숫자 상수에 대한 소수점으로 사용되는 값은 마침표입니다.

***SYSVAL:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 QDECFMT 시스템 값입니다.

주: QDECFMT가 소수점으로 사용되는 값을 쉼표로 지정하면 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)를 뒤에 공백이 오는 쉼표로 분리해야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 소수점이 마침표인 VALUES(1.1,2.23,4.1)와 동일합니다.

***COMMA:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 쉼표입니다.

주: 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)는 뒤에 공백이 오는 쉼표로 분리해야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 마침표가 소수점으로 지정된 VALUES(1.1,2.23,4.1)와 동일합니다.

요소 5: 명명 규칙 옵션

***SYS:** 시스템 명명 규칙(library-name/file-name)이 사용됩니다.

***SQL:** SQL 명명 규칙(schema-name.table-name)이 사용됩니다. iSeries 시스템 이외의 리모트 데이터베이스에서 프로그램을 작성할 때 *SQL은 명명 규칙을 따라 지정해야 합니다.

요소 6: 2차 레벨 메시지 텍스트 옵션

***NOSECLVL:** 2차 레벨 텍스트 설명이 인쇄 출력에 추가되지 않습니다.

***SECLVL:** 대체 자료를 가진 2차 레벨 텍스트가 리스트의 모든 메시지에 대한 인쇄 출력에 추가됩니다.

요소 7: DRDA 옵션에 대한 대형 오브젝트 최적화

***OPTLOB:** 커서의 첫 번째 FETCH는 모든 후속 FETCH에서 커서가 LOB(대형 오브젝트)에 대해 사용될 방법을 결정합니다. 이 옵션은 커서가 닫힐 때까지 적용됩니다.

첫 번째 FETCH가 LOB 위치 지정자를 사용하여 LOB 열에 액세스할 경우 그 커서에 대한 어떤 후속 FETCH도 그 LOB 열을 LOB 호스트 변수에 페치할 수 없습니다.

첫 번째 FETCH가 LOB 열을 LOB 호스트 변수에 배치할 경우 그 커서에 대한 어떤 후속 FETCH도 그 열에 대해 LOB 위치 지정자를 사용할 수 없습니다.

***NOOPTLOB:** 열이 LOB 위치 지정자나 LOB 호스트 변수에 검색되는지에 대해서는 제한이 없습니다. 이 옵션을 사용하면 성능이 떨어질 수도 있습니다.

TGTRLS

작성될 오브젝트를 사용하려고 하는 오퍼레이팅 시스템의 릴리스 레벨을 지정합니다.

***CURRENT**와 ***PRV** 값이 주어질 때와 릴리스 레벨 값을 지정할 때에는 VxRxMx의 형식으로 릴리스를 지정하게 되며, 여기서 Vx는 버전, Rx는 릴리스, Mx는 수정 레벨을 나타냅니다. 예를 들어, V2R3M0은 버전 2, 릴리스 3, 수정 레벨 0을 나타냅니다.

***CURRENT:** 오브젝트는 사용자 시스템에서 현재 실행 중인 오퍼레이팅 시스템의 릴리스에 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, ***CURRENT**는 V2R3M5가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

주: 시스템에서 V2R3M5가 실행되고 있으며, V2R3M0이 설치된 시스템에서 오브젝트를 사용하려는 경우에는 TGTRLS(***CURRENT**)가 아니고 TGTRLS(V2R3M0)를 지정해야 합니다.

***PRV:** 오브젝트는 오퍼레이팅 시스템의 이전 릴리스, 수정 레벨 0에서 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, ***PRV**는 V2R2M0이 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

release-level: VxRxMx 형식의 릴리스를 지정하십시오. 지정된 릴리스의 시스템이 나 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스의 시스템에서도 오브젝트를 사용할 수 있습니다.

유효한 값들은 현재 버전, 릴리스, 수정 레벨에 기초하고 있으며, 새로운 릴리스로 변경될 수 있습니다. 이 명령이 지원하는 가장 이전의 릴리스 레벨보다 더 앞의 릴리스 레벨을 지정하면 지원되는 릴리스 레벨 가운데 가장 이전의 릴리스가 표시된 오류 메시지를 받게 됩니다.

INCFILE

SQL INCLUDE문을 사용하여 프로그램에 포함시킨 멤버에 들어 있는 소스 파일의 규정화명을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

***SRCFILE:** SRCFILE 매개변수에 지정된 규정된 소스 파일에는 모든 SQL INCLUDE문에 지정된 소스 파일 멤버가 포함됩니다.

source-file-name: 모든 SQL INCLUDE문에 지정된 소스 파일명을 포함하는 소스 파일의 이름을 지정하십시오. 지정된 소스 파일의 레코드 길이가 SRCFILE 매개변수에 지정된 소스 파일의 레코드 길이보다 짧아야 합니다.

COMMIT

컴파일된 프로그램의 SQL문이 확약 제어하에서 실행되는지 여부를 지정합니다. 호스트 언어 소스문에 참조된 파일은 이 옵션의 영향을 받지 않습니다. SQL문에 참조된 SQL 표, SQL 뷰 및 SQL 패키지만이 영향을 받습니다.

***CHG 또는 *UR:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON 및 REVOKE문에서 참조된 오브젝트들을 지정하며, 갱신, 삭제 및 삽입된 행들은 작업 단위(트랜잭션)가 종료될 때까지 잠겨 있습니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다.

***ALL 또는 *RS:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입된 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***CS:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 선택은 되었지만 갱신되지 않은 행은 다음 행이 선택될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***NONE 또는 *NC:** 확약 제어가 사용되지 않음을 지정합니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다. SQL DROP SCHEMA문이 프로그램에 포함되는 경우에는 *NONE 또는 *NC가 반드시 사용되어야 합니다. RDB 매개변수에 관계형 데이터베이스가 지정되어 있고 관계형 데이터베이스가 iSeries가 아닌 시스템에 있으면, *NONE 또는 *NC를 지정할 수 없습니다.

***RR:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다. SELECT, UPDATE, DELETE 및 INSERT문에 참조된 모든 표는 작업 단위(트랜잭션)가 종료될 때까지 배타적으로 잠겨 있습니다.

CLOSQLCSR

SQL 커서가 내재적으로 닫히고 SQL 준비 명령문이 내재적으로 취소되고 LOCK TABLE 잠금이 해제되는 시기를 지정합니다. SQL 커서는 CLOSE, COMMIT 또는 ROLLBACK(HOLD 없이) SQL문을 발행할 때 명시적으로 닫힙니다.

***ENDPGM:** 프로그램 종료시 SQL 커서는 닫히고 SQL 준비 명령문은 취소됩니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 LOCK TABLE 잠금이 해제됩니다.

***ENDSQL:** SQL 커서가 호출 사이에 열린 상태로 남아 있고 다른 SQL OPEN을 실행하지 않고 폐치될 수 있습니다. 호출 스택상의 보다 높은 프로그램 중 하나가 최소한 하나의 SQL문을 실행해야 합니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 SQL 커서가 닫히고 준비된 SQL문이 취소되며, LOCK TABLE 잠금이 해제됩니다. *ENDSQL이 호출된 첫 번째 SQL 프로그램(호출 스택의 첫 번째 SQL 프로그램)인 프로그램에 대해 지정되는 경우 프로그램은 *ENDPGM이 지정된 것처럼 처리됩니다.

***ENDJOB:** SQL 커서가 호출 사이에 열린 상태로 남아 있고 다른 SQL OPEN을 실행하지 않고 폐치될 수 있습니다. 호출 스택에 나중에 들어간 프로그램은 SQL문을 실행할 필요가 없습니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 SQL 커서는 열린 상태로 남아있고, SQL 준비 명령문이 보존되며, LOCK TABLE 잠금이 보유됩니다. SQL 커서가 닫히고, SQL 준비 명령문은 취소되며 작업 종료시 LOCK TABLE 잠금이 해제됩니다.

ALWCOPYDTA

자료의 사본이 SELECT문에 사용될 수 있는지 여부를 지정합니다.

***OPTIMIZE:** 시스템은 데이터베이스에서 직접 검색된 자료를 사용할지 또는 자료의 사본을 사용할지의 여부를 판별합니다. 결정은 최상의 성능을 제공할 수 있는 방법에 근거하여 이루어집니다. COMMIT가 *CHG 또는 *CS이면 ALWBLK가 *ALLREAD가 아니거나, COMMIT가 *ALL 또는 *RR이면 조회 수행에 필요한 경우에만 자료의 사본이 사용됩니다.

***YES:** 필요할 때만 자료의 사본이 사용됩니다.

***NO:** 자료의 사본은 허용되지 않습니다. 조회를 수행하는 데 자료의 임시 사본이 필요하면 오류 메시지가 표시됩니다.

ALWBLK

데이터베이스 관리자가 레코드 블록화를 사용할 수 있는지 여부와 블록화가 읽기 전용 커서에 사용될 수 있는 확장 범위를 지정합니다.

***ALLREAD:** *NONE 또는 *CHG가 COMMIT 매개변수에 지정된 경우 읽기 전용 커서에 대해 행이 블록화됩니다. 명시적으로 갱신될 수 없는 프로그램 내의 모든 커서들은 EXECUTE 또는 EXECUTE IMMEDIATE문이 프로그램 내에 있을 지라도 읽기 전용으로 열립니다.

*ALLREAD 지정:

- *READ에 대해 허용된 블록화 이외의 확약 제어 레벨 *CHG하에서 레코드의 블록화를 허용합니다.
- 프로그램에서 거의 모든 읽기 전용 커서의 성능이 향상될 수 있으나 다음과 같은 방법으로 조회를 제한합니다.
 - 롤백(ROLLBACK) 명령, 호스트 언어의 ROLLBACK문 또는 ROLLBACK HOLD SQL문은 *ALLREAD가 지정된 경우 읽기 전용 커서의 위치를 재 지정하지 않습니다.
 - 커서에 대한 DECLARE문에 FOR UPDATE절이 없는 한, 지정된 UPDATE 또는 DELETE문의 동적 수행(예를 들면, EXECUTE IMMEDIATE 사용)은 커서의 행을 갱신하는 데 사용될 수 없습니다.

*NONE: 커서에 대한 자료의 검색을 위해 행이 블록화되지 않습니다.

*NONE을 지정하는 경우:

- 검색된 자료가 현재의 자료임을 보장합니다.
- 조회시 자료의 첫 번째 행을 검색하는 데 필요한 시간을 줄일 수 있습니다.
- 조회가 닫히기 전에 조회의 처음 몇 행만이 검색될 때 프로그램에 의해 사용되지 않은 자료 행 블록의 검색으로부터 데이터베이스 관리자를 중단시킵니다.
- 많은 행을 검색하는 조회의 전반적인 성능이 저하될 수 있습니다.

*READ: 레코드는 다음의 경우 커서에 대한 자료의 읽기 전용 검색에 대해 블록화됩니다.

- *NONE이 COMMIT 매개변수에 지정되어 확약 제어가 사용되지 않음을 표시하는 경우
- FOR READ ONLY절로 커서가 선언되었거나 커서에 대해 지정된 UPDATE 또는 DELETE문을 수행할 수 있는 동적 명령문이 없는 경우

*READ를 지정하면 위의 조건을 만족하는 조회의 전반적인 성능이 향상되고 많은 수의 레코드를 검색할 수 있습니다.

DLYPRP

PREPARE문에 대한 동적 명령문 유효성 검사가 OPEN, EXECUTE 또는 DESCRIBE문 실행시까지 지연되는지 여부를 지정합니다. 유효성 검사가 지연되면 불필요한 확인이 제거되므로 성능이 향상됩니다.

***NO:** 동적 명령문 유효성 검사가 지연되지 않습니다. 동적 명령문이 준비될 때 액세스 계획의 유효성이 확인됩니다. 동적 명령문이 OPEN 또는 EXECUTE문에 사용될 때 액세스 계획의 유효성이 재확인됩니다. 동적 명령문에 의해 참조되는 오브

CRTSQLPLI

젝트의 존재 또는 권한으로 인해, 동적 명령문이 아직 유효한지 확인하기 위해 OPEN 또는 EXECUTE문을 발행하기 전에 SQLCODE 또는 SQLSTATE를 검사해야 합니다.

***YES:** 동적 명령문 유효성 검사는 동적 명령문이 OPEN, EXECUTE 또는 DESCRIBE SQL문에 사용될 때까지 지연됩니다. 동적 명령문이 사용될 때 유효성 검사가 완료되고 액세스 계획이 수립됩니다. 이 매개변수에 *YES를 지정하면 OPEN, EXECUTE 또는 DESCRIBE문을 실행한 후에 SQLCODE와 SQLSTATE를 체크하여 동적 명령문이 유효한지 확인해야 합니다.

주: *YES를 지정하면 PREPARE에 INTO절이 사용되거나, 동적 명령문에 대한 OPEN문이 발행되기 전에 DESCRIBE문에서 그 동적 명령문을 사용하는 경우 성능은 향상되지 않습니다.

GENLVL

변경되지 않는 프로그램의 오류 심각도 레벨을 지정합니다. 프로그램에서 SQL문 준비 동안에 이 매개변수에 지정된 값 이상의 심각도 레벨 오류가 발생한다면, 프로그램은 변경될 수 없습니다.

10: 디폴트 심각도 레벨은 10입니다.

심각도 레벨: 0-40 범위 내의 값을 지정하십시오.

MARGINS

소스 텍스트가 들어 있는 사전컴파일러 입력 레코드의 부분을 지정합니다.

***SRCFILE:** 사용자가 SRCMBR 매개변수에 지정한 파일 멤버의 여백 값이 사용됩니다. 멤버가 SQLPLI 소스 유형이면 여백 값은 SEU 서비스 화면에 지정된 값입니다. 멤버가 다른 소스 유형이면 여백값은 2-72 사이의 디폴트 값입니다.

요소 1: 좌측 여백

왼쪽: 명령문의 시작 위치를 지정합니다. 유효한 값의 범위는 1-80 사이입니다.

요소 2: 우측 여백

오른쪽: 명령문의 끝 위치를 지정합니다. 유효한 값의 범위는 1-80 사이입니다.

DATFMT

결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 날짜 필드가 지정된 형식으로 리턴됩니다. 입력 날짜 스트링의 경우 지정된 값은 유효한 형식으로 날짜가 지정되었는지 여부를 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 iSeries 시스템이 아닌 시스템에 있으면, *USA, *ISO, *EUR 또는 *JIS가 지정되어야 합니다.

***JOB:** 작업에 지정된 형식이 사용됩니다. 작업 표시(DSPJOB) 명령을 사용하여 작업에 대한 현재 날짜 형식을 판별하십시오.

***USA:** 미국식 날짜 형식(mm/dd/yyyy)이 사용됩니다.

***ISO:** 국제 표준화 기구(ISO) 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***EUR:** 유럽식 날짜 형식(dd.mm.yyyy)이 사용됩니다.

***JIS:** 일본 산업 표준 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***MDY:** 날짜 형식(mm/dd/yy)이 사용됩니다.

***DMY:** 날짜 형식(dd/mm/yy)이 사용됩니다.

***YMD:** 날짜 형식(yy/mm/dd)이 사용됩니다.

***JUL:** 율리우스력 날짜 형식(yy/ddd)이 사용됩니다.

DATSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *JOB, *MDY, *DMY, *YMD 또는 *JUL이 DATFMT 매개변수에 지정되어 있을 때만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 날짜 분리자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘/’: 슬래시(/)가 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘-’: 붙임표(-)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

TIMFMT

결과 열에 액세스하기 위해 사용되는 형식을 지정합니다. 입력 시간 스트링의 경우 지정된 값은 시간이 유효한 형식으로 지정되었는지 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 또 다른 iSeries 시스템이 아닌 시스템에 있으면, 시간 형식이 콜론이나 마침표의 시간 분리문자가 있는 *USA, *ISO, *EUR, *JIS 또는 *HMS이어야 합니다.

***HMS:** (hh:mm:ss) 형식이 사용됩니다.

***USA:** 미국식 시간 형식(hh:mm xx)이 사용되는데, 여기서 xx는 AM(오전) 또는 PM(오후)입니다.

***ISO:** 국제 표준화 기구(ISO) 시간 형식(hh.mm.ss)이 사용됩니다.

***EUR:** 유럽식 시간 형식(hh.mm.ss)이 사용됩니다.

***JIS:** 일본 산업 표준 시간 형식(hh:mm:ss)이 사용됩니다.

TIMSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *HMS가 TIMFMT 매개변수에 지정되어 있을 때에만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 시간 분리 문자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘.’: 콜론(:)이 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘.’: 쉼표(,)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

REPLACE

동일한 이름의 프로그램 또는 SQL 패키지가 동일한 라이브러리에 있을 때 새로운 프로그램 또는 SQL 패키지가 작성되는지를 지정합니다. 이 매개변수 값은 CRTPLIPGM 명령으로 전달됩니다. 이 매개변수에 대한 자세한 정보는 CL 참조서의 부록 A, "확장 매개변수 설명"을 참조하십시오.

***YES:** 새로운 패키지나 SQL 패키지가 작성되고, 지정된 라이브러리 내에서 같은 이름 및 유형을 가지는 기존의 프로그램 또는 SQL 패키지는 QRPLOBJ로 이동됩니다.

***NO:** 새로운 프로그램 또는 SQL 패키지는 같은 이름과 유형의 오브젝트가 이미 지정된 라이브러리에 존재하는 경우 작성되지 않습니다.

RDB

SQL 패키지 오브젝트가 작성되는 관계형 데이터베이스의 이름을 지정합니다.

***LOCAL:** 프로그램은 분산 SQL 프로그램으로서 작성됩니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다. SQL 패키지 오브젝트는 사전컴파일 프로세스의 일부분으로서 작성되지 않습니다. 구조화 조회 언어 패키지 작성(CRTSQLPKG) 명령이 사용될 수 있습니다.

relational-database-name: 새로운 SQL 패키지 오브젝트가 작성될 때 관계형 데이터베이스의 이름을 지정하십시오. 로컬 관계형 데이터베이스명이 지정되면 작성된 프로그램은 계속 분산 SQL 프로그램입니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다.

***NONE:** SQL 패키지 오브젝트가 작성되지 않습니다. 프로그램 오브젝트는 분산 프로그램이 아니며 CRTSQLPKG(구조화 조회 언어 패키지 작성) 명령은 사용될 수 없습니다.

USER

변환 시작시 리모트 시스템으로 송신되는 사용자명을 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***CURRENT:** 현재 작업이 실행 중인 사용자 프로파일이 사용됩니다.

user-name: 어플리케이션 서버 작업에 사용되는 사용자명을 지정하십시오.

PASSWORD

리모트 시스템에 사용될 암호를 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***NONE:** 암호가 송신되지 않습니다. 이 값이 지정되면 USER(*CURRENT) 또한 지정되어야 합니다.

password: USER 매개변수에 지정된 사용자명의 암호를 지정하십시오.

RDBCNNMTH

CONNECT문에 사용되는 의미를 지정합니다. 자세한 정보는 SQL 참조서에서 CONNECT(유형 1) 및 CONNECT(유형 2)를 참조하십시오.

***DUW:** CONNECT(유형 2) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 추가 관계형 데이터베이스에 대한 연속된 CONNECT문이 이전의 연결을 단절시키지는 않습니다.

***RUW:** 관계형 데이터베이스로의 연결은 단 하나만 허용됩니다. 연속 CONNECT 문은 새로운 연결이 설정되기 전에 이전의 연결을 단절시킵니다.

DFTRDBCOL

표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용되는 스키마명을 지정합니다. 이 매개변수는 정적 SQL문에만 적용됩니다.

***NONE:** OPTION 매개변수에 정의된 명명 규칙이 사용됩니다.

schema-name: 스키마 ID의 이름을 지정하십시오. 이 값은 OPTION 매개변수에 지정된 명명 규칙 대신 사용됩니다.

DYNDFTCOL

DFTRDBCOL 매개변수에 대해 지정된 디폴트 스키마명도 동적 명령문에 사용되는지를 지정합니다.

***NO:** 표, 뷰, 색인 및 동적 SQL문에 대한 SQL 패키지의 규정되지 않은 이름으로 DFTRDBCOL 매개변수에 지정된 값을 사용하지 마십시오. OPTION 매개변수에 지정된 명명 규칙이 사용됩니다.

***YES:** DFTRDBCOL 매개변수에 지정된 컬렉션명은 표, 뷰, 색인 및 동적 SQL문의 규정되지 않은 이름으로 사용됩니다.

SQLPKG

이 명령의 RDB 매개변수에 지정된 관계형 데이터베이스에 작성된 SQL 패키지의 규정된 이름을 지정합니다.

가능한 라이브러리 값은 다음과 같습니다.

***PGMLIB:** 패키지가 프로그램을 포함하는 라이브러리와 같은 이름의 라이브러리에 작성됩니다.

library-name: 패키지가 작성된 라이브러리의 이름을 지정하십시오.

***PGM:** 패키지의 이름이 프로그램명과 같습니다.

package-name: RDBNAME 매개변수에 지정된 리모트 데이터베이스에 작성된 패키지의 이름을 지정하십시오.

SQLPATH

정적 SQL문의 사용자 정의 유형, 프로시저 및 함수 등을 찾기 위해 사용될 경로를 지정합니다.

***NAMING:** 사용되는 경로는 OPTION 매개변수에 지정된 명명 규칙에 따릅니다.

***SYS** 명명의 경우 사용되는 경로는 *LIBL로, 실행시 현재 라이브러리 리스트입니다.

***SQL** 명명의 경우 사용되는 경로는 "QSYS", "QSYS2", "userid"입니다. 여기서 "userid"는 USER 특수 레지스터의 값입니다. DFTRDBCOL 매개변수에 schema-name을 지정한 경우 schema-name이 사용자 ID 대신 사용됩니다.

***LIBL:** 사용되는 경로는 실행시 라이브러리 리스트입니다.

schema-name: 하나 이상의 스키마명 리스트를 지정하십시오. 최대 268개의 개별 스키마를 지정할 수 있습니다.

SQLCURRULE

SQL문에 사용되는 의미를 지정합니다.

***DB2:** 모든 SQL문의 의미에 대한 디폴트는 DB2에 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 문자 자료로 처리됩니다.

***STD:** 모든 SQL문의 의미에 대한 디폴트는 ISO 및 ANSI SQL 표준에서 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 2진 자료로 처리됩니다.

SAAFLAG

IBM SQL 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 IBM SQL 구문을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다. IBM SQL 구문을 생성하는 IBM 데이터베이스에 대한 정보는 *DRDA IBM SQL Reference, SC26-3255-00* 책을 참조하십시오.

***NOFLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사하지 않습니다.

***FLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사합니다.

FLAGSTD

미국 표준 협회(ANSI) 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 다음과 같은 표준을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다.

ANSI X3.135-1992 항목
ISO 9075-1992 항목
FIPS 127.2 항목

미국 표준 협회(ANSI) 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 다음과 같은 표준을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다.

ANSI X3.135-1992 항목
ISO 9075-1992 항목
FIPS 127.2 항목

***NONE:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크하지 않습니다.

***ANS:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크합니다.

PRTFILE

리스트가 전달되는 프린터 파일의 규정화명을 지정합니다. 파일의 최소 레코드 길이는 132바이트여야 하며, 그렇지 않으면 정보가 유실됩니다.

프린터 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

CRTSQLPLI

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QSYSPRT: 파일명이 지정되지 않으면 사전컴파일러 인쇄 출력이 IBM 제공 프린터 파일 QSYSPRT로 전달됩니다.

printer-file-name: 사전컴파일러 인쇄 출력이 전달될 프린터 파일의 이름을 지정하십시오.

SRTSEQ

SQL문의 스트링 비교에 사용될 정렬 순서표를 지정합니다.

주: *HEX는 어플리케이션 서버가 iSeries 시스템에 없거나 릴리스 레벨이 V2R3M0 이전인 분산 어플리케이션의 매개변수에 대해서만 지정되어야 합니다

***JOB:** 작업에 대한 SRTSEQ 값이 사전컴파일 중에 검색됩니다.

***JOBRUN:** 작업에 대한 SRTSEQ 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 SRTSEQ(*JOBRUN)는 LANGID(*JOBRUN)가 지정될 때만 유효합니다.

***LANGIDUNQ:** 정렬 순서표에는 코드 페이지의 각 문자에 대한 고유한 가중치가 있어야 합니다.

***LANGIDSHR:** LANGID 매개변수에 지정된 공유 가중치 정렬표가 사용됩니다.

***HEX:** 정렬 순서표가 사용되지 않습니다. 문자의 16진수 값이 정렬 순서를 결정하기 위해 사용됩니다.

정렬 순서표는 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

table-name: 사용될 정렬 순서표의 이름을 지정하십시오.

LANGID

SRTSEQ(*LANGIDUNQ) 또는 SRTSEQ(*LANGIDSHR) 지정시 사용될 언어 ID를 지정합니다.

***JOB:** 작업에 대한 LANGID 값이 사전컴파일 중에 검색됩니다.

***JOBRUN:** 작업에 대한 LANGID 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 LANGID(*JOBRUN)는 SRTSEQ(*JOBRUN)가 지정될 때만 유효합니다.

language-id: 프로그램에 사용될 언어 ID를 지정하십시오.

USRPRF

정적 SQL문에서 각 오브젝트에 대해 프로그램 오브젝트가 가지는 권한을 포함하여 컴파일된 프로그램 오브젝트를 실행할 때 사용되는 사용자 프로파일을 지정합니다. 프로그램 소유자 또는 프로그램 사용자의 프로파일이 프로그램 오브젝트가 사용할 수 있는 오브젝트를 제어하는 데 사용됩니다.

***NAMING:** 사용자 프로파일은 명명 규칙에 의해 판별됩니다. 명명 규칙이 *SQL이면 USRPRF(*OWNER)가 사용됩니다. 명명 규칙이 *SYS이면 USRPRF(*USER)가 사용됩니다.

***USER:** 프로그램 오브젝트를 실행 중인 사용자의 프로파일이 사용됩니다.

***OWNER:** 프로그램 소유자와 프로그램 사용자 모두의 사용자 프로파일이 프로그램 실행시 사용됩니다.

DYNUSRPRF

SQL문에 사용되는 사용자 프로파일을 지정합니다.

***USER:** 프로그램은 프로그램 사용자의 사용자 프로파일하에서 수행됩니다.

***OWNER:** 로컬 동적 SQL문은 프로그램 소유자의 사용자 프로파일하에서 실행됩니다. 분산 동적 SQL문은 SQL 패키지 소유자의 사용자 프로파일하에서 실행됩니다.

TOSRCFILE

SQL 사전컴파일러에 의해 처리된 출력 소스 멤버가 있게 될 소스 파일의 규정화명을 지정합니다. 지정된 소스 파일을 찾을 수 없는 경우에는 이를 작성합니다. 출력 멤버는 SRCMBR 매개변수용으로 지정한 이름과 동일한 이름을 갖게 됩니다.

가능한 라이브러리 값은 다음과 같습니다.

QTEMP: 라이브러리 QTEMP가 사용될 것입니다.

***LIBL:** 작업 라이브러리 리스트가 지정 파일에 대해 탐색됩니다. 만일 라이브러리 리스트의 어떠한 라이브러리에서도 이 파일이 없는 경우에 파일은 현재 라이브러리에서 작성됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 사용될 것입니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없는 경우 QGPL 라이브러리가 사용될 것입니다.

library-name: 출력 소스 파일이 작성될 라이브러리명을 지정하십시오.

CRTSQLPLI

QSQLTEMP: 소스 파일 QSQLTEMP가 사용될 것입니다.

source-file-name: 출력 소스 멤버가 들어 있는 소스 파일의 이름을 지정하십시오.

TEXT

프로그램과 그 기능을 간단하게 설명하는 텍스트를 지정합니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 TEXT 매개변수에 나와 있습니다.

***SRCMBRTXT:** PL/I 프로그램을 작성하는 데 사용되는 소스 파일 멤버로부터 텍스트를 취합니다. 소스 입력 유틸리티 시작(STRSEU) 명령을 사용하거나 실제 파일 멤버 추가(ADDPFM) 또는 실제 파일 멤버 변경(CHGPFM) 명령을 사용하여 데이터베이스 소스 멤버에 대한 텍스트를 추가하거나 변경할 수 있습니다. 소스 파일이 인라인 파일이나 장치 파일일 경우 텍스트는 공백입니다.

***BLANK:** 텍스트가 지정되지 않습니다.

'description': 50자 미만의 텍스트를 어포스트로피로 묶어 지정하십시오.

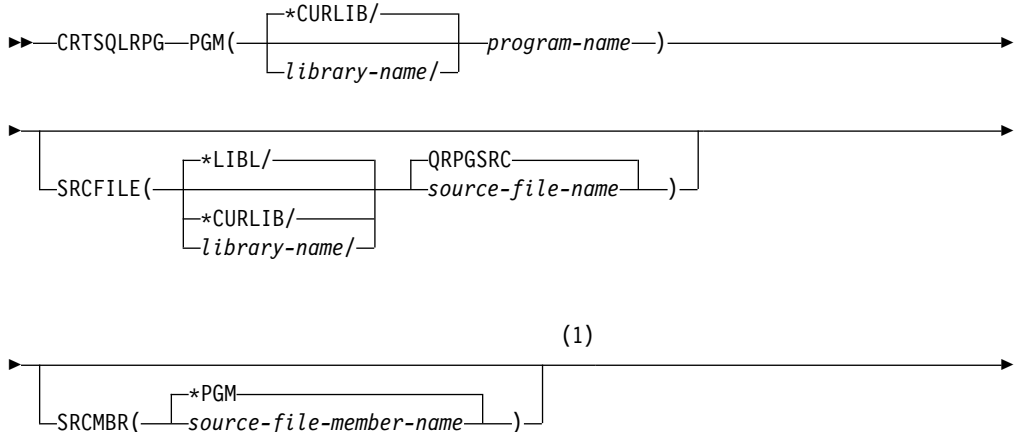
예:

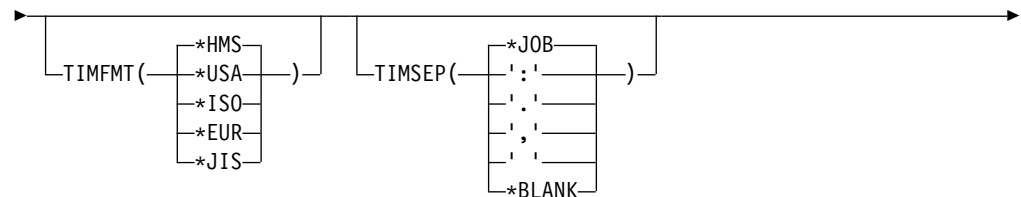
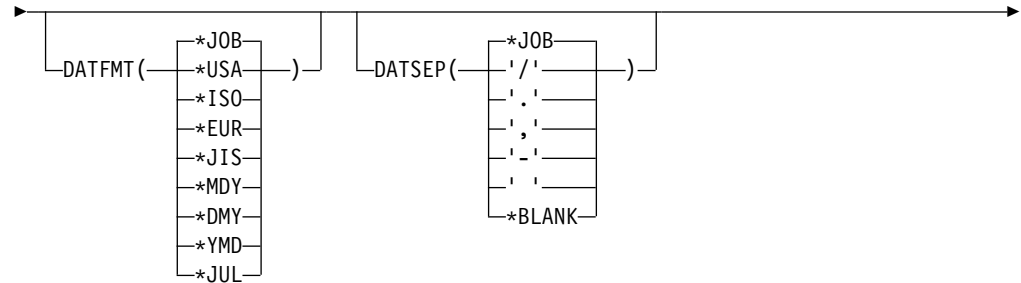
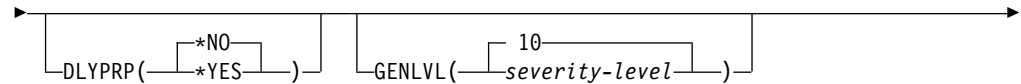
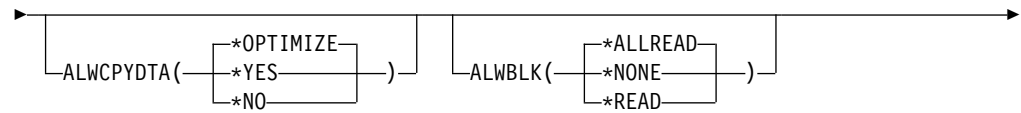
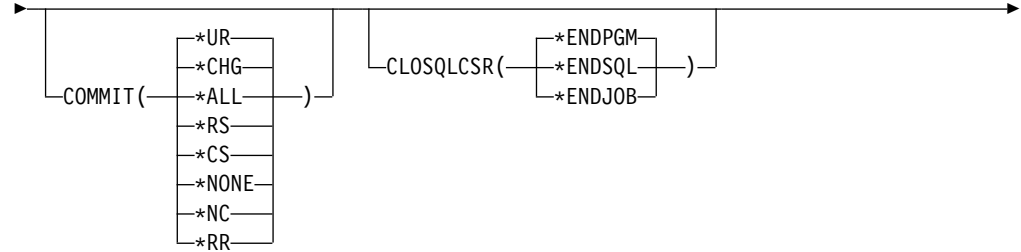
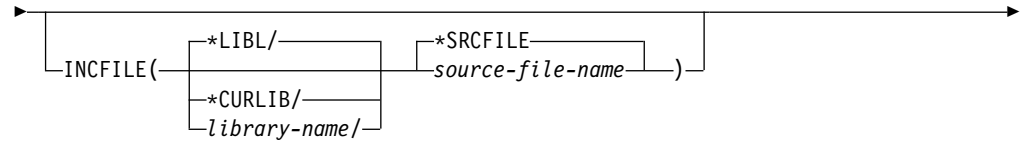
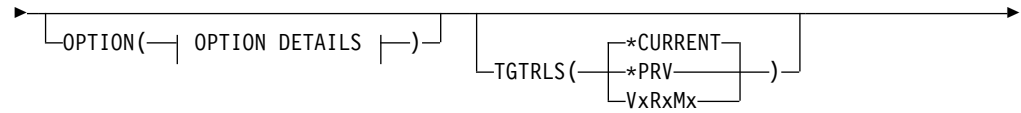
```
CRTSQLPLI  PAYROLL  TEXT('Payroll Program')
```

이 명령은 소스문을 컴파일하는 SQL 컴파일러를 수행하여 변경된 소스문을 라이브러리 QTEMP에 있는 파일 QSQLTEMP의 멤버 PAYROLL에 저장합니다. SQL 사전 컴파일러에 의해 작성된 소스 멤버를 사용하여 현재 라이브러리에 프로그램 PAYROLL을 작성하기 위해서 PL/I 컴파일러가 호출됩니다.

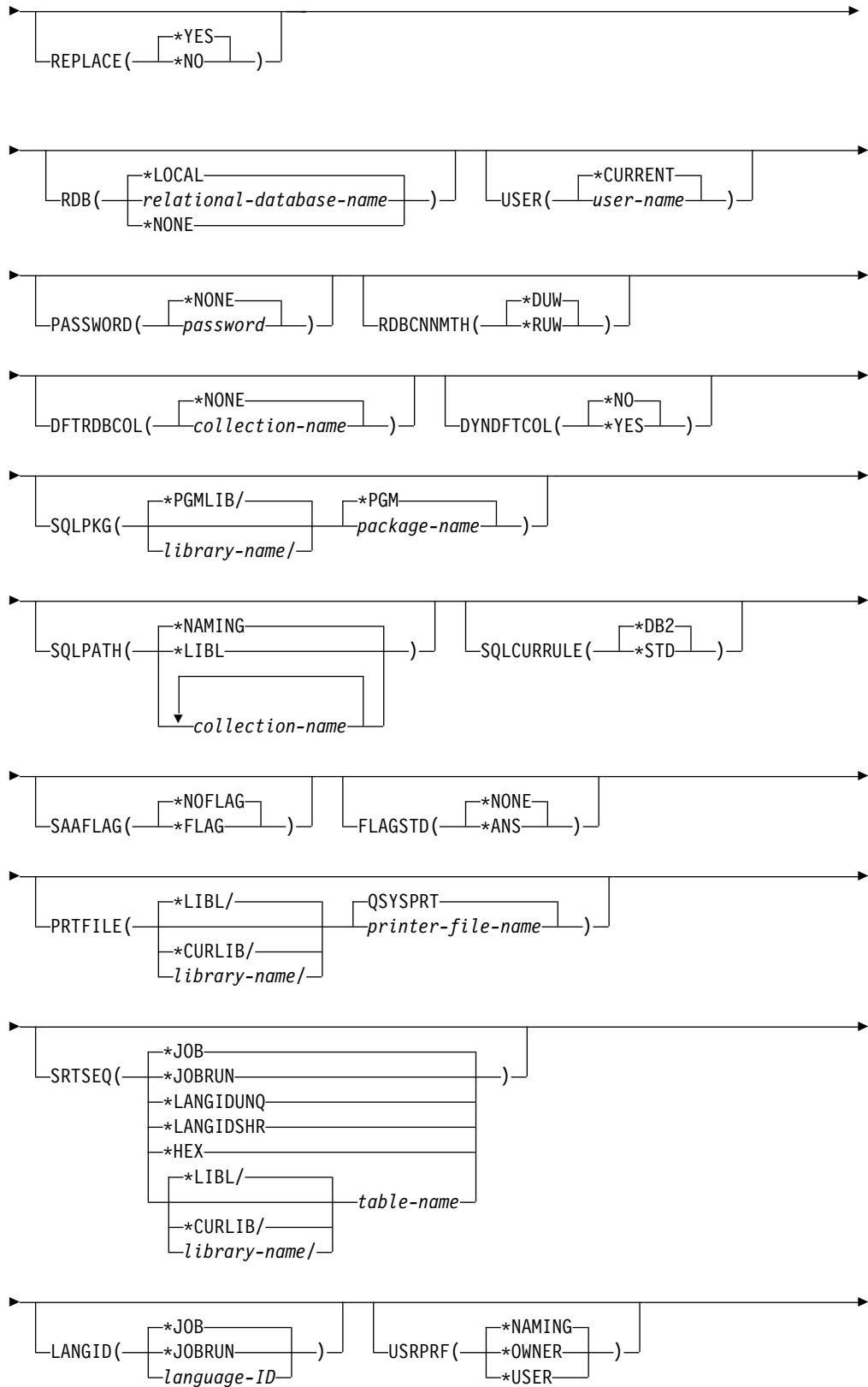
CRTSQLRPG(구조화 조회 언어 RPG 작성) 명령

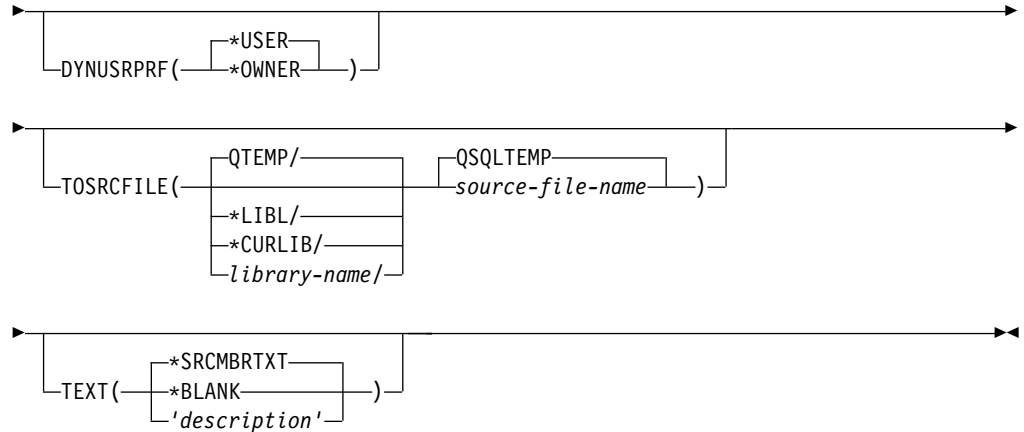
Job: B,I Pgm: B,I REXX: B,I Exec



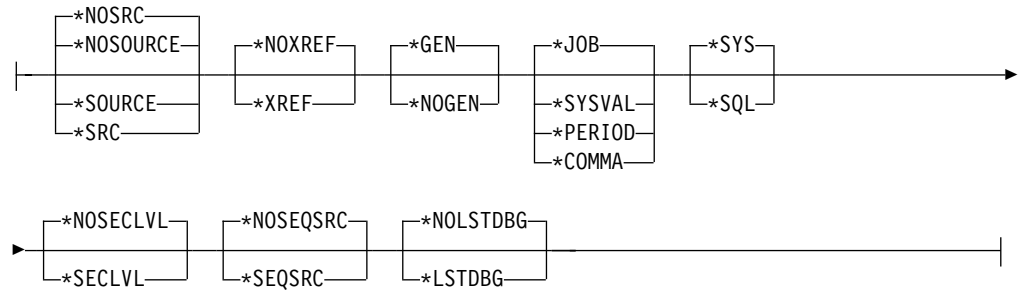


CRTSQLRPG





OPTION 세부사항:



주:

1 이 앞에 오는 모든 매개변수는 앞뒤 관계에 따라 지정할 수 있습니다.

목적:

CRTSQLRPG(구조화 조회 언어 RPG 작성) 명령은 SQL문이 들어 있는 RPG 소스 문을 사전컴파일하는 구조화 조회 언어(SQL) 사전컴파일러를 호출하여 임시 소스 멤버를 작성한 후 선택적으로 프로그램을 컴파일하기 위해 PL/I 컴파일러를 호출합니다.

매개변수:

PGM

컴파일된 프로그램의 규정화명을 지정합니다.

컴파일된 RPG 프로그램의 이름은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***CURLIB:** 컴파일된 RPG 프로그램이 해당 작업에 대한 현재 라이브러리에 작성됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: RPG 프로그램이 작성될 라이브러리의 이름을 지정하십시오.

program-name: 컴파일된 프로그램의 이름을 지정하십시오.

SRCFILE

SQL문과 함께 RPG 소스가 들어 있는 소스 파일의 규정된 이름을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QRPGSRC: 소스 파일명이 지정되지 않은 경우 IBM 제공 소스 파일 QRPGSRC에 RPG 소스가 포함됩니다.

source-file-name: RPG 소스가 들어 있는 소스 파일의 이름을 지정하십시오.

SRCMBR

RPG 소스가 들어 있는 소스 파일 멤버명을 지정합니다. 이 매개변수는 SRCFILE 매개변수의 소스 파일명이 데이터베이스 파일일 경우에만 지정됩니다. 이 매개변수가 지정되지 않으면 PGM 매개변수에 지정된 PGM명이 사용됩니다.

***PGM:** RPG 소스가 PGM 매개변수에 지정된 것과 같은 이름의 소스 파일 멤버에 있음을 지정합니다.

source-file-member-name: RPG 소스가 들어 있는 멤버의 이름을 지정하십시오.

OPTION

RPG 소스가 사전컴파일될 때 다음 옵션 중 하나 또는 여러 개가 사용되는지의 여부를 지정합니다. 옵션이 두 번 이상 지정되면 마지막에 지정된 옵션이 사용됩니다.

요소 1: 소스 리스팅 옵션

***NOSOURCE** 또는 ***NOSRC:** 사전컴파일러나 패키지 작성 중에 오류가 검출되지 않는 한, 소스 인쇄 출력이 사전컴파일러에 의해 작성되지 않습니다.

***SOURCE** 또는 ***SRC:** 사전컴파일러가 RPG 소스 입력으로 구성된 소스 인쇄 출력을 작성합니다.

요소 2: 상호 참조 옵션

***NOXREF:** 사전컴파일러가 상호 참조명을 지정하지 않습니다.

***XREF:** 사전컴파일러는 그러한 항목을 참조하는 프로그램 내의 명령문 번호 항목을 상호 참조합니다.

요소 3: 프로그램 작성 옵션

***GEN:** 컴파일러는 프로그램 컴파일 후 수행될 수 있는 프로그램을 작성합니다. SQL 패키지 오브젝트는 데이터베이스명이 RDB 매개변수에 지정된 경우에 작성됩니다.

***NOGEN:** 사전컴파일러가 RPG 컴파일러를 호출하지 않고, 프로그램과 SQL 패키지가 작성되지 않습니다.

요소 4: 소수점 옵션

***JOB:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 사전컴파일시 작업에 대해 지정된 소수점 표현입니다.

***SYSVAL:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 QDECFMT 시스템 값입니다.

주: QDECFMT에 소수점으로 사용되는 값이 쉼표로 지정되면 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)가 쉼표로 분리되고, 그 뒤에는 공백이 와야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 마침표가 소수점으로 지정된 VALUES(1.1,2.23,4.1)와 동일합니다.

***PERIOD:** SQL문에 사용된 숫자 상수에 대한 소수점으로 사용되는 값은 마침표입니다.

***COMMA:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 쉼표입니다.

주: 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)가 쉼표로 분리되고, 그 뒤에는 공백이 와야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 마침표가 소수점으로 지정된 VALUES(1.1,2.23,4.1)와 동일합니다.

요소 5: 명명 규칙 옵션

***SYS:** 시스템 명명 규칙(library-name/file-name)이 사용됩니다.

***SQL:** SQL 명명 규칙(schema-name.table-name)이 사용됩니다. iSeries 시스템 이외의 리모트 데이터베이스에서 프로그램을 작성할 때 *SQL은 명명 규칙을 따라 지정해야 합니다.

요소 6: 2차 레벨 메시지 텍스트 옵션

***NOSECLVL:** 2차 레벨 텍스트 설명이 인쇄 출력에 추가되지 않습니다.

***SECLVL:** 대체 자료를 가진 2차 레벨 텍스트가 리스팅의 모든 메시지에 대해 추가됩니다.

요소 7: 소스 순번 옵션

***NOSEQSRC:** QSQLTEMP 내에 새로운 소스 멤버를 작성할 때 입력 소스 파일에서 가져온 소스 순번이 사용됩니다.

***SEQSRC:** QSQLTEMP에서 새로운 소스 멤버로 작성된 소스 레코드는 000001 부터 시작 번호가 매겨집니다.

요소 8: 디버그 리스팅 뷰 옵션

***NOLSTDBG:** 오류 및 디버그 정보가 생성되지 않습니다.

***LSTDBG:** SQL 사전컴파일러는 리스트 뷰, 오류 및 이 뷰에 필요한 디버그 정보를 생성합니다. 프로그램을 컴파일하기 위해 CODE/400 제품을 사용중인 경우에만 *LSTDBG를 사용할 수 있습니다.

TGTRLS

작성될 오브젝트를 사용하려고 하는 오퍼레이팅 시스템의 릴리스 레벨을 지정합니다.

***CURRENT**와 ***PRV** 값이 주어질 때와 릴리스 레벨 값을 지정할 때에는 VxRxMx의 형식으로 릴리스를 지정하게 되며, 여기서 Vx는 버전, Rx는 릴리스, Mx는 수정 레벨을 나타냅니다. 예를 들어, V2R3M0은 버전 2, 릴리스 3, 수정 레벨 0을 나타냅니다.

***CURRENT:** 오브젝트는 사용자 시스템에서 현재 실행 중인 오퍼레이팅 시스템의 릴리스에 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *CURRENT는 V2R3M5가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

주: 시스템에서 V2R3M5가 실행되고 있으며, V2R3M0이 설치된 시스템에서 오브젝트를 사용하려는 경우에는 TGTRLS(*CURRENT)가 아니고 TGTRLS(V2R3M0)를 지정해야 합니다.

***PRV:** 오브젝트는 오퍼레이팅 시스템의 이전 릴리스, 수정 레벨 0에서 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *PRV는 V2R2M0이 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

release-level: VxRxMx 형식의 릴리스를 지정하십시오. 지정된 릴리스의 시스템이나 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스의 시스템에서도 오브젝트를 사용할 수 있습니다.

유효한 값들은 현재 버전, 릴리스, 수정 레벨에 기초하고 있으며, 새로운 릴리스 각각으로 변경될 수 있습니다. 이 명령이 지원하는 가장 이전의 릴리스 레벨보다 더 앞의 릴리스 레벨을 지정하면 지원되는 릴리스 레벨 가운데 가장 이전의 릴리스가 표시된 오류 메시지를 받게 됩니다.

INCFILE

SQL INCLUDE문을 사용하여 프로그램에 포함시킨 멤버에 들어 있는 소스 파일의 규정화명을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

***SRCFILE:** SRCFILE 매개변수에 지정된 규정된 소스 파일에는 모든 SQL INCLUDE문에 지정된 소스 파일 멤버가 포함됩니다.

source-file-name: 모든 SQL INCLUDE문에 지정된 소스 파일명을 포함하는 소스 파일의 이름을 지정하십시오. 여기에서 지정된 소스 파일의 레코드 길이는 SRCFILE 매개변수에 지정된 소스 파일의 레코드 길이보다 짧아야 합니다.

COMMIT

컴파일된 프로그램의 SQL문이 확약 제어하에서 실행되는지 여부를 지정합니다. 호스트 언어 소스문에 참조된 파일은 이 옵션의 영향을 받지 않습니다. SQL문에 참조된 SQL 표, SQL 뷰 및 SQL 패키지만이 영향을 받습니다.

주: RPG 소스문에 언급된 파일은 이 옵션의 영향을 받지 않습니다.

***CHG 또는 *UR:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON 및 REVOKE문에서 참조된 오브젝트들을 지정하며, 갱신, 삭제 및 삽입된 행들은 작업 단위(트랜잭션)가 종료될 때까지 잠겨 있습니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다.

***ALL 또는 *RS:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입된 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***CS:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 선택은 되었지만 갱신되지 않은 행은 다음 행이 선택될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***NONE 또는 *NC:** 확약 제어가 사용되지 않음을 지정합니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다. SQL DROP SCHEMA문이 프

로그래밍에 포함되는 경우에는 *NONE 또는 *NC가 반드시 사용되어야 합니다. RDB 매개변수에 관계형 데이터베이스가 지정되어 있고 관계형 데이터베이스가 iSeries가 아닌 시스템에 있으면, *NONE 또는 *NC를 지정할 수 없습니다.

***RR:** SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 예약되지 않은 변경 내용을 볼 수 없습니다. SELECT, UPDATE, DELETE 및 INSERT문에 참조된 모든 표는 작업 단위(트랜잭션)가 종료될 때까지 배타적으로 잠겨 있습니다.

CLOSQLCSR

SQL 커서가 내재적으로 닫히고 SQL 준비 명령문이 내재적으로 취소되고 LOCK TABLE 잠금이 해제되는 시기를 지정합니다. SQL 커서는 CLOSE, COMMIT 또는 ROLLBACK(HOLD 없이) SQL문을 발행할 때 명시적으로 닫힙니다.

***ENDPGM:** 프로그램 종료시 SQL 커서는 닫히고 SQL 준비 명령문은 취소됩니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 LOCK TABLE 잠금이 해제됩니다.

***ENDSQL:** SQL 커서가 호출 사이에 열린 상태로 남아 있고 다른 SQL OPEN을 실행하지 않고 폐치될 수 있습니다. 호출 스택상의 보다 높은 프로그램 중 하나가 최소한 하나의 SQL문을 실행해야 합니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 SQL 커서가 닫히고 준비된 SQL문이 취소되며, LOCK TABLE 잠금이 해제됩니다. *ENDSQL이 호출된 첫 번째 SQL 프로그램(호출 스택의 첫 번째 SQL 프로그램)인 프로그램에 대해 지정되는 경우 프로그램은 *ENDPGM이 지정된 것처럼 처리됩니다.

***ENDJOB:** SQL 커서가 호출 사이에 열린 상태로 남아 있고 다른 SQL OPEN을 실행하지 않고 폐치될 수 있습니다. 호출 스택상의 보다 높은 프로그램 중 하나가 최소한 하나의 SQL문을 실행해야 합니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 SQL 커서가 닫히고 준비된 SQL문이 취소되며, LOCK TABLE 잠금이 해제됩니다. *ENDSQL이 호출된 첫 번째 SQL 프로그램(호출 스택의 첫 번째 SQL 프로그램)인 프로그램에 대해 지정되는 경우 프로그램은 *ENDPGM이 지정된 것처럼 처리됩니다.

ALWCPYDTA

자료의 사본이 SELECT문에 사용될 수 있는지 여부를 지정합니다.

***OPTIMIZE:** 시스템은 데이터베이스에서 직접 검색된 자료를 사용할지 또는 자료의 사본을 사용할지의 여부를 판별합니다. 결정은 최상의 성능을 제공할 수 있는 방법에 근거하여 이루어집니다. COMMIT가 *CHG 또는 *CS이면서 ALWBLK가 *ALLREAD가 아니거나, COMMIT가 *ALL 또는 *RR이면 조회 수행에 필요한 경우에만 자료의 사본이 사용됩니다.

***YES:** 필요할 때만 자료의 사본이 사용됩니다.

***NO:** 자료의 사본은 허용되지 않습니다. 조회를 수행하는 데 자료의 임시 사본이 필요하면 오류 메시지가 표시됩니다.

ALWBLK

데이터베이스 관리자가 레코드 블록화를 사용할 수 있는지 여부와 블록화가 읽기 전용 커서에 사용될 수 있는 확장 범위를 지정합니다.

***ALLREAD:** *NONE 또는 *CHG가 COMMIT 매개변수에 지정된 경우 읽기 전용 커서에 대해 행이 블록화됩니다. 명시적으로 갱신될 수 없는 프로그램 내의 모든 커서들은 EXECUTE 또는 EXECUTE IMMEDIATE문이 프로그램 내에 있을 지라도 읽기 전용으로 열립니다.

***ALLREAD** 지정:

- *READ에 대해 허용된 블록화 이외의 확약 제어 레벨 *CHG하에서 레코드의 블록화를 허용합니다.
- 프로그램에서 거의 모든 읽기 전용 커서의 성능이 향상될 수 있으나 다음과 같은 방법으로 조회를 제한합니다.
 - 롤백(ROLLBACK) 명령, 호스트 언어의 ROLLBACK문 또는 ROLLBACK HOLD SQL문은 *ALLREAD가 지정된 경우 읽기 전용 커서의 위치를 재 지정하지 않습니다.
 - 커서에 대한 DECLARE문에 FOR UPDATE절이 없는 한, 지정된 UPDATE 또는 DELETE문의 동적 수행(예를 들면, EXECUTE IMMEDIATE 사용)은 커서의 행을 갱신하는 데 사용될 수 없습니다.

***NONE:** 커서에 대한 자료의 검색을 위해 행이 블록화되지 않습니다.

***NONE**을 지정하는 경우:

- 검색된 자료가 현재의 자료임을 보장합니다.
- 조회시 자료의 첫 번째 행을 검색하는 데 필요한 시간을 줄일 수 있습니다.
- 조회가 닫히기 전에 조회의 처음 몇 행만이 검색될 때 프로그램에 의해 사용되지 않은 자료 행 블록의 검색으로부터 데이터베이스 관리자를 중단시킵니다.
- 많은 행을 검색하는 조회의 전반적인 성능이 저하될 수 있습니다.

***READ:** 레코드는 다음의 경우 커서에 대한 자료의 읽기 전용 검색에 대해 블록화됩니다.

- *NONE이 COMMIT 매개변수에 지정되어 확약 제어가 사용되지 않음을 표시하는 경우
- FOR READ ONLY절로 커서가 선언되었거나 커서에 대해 지정된 UPDATE 또는 DELETE문을 수행할 수 있는 동적 명령문이 없는 경우

*READ를 지정하면 위의 조건을 만족하는 조회의 전반적인 성능이 향상되고 많은 수의 레코드를 검색할 수 있습니다.

DLYPRP

PREPARE문에 대한 동적 명령문 유효성 검사가 OPEN, EXECUTE 또는 DESCRIBE문 실행시까지 지연되는지 여부를 지정합니다. 유효성 검사가 지연되면 불필요한 확인이 제거되므로 성능이 향상됩니다.

***NO:** 동적 명령문 유효성 검사가 지연되지 않습니다. 동적 명령문이 준비될 때 액세스 계획의 유효성이 확인됩니다. 동적 명령문이 OPEN 또는 EXECUTE문에 사용될 때 액세스 계획의 유효성이 재확인됩니다. 동적 명령문에 의해 참조되는 오브젝트의 존재 또는 권한으로 인해, 동적 명령문이 아직 유효한지 확인하기 위해 OPEN 또는 EXECUTE문을 발행하기 전에 SQLCODE 또는 SQLSTATE를 검사해야 합니다.

***YES:** 동적 명령문 유효성 검사는 동적 명령문이 OPEN, EXECUTE 또는 DESCRIBE SQL문에 사용될 때까지 지연됩니다. 동적 명령문이 사용될 때 유효성 검사가 완료되고 액세스 계획이 수립됩니다. 이 매개변수에 *YES를 지정하면 OPEN, EXECUTE 또는 DESCRIBE문을 실행한 후에 SQLCODE와 SQLSTATE를 체크하여 동적 명령문이 유효한지 확인해야 합니다.

주: *YES를 지정하면 PREPARE에 INTO절이 사용되거나, 동적 명령문에 대한 OPEN문이 발행되기 전에 DESCRIBE문에서 그 동적 명령문을 사용하는 경우 성능은 향상되지 않습니다.

GENLVL

작성 조작이 실패하는 심각도 레벨을 지정합니다. 이 값 이상의 심각도 레벨을 가지는 오류가 발생하면 조작이 종료됩니다.

10: 디폴트 심각도 레벨은 10입니다.

심각도 레벨: 0-40 범위 내의 값을 지정하십시오.

DATFMT

결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 날짜 필드가 지정된 형식으로 리턴됩니다. 입력 날짜 스트링의 경우 지정된 값은 유효한 형식으로 날짜가 지정되었는지 여부를 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 iSeries 시스템이 아닌 시스템에 있으면, *USA, *ISO, *EUR 또는 *JIS가 지정되어야 합니다.

***JOB:** 작업에 지정된 형식이 사용됩니다. 작업 표시(DSPJOB) 명령을 사용하여 작업에 대한 현재 날짜 형식을 판별하십시오.

***USA:** 미국식 날짜 형식(mm/dd/yyyy)이 사용됩니다.

***ISO:** 국제 표준화 기구(ISO) 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***EUR:** 유럽식 날짜 형식(dd.mm.yyyy)이 사용됩니다.

***JIS:** 일본 산업 표준 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***MDY:** 날짜 형식(mm/dd/yy)이 사용됩니다.

***DMY:** 날짜 형식(dd/mm/yy)이 사용됩니다.

***YMD:** 날짜 형식(yy/mm/dd)이 사용됩니다.

***JUL:** 율리우스력 날짜 형식(yy/ddd)이 사용됩니다.

DATSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *JOB, *MDY, *DMY, *YMD 또는 *JUL이 DATFMT 매개변수에 지정되어 있을 때만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 날짜 분리자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘/’: 슬래시(/)가 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘-’: 붙임표(-)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

TIMFMT

결과 열에 액세스하기 위해 사용되는 형식을 지정합니다. 입력 시간 스트링의 경우 지정된 값은 시간이 유효한 형식으로 지정되었는지 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 또 다른 iSeries 시스템이 아닌 시스템에 있으면, 시간 형식이 콜론이나 마침표의 시간 분리문자가 있는 *USA, *ISO, *EUR, *JIS 또는 *HMS이어야 합니다.

***HMS:** (hh:mm:ss) 형식이 사용됩니다.

***USA:** 미국식 시간 형식(hh:mm xx)이 사용되는데, 여기서 xx는 AM(오전) 또는 PM(오후)입니다.

***ISO:** 국제 표준화 기구(ISO) 시간 형식(hh.mm.ss)이 사용됩니다.

***EUR:** 유럽식 시간 형식(hh.mm.ss)이 사용됩니다.

***JIS:** 일본 산업 표준 시간 형식(hh:mm:ss)이 사용됩니다.

TIMSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *HMS가 TIMFMT 매개변수에 지정되어 있을 때에만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 시간 분리 문자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘:’: 콜론(:)이 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

REPLACE

동일한 라이브러리에 동일한 이름을 가진 SQL 패키지가 존재할 때 새로운 SQL 패키지가 작성되는지의 여부를 지정합니다. 이 매개변수의 값은 C 명령으로 전달됩니다. 이 매개변수에 대한 정보는 CL Reference에서 부록A, "확장 매개변수 설명"을 참조하십시오.

***YES:** 새로운 패키지나 SQL 패키지가 작성되고, 지정된 라이브러리 내에서 같은 이름 및 유형을 가지는 기존의 프로그램 또는 SQL 패키지는 QRPLOBJ로 이동됩니다.

***NO:** 새로운 프로그램 또는 SQL 패키지는 같은 이름과 유형의 오브젝트가 이미 지정된 라이브러리에 존재하는 경우 작성되지 않습니다.

RDB

SQL 패키지 오브젝트가 작성되는 관계형 데이터베이스의 이름을 지정합니다.

***LOCAL:** 프로그램은 분산 SQL 프로그램으로서 작성됩니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다. SQL 패키지 오브젝트는 사전컴파일 프로세스의 일부분으로서 작성되지 않습니다. 구조화 조회 언어 패키지 작성(CRTSQLPKG) 명령이 사용될 수 있습니다.

relational-database-name: 새로운 SQL 패키지 오브젝트가 작성될 때 관계형 데이터베이스의 이름을 지정하십시오. 로컬 관계형 데이터베이스명이 지정되면 작성된 프로그램은 계속 분산 SQL 프로그램입니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다.

***NONE:** SQL 패키지 오브젝트가 작성되지 않습니다. 프로그램 오브젝트는 분산 프로그램이 아니며 CRTSQLPKG(구조화 조회 언어 패키지 작성) 명령은 사용될 수 없습니다.

USER

변환 시작시 리모트 시스템으로 송신되는 사용자명을 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***CURRENT:** 현재 작업이 실행 중인 사용자 프로파일이 사용됩니다.

user-name: 어플리케이션 리퀘스터 작업에 사용되는 사용자명을 지정하십시오.

PASSWORD

리모트 시스템에 사용될 암호를 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***NONE:** 암호가 송신되지 않습니다. 이 값이 지정되면 USER(*CURRENT) 또한 지정되어야 합니다.

password: USER 매개변수에 지정된 사용자명의 암호를 지정하십시오.

RDBCNNMTH

CONNECT문에 사용되는 의미를 지정합니다. 자세한 정보는 SQL 참조서를 참조하십시오.

***DUW:** CONNECT(유형 2) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 추가 관계형 데이터베이스에 대한 연속된 CONNECT문이 이전의 연결을 단절시키지는 않습니다.

***RUW:** CONNECT(유형 1) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 연속된 CONNECT문은 새로운 연결이 설정되기 전에 이전의 연결을 단절시킵니다.

DFTRDBCOL

표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용되는 스키마명을 지정합니다. 이 매개변수는 정적 SQL문에만 적용됩니다.

***NONE:** OPTION 매개변수에 정의된 명명 규칙이 사용됩니다.

schema-name: 스키마 ID의 이름을 지정하십시오. 이 값은 OPTION 매개변수에 지정된 명명 규칙 대신 사용됩니다.

DYNDFTCOL

DFTRDBCOL 매개변수에 대해 지정된 디폴트 스키마명도 동적 명령문에 사용되는지를 지정합니다.

***NO:** 표, 뷰, 색인 및 동적 SQL문에 대한 SQL 패키지의 규정되지 않은 이름으로 DFTRDBCOL 매개변수에 지정된 값을 사용하지 마십시오. OPTION 매개변수에 지정된 명명 규칙이 사용됩니다.

***YES:** DFTRDBCOL 매개변수에 지정된 컬렉션명은 표, 뷰, 색인 및 동적 SQL문의 규정되지 않은 이름으로 사용됩니다.

SQLPKG

이 명령의 RDB 매개변수에 지정된 관계형 데이터베이스에 작성된 SQL 패키지의 규정된 이름을 지정합니다.

가능한 라이브러리 값은 다음과 같습니다.

***PGMLIB:** 패키지가 프로그램을 포함하는 라이브러리와 같은 이름의 라이브러리에 작성됩니다.

library-name: 패키지가 작성된 라이브러리의 이름을 지정하십시오.

***PGM:** 패키지의 이름이 프로그램명과 같습니다.

package-name: RDBNAME 매개변수에 지정된 리모트 데이터베이스에 작성된 패키지의 이름을 지정하십시오.

SQLPATH

정적 SQL문의 사용자 정의 유형, 프로시저 및 함수 등을 찾기 위해 사용될 경로를 지정합니다.

***NAMING:** 사용되는 경로는 OPTION 매개변수에 지정된 명명 규칙에 따릅니다.

***SYS** 명명의 경우 사용되는 경로는 *LIBL로, 실행시 현재 라이브러리 리스트입니다.

***SQL** 명명의 경우 사용되는 경로는 "QSYS", "QSYS2", "userid"입니다. 여기서 "userid"는 USER 특수 레지스터의 값입니다. DFTRDBCOL 매개변수에 schema-name을 지정한 경우 schema-name이 사용자 ID 대신 사용됩니다.

***LIBL:** 사용되는 경로는 실행시 라이브러리 리스트입니다.

schema-name: 하나 이상의 스키마명 리스트를 지정하십시오. 최대 268개의 개별 스키마를 지정할 수 있습니다.

SQLCURRULE

SQL문에 사용되는 의미를 지정합니다.

***DB2:** 모든 SQL문의 의미에 대한 디폴트는 DB2에 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 문자 자료로 처리됩니다.

***STD:** 모든 SQL문의 의미에 대한 디폴트는 ISO 및 ANSI SQL 표준에서 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 2진 자료로 처리됩니다.

SAAFLAG

IBM SQL 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 IBM SQL 구문을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다. IBM SQL 구문을 생성하는 IBM 데이터베이스에 대한 정보는 *DRDA IBM SQL Reference, SC26-3255-00* 책을 참조하십시오.

***NOFLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사하지 않습니다.

***FLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사합니다.

FLAGSTD

미국 표준 협회(ANSI) 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 다음과 같은 표준을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다.

ANSI X3.135-1992 항목

ISO 9075-1992 항목

FIPS 127.2 항목

***NONE:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크하지 않습니다.

***ANS:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크합니다.

PRTFILE

리스트가 전달되는 프린터 파일의 규정화명을 지정합니다. 파일의 최소 레코드 길이는 132바이트여야 하며, 그렇지 않으면 정보가 유실됩니다.

프린터 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

CRTSQLRPG

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QSYSPRT: 파일명이 지정되지 않으면 사전컴파일러 인쇄 출력이 IBM 제공 프린터 파일 QSYSPRT로 전달됩니다.

printer-file-name: 컴파일러 인쇄 출력이 전달될 프린터 파일의 이름을 지정하십시오.

SRTSEQ

SQL문의 스트링 비교에 사용될 정렬 순서표를 지정합니다.

주: *HEX는 어플리케이션 서버가 iSeries 시스템에 없거나 릴리스 레벨이 V2R3M0 이전인 분산 어플리케이션의 매개변수에 대해서만 지정되어야 합니다

***JOB**: 작업에 대한 SRTSEQ 값이 사전컴파일 중에 검색됩니다.

***JOB RUN**: 작업에 대한 SRTSEQ 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 SRTSEQ(*JOB RUN)는 LANGID(*JOB RUN)가 지정될 때만 유효합니다.

***LANGID UNQ**: LANGID 매개변수에 지정된 고유 가중치 정렬표가 사용됩니다.

***LANGID SHR**: LANGID 매개변수에 지정된 공유 가중치 정렬표가 사용됩니다.

***HEX**: 정렬 순서표가 사용되지 않습니다. 문자의 16진수 값이 정렬 순서를 결정하기 위해 사용됩니다.

정렬 순서표의 이름은 다음 값 중 하나로 규정될 수 있습니다.

***LIBL**: 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB**: 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

table-name: 사용될 정렬 순서표의 이름을 지정하십시오.

LANGID

SRTSEQ(*LANGID UNQ) 또는 SRTSEQ(*LANGID SHR) 지정시 사용될 언어 ID를 지정합니다.

***JOB**: 작업에 대한 LANGID 값이 사전컴파일 중에 검색됩니다.

***JOB RUN**: 작업에 대한 LANGID 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 LANGID(*JOB RUN)는 SRTSEQ(*JOB RUN)가 지정될 때만 유효합니다.

language-id: 프로그램에 사용될 언어 ID를 지정하십시오.

USRPRF

정적 SQL문에서 각 오브젝트에 대해 프로그램 오브젝트가 가지는 권한을 포함하여 컴파일된 프로그램 오브젝트를 실행할 때 사용되는 사용자 프로파일을 지정합니다. 프로그램 소유자 또는 프로그램 사용자의 프로파일이 프로그램 오브젝트가 사용할 수 있는 오브젝트를 제어하는 데 사용됩니다.

***NAMING:** 사용자 프로파일은 명명 규칙에 의해 판별됩니다. 명명 규칙이 *SQL이면 USRPRF(*OWNER)가 사용됩니다. 명명 규칙이 *SYS이면 USRPRF(*USER)가 사용됩니다.

***USER:** 프로그램 오브젝트를 실행 중인 사용자의 프로파일이 사용됩니다.

***OWNER:** 프로그램 소유자와 프로그램 사용자 모두의 사용자 프로파일이 프로그램 실행시 사용됩니다.

DYNUSRPRF

SQL문에 사용되는 사용자 프로파일을 지정합니다.

***USER:** 로컬 동적 SQL문은 프로그램 사용자의 파일하에서 실행됩니다. 분산 동적 SQL문은 어플리케이션 서버 작업의 사용자 프로파일하에서 실행됩니다.

***OWNER:** 로컬 동적 SQL문은 프로그램 소유자의 사용자 프로파일하에서 실행됩니다. 분산 동적 SQL문은 SQL 패키지 소유자의 사용자 프로파일하에서 실행됩니다.

TOSRCFILE

SQL 사전컴파일러에 의해 처리된 출력 소스 멤버가 있게 될 소스 파일의 규정화명을 지정합니다. 지정된 소스 파일을 찾을 수 없는 경우에는 이를 작성합니다. 출력 멤버는 SRCMBR 매개변수용으로 지정한 이름과 동일한 이름을 갖게 됩니다.

가능한 라이브러리 값은 다음과 같습니다.

QTEMP: 라이브러리 QTEMP가 사용될 것입니다.

***LIBL:** 작업 라이브러리 리스트가 지정 파일에 대해 탐색됩니다. 만일 라이브러리 리스트의 어떠한 라이브러리에서도 이 파일이 없는 경우에 파일은 현재 라이브러리에서 작성됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 사용될 것입니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없는 경우 QGPL 라이브러리가 사용될 것입니다.

library-name: 출력 소스 파일이 작성될 라이브러리명을 지정하십시오.

QSQLTEMP: 소스 파일 QSQLTEMP가 사용될 것입니다.

source-file-name: 출력 소스 멤버가 들어 있는 소스 파일의 이름을 지정하십시오.

CRTSQLRPG

TEXT

프로그램과 그 기능을 간단하게 설명하는 텍스트를 지정합니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 TEXT 매개변수에 나와 있습니다.

***SRCMBRTXT:** RPG 프로그램을 작성하는 데 사용되는 소스 파일 멤버로부터 텍스트를 취합니다. 데이터베이스 소스 멤버에 대한 텍스트는 소스 입력 유틸리티 시작(STRSEU) 명령을 사용하거나 실제 파일 멤버 추가(ADDPFM) 명령 또는 실제 파일 멤버 변경(CHGPFM) 명령을 사용하여 추가되거나 변경될 수 있습니다. 소스 파일이 인라인 파일이나 장치 파일일 경우 텍스트는 공백입니다.

***BLANK:** 텍스트가 지정되지 않습니다.

'description': 50자 미만의 텍스트를 어포스트로피로 묶어 지정하십시오.

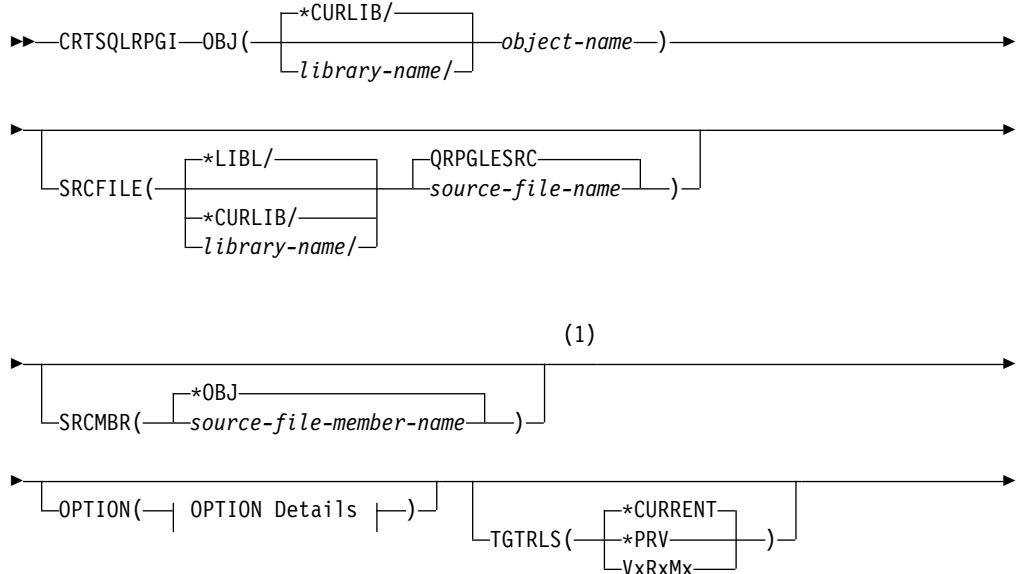
예:

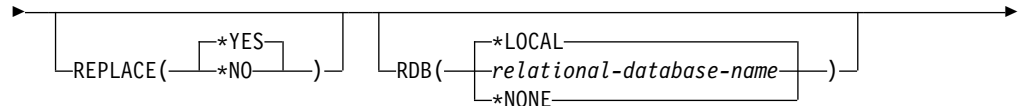
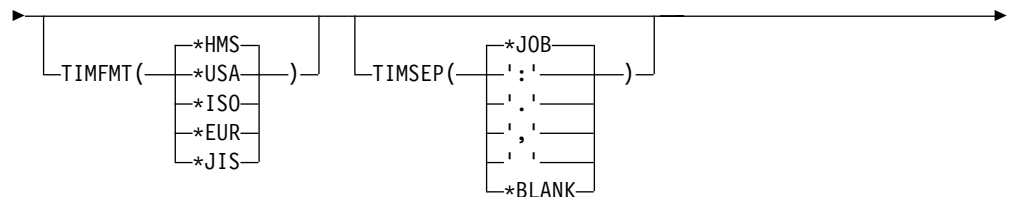
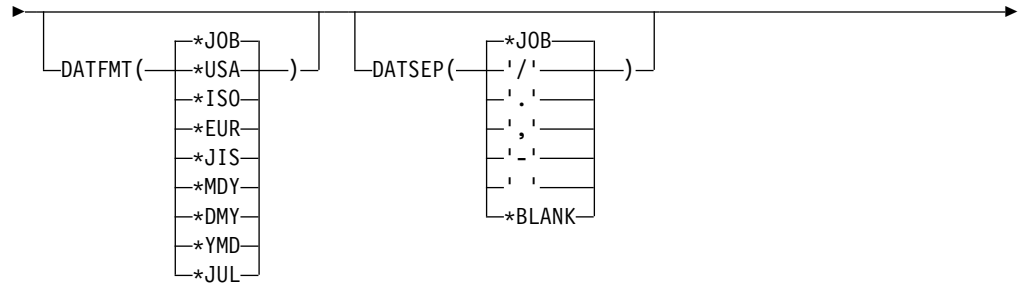
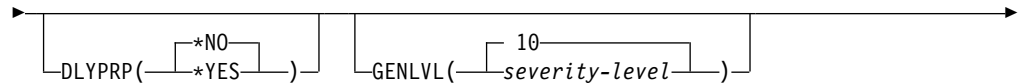
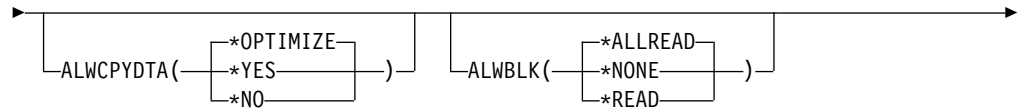
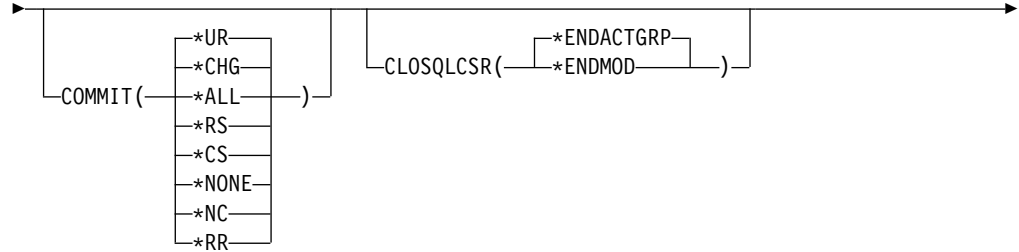
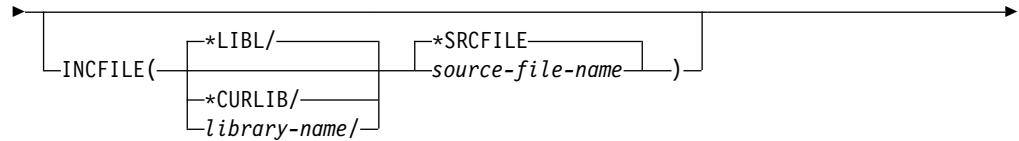
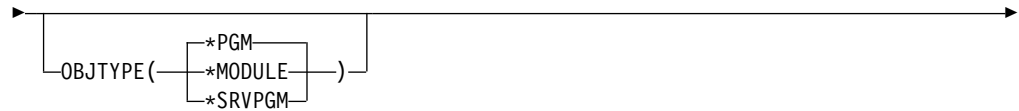
```
CRTSQLRPG PGM(JONES/ARBR5)
TEXT('Accounts Receivable Branch 5')
```

이 명령은 소스문을 컴파일하는 SQL 컴파일러를 수행하여 변경된 소스문을 라이브러리 QTEMP에 있는 파일 QSQLTEMP의 멤버 ARBR5에 저장합니다. SQL 사전컴파일러에 의해 작성된 소스 멤버를 사용하여 현재 라이브러리에 프로그램 ARBR5를 작성하기 위해서 RPG 컴파일러가 호출됩니다.

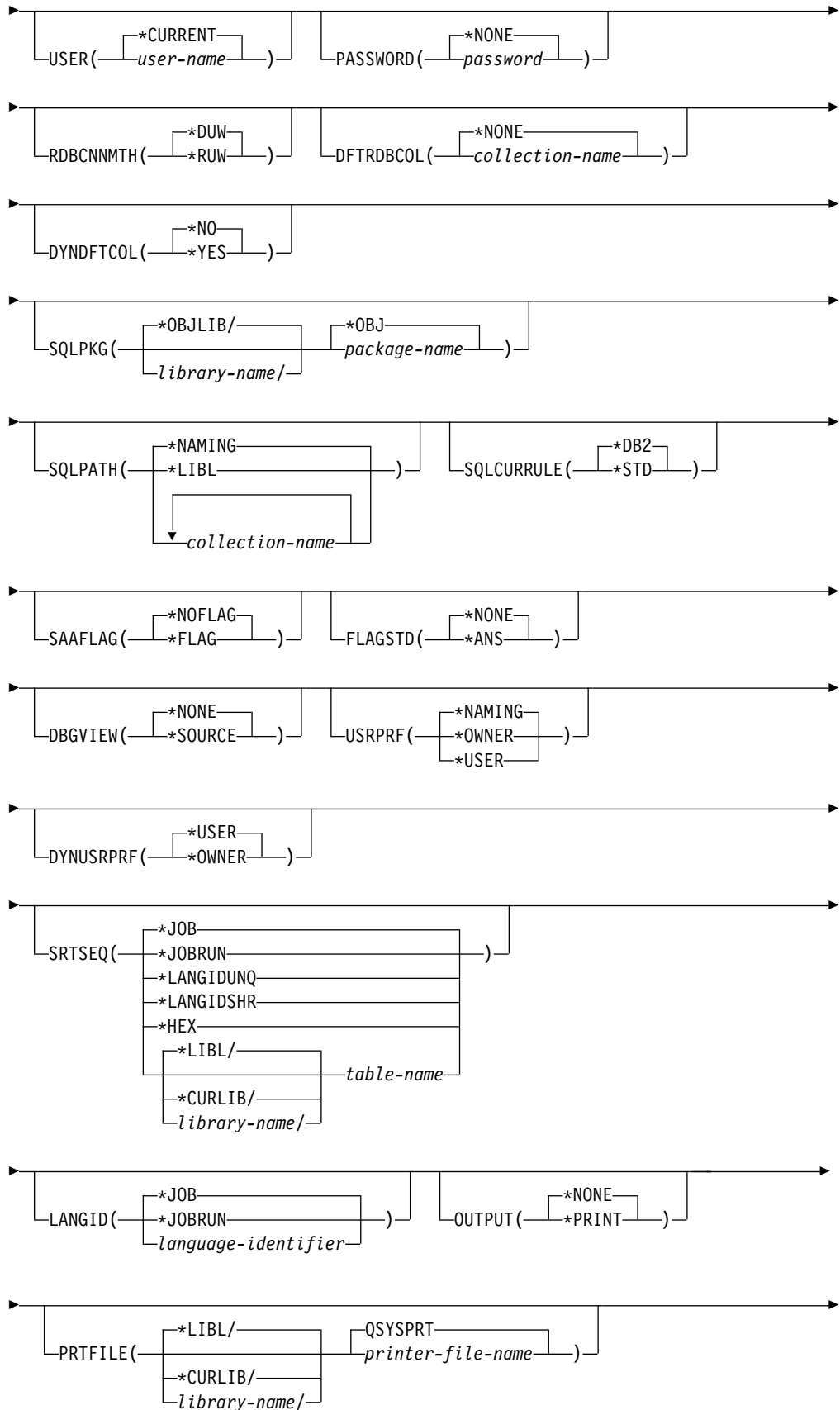
CRTSQLRPGI(SQL ILE RPG 오브젝트 작성) 명령

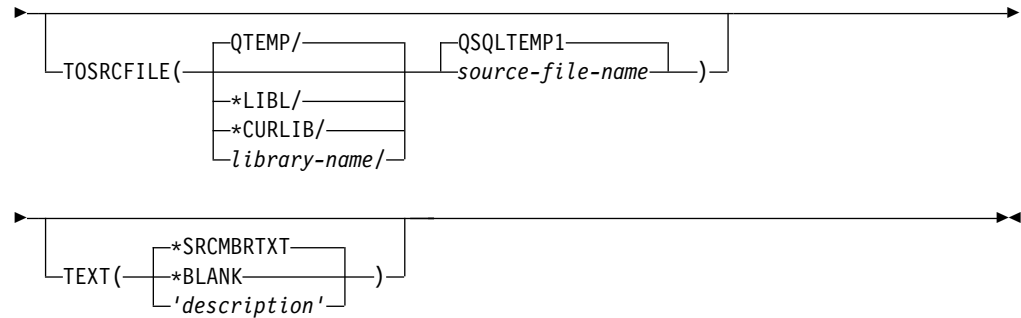
Job: B,I Pgm: B,I REXX: B,I Exec



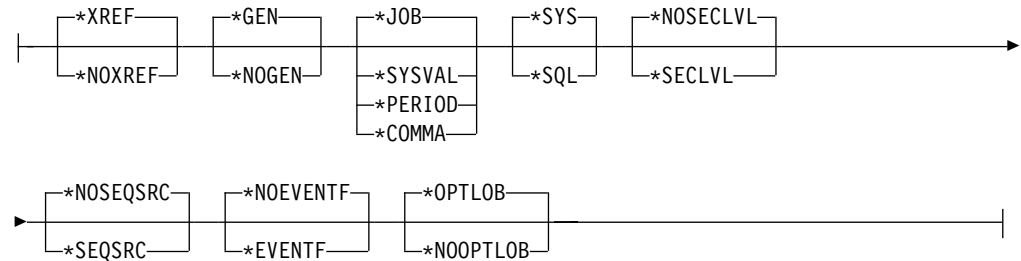


CRTSQLRPGI





OPTION 세부사항:



주:

1 이 앞에 오는 모든 매개변수는 앞뒤 관계에 따라 지정할 수 있습니다.

목적:

CRTSQLRPGI(구조화 조회 언어 ILE RPG 오브젝트) 명령은 SQL문이 들어 있는 RPG 소스를 사전컴파일하는 구조화 조회 언어(SQL) 사전컴파일러를 호출하고, 임시 소스 멤버를 작성하며 모듈, 프로그램 또는 서비스 프로그램을 작성하기 위해 ILE RPG 컴파일러를 선택적으로 호출합니다.

매개변수:

OBJ

작성된 오브젝트의 규정화명을 지정합니다.

***CURLIB:** 새로운 오브젝트가 해당 작업에 대한 현재 라이브러리에 작성됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 오브젝트가 작성된 라이브러리의 이름을 지정하십시오.

object-name: 작성 중인 오브젝트의 이름을 지정하십시오.

SRCFILE

SQL문과 함께 RPG 소스가 들어 있는 소스 파일의 규정된 이름을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QRPGLESRC: 소스 파일명이 지정되지 않은 경우 IBM 제공 소스 파일 QRPGLESRC에 RPG 소스가 포함됩니다.

source-file-name: RPG 소스가 들어 있는 소스 파일의 이름을 지정하십시오.

SRCMBR

RPG 소스가 들어 있는 소스 파일 멤버명을 지정합니다. 이 매개변수는 SRCFILE 매개변수의 소스 파일명이 데이터베이스 파일일 경우에만 지정됩니다. 이 매개변수가 지정되지 않으면 OBJ 매개변수에 지정된 PGM명이 사용됩니다.

***OBJ:** RPG 소스가 OBJ 매개변수에 지정된 것과 같은 이름의 소스 파일 멤버에 있음을 지정합니다.

source-file-member-name: RPG 소스가 들어 있는 멤버의 이름을 지정하십시오.

OPTION

RPG 소스가 사전컴파일될 때 다음 옵션 중 하나 또는 여러 개가 사용되는지의 여부를 지정합니다. 옵션이 두 번 이상 지정되면 마지막에 지정된 옵션이 사용됩니다.

요소 1: 상호 참조 옵션

***XREF:** 사전컴파일러는 그러한 항목을 참조하는 프로그램 내의 명령문 번호 항목을 상호 참조합니다.

***NOXREF:** 사전컴파일러는 이름을 상호 참조하지 않습니다.

요소 2: 프로그램 작성 옵션

***GEN:** 사전컴파일러가 OBJTYPE 매개변수에 의해 지정된 오브젝트를 작성합니다.

***NOGEN:** 사전컴파일러가 RPG 컴파일러를 호출하지 않으며, 모듈, 프로그램, 서비스 프로그램 또는 SQL 패키지가 작성되지 않습니다.

요소 3: 소수점 옵션

***JOB:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 사전컴파일 작업에 대해 지정된 소수점 표현입니다.

***SYSVAL:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 QDECFMT 시스템 값입니다.

주: QDECFMT가 소수점으로 사용될 값을 쉼표로 지정하면 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)가 쉼표로 분리되고 그 뒤에 공백이 와야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 소수점이 마침표인 VALUES(1.1,2.23,4.1)와 동일합니다.

***PERIOD:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 마침표(.)입니다.

***COMMA:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 쉼표(,)입니다.

주: 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)는 쉼표로 분리되고 그 뒤에 공백이 와야 합니다. 예를 들어, VALUES(1,1, 2,23, 4,1)는 마침표가 소수점으로 지정된 VALUES(1.1,2.23,4.1)와 동일합니다.

요소 4: 명명 규칙 옵션

***SYS:** 시스템 명명 규칙(library-name/file-name)이 사용됩니다.

***SQL:** SQL 명명 규칙(schema-name.table-name)이 사용됩니다. iSeries 시스템 이외의 리모트 데이터베이스에서 프로그램을 작성할 때 *SQL은 명명 규칙을 따라 지정해야 합니다.

요소 5: 1차 레벨 메세지 텍스트 옵션

***NOSECLVL:** 2차 레벨 텍스트 설명이 인쇄 출력에 추가되지 않습니다.

***SECLVL:** 대체 자료를 가진 2차 레벨 텍스트가 리스팅의 모든 메세지에 대해 추가됩니다.

요소 6: 소스 순번 옵션

***NOSEQSRC:** QSQLTEMP1에 작성된 소스 파일 멤버는 사전컴파일러가 읽은 소스 원본과 동일한 순서 번호를 가집니다.

***SEQSRC:** QSQLTEMP1에서 작성된 소스 파일 멤버는 000001부터 시작하는 순서 번호를 가집니다(000001씩 증가함).

요소 7: 이벤트 파일 작성

***NOEVENTF:** 컴파일러가 CoOperative Development Environment/400(CODE/400)에서 사용할 이벤트 파일을 작성하지 않습니다.

***EVENTF:** 컴파일러가 CoOperative Development Environment/400(CODE/400)에서 사용할 이벤트 파일을 작성합니다. 이벤트 파일은 사용자 소스 라이브러리에 있는 파일 EVFEVENT의 멤버로 생성됩니다. CODE/400은 이 파일을 이용하여

CODE/400 에디터와 함께 통합된 오류 피드백을 제공합니다. 이 옵션은 일반적으로 사용자를 대신해서 CODE/400에 의해 지정됩니다.

요소 8: 날짜 변환

***NOCVTDT:** 외부적으로 서술된 파일로부터 검색되는 자료, 시간 및 시간소인 자료 유형이 고유의 RPG 언어를 사용하여 처리됩니다

***CVTDT:** 외부적으로 서술된 파일로부터 검색되는 자료, 시간 및 시간소인 자료 유형이 고정 길이 문자로서 처리됩니다

요소 9: DRDA에 대한 대형 오브젝트 최적화

***OPTLOB:** 커서의 첫 번째 FETCH는 모든 후속 FETCH에서 커서가 LOB(대형 오브젝트)에 대해 사용될 방법을 결정합니다. 이 옵션은 커서가 닫힐 때까지 적용됩니다.

첫 번째 FETCH가 LOB 위치 지정자를 사용하여 LOB 열에 액세스할 경우 그 커서에 대한 어떤 후속 FETCH도 그 LOB 열을 LOB 호스트 변수에 폐치할 수 없습니다.

첫 번째 FETCH가 LOB 열을 LOB 호스트 변수에 배치할 경우 그 커서에 대한 어떤 후속 FETCH도 그 열에 대해 LOB 위치 지정자를 사용할 수 없습니다.

***NOOPTLOB:** 열이 LOB 위치 지정자나 LOB 호스트 변수에 검색되는지에 대해서는 제한이 없습니다. 이 옵션을 사용하면 성능이 떨어질 수도 있습니다.

TGTRLS

작성될 오브젝트를 사용하려고 하는 오퍼레이팅 시스템의 릴리스 레벨을 지정합니다.

***CURRENT**와 ***PRV** 값이 주어질 때와 릴리스 레벨 값을 지정할 때에는 VxRxMx의 형식으로 릴리스를 지정하게 되며, 여기서 Vx는 버전, Rx는 릴리스, Mx는 수정 레벨을 나타냅니다. 예를 들어, V2R3M0은 버전 2, 릴리스 3, 수정 레벨 0을 나타냅니다.

***CURRENT:** 오브젝트는 사용자 시스템에서 현재 실행 중인 오퍼레이팅 시스템의 릴리스에 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *CURRENT는 V2R3M5가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

주: 시스템에서 V2R3M5가 실행되고 있으며, V2R3M0가 설치된 시스템에서 오브젝트를 사용하려는 경우에는 TGTRLS(*CURRENT)가 아니고 TGTRLS(V2R3M0)를 지정해야 합니다.

***PRV:** 오브젝트는 오퍼레이팅 시스템의 이전 릴리스, 수정 레벨 0에서 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *PRV는 V2R2M0가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

release-level: VxRxMx 형식의 릴리스를 지정하십시오. 지정된 릴리스의 시스템이 나 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스의 시스템에서도 오브젝트를 사용할 수 있습니다.

유효한 값들은 현재 버전, 릴리스, 수정 레벨에 기초하고 있으며, 새로운 릴리스로 변경될 수 있습니다. 이 명령이 지원하는 가장 이전의 릴리스 레벨보다 더 앞의 릴리스 레벨을 지정하면 지원되는 릴리스 레벨 가운데 가장 이전의 릴리스가 표시된 오류 메시지를 받게 됩니다.

OBJTYPE

작성될 오브젝트 유형을 지정합니다.

***PGM:** SQL 사전컴파일러가 CRTBNDRPG 명령을 사용하여 바인드된 프로그램을 작성합니다.

***MODULE:** SQL 사전컴파일러가 CRTRPGMOD 명령을 발행하여 모듈을 작성합니다.

***SRVPGM:** SQL 사전컴파일러가 CRTRPGMOD와 CRTSRVPGM 명령을 사용하여 서비스 프로그램을 작성합니다.

주:

1. OBJTYPE(*PGM), OBJTYPE(*SRVPGM) 또는 RDB 매개변수 지정시 CRTSQLPKG 명령은 프로그램이 작성된 후 SQL 사전컴파일러에 의해 발행됩니다. OBJTYPE(*MODULE) 지정시, SQL 패키지는 작성되지 않으며 CRTPGM 또는 CRTSRVPGM 명령으로 프로그램을 작성한 후 CRTSQLPKG 명령을 발행해야 합니다.
2. *NOGEN이 지정되는 경우 SQL 임시 소스 멤버만이 생성되며 모듈, 프로그램, 서비스 프로그램 또는 SQL 패키지는 작성되지 않습니다.

INCFILE

SQL INCLUDE문을 사용하여 프로그램에 포함시킨 멤버에 들어 있는 소스 파일의 규정화명을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

***SRCFILE**: SRCFILE 매개변수에 지정된 규정된 소스 파일에는 모든 SQL INCLUDE문에 지정된 소스 파일 멤버가 포함됩니다.

source-file-name: 모든 SQL INCLUDE문에 지정된 소스 파일명을 포함하는 소스 파일의 이름을 지정하십시오. 여기에 지정된 소스 파일의 레코드 길이는 SRCFILE 매개변수에 지정된 소스 파일의 길이보다 짧아야 합니다.

COMMIT

컴파일된 장치에서 SQL문이 확약 제어하에 실행되는지를 지정합니다. 호스트 언어 소스문에 참조된 파일은 이 옵션의 영향을 받지 않습니다. SQL문에 참조된 SQL 표, SQL 뷰 및 SQL 패키지만이 영향을 받습니다.

***CHG** 또는 ***UR**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON 및 REVOKE문에서 참조된 오브젝트들을 지정하며, 갱신, 삭제 및 삽입된 행들은 작업 단위(트랜잭션)가 종료될 때까지 잠겨 있습니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다.

***ALL** 또는 ***RS**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입된 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***CS**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 선택은 되었지만 갱신되지 않은 행은 다음 행이 선택될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***NONE** 또는 ***NC**: 확약 제어가 사용되지 않음을 지정합니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다. SQL DROP SCHEMA문이 프로그램에 포함되는 경우에는 *NONE 또는 *NC가 반드시 사용되어야 합니다. RDB 매개변수에 관계형 데이터베이스가 지정되어 있고 관계형 데이터베이스가 iSeries가 아닌 시스템에 있으면, *NONE 또는 *NC를 지정할 수 없습니다.

***RR**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다. SELECT, UPDATE, DELETE 및 INSERT문에 참조된 모든 표는 작업 단위(트랜잭션)가 종료될 때까지 배타적으로 잠겨 있습니다.

CLOSQCSR

SQL 커서가 내재적으로 닫히고 SQL 준비 명령문이 내재적으로 취소되고 LOCK

TABLE 잠금이 해제되는 시기를 지정합니다. SQL 커서는 CLOSE, COMMIT 또는 ROLLBACK(HOLD 없이) SQL문을 발행할 때 명시적으로 닫힙니다.

***ENDACTGRP:** 활성 그룹 종료시, SQL 커서는 닫히고 SQL 준비 명령문은 내재적으로 취소되며 LOCK TABLE 잠금은 해제됩니다.

***ENDMOD:** 모듈 종료시 SQL 커서는 닫히고 SQL 준비 명령문은 내재적으로 취소됩니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 LOCK TABLE 잠금이 해제됩니다.

ALWCOPYDTA

자료의 사본이 SELECT문에 사용될 수 있는지 여부를 지정합니다.

***OPTIMIZE:** 시스템은 데이터베이스에서 직접 검색된 자료를 사용할지 또는 자료의 사본을 사용할지의 여부를 판별합니다. 결정은 최상의 성능을 제공할 수 있는 방법에 근거하여 이루어집니다. COMMIT가 *CHG 또는 *CS이면서 ALWBLK가 *ALLREAD가 아니거나, COMMIT가 *ALL 또는 *RR이면 조회 수행에 필요한 경우에만 자료의 사본이 사용됩니다.

***YES:** 필요할 때만 자료의 사본이 사용됩니다.

***NO:** 자료의 사본은 허용되지 않습니다. 조회를 수행하는 데 자료의 임시 사본이 필요하면 오류 메시지가 표시됩니다.

ALWBLK

데이터베이스 관리자가 레코드 블록화를 사용할 수 있는지 여부와 블록화가 읽기 전용 커서에 사용될 수 있는 확장 범위를 지정합니다.

***ALLREAD:** *NONE 또는 *CHG가 COMMIT 매개변수에 지정된 경우 읽기 전용 커서에 대해 행이 블록화됩니다. 명시적으로 갱신될 수 없는 프로그램 내의 모든 커서들은 EXECUTE 또는 EXECUTE IMMEDIATE문이 프로그램 내에 있을 지라도 읽기 전용으로 열립니다.

*ALLREAD 지정:

- *READ에 대해 허용된 블록화 이외의 확약 제어 레벨 *CHG하에서 레코드의 블록화를 허용합니다.
- 프로그램에서 거의 모든 읽기 전용 커서의 성능이 향상될 수 있으나 다음과 같은 방법으로 조회를 제한합니다.
 - 롤백(ROLLBACK) 명령, 호스트 언어의 ROLLBACK문 또는 ROLLBACK HOLD SQL문은 *ALLREAD가 지정된 경우 읽기 전용 커서의 위치를 재 지정하지 않습니다.
 - 커서에 대한 DECLARE문에 FOR UPDATE절이 없는 한, 지정된 UPDATE 또는 DELETE문의 동적 수행(예를 들면, EXECUTE IMMEDIATE 사용)은 커서의 행을 갱신하는 데 사용될 수 없습니다.

***NONE:** 커서에 대한 자료의 검색을 위해 행이 블록화되지 않습니다.

CRTSQLRPGI

*NONE을 지정하는 경우:

- 검색된 자료가 현재의 자료임을 보장합니다.
- 조회시 자료의 첫 번째 행을 검색하는 데 필요한 시간을 줄일 수 있습니다.
- 조회가 닫히기 전에 조회의 처음 몇 행만이 검색될 때 프로그램에 의해 사용되지 않은 자료 행 블록의 검색으로부터 데이터베이스 관리자를 중단시킵니다.
- 많은 행을 검색하는 조회의 전반적인 성능이 저하될 수 있습니다.

***READ:** 레코드는 다음의 경우 커서에 대한 자료의 읽기 전용 검색에 대해 블록화됩니다.

- *NONE이 COMMIT 매개변수에 지정되어 확약 제어가 사용되지 않음을 표시하는 경우
- FOR READ ONLY절로 커서가 선언되었거나 커서에 대해 지정된 UPDATE 또는 DELETE문을 수행할 수 있는 동적 명령문이 없는 경우

*READ를 지정하면 위의 조건을 만족하는 조회의 전반적인 성능이 향상되고 많은 수의 레코드를 검색할 수 있습니다.

DLYPRP

PREPARE문에 대한 동적 명령문 유효성 검사가 OPEN, EXECUTE 또는 DESCRIBE문 실행시까지 지연되는지 여부를 지정합니다. 유효성 검사가 지연되면 불필요한 확인이 제거되므로 성능이 향상됩니다.

***NO:** 동적 명령문 유효성 검사가 지연되지 않습니다. 동적 명령문이 준비될 때 액세스 계획의 유효성이 확인됩니다. 동적 명령문이 OPEN 또는 EXECUTE문에 사용될 때 액세스 계획의 유효성이 재확인됩니다. 동적 명령문에 의해 참조되는 오브젝트의 존재 또는 권한으로 인해, 동적 명령문이 아직 유효한지 확인하기 위해 OPEN 또는 EXECUTE문을 발행하기 전에 SQLCODE 또는 SQLSTATE를 검사해야 합니다.

***YES:** 동적 명령문 유효성 검사는 동적 명령문이 OPEN, EXECUTE 또는 DESCRIBE SQL문에 사용될 때까지 지연됩니다. 동적 명령문이 사용될 때 유효성 검사가 완료되고 액세스 계획이 수립됩니다. 이 매개변수에 *YES를 지정하면 OPEN, EXECUTE 또는 DESCRIBE문을 실행한 후에 SQLCODE와 SQLSTATE를 체크하여 동적 명령문이 유효한지 확인해야 합니다.

주: *YES를 지정하면 PREPARE에 INTO절이 사용되거나, 동적 명령문에 대한 OPEN문이 발행되기 전에 DESCRIBE문에서 그 동적 명령문을 사용하는 경우 성능은 향상되지 않습니다.

GENLVL

작성 조작이 실패하는 심각도 레벨을 지정합니다. 이 값 이상의 심각도를 가진 오류가 발생하면 조작이 종료됩니다.

10: 디폴트 심각도 레벨은 10입니다.

심각도 레벨: 0-40 범위 내의 값을 지정하십시오.

DATEFMT

결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 날짜 필드가 지정된 형식으로 리턴됩니다. 입력 날짜 스트링의 경우 지정된 값은 유효한 형식으로 날짜가 지정되었는지 여부를 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 iSeries 시스템이 아닌 시스템에 있으면, *USA, *ISO, *EUR 또는 *JIS가 지정되어야 합니다.

***JOB:** 작업에 지정된 형식이 사용됩니다. 작업 표시(DSPJOB) 명령을 사용하여 작업에 대한 현재 날짜 형식을 판별하십시오.

***USA:** 미국식 날짜 형식(mm/dd/yyyy)이 사용됩니다.

***ISO:** 국제 표준화 기구(ISO) 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***EUR:** 유럽식 날짜 형식(dd.mm.yyyy)이 사용됩니다.

***JIS:** 일본 산업 표준 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***MDY:** 날짜 형식(mm/dd/yy)이 사용됩니다.

***DMY:** 날짜 형식(dd/mm/yy)이 사용됩니다.

***YMD:** 날짜 형식(yy/mm/dd)이 사용됩니다.

***JUL:** 율리우스력 날짜 형식(yy/ddd)이 사용됩니다.

DATSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *JOB, *MDY, *DMY, *YMD 또는 *JUL이 DATEFMT 매개변수에 지정되어 있을 때만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 날짜 분리자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘/’: 슬래시(/)가 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

CRTSQLRPGI

‘,’: 쉼표(,)가 사용됩니다.

‘-’: 붙임표(-)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

TIMFMT

결과 열에 액세스하기 위해 사용되는 형식을 지정합니다. 입력 시간 스트링의 경우 지정된 값은 시간이 유효한 형식으로 지정되었는지 판별하는 데 사용됩니다.

주: *USA, *ISO, *EUR 또는 *JIS 형식을 사용하는 입력 시간 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 또 다른 iSeries 시스템이 아닌 시스템에 있으면, 시간 형식이 콜론이나 마침표의 시간 분리문자가 있는 *USA, *ISO, *EUR, *JIS 또는 *HMS이어야 합니다.

***HMS:** **hh:mm:ss** 형식이 사용됩니다.

***USA:** 미국식 시간 형식 **hh:mm xx**가 사용되는데, 여기서 **xx**는 AM(오전) 또는 PM(오후)입니다

***ISO:** 국제 표준화 기구(ISO) 시간 형식 **hh.mm.ss**가 사용됩니다.

***EUR:** 유럽식 시간 형식 **hh.mm.ss**가 사용됩니다.

***JIS:** 일본 산업 표준 시간 형식인 **hh:mm:ss**가 사용됩니다.

TIMSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *HMS가 TIMFMT 매개변수에 지정되어 있을 때에만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 시간 분리 문자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘.’: 콜론(:)이 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

REPLACE

동일한 이름의 기존 SQL 모듈, 프로그램, 서비스 프로그램 또는 패키지가 있고 동일한 라이브러리에 입력되었을 때 SQL 모듈, 프로그램, 서비스 프로그램 또는 패키지가 작성되는지를 지정합니다. 이 매개변수의 값은 CRTMODC, CRTBND, CRTSRVPGM 및 CRTSQLPKG 명령에 전달됩니다.

***YES:** 새로운 SQL 모듈, 서비스 프로그램 또는 패키지가 작성되고, 지정된 라이브러리 내에서 같은 이름과 유형을 가지는 기존의 모든 SQL 오브젝트는 QRPLOBJ 로 이동됩니다.

***NO:** 새로운 SQL 모듈, 프로그램, 서비스 프로그램 또는 패키지가 지정된 라이브러리에 동일한 이름과 유형의 SQL 오브젝트가 이미 있는 경우에는 작성되지 않습니다.

RDB

SQL 패키지 오브젝트가 작성되는 관계형 데이터베이스의 이름을 지정합니다.

***LOCAL:** 프로그램은 분산 SQL 프로그램으로서 작성됩니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다. SQL 패키지 오브젝트는 사전컴파일 프로세스의 일부분으로서 작성되지 않습니다. 구조화 조회 언어 패키지 작성(CRTSQLPKG) 명령이 사용될 수 있습니다.

relational-database-name: 새로운 SQL 패키지 오브젝트가 작성될 때 관계형 데이터베이스의 이름을 지정하십시오. 로컬 관계형 데이터베이스명이 지정되면 작성된 프로그램은 계속 분산 SQL 프로그램입니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다.

***NONE:** SQL 패키지 오브젝트가 작성되지 않습니다. 프로그램 오브젝트는 분산 프로그램이 아니며 CRTSQLPKG(구조화 조회 언어 패키지 작성) 명령은 사용될 수 없습니다.

USER

변환 시작시 리모트 시스템으로 송신되는 사용자명을 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***CURRENT:** 현재 작업이 실행 중인 사용자 프로파일이 사용됩니다.

user-name: 어플리케이션 서버 작업에 사용되는 사용자명을 지정하십시오.

PASSWORD

리모트 시스템에 사용될 암호를 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***NONE:** 암호가 송신되지 않습니다. 이 값이 지정되면 USER(*CURRENT) 또한 지정되어야 합니다.

password: USER 매개변수에 지정된 사용자명의 암호를 지정하십시오.

RDBCNNMTH

CONNECT문에 사용되는 의미를 지정합니다. 자세한 정보는 SQL 참조서를 참조하십시오.

***DUW:** CONNECT(유형 2) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 추가 관계형 데이터베이스에 대한 연속된 CONNECT문이 이전의 연결을 단절시키지는 않습니다.

***RUW:** CONNECT(유형 1) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 연속된 CONNECT문은 새로운 연결이 설정되기 전에 이전의 연결을 단절시킵니다.

DFTRDBCOL

표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용되는 스키마명을 지정합니다. 이 매개변수는 정적 SQL문에만 적용됩니다.

***NONE:** OPTION 매개변수에 정의된 명명 규칙이 사용됩니다.

schema-name: 스키마 ID의 이름을 지정하십시오. 이 값은 OPTION 매개변수에 지정된 명명 규칙 대신 사용됩니다.

DYNDFTCOL

DFTRDBCOL 매개변수에 대해 지정된 디폴트 스키마명도 동적 명령문에 사용되는지를 지정합니다.

***NO:** 표, 뷰, 색인 및 동적 SQL문에 대한 SQL 패키지의 규정되지 않은 이름으로 DFTRDBCOL 매개변수에 지정된 값을 사용하지 마십시오. OPTION 매개변수에 지정된 명명 규칙이 사용됩니다.

***YES:** DFTRDBCOL 매개변수에 지정된 콜렉션명은 표, 뷰, 색인 및 동적 SQL문의 규정되지 않은 이름으로 사용됩니다.

SQLPKG

이 명령의 RDB 매개변수에 지정된 관계형 데이터베이스에 작성된 SQL 패키지의 규정된 이름을 지정합니다.

가능한 라이브러리 값은 다음과 같습니다.

***OBJLIB:** 패키지가 OBJ 매개변수에 지정된 라이브러리와 같은 이름의 라이브러리에 작성됩니다.

library-name: 패키지가 작성된 라이브러리의 이름을 지정하십시오.

***OBJ:** SQL 패키지의 이름은 OBJ 매개변수에 지정된 오브젝트명과 같습니다.

package-name: SQL 패키지의 이름을 지정하십시오. 리모트 시스템이 iSeries 시스템이 아니면, 9자 이상을 지정할 수 없습니다.

SQLPATH

정적 SQL문의 사용자 정의 유형, 프로시저 및 함수 등을 찾기 위해 사용될 경로를 지정합니다.

***NAMING:** 사용되는 경로는 OPTION 매개변수에 지정된 명명 규칙에 따릅니다.

***SYS** 명명의 경우 사용되는 경로는 *LIBL로, 실행시 현재 라이브러리 리스트입니다.

***SQL** 명명의 경우 사용되는 경로는 "QSYS", "QSYS2", "userid"입니다. 여기서 "userid"는 USER 특수 레지스터의 값입니다. DFTRDBCOL 매개변수에 schema-name을 지정한 경우 schema-name이 사용자 ID 대신 사용됩니다.

***LIBL:** 사용되는 경로는 실행시 라이브러리 리스트입니다.

schema-name: 하나 이상의 스키마명 리스트를 지정하십시오. 최대 268개의 개별 스키마를 지정할 수 있습니다.

SQLCURRULE

SQL문에 사용되는 의미를 지정합니다.

***DB2:** 모든 SQL문의 의미에 대한 디폴트는 DB2에 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 문자 자료로 처리됩니다.

***STD:** 모든 SQL문의 의미에 대한 디폴트는 ISO 및 ANSI SQL 표준에서 설정된 규칙입니다. 다음 의미는 이 옵션에서 제어됩니다.

- 16진 상수는 2진 자료로 처리됩니다.

SAAFLAG

IBM SQL 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 IBM SQL 구문을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다. IBM 데이터베이스에 있는 IBM SQL 구문에 대한 정보는 *DRDA IBM SQL Reference, SC26-3255-00* 책을 참조하십시오.

***NOFLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사하지 않습니다.

***FLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 검사합니다.

FLAGSTD

미국 표준 협회(ANSI) 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 다음과 같은 표준을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다.

ANSI X3.135-1992 항목

ISO 9075-1992 항목

FIPS 127.2 항목

***NONE:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크하지 않습니다.

CRTSQLRPGI

***ANS:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크합니다.

DBGVIEW

처리되는 테이프 자료 파일의 레코드 유형과 블록화 속성을 지정합니다.

***NONE:** 소스 뷰가 생성되지 않습니다.

***SOURCE:** SQL 사전컴파일러가 루트와 함께, 필요에 따라 SQL INCLUDE문에 대한 소스 뷰도 제공합니다. 사전컴파일러에 의해 생성된 명령문이 있는 뷰가 제공됩니다.

USRPRF

정적 SQL문에서 각 오브젝트에 대해 프로그램 오브젝트가 가지는 권한을 포함하여 컴파일된 프로그램 오브젝트를 실행할 때 사용되는 사용자 프로파일을 지정합니다. 프로그램 소유자 또는 프로그램 사용자의 프로파일이 프로그램 오브젝트가 사용할 수 있는 오브젝트를 제어하는 데 사용됩니다.

***NAMING:** 사용자 프로파일은 명명 규칙에 의해 판별됩니다. 명명 규칙이 *SQL이면 USRPRF(*OWNER)가 사용됩니다. 명명 규칙이 *SYS이면 USRPRF(*USER)가 사용됩니다.

***USER:** 프로그램 오브젝트를 실행 중인 사용자의 프로파일이 사용됩니다.

***OWNER:** 프로그램 소유자와 프로그램 사용자 모두의 사용자 프로파일이 프로그램 실행시 사용됩니다.

DYNUSRPRF

동적 SQL문에 대해 사용될 사용자 프로파일을 지정합니다.

***USER:** 로컬 동적 SQL문은 프로그램 사용자의 프로파일하에서 실행됩니다. 분산 동적 SQL문은 SQL 패키지의 사용자 프로파일하에서 실행됩니다.

***OWNER:** 로컬 동적 SQL문은 프로그램 소유자의 프로파일하에서 실행됩니다. 분산 동적 SQL문은 SQL 패키지 소유자의 프로파일하에서 실행됩니다.

SRTSEQ

SQL문의 스트링 비교에 사용될 정렬 순서표를 지정합니다.

주: *HEX는 어플리케이션 서버가 iSeries 시스템에 없거나 릴리스 레벨이 V2R3M0 이전인 분산 어플리케이션의 매개변수에 대해서만 지정되어야 합니다

***JOB:** 작업에 대한 SRTSEQ 값이 사전컴파일 중에 검색됩니다.

***JOB RUN:** 작업에 대한 SRTSEQ 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 SRTSEQ(*JOB RUN)는 LANGID(*JOB RUN)가 지정될 때만 유효합니다.

***LANGID UNQ:** LANGID 매개변수에 지정된 고유 가중치 정렬표가 사용됩니다.

***LANGIDSHR:** 정렬 순서표는 여러 문자에 대해 동일한 가중치를 사용하며, LANGID 매개변수에 지정된 언어와 연관된 공유 가중치 정렬 순서표입니다.

***HEX:** 정렬 순서표가 사용되지 않습니다. 문자의 16진수 값이 정렬 순서를 결정하기 위해 사용됩니다.

정렬 순서표의 이름은 다음 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

table-name: 사용될 정렬 순서표의 이름을 지정하십시오.

LANGID

SRTSEQ(*LANGIDUNQ) 또는 SRTSEQ(*LANGIDSHR) 지정시 사용될 언어 ID를 지정합니다.

***JOB:** 작업에 대한 LANGID 값이 사전컴파일 중에 검색됩니다.

***JOBRUN:** 작업에 대한 LANGID 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 LANGID(*JOBRUN)는 SRTSEQ(*JOBRUN)가 지정될 때만 유효합니다.

language-identifier: 언어 ID를 지정하십시오.

OUTPUT

사전컴파일러 리스트가 생성되는지를 지정합니다.

***NONE:** 사전컴파일러 리스트가 생성되지 않습니다.

***PRINT:** 사전컴파일러 리스트가 생성됩니다.

PRTFILE

사전컴파일러 인쇄 출력을 지정한 프린터 파일의 규정화명을 지정합니다. 파일의 길이는 최소 132바이트이어야 합니다. 132바이트보다 적은 레코드 길이로 파일이 지정되는 경우 정보는 유실됩니다.

프린터 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

CRTSQLRPGI

QSYSPRT: 파일명이 지정되지 않으면 사전컴파일러 인쇄 출력이 IBM 제공 프린터 파일 QSYSPRT로 전달됩니다.

printer-file-name: 사전컴파일러 인쇄 출력이 전달될 프린터 파일의 이름을 지정하십시오.

TOSRCFILE

SQL 사전컴파일러에 의해 처리된 출력 소스 멤버가 있게 될 소스 파일의 규정화명을 지정합니다. 지정된 소스 파일을 찾을 수 없는 경우에는 이를 작성합니다. 출력 멤버는 SRCMBR 매개변수용으로 지정한 이름과 동일한 이름을 갖게 됩니다.

가능한 라이브러리 값은 다음과 같습니다.

QTEMP: 라이브러리 QTEMP가 사용될 것입니다.

***LIBL:** 작업 라이브러리 리스트가 지정 파일에 대해 탐색됩니다. 만일 라이브러리 리스트의 어떠한 라이브러리에서도 이 파일이 없는 경우에 파일은 현재 라이브러리에서 작성됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 사용될 것입니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없는 경우 QGPL 라이브러리가 사용될 것입니다.

library-name: 출력 소스 파일이 작성될 라이브러리명을 지정하십시오.

QSQLTEMP1: 소스 파일 QSQLTEMP1이 사용될 것입니다.

source-file-name: 출력 소스 멤버가 들어 있는 소스 파일의 이름을 지정하십시오.

TEXT

기능을 간략하게 설명하는 텍스트를 지정합니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 TEXT 매개변수에 나와 있습니다.

***SRCMBRTXT:** RPG 프로그램을 작성하는 데 사용되는 소스 파일 멤버로부터 텍스트를 취합니다. STRSEU(소스 입력 유틸리티 시작) 명령을 사용하거나 ADDPFM(실제 파일 멤버 추가) 또는 CHGPFM(실제 파일 멤버 변경) 명령을 사용하여 데이터베이스 소스 멤버에 텍스트를 추가하거나 데이터베이스 소스 멤버의 텍스트를 변경할 수 있습니다. 소스 파일이 인라인 파일이나 장치 파일일 경우 텍스트는 공백입니다.

***BLANK:** 텍스트가 지정되지 않습니다.

'description': 50자 미만의 텍스트를 어포스트로피로 묶어 지정하십시오.

예:

```
CRTSQLRPGI PAYROLL OBJTYPE(*PGM) TEXT('Payroll Program')
```

CRTSQLRPGI

이 명령은 소스문을 사전컴파일하는 SQL 사전컴파일러를 실행하여 변경된 소스문을 라이브러리 QTEMP에 있는 파일 QSQLTEMP1의 멤버 PAYROLL에 저장합니다. SQL 사전컴파일러에 의해 작성된 소스 멤버를 사용함으로써 현재 라이브러리에 있는 프로그램 PAYROLL을 작성하기 위해 ILE RPG 컴파일러가 호출됩니다.

CRTSQLRPGI

부록 C. iSeries용 FORTRAN 사전컴파일러 사용

이 부록에는 이 컴파일러가 더 이상 iSeries에서 지원되지 않지만, iSeries용 FORTRAN에 대한 구문 다이어그램이 나와 있습니다. 다른 부록 349 페이지의 부록 D 『FORTRAN 어플리케이션에서 SQL문 코딩』에서는 FORTRAN/400 프로그램에서 SQL문을 삽입할 때 고유한 어플리케이션 및 코딩 요구사항을 설명합니다.

자세한 내용은 『FORTRAN/400 사전컴파일러 사용』을 참조하십시오.

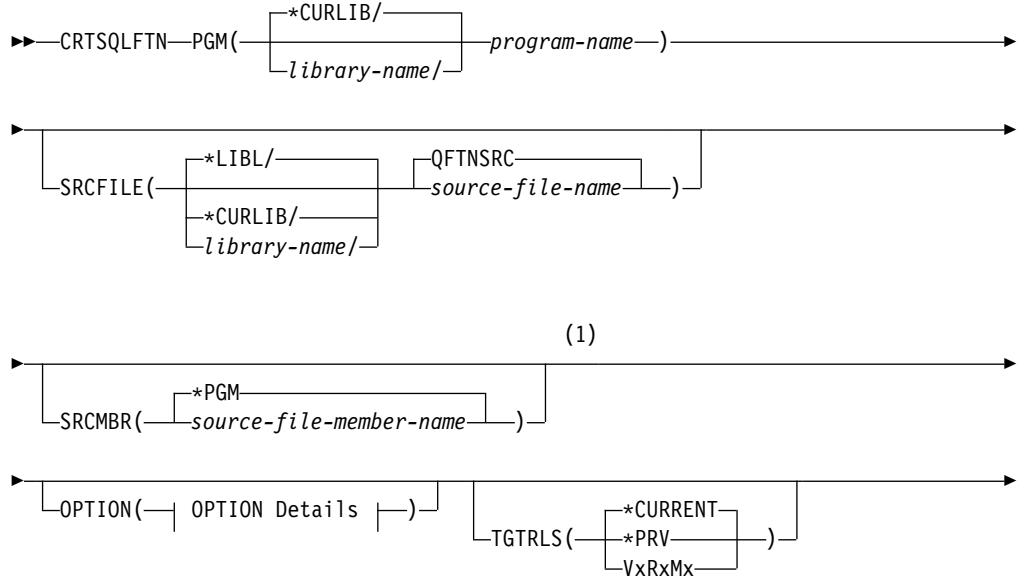
주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

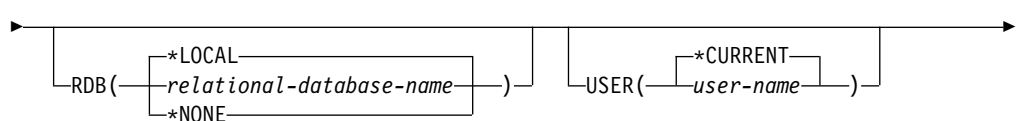
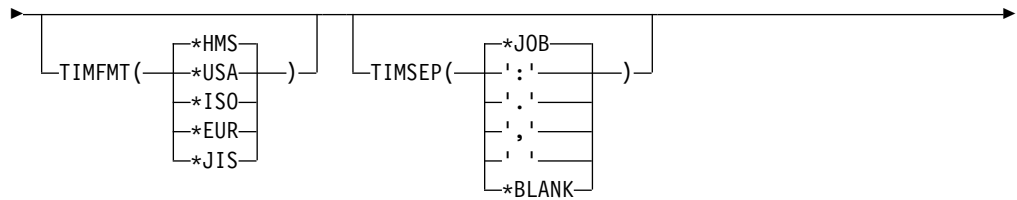
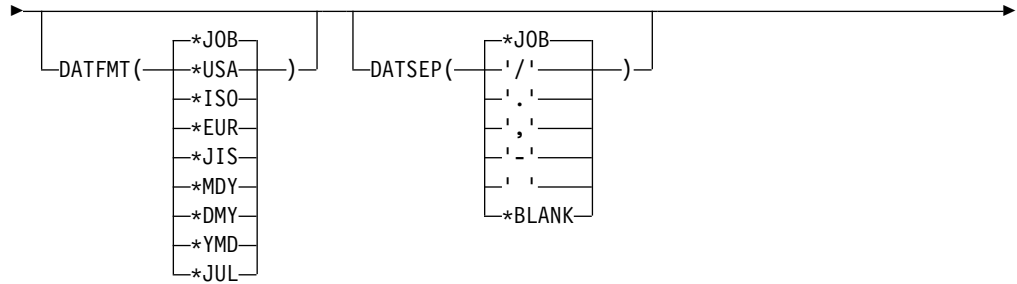
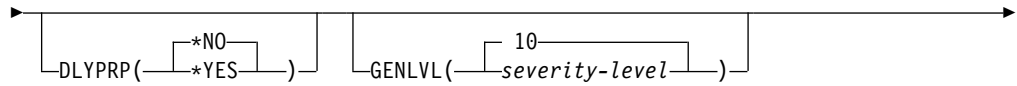
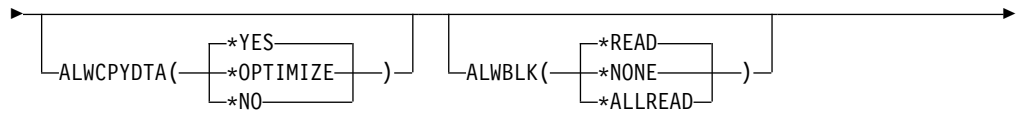
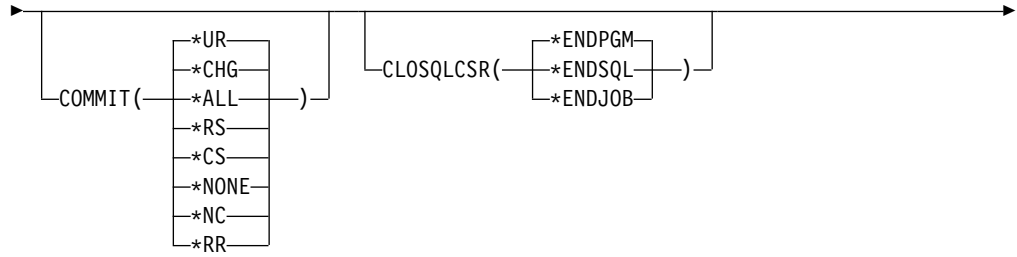
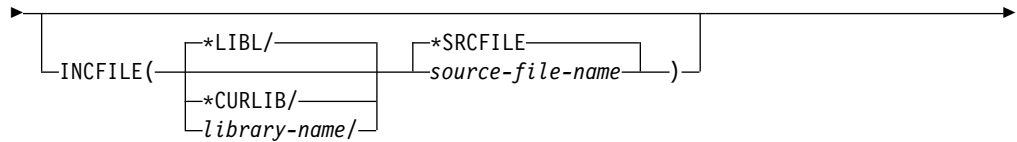
FORTRAN/400 사전컴파일러 사용

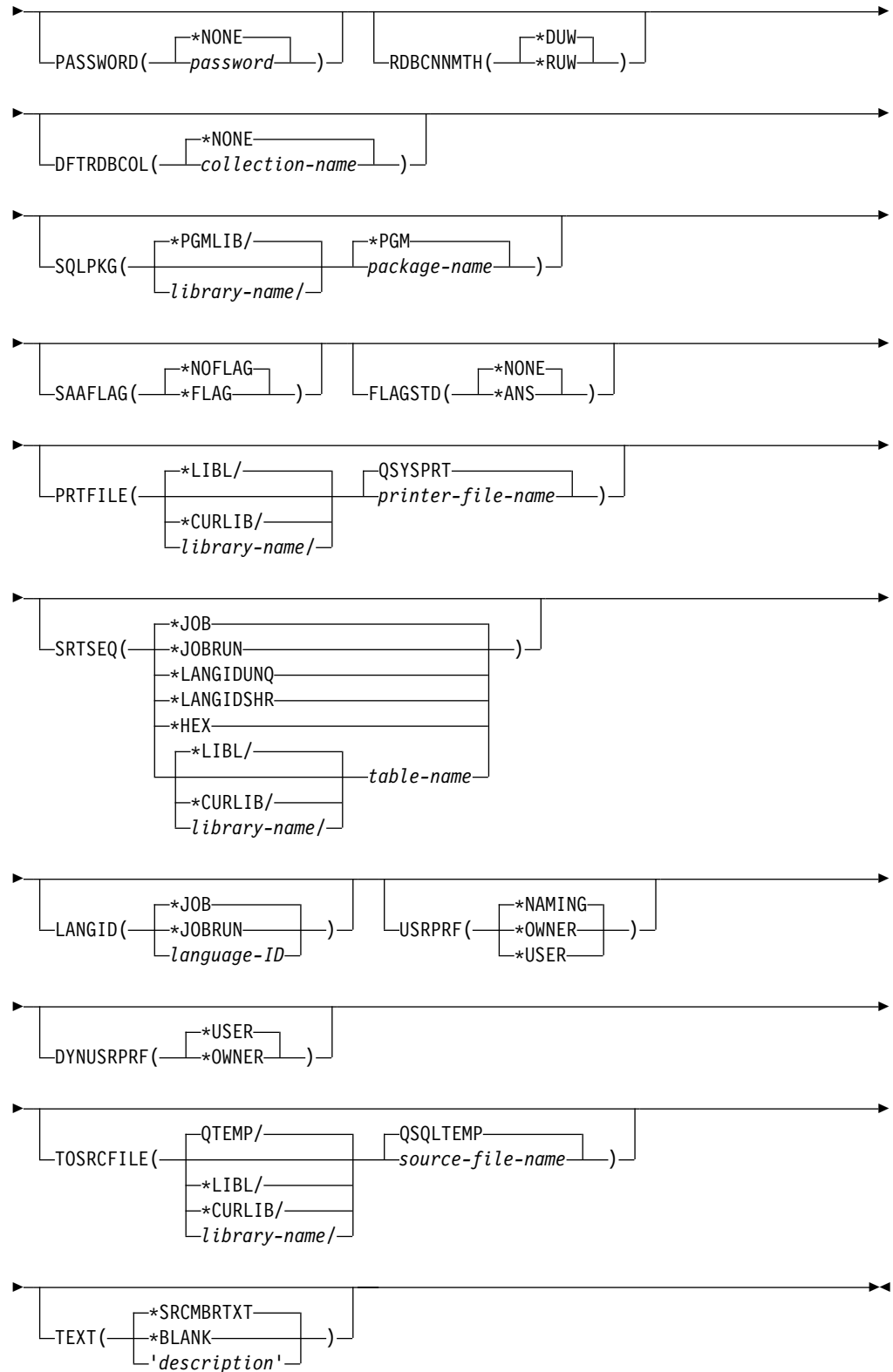
FORTRAN/400 컴파일러는 더 이상 iSeries 시스템에서 지원되지 않습니다. 이 부록은 IBM 제품이 아닌 기타 FORTRAN 컴파일러와 함께 SQL FORTRAN 사전컴파일러를 사용하고 있는 고객을 돕기 위해 작성되었습니다. FORTRAN 사전컴파일러 사용에 대한 설명은 CRTSQLFTN(구조화 조회 언어 FORTRAN 작성) 명령을 참조하십시오. 자세한 정보는 『CRTSQLFTN(구조화 조회 언어 FORTRAN 작성) 명령』을 참조하십시오.

CRTSQLFTN(구조화 조회 언어 FORTRAN 작성) 명령

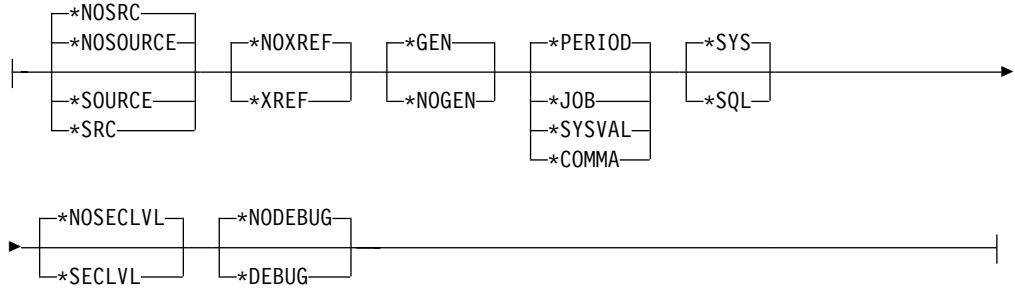
Job: B,I Pgm: B,I REXX: B,I Exec







OPTION 세부사항:



주:

- 1 이 앞에 오는 모든 매개변수는 앞뒤 관계에 따라 지정할 수 있습니다.

CRTSQLFTN 명령의 목적

CRTSQLFTN(구조화 조회 언어 FORTRAN 작성) 명령은 SQM문이 들어 있는 FORTRAN 소스를 사전컴파일하는 구조화 조회 언어(SQL) 사전컴파일러를 호출하여 임시 소스 멤버를 작성한 후 선택적으로 FORTRAN 컴파일러를 호출하여 프로그램을 컴파일합니다.

CRTSQLFTN 명령의 매개변수

PGM

컴파일된 프로그램의 규정화명을 지정합니다.

컴파일된 FORTRAN 프로그램의 이름은 다음 라이브러리 값 중 하나로 규정될 수 있습니다.

***CURLIB:** 컴파일된 FORTRAN 프로그램은 작업의 현재 라이브러리에서 작성됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 컴파일된 FORTRAN 프로그램이 작성된 라이브러리의 이름을 지정하십시오.

program-name: 컴파일된 FORTRAN 프로그램의 이름을 지정합니다.

SRCFILE

SQL문이 있는 FORTRAN 소스가 들어 있는 소스 파일의 규정된 이름을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QFTNSRC: 소스 파일명이 지정되지 않은 경우 IBM 제공 소스 파일 QFTNSRC에 FORTRAN 소스가 포함됩니다.

source-file-name: FORTRAN 소스가 들어 있는 소스 파일의 이름을 지정합니다.

SRCMBR

FORTRAN 소스가 들어 있는 소스 파일 멤버명을 지정합니다. 이 매개변수는 SRCFILE 매개변수의 소스 파일명이 데이터베이스 파일일 경우에만 지정됩니다. 이 매개변수가 지정되지 않으면 PGM 매개변수에 지정된 PGM명이 사용됩니다.

***PGM:** FORTRAN 소스가 PGM 매개변수에 지정된 이름과 같은 이름의 소스 파일의 멤버에 있다고 지정합니다.

source-file-member-name: FORTRAN 소스가 들어 있는 멤버의 이름을 지정합니다.

OPTION

FORTRAN 소스문이 사전컴파일될 때 다음 옵션 중 하나 또는 여러 개가 사용되는지의 여부를 지정합니다. 옵션이 두 번 이상 지정되면 마지막에 지정된 옵션이 사용됩니다.

요소 1: 소스 리스팅 옵션

***NOSOURCE** 또는 ***NOSRC:** 사전컴파일러나 패키지 작성 중에 오류가 검출되지 않는 한, 소스 인쇄 출력이 사전컴파일러에 의해 작성되지 않습니다.

***SOURCE** or ***SRC:** 사전컴파일러가 FORTRAN 소스 입력으로 구성된 소스 인쇄 출력을 작성합니다.

요소 2: 상호 참조 옵션

***NOXREF:** 사전컴파일러가 상호 참조명을 지정하지 않습니다.

***XREF:** 사전컴파일러는 그러한 항목을 참조하는 프로그램 내의 명령문 번호 항목을 상호 참조합니다.

요소 3: 프로그램 작성 옵션

***GEN:**

***NOGEN:** 사전컴파일러가 FORTRAN 컴파일러를 호출하지 않고, 프로그램과 SQL 패키지가 작성되지 않습니다.

요소 4: 소수점 옵션

***PERIOD:** SQL문에 사용된 숫자 상수에 대한 소수점으로 사용되는 값은 마침표입니다.

***JOB:** SQL에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 사전컴파일시 작업에 대해 지정된 소수점 표현입니다.

***SYSVAL:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 QDECFMT 시스템 값입니다.

주: QDECFMT가 소수점으로 사용되는 값을 쉼표로 지정하면 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)를 뒤에 공백이 오는 쉼표로 분리해야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 소수점이 마침표인 VALUES(1.1,2.23,4.1)와 동일합니다.

***COMMA:** SQL문에 있는 숫자 상수에 대한 소수점으로 사용되는 값이 쉼표입니다.

주: 리스트에 있는 숫자 상수(SELECT절 또는 VALUES절에서와 같이)는 뒤에 공백이 오는 쉼표로 분리해야 합니다. 예를 들어 VALUES(1,1, 2,23, 4,1)는 마침표가 소수점으로 지정된 VALUES(1.1,2.23,4.1)와 동일합니다.

요소 5: 명명 규칙 옵션

***SYS:** 시스템 명명 규칙(library-name/file-name)이 사용됩니다.

***SQL:** SQL 명명 규칙(schema-name.table-name)이 사용됩니다. iSeries 서버 이외의 리모트 데이터베이스에서 프로그램을 작성할 때 *SQL은 명명 규칙을 따라 지정되어야 합니다.

요소 6: 2차 레벨 메시지 텍스트 옵션

***NOSECLVL:** 2차 레벨 텍스트 설명이 인쇄 출력에 추가되지 않습니다.

***SECLVL:** 대체 자료를 가진 2차 레벨 텍스트가 리스팅의 모든 메시지에 대해 추가됩니다.

요소 7: 디버그 옵션

***NODEBUG:** 기호 확장 프로그램 모델(EPM)의 디버그 정보가 프로그램과 함께 저장되지 않습니다. 이 옵션은 컴파일러에 전달되고, SQL 사전컴파일러에는 영향을 미치지 않습니다.

***DEBUG:** 기호 EPM 디버그 정보가 프로그램과 함께 저장됩니다. 이 옵션은 컴파일러에 전달되고, SQL 사전컴파일러에는 영향을 미치지 않습니다.

TGTRLS

작성될 오브젝트를 사용하려고 하는 오퍼레이팅 시스템의 릴리스 레벨을 지정합니다.

*CURRENT와 *PRV 값이 주어질 때와 릴리스 레벨 값을 지정할 때에는 VxRxMx의 형식으로 릴리스를 지정하게 되며, 여기서 Vx는 버전, Rx는 릴리스, Mx는 수정 레벨을 나타냅니다. 예를 들어, V2R3M0은 버전 2, 릴리스 3, 수정 레벨 0을 나타냅니다.

***CURRENT:** 오브젝트는 사용자 시스템에서 현재 실행 중인 오퍼레이팅 시스템의 릴리스에 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *CURRENT는 V2R3M5가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

주: 시스템에서 V2R3M5가 실행되고 있으며, V2R3M0가 설치된 시스템에서 오브젝트를 사용하려는 경우에는 TGTRLS(*CURRENT)가 아니고 TGTRLS(V2R3M0)를 지정해야 합니다.

***PRV:** 오브젝트는 오퍼레이팅 시스템의 이전 릴리스, 수정 레벨 0에서 사용됩니다. 예를 들어, 시스템에서 V2R3M5가 실행되고 있으면, *PRV는 V2R2M0가 설치된 시스템에서 오브젝트를 사용할 것임을 의미합니다. 또한 사용자는 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스에서도 오브젝트를 사용할 수 있습니다.

release-level: VxRxMx 형식의 릴리스를 지정하십시오. 지정된 릴리스의 시스템이나 설치되어 있는 오퍼레이팅 시스템의 어느 후속 릴리스의 시스템에서도 오브젝트를 사용할 수 있습니다.

유효한 값들은 현재 버전, 릴리스, 수정 레벨에 기초하고 있으며, 새로운 릴리스로 변경될 수 있습니다. 이 명령이 지원하는 가장 이전의 릴리스 레벨보다 더 앞의 릴리스 레벨을 지정하면 지원되는 릴리스 레벨 가운데 가장 이전의 릴리스가 표시된 오류 메시지를 받게 됩니다.

INCFILE

SQL INCLUDE문을 사용하여 프로그램에 포함시킨 멤버에 들어 있는 소스 파일의 규정화명을 지정합니다.

소스 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

***SRCFILE:** SRCFILE 매개변수에 지정된 규정된 소스 파일에는 모든 SQL INCLUDE문에 지정된 소스 파일 멤버가 포함됩니다.

source-file-name: 모든 SQL INCLUDE문에 지정된 소스 파일명을 포함하는 소스 파일의 이름을 지정하십시오. 여기에서 지정된 소스 파일의 레코드 길이는 SRCFILE 매개변수에 지정된 소스 파일의 레코드 길이보다 짧아야 합니다.

COMMIT

컴파일된 프로그램의 SQL문이 확약 제어하에서 실행되는지 여부를 지정합니다. 호스트 언어 소스문에 참조된 파일은 이 옵션의 영향을 받지 않습니다. SQL문에 참조된 SQL 표, SQL 뷰 및 SQL 패키지만이 영향을 받습니다.

***CHG 또는 *UR**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON 및 REVOKE문에서 참조된 오브젝트들을 지정하며, 갱신, 삭제 및 삽입된 행들은 작업 단위(트랜잭션)가 종료될 때까지 잠겨 있습니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다.

***ALL 또는 *RS**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입된 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***CS**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 선택은 되었지만 갱신되지 않은 행은 다음 행이 선택될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다.

***NONE 또는 *NC**: 확약 제어가 사용되지 않음을 지정합니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 있습니다. SQL DROP SCHEMA문이 프로그램에 포함되는 경우에는 *NONE 또는 *NC가 반드시 사용되어야 합니다. RDB 매개변수에 관계형 데이터베이스가 지정되어 있고 관계형 데이터베이스가 iSeries가 아닌 시스템에 있으면, *NONE 또는 *NC를 지정할 수 없습니다.

***RR**: SQL ALTER, CALL, COMMENT ON, CREATE, DROP, GRANT, LABEL ON, RENAME 및 REVOKE문에서 참조되는 오브젝트들을 지정하며, 선택, 갱신, 삭제 및 삽입되는 행은 작업 단위(트랜잭션)가 종료될 때까지 잠깁니다. 사용자는 다른 작업의 확약되지 않은 변경 내용을 볼 수 없습니다. SELECT, UPDATE, DELETE 및 INSERT문에 참조된 모든 표는 작업 단위(트랜잭션)가 종료될 때까지 배타적으로 잠겨 있습니다.

CLOSE

SQL 커서가 내재적으로 닫히고 SQL 준비 명령문이 내재적으로 취소되고 LOCK TABLE 잠금이 해제되는 시기를 지정합니다. SQL 커서는 CLOSE, COMMIT 또는 ROLLBACK(HOLD 없이) SQL문을 발행할 때 명시적으로 닫힙니다.

***ENDPGM:** 프로그램 종료시 SQL 커서는 닫히고 SQL 준비 명령문은 취소됩니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 LOCK TABLE 잠금이 해제됩니다.

***ENDSQL:** SQL 커서가 호출 사이에 열린 상태로 남아 있고 다른 SQL OPEN을 실행하지 않고 폐지될 수 있습니다. 호출 스택상의 보다 높은 프로그램 중 하나가 최소한 하나의 SQL문을 실행해야 합니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 SQL 커서가 닫히고 준비된 SQL문이 취소되며, LOCK TABLE 잠금이 해제됩니다. *ENDSQL이 호출된 첫 번째 SQL 프로그램(호출 스택의 첫 번째 SQL 프로그램)인 프로그램에 대해 지정되는 경우 프로그램은 *ENDPGM이 지정된 것처럼 처리됩니다.

***ENDJOB:** SQL 커서가 호출 사이에 열린 상태로 남아 있고 다른 SQL OPEN을 실행하지 않고 폐지될 수 있습니다. 호출 스택에 나중에 들어간 프로그램은 SQL문을 실행할 필요가 없습니다. 호출 스택의 첫 번째 SQL 프로그램이 종료할 때 SQL 커서는 열린 상태로 남아있고, SQL 준비 명령문이 보존되며, LOCK TABLE 잠금이 보유됩니다. SQL 커서가 닫히고, SQL 준비 명령문은 취소되며 작업 종료시 LOCK TABLE 잠금이 해제됩니다.

ALWCPYDTA

자료의 사본이 SELECT문에 사용될 수 있는지 여부를 지정합니다.

***OPTIMIZE:** 시스템은 데이터베이스에서 직접 검색된 자료를 사용할지 또는 자료의 사본을 사용할지의 여부를 판별합니다. 결정은 최상의 성능을 제공할 수 있는 방법에 근거하여 이루어집니다. COMMIT가 *CHG 또는 *CS이면 ALWBLK가 *ALLREAD가 아니거나, COMMIT가 *ALL 또는 *RR이면 조회 수행에 필요한 경우에만 자료의 사본이 사용됩니다.

***YES:** 필요할 때만 자료의 사본이 사용됩니다.

***NO:** 자료의 사본은 허용되지 않습니다. 조회를 수행하는 데 자료의 임시 사본이 필요하면 오류 메시지가 표시됩니다.

ALWBLK

데이터베이스 관리자가 레코드 블록화를 사용할 수 있는지 여부와 블록화가 읽기 전용 커서에 사용될 수 있는 확장 범위를 지정합니다.

***ALLREAD:** *NONE 또는 *CHG가 COMMIT 매개변수에 지정된 경우 읽기 전용 커서에 대해 행이 블록화됩니다. 명시적으로 갱신될 수 없는 프로그램 내의 모든 커서들은 EXECUTE 또는 EXECUTE IMMEDIATE문이 프로그램 내에 있을 지라도 읽기 전용으로 열립니다.

***ALLREAD 지정:**

- *READ에 대해 허용된 블록화 이외의 확약 제어 레벨 *CHG하에서 레코드의 블록화를 허용합니다.

- 프로그램에서 거의 모든 읽기 전용 커서의 성능이 향상될 수 있으나 다음과 같은 방법으로 조회를 제한합니다.
 - 롤백(ROLLBACK) 명령, 호스트 언어의 ROLLBACK문 또는 ROLLBACK HOLD SQL문은 *ALLREAD가 지정된 경우 읽기 전용 커서의 위치를 재 지정하지 않습니다.
 - 커서에 대한 DECLARE문에 FOR UPDATE절이 없는 한, 지정된 UPDATE 또는 DELETE문의 동적 수행(예를 들면, EXECUTE IMMEDIATE 사용)은 커서의 행을 갱신하는 데 사용될 수 없습니다.

***NONE:** 커서에 대한 자료의 검색을 위해 행이 블록화되지 않습니다.

***NONE**을 지정하는 경우:

- 검색된 자료가 현재의 자료임을 보장합니다.
- 조회시 자료의 첫 번째 행을 검색하는 데 필요한 시간을 줄일 수 있습니다.
- 조회가 닫히기 전에 조회의 처음 몇 행만이 검색될 때 프로그램에 의해 사용되지 않은 자료 행 블록의 검색으로부터 데이터베이스 관리자를 중단시킵니다.
- 많은 행을 검색하는 조회의 전반적인 성능이 저하될 수 있습니다.

***READ:** 다음의 경우 레코드가 커서에 대한 자료의 읽기 전용 검색에 대해 블록화됩니다.

- *NONE이 COMMIT 매개변수에 지정되어 확약 제어가 사용되지 않음을 표시하는 경우
- FOR FETCH ONLY절로 커서가 선언되었거나 커서에 대해 지정된 UPDATE 또는 DELETE문을 실행할 수 있는 동적 명령문이 없는 경우

***READ**를 지정하면 위의 조건을 만족하는 조회의 전반적인 성능이 향상되고 많은 수의 레코드를 검색할 수 있습니다.

DLYPRP

PREPARE문에 대한 동적 명령문 유효성 검사가 OPEN, EXECUTE 또는 DESCRIBE문 실행시까지 지연되는지 여부를 지정합니다. 유효성 검사가 지연되면 불필요한 확인이 제거되므로 성능이 향상됩니다.

***NO:** 동적 명령문 유효성 검사가 지연되지 않습니다. 동적 명령문이 준비될 때 액세스 계획의 유효성이 확인됩니다. 동적 명령문이 OPEN 또는 EXECUTE문에 사용될 때 액세스 계획의 유효성이 재확인됩니다. 동적 명령문에 의해 참조되는 오브젝트의 존재 또는 권한으로 인해, 동적 명령문이 아직 유효한지 확인하기 위해 OPEN 또는 EXECUTE문을 발행하기 전에 SQLCODE 또는 SQLSTATE를 검사해야 합니다.

***YES:** 동적 명령문 유효성 검사는 동적 명령문이 OPEN, EXECUTE 또는 DESCRIBE SQL문에 사용될 때까지 지연됩니다. 동적 명령문이 사용될 때 유효

성 검사가 완료되고 액세스 계획이 수립됩니다. 이 매개변수에 *YES를 지정하면 OPEN, EXECUTE 또는 DESCRIBE문을 실행한 후에 SQLCODE와 SQLSTATE를 체크하여 동적 명령문이 유효한지 확인해야 합니다.

주: *YES를 지정하면 PREPARE에 INTO절이 사용되거나, 동적 명령문에 대한 OPEN문이 발행되기 전에 DESCRIBE문에서 그 동적 명령문을 사용하는 경우 성능은 향상되지 않습니다.

GENLVL

작성 조작이 실패하는 심각도 레벨을 지정합니다. 이 값 이상의 심각도 레벨을 가지는 오류가 발생하면 조작이 종료됩니다.

10: 디폴트 심각도 레벨은 10입니다.

심각도 레벨: 0-40 범위 내의 값을 지정하십시오.

DATFMT

결과 열에 액세스할 때 사용되는 형식을 지정합니다. 모든 출력 날짜 필드가 지정된 형식으로 리턴됩니다. 입력 날짜 스트링의 경우 지정된 값은 유효한 형식으로 날짜가 지정되었는지 여부를 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 iSeries 서버이 아닌 시스템에 있으면, *USA, *ISO, *EUR 또는 *JIS가 지정되어야 합니다.

***JOB:** 작업에 지정된 형식이 사용됩니다. 작업 표시(DSPJOB) 명령을 사용하여 작업에 대한 현재 날짜 형식을 판별하십시오.

***USA:** 미국식 날짜 형식(mm/dd/yyyy)이 사용됩니다.

***ISO:** 국제 표준화 기구(ISO) 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***EUR:** 유럽식 날짜 형식(dd.mm.yyyy)이 사용됩니다.

***JIS:** 일본 산업 표준 날짜 형식(yyyy-mm-dd)이 사용됩니다.

***MDY:** 날짜 형식(mm/dd/yy)이 사용됩니다.

***DMY:** 날짜 형식(dd/mm/yy)이 사용됩니다.

***YMD:** 날짜 형식(yy/mm/dd)이 사용됩니다.

***JUL:** 율리우스력 날짜 형식(yy/ddd)이 사용됩니다.

DATSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *JOB, *MDY, *DMY, *YMD 또는 *JUL이 DATFMT 매개변수에 지정되어 있을 때만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 날짜 분리자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘/’: 슬래시(/)가 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘-’: 붙임표(-)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

TIMFMT

결과 열에 액세스하기 위해 사용되는 형식을 지정합니다. 입력 시간 스트링의 경우 지정된 값은 시간이 유효한 형식으로 지정되었는지 판별하는 데 사용됩니다.

주: 형식 *USA, *ISO, *EUR 또는 *JIS를 사용하는 입력 날짜 스트링은 항상 유효합니다.

관계형 데이터베이스가 RDB 매개변수에 지정되고 데이터베이스가 iSeries 서버가 아닌 시스템에 있으면 날짜 형식은 콜론이나 마침표의 시간 분리자가 있는 *USA, *ISO, *EUR, *JIS 또는 *HMS여야 합니다.

***HMS:** (hh:mm:ss) 형식이 사용됩니다.

***USA:** 미국식 시간 형식(hh:mm xx)이 사용되는데, 여기서 xx는 AM(오전) 또는 PM(오후)입니다.

***ISO:** 국제 표준화 기구(ISO) 시간 형식(hh.mm.ss)이 사용됩니다.

***EUR:** 유럽식 시간 형식(hh.mm.ss)이 사용됩니다.

***JIS:** 일본 산업 표준 시간 형식(hh:mm:ss)이 사용됩니다.

TIMSEP

결과 열에 액세스하기 위해 사용되는 분리자를 지정합니다.

주: 이 매개변수는 *HMS가 TIMFMT 매개변수에 지정되어 있을 때에만 적용됩니다.

***JOB:** 사전컴파일시 작업에 지정된 시간 분리 문자가 사용됩니다. 작업에 대한 현재 값을 결정하려면 DSPJOB(작업 표시) 명령을 사용하십시오.

‘:’: 콜론(:)이 사용됩니다.

‘.’: 마침표(.)가 사용됩니다.

‘,’: 쉼표(,)가 사용됩니다.

‘ ’: 공백()이 사용됩니다.

***BLANK:** 공백()이 사용됩니다.

REPLACE

동일한 이름의 프로그램 또는 SQL 패키지가 동일한 라이브러리에 있을 때 새로운 프로그램 또는 SQL 패키지가 작성되는지를 지정합니다. 이 매개변수 값은 CRTFTNPGM 명령으로 전달됩니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 REPLACE 매개변수에 나와 있습니다.

***YES:** 새로운 패키지나 SQL 패키지가 작성되고, 지정된 라이브러리 내에서 같은 이름 및 유형을 가지는 기존의 프로그램 또는 SQL 패키지는 QRPLOBJ로 이동됩니다.

***NO:** 새로운 프로그램 또는 SQL 패키지는 같은 이름과 유형의 오브젝트가 이미 지정된 라이브러리에 존재하는 경우 작성되지 않습니다.

RDB

SQL 패키지 오브젝트가 작성되는 관계형 데이터베이스의 이름을 지정합니다.

***LOCAL:** 프로그램은 분산 SQL 프로그램으로서 작성됩니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다. SQL 패키지 오브젝트는 사전컴파일 프로세스의 일부분으로서 작성되지 않습니다. 구조화 조회 언어 패키지 작성(CRTSQLPKG) 명령이 사용될 수 있습니다.

relational-database-name: 새로운 SQL 패키지 오브젝트가 작성될 때 관계형 데이터베이스의 이름을 지정하십시오. 로컬 관계형 데이터베이스명이 지정되면 작성된 프로그램은 계속 분산 SQL 프로그램입니다. SQL문이 로컬 데이터베이스에 액세스하게 됩니다.

***NONE:** SQL 패키지 오브젝트가 작성되지 않습니다. 프로그램 오브젝트는 분산 프로그램이 아니며 CRTSQLPKG(구조화 조회 언어 패키지 작성) 명령은 사용될 수 없습니다.

USER

변환 시작시 리모트 시스템으로 송신되는 사용자명을 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***CURRENT:** 현재 작업이 실행 중인 사용자 프로파일이 사용됩니다.

user-name: 어플리케이션 서버 작업에 사용되는 사용자명을 지정하십시오.

PASSWORD

리모트 시스템에 사용될 암호를 지정합니다. 이 매개변수는 RDB가 지정된 경우에만 유효합니다.

***NONE**: 암호가 송신되지 않습니다. 이 값이 지정되면 USER(*CURRENT) 또한 지정되어야 합니다.

password: USER 매개변수에 지정된 사용자명의 암호를 지정하십시오.

RDBCNNMTH

CONNECT문에 사용되는 의미를 지정합니다. 자세한 정보는 SQL 참조서에서 CONNECT(유형 1) 및 CONNECT(유형 2)를 참조하십시오.

***DUW**: CONNECT(유형 2) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 추가 관계형 데이터베이스에 대한 연속된 CONNECT문이 이전의 연결을 단절시키지는 않습니다.

***RUW**: CONNECT(유형 1) 의미는 분산 작업 단위를 지원하는 데 사용됩니다. 연속된 CONNECT문은 새로운 연결이 설정되기 전에 이전의 연결을 단절시킵니다.

DFTRDBCOL

표, 뷰, 색인 및 SQL 패키지의 규정되지 않은 이름에 사용되는 스키마명을 지정합니다. 이 매개변수는 정적 SQL문에만 적용됩니다.

***NONE**: OPTION 매개변수에 정의된 명명 규칙이 사용됩니다.

schema-name: 스키마 ID의 이름을 지정하십시오. 이 값은 OPTION 매개변수에 지정된 명명 규칙 대신 사용됩니다.

SQLPKG

이 명령의 RDB 매개변수에 지정된 관계형 데이터베이스에 작성된 SQL 패키지의 규정된 이름을 지정합니다.

가능한 라이브러리 값은 다음과 같습니다.

***PGMLIB**: 패키지가 프로그램을 포함하는 라이브러리와 같은 이름의 라이브러리에 작성됩니다.

library-name: 패키지가 작성된 라이브러리의 이름을 지정하십시오.

***PGM**: 패키지의 이름이 프로그램명과 같습니다.

package-name: RDB 매개변수에 지정된 리모트 데이터베이스에 작성된 패키지의 이름을 지정하십시오.

SAAFLAG

IBM SQL 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 IBM SQL 구문을

따르고 있는지 확인하기 위해 SQL문을 플래그합니다. IBM SQL 구문을 생성하는 IBM 데이터베이스에 대한 정보는 *DRDA IBM SQL Reference, SC26-3255-00* 책을 참조하십시오.

***NOFLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 체크하지 않습니다.

***FLAG:** 사전컴파일러가 SQL문이 IBM SQL 구문을 따르는지 알아보기 위해 체크합니다.

FLAGSTD

미국 표준 협회(ANSI) 플래그 기능을 지정합니다. 이 매개변수는 SQL문이 다음과 같은 표준을 따르고 있는지 확인하기 위해 SQL문을 플래그합니다.

ANSI X3.135-1992 항목
ISO 9075-1992 항목
FIPS 127.2 항목

***NONE:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크하지 않습니다.

***ANS:** 사전컴파일러가 SQL문이 ANSI 표준에 따르는지 알아보기 위해 체크합니다.

PRTFILE

리스트가 전달되는 프린터 파일의 규정화명을 지정합니다. 파일의 최소 레코드 길이는 132바이트여야 하며, 그렇지 않으면 정보가 유실됩니다.

프린터 파일명은 다음의 라이브러리 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

QSYSPRT: 파일명이 지정되지 않으면 사전컴파일러 인쇄 출력이 IBM 제공 프린터 파일 QSYSPRT로 전달됩니다.

printer-file-name: 사전컴파일러 인쇄 출력이 전달될 프린터 파일의 이름을 지정하십시오.

SRTSEQ

SQL문의 스트링 비교에 사용될 정렬 순서표를 지정합니다.

주: *HEX는 어플리케이션 서버가 iSeries 서버에 없거나 릴리스 레벨이 V2R3M0 이전인 분산 어플리케이션의 매개변수에 대해서만 지정되어야 합니다.

***JOB:** 작업에 대한 SRTSEQ 값이 사전컴파일 중에 검색됩니다.

***JOBRUN:** 작업에 대한 SRTSEQ 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 SRTSEQ(*JOBRUN)는 LANGID(*JOBRUN)가 지정될 때만 유효합니다.

***LANGIDUNQ:** LANGID 매개변수에 지정된 고유 가중치 정렬표가 사용됩니다.

***LANGIDSHR:** LANGID 매개변수에 지정된 공유 가중치 정렬표가 사용됩니다.

***HEX:** 정렬 순서표가 사용되지 않습니다. 문자의 16진수 값이 정렬 순서를 결정하기 위해 사용됩니다.

정렬 순서표의 이름은 다음 값 중 하나로 규정될 수 있습니다.

***LIBL:** 일치되는 첫 번째 항목이 발견될 때까지 작업의 라이브러리 리스트에 있는 모든 라이브러리가 탐색됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 탐색됩니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없으면 QGPL 라이브러리가 사용됩니다.

library-name: 탐색될 라이브러리의 이름을 지정하십시오.

table-name: 사용될 정렬 순서표의 이름을 지정하십시오.

LANGID

SRTSEQ(*LANGIDUNQ) 또는 SRTSEQ(*LANGIDSHR) 지정시 사용될 언어 ID를 지정합니다.

***JOB:** 작업에 대한 LANGID 값이 사전컴파일 중에 검색됩니다.

***JOBRUN:** 작업에 대한 LANGID 값이 프로그램 실행시 검색됩니다. 분산 어플리케이션의 경우 LANGID(*JOBRUN)는 SRTSEQ(*JOBRUN)가 지정될 때만 유효합니다.

language-id: 프로그램에 사용될 언어 ID를 지정하십시오.

USRPRF

정적 SQL문에서 각 오브젝트에 대해 프로그램 오브젝트가 가지는 권한을 포함하여 컴파일된 프로그램 오브젝트를 실행할 때 사용되는 사용자 프로파일을 지정합니다. 프로그램 소유자 또는 프로그램 사용자의 프로파일이 프로그램 오브젝트가 사용할 수 있는 오브젝트를 제어하는 데 사용됩니다.

***NAMING:** 사용자 프로파일은 명명 규칙에 의해 판별됩니다. 명명 규칙이 *SQL이면 USRPRF(*OWNER)가 사용됩니다. 명명 규칙이 *SYS이면 USRPRF(*USER)가 사용됩니다.

***USER:** 프로그램 오브젝트를 실행 중인 사용자의 프로파일이 사용됩니다.

***OWNER:** 프로그램 소유자와 프로그램 사용자 모두의 사용자 프로파일이 프로그램 실행시 사용됩니다.

DYNUSRPRF

SQL문에 사용되는 사용자 프로파일을 지정합니다.

***USER:** 로컬 동적 SQL문은 프로그램 사용자의 파일하에서 실행됩니다. 분산 동적 SQL문은 어플리케이션 서버 작업의 사용자 프로파일하에서 실행됩니다.

***OWNER:** 로컬 동적 SQL문은 프로그램 소유자의 사용자 프로파일하에서 실행됩니다. 분산 동적 SQL문은 SQL 패키지 소유자의 사용자 프로파일하에서 실행됩니다.

TOSRCFILE

SQL 사전컴파일러에 의해 처리된 출력 소스 멤버가 있게 될 소스 파일의 규정화명을 지정합니다. 지정된 소스 파일을 찾을 수 없는 경우에는 이를 작성합니다. 출력 멤버는 SRCMBR 매개변수용으로 지정한 이름과 동일한 이름을 갖게 됩니다.

가능한 라이브러리 값은 다음과 같습니다.

QTEMP: 라이브러리 QTEMP가 사용될 것입니다.

***LIBL:** 작업 라이브러리 리스트가 지정 파일에 대해 탐색됩니다. 만일 라이브러리 리스트의 어떠한 라이브러리에서도 이 파일이 없는 경우에 파일은 현재 라이브러리에서 작성됩니다.

***CURLIB:** 작업에 대한 현재 라이브러리가 사용될 것입니다. 작업에 대한 현재 라이브러리로 지정된 라이브러리가 없는 경우 QGPL 라이브러리가 사용될 것입니다.

library-name: 출력 소스 파일이 작성될 라이브러리명을 지정하십시오.

QSQLTEMP: 소스 파일 QSQLTEMP가 사용될 것입니다.

source-file-name: 출력 소스 멤버가 들어 있는 소스 파일의 이름을 지정하십시오.

TEXT

LANGID를 간략하게 설명하는 텍스트를 지정합니다. 이 매개변수에 대한 자세한 정보는 Information Center의 CL 참조서 절에 있는 TEXT 매개변수에 나와 있습니다.

***SRCMBRTXT:** 텍스트는 FORTRAN 프로그램을 작성하는 데 사용되는 소스 파일 멤버에서 취해집니다. STRSEU(소스 입력 유틸리티 시작) 명령을 사용하거나 ADDPFM(실제 파일 멤버 추가) 또는 CHGPFM(실제 파일 멤버 변경) 명령을 사용하여 데이터베이스 소스 멤버에 텍스트를 추가하거나 데이터베이스 소스 멤버의 텍스트를 변경할 수 있습니다. 소스 파일이 인라인 파일이나 장치 파일일 경우 텍스트는 공백입니다.

***BLANK:** 텍스트가 지정되지 않습니다.

'description': 50자 미만의 텍스트를 어포스트로피로 묶어 지정하십시오.

CRTSQLFTN 명령의 목적

```
CRTSQLFTN PAYROLL TEXT('Payroll Program')
```

이 명령은 소스문을 컴파일하는 SQL 컴파일러를 수행하여 변경된 소스문을 라이브러리 QTEMP에 있는 파일 QSQLTEMP의 멤버 PAYROLL에 저장합니다. FORTRAN 컴파일러는 SQL 사전컴파일러에서 작성된 소스 멤버를 사용하여 현재 라이브러리에 프로그램 PAYROLL을 작성하기 위해 호출됩니다.

부록 D. FORTRAN 어플리케이션에서 SQL문 코딩

이 부록에서는 FORTRAN 프로그램에서 SQL문을 삽입할 때 고유한 어플리케이션 및 코딩 요구사항을 설명합니다. 호스트 변수에 대한 요구사항도 정의됩니다.

자세한 내용은 다음 절을 참조하십시오.

- 『FORTRAN 어플리케이션에서 SQL 통신 영역 정의』
- 351 페이지의 『FORTRAN 어플리케이션에서 SQL 설명자 영역 정의』
- 351 페이지의 『FORTRAN 어플리케이션에서 SQL문 삽입』
- 354 페이지의 『FORTRAN 어플리케이션에서 호스트 변수 사용』
- 356 페이지의 『동일한 SQL 및 FORTRAN 자료 유형 판별』
- 357 페이지의 『FORTRAN 어플리케이션에서 인디케이터 변수 사용』

주: 코드 예에 관한 정보는 x 페이지의 『코드 면책사항 정보』를 참조하십시오.

FORTRAN 어플리케이션에서 SQL 통신 영역 정의

SQL문이 들어 있는 FORTRAN 프로그램에는 다음 중 하나 또는 둘 모두가 포함되어 있어야 합니다.

- INTEGER로 선언된 SQLCOD 변수
- CHARACTER*5로 선언된 SQLSTA(또는 SQLSTATE) 변수

또는

- SQLCA(SQLCOD 및 SQLSTA 변수가 들어 있는)

SQLCOD 값은 각 SQL문이 실행된 후, 데이터베이스 관리자에 의해 설정됩니다. 어플리케이션은 SQLCOD 값을 체크하여 최종 SQL문이 제대로 수행되었는지의 여부를 판별할 수 있습니다.

SQLCA는 직접 또는 SQL INCLUDE문을 사용하여 FORTRAN 프로그램에서 작성될 수 있습니다. SQL INCLUDE문을 사용할 때에는 다음의 표준 선언 부분이 포함되어야 합니다.

```
EXEC SQL INCLUDE SQLCA
```

SQLCA에 대해 포함된 FORTRAN 소스문은 다음과 같습니다.

```
*  
*   The SQL communications area  
*  
CHARACTER SQLCA(136)  
CHARACTER SQLCAID*8
```

```

INTEGER*4 SQLCABC
INTEGER*4 SQLCODE
INTEGER*2 SQLERRML
CHARACTER SQLERRMC*70
CHARACTER SQLERRP*8
INTEGER*4 SQLERRD(6)
CHARACTER SQLWARN*11
CHARACTER SQLSTATE*5
EQUIVALENCE (SQLCA( 1), SQLCAID)
EQUIVALENCE (SQLCA( 9), SQLCABC)
EQUIVALENCE (SQLCA( 13), SQLCODE)
EQUIVALENCE (SQLCA( 17), SQLERRML)
EQUIVALENCE (SQLCA( 19), SQLERRMC)
EQUIVALENCE (SQLCA( 89), SQLERRP)
EQUIVALENCE (SQLCA( 97), SQLERRD)
EQUIVALENCE (SQLCA(121), SQLWARN)
EQUIVALENCE (SQLCA(132), SQLSTATE)

```

*

```

INTEGER*4 SQLCOD
C      SQLERR(6)
INTEGER*2 SQLTXL
CHARACTER SQLERP*8,
C      SQLWRN(0:7)*1,
C      SQLWRX(1:3)*1,
C      SQLTXT*70,
C      SQLSTT*5,
C      SQLWRNWK*8,
C      SQLWRXWK*3,
C      SQLERRWK*24,
C      SQLERRDWK*24
EQUIVALENCE (SQLWRN(1), SQLWRNWK)
EQUIVALENCE (SQLWRX(1), SQLWRXWK)
EQUIVALENCE (SQLCA(97), SQLERRDWK)
EQUIVALENCE (SQLERR(1), SQLERRWK)
COMMON /SQLCA1/SQLCOD,SQLERR,SQLTXL
COMMON /SQLCA2/SQLERP,SQLWRN,SQLTXT,SQLWRX,SQLSTT

```

SQLSTATE에 대한 선언이 프로그램에서 발견되고 SQLCA가 컴파일러에 의해 제공 될 경우 SQLSTATE는 SQLSTOTE로 대체됩니다. 다른 IBM SQL 구현과의 호환성이 주요 고려사항이 아니면, 프로그램에서 FORTRAN 변수 SQLCOD, SQLSTA 또는 SQLSTATE를 코딩하여 SQLSTA를 포함시키는 것이 좋습니다. 이 방법은 성능은 향상시키지만, 호환되는 SQLCA를 생성하지는 못합니다.

SQLCA에 대한 자세한 정보는 SQL 참조서의 SQL 통신 영역을 참조하십시오.

SQLCOD, SQLSTA, SQLSTATE 및 SQLCA 변수는 첫 번째의 실행 가능한 SQL 문 앞에 위치해야 합니다. 프로그램에서 실행될 수 있는 모든 SQL문은 SQLCOD, SQLSTA, SQLSTATE 및 SQLCA 변수 선언의 범위 내에 있어야 합니다.

프로그램에서 실행될 수 있는 모든 SQL문은 SQLCOD 변수 또는 SQLCA 변수 선언 부분의 범위 내에 있어야 합니다.

FORTRAN 어플리케이션에서 SQL 설명자 영역 정의

다음 명령문은 SQLDA를 필요로 합니다.

EXECUTE...USING DESCRIPTOR *descriptor-name*

FETCH...USING DESCRIPTOR *descriptor-name*

OPEN...USING DESCRIPTOR *descriptor-name*

CALL...USING DESCRIPTOR *descriptor-name*

DESCRIBE *statement-name* INTO *descriptor-name*

DESCRIBE TABLE *host-variable* INTO *descriptor-name*

PREPARE *statement-name* INTO *descriptor-name*

SQLCA와는 달리 프로그램에 두 개 이상의 SQLDA가 있을 수 있으며 SQLDA는 유효한 모든 이름을 가질 수 있습니다.

동적 SQL은 iSeries용 DB2 UDB 프로그램 개념 정보의 동적 SQL 어플리케이션에 설명되어 있는 첨단 프로그래밍 기술입니다. 동적 SQL을 사용하면 프로그램 수행 중 사용자 프로그램이 SQL문을 개발 및 수행할 수 있습니다. 동적으로 수행하는 가변 SELECT 리스트(조회)의 한 부분으로 리턴될 자료의 리스트임)가 있는 SELECT문은 SQL 설명자 영역(SQLDA)을 필요로 합니다. 그 이유는 사용자가 SELECT의 결과를 수신하기 위해 지정된 변수의 수와 유형을 미리 알 수 없기 때문입니다. SQLDA가 FORTRAN에서 지원되지 않는 포인터 변수를 사용하므로 INCLUDE SQLDA문은 FORTRAN 프로그램에 지정될 수 없습니다. SQLDA가 C, COBOL, PL/I 또는 ILE RPG 프로그램에서 설정되어 FORTRAN 프로그램으로 전달되지 않는 한, SQLDA를 사용할 수 없습니다.

SQLDA에 대한 자세한 정보는 SQL 참조서의 SQL 설명자 영역을 참조하십시오.

행 기억영역을 사용하여 복수 행 FETCH문에 SQLDA를 코딩하는 것은 각 FETCH문에 있는 복수 행을 검색하기 위한 기술을 제공합니다. 이 기법은 어플리케이션에서 다수의 행을 읽을 경우 어플리케이션의 성능을 향상시킬 수 있습니다. FETCH문 사용에 대한 자세한 내용은 SQL 참조서를 참조하십시오.

FORTRAN 어플리케이션에서 SQL문 삽입

SQL문은 실행될 수 있는 명령문이 표시되는 모든 FORTRAN 프로그램에서 코딩될 수 있습니다. SQL문이 IF문 내에 있으면, 필요한 THEN 및 END IF문이 생성됩니다.

FORTRAN 프로그램의 각 SQL문은 EXEC SQL로 시작되어야 합니다. EXEC SQL 키워드는 모두 한 행에 표시되어야 하나 명령문의 나머지 부분은 같은 행 및 후속 행에 표시될 수 있습니다.

예:

FORTRAN 프로그램에 작성된 UPDATE 명령문은 다음과 같습니다.

```
EXEC SQL
C  UPDATE DEPARTMENT
C  SET MGRNO = :MGRNUM
C  WHERE DEPTNO = :INTDEPT
```

동일한 행에서 SQL문 다음에는 또 다른 SQL문이나 FORTRAN문이 올 수 없습니다.

FORTRAN에서는 명령문 내의 단어를 구분하기 위해 공백을 사용할 필요가 없으나 SQL 언어에서는 필요합니다. 삽입된 SQL 규칙은 한 개 이상의 공백을 분리문자로 사용하는 SQL 구문 규칙을 따릅니다.

자세한 내용은 다음 절을 참조하십시오.

- 『SQL을 사용하는 FORTRAN 어플리케이션에서 주석』
- 『SQL을 사용하는 FORTRAN 어플리케이션에서 행 디버그』
- 『SQL을 사용하는 FORTRAN 어플리케이션에서 SQL문의 연속』
- 353 페이지의 『SQL을 사용하는 FORTRAN 어플리케이션에서 코드 포함』
- 353 페이지의 『SQL을 사용하는 FORTRAN 어플리케이션에서 여백』
- 353 페이지의 『SQL을 사용하는 FORTRAN 어플리케이션에서 이름』
- 353 페이지의 『SQL을 사용하는 FORTRAN 어플리케이션에서 명령문 레이블』
- 354 페이지의 『SQL을 사용하는 FORTRAN 어플리케이션에서 WHENEVER문』
- 354 페이지의 『SQL 사전컴파일러에서 FORTRAN 컴파일시 옵션』

SQL을 사용하는 FORTRAN 어플리케이션에서 주석

SQL 주석(--) 외에도 FORTRAN 주석을 키워드 EXEC 및 SQL 사이를 제외하고 공백이 허용되는 모든 삽입 SQL문에 포함시킬 수 있습니다.

주석은 행의 끝까지 확장시킬 수 있습니다. 주석 행은 계속되는 SQL문의 행 사이에 표시될 수 있습니다. 문자 !는 문자 문맥이나 열 6에서 나타날 때를 제외하면 그 문장이 주석임을 나타냅니다.

SQL을 사용하는 FORTRAN 어플리케이션에서 행 디버그

디버그 명령문(1열에 'D' 또는 'd')이 들어 있는 행은 사전컴파일러에 의해 주석 행으로 간주됩니다.

SQL을 사용하는 FORTRAN 어플리케이션에서 SQL문의 연속

SQL문의 행 계속 규칙은 EXEC SQL이 한 행 내에 지정되어야 한다는 것을 제외하면 다른 FORTRAN문의 규칙과 동일합니다.

DBCS 자료가 들어 있는 상수는 계속되는 행의 73열에 SI 문자를 위치시키고 계속 행의 6열에 SO 문자를 위치시켜 복수 행에 걸쳐 계속될 수 있습니다.

이 예에서 SQL문에 G'<AABBCCDDEEFFGGHHIIJJKK>'의 유효한 그래픽 상수가 있습니다.

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
EXEC SQL SELECT * FROM GRAPHTAB WHERE GRAPHCOL = G'<AABBCC>
<DDEEFFGGHHIIJJKK>'
```

SQL을 사용하는 FORTRAN 어플리케이션에서 코드 포함

SQL문이나 FORTRAN문은 명령문이 삽입될 소스 코드 내의 한 지점에서 다음 SQL문을 삽입하여 포함될 수 있습니다.

```
EXEC SQL INCLUDE member-name
```

FORTRAN INCLUDE 컴파일러 지시문은 SQL문이나 SQL문에 사용될 FORTRAN 호스트 변수 선언을 포함시키는 데 사용할 수 없습니다.

SQL을 사용하는 FORTRAN 어플리케이션에서 여백

7-72 열 사이에 SQL문(EXEC SQL로 시작하는)을 작성하십시오.

SQL을 사용하는 FORTRAN 어플리케이션에서 이름

유효한 모든 FORTRAN 변수명은 호스트 변수에 사용될 수 있으며, 다음 제한사항을 따라야 합니다.

'SQ', 'SQL', 'RDI' 또는 'DSN'으로 시작하는 호스트 변수명이나 외부 항목명을 사용하지 마십시오. 이러한 이름들은 데이터베이스 관리자 예약어입니다.

호스트 변수를 식별하려면 다음 키워드를 사용하지 마십시오.

```
FUNCTION
IMPLICIT
PROGRAM
SUBROUTINE
```

SQL을 사용하는 FORTRAN 어플리케이션에서 명령문 레이블

실행가능 SQL문은 1-5 열에 지정된 그와 연관된 명령문 번호를 가질 수 있습니다. 그러나 프로그램 준비중에 레이블된 SQL문은 레이블이 있는 CONTINUE문을 코드가 실행하기 전에 생성하도록 합니다. 레이블된 SQL문은 DO 루프 내에서 최종 명령문이 될 수 없습니다. CONTINUE문이 실행될 수 있으므로 FORTRAN 프로그램에서 실행될 수 있는 첫 번째 명령문 앞에 발생하는 SQL문(예: INCLUDE 및 BEGIN DECLARE SECTION)은 레이블되어서는 안됩니다.

SQL을 사용하는 FORTRAN 어플리케이션에서 WHENEVER문

SQL WHENEVER문에서 GOTO절의 목표는 FORTRAN 소스의 레이블이어야 하며, 동일한 서브프로그램 내의 명령문을 참조해야 합니다. WHENEVER문은 동일한 서브프로그램 내의 SQL문에만 적용됩니다.

SQL 사전컴파일러에서 FORTRAN 컴파일시 옵션

FORTRAN PROCESS문은 FORTRAN 컴파일러에 컴파일시 옵션을 지정하는 데 사용할 수 있습니다. PROCESS문은 프로그램을 작성하기 위해 사전컴파일러에서 호출될 때 FORTRAN 컴파일러에서 인식되더라도, SQL 사전컴파일러 자체는 PROCESS문을 인식하지 못합니다.

FORTRAN 어플리케이션에서 호스트 변수 사용

SQL문에서 사용된 모든 호스트 변수는 명시적으로 선언되어야 합니다. 디폴트 입력을 통한 또는 IMPLICIT문에 의한 호스트 변수의 내재적 선언 부분은 제공되지 않습니다. SQL문에서 사용된 호스트 변수는 SQL문에서 호스트 변수를 처음 사용하기 전에 선언되어야 합니다.

호스트 변수 정의에 사용된 FORTRAN문 앞에는 BEGIN DECLARE SECTION문이 오고 뒤에는 END DECLARE SECTION문이 와야 합니다. BEGIN DECLARE SECTION 및 END DECLARE SECTION이 지정되면 SQL문에서 사용된 모든 호스트 변수 선언은 BEGIN DECLARE SECTION과 END DECLARE SECTION문 사이에 있어야 합니다.

주: LOB 및 ROWID 호스트 변수는 FORTRAN에서 지원되지 않습니다.

SQL문 내의 모든 호스트 변수 앞에는 콜론(:)이 와야 합니다.

호스트 변수가 다른 블록이나 프로시저에 내에 있어도 호스트 변수명은 프로그램 내에서 고유해야 합니다.

문자 호스트 변수 선언 부분은 문자 변수의 길이를 정의하는 표현식을 사용해서는 안됩니다. 문자 호스트 변수의 선언은 길이가 정의되어 있지 않아야 합니다(예: CHARACTER(*)).

호스트 변수를 사용하는 SQL문은 변수가 선언된 명령문의 범위 내에 있어야 합니다.

호스트 변수는 스칼라 변수이어야 하며, 배열의 요소(아래 첨자가 붙은 변수)는 될 수 없습니다.

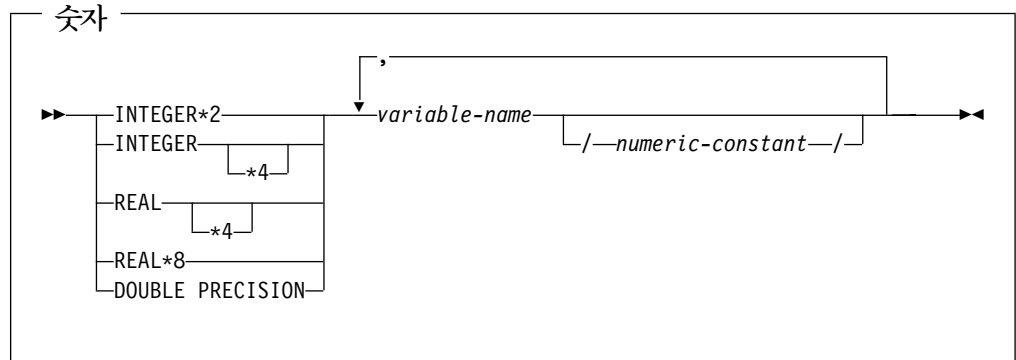
자세한 내용은 355 페이지의 『FORTRAN 어플리케이션에서 호스트 변수 선언』를 참조하십시오.

FORTRAN 어플리케이션에서 호스트 변수 선언

FORTRAN 사전컴파일러는 유효한 호스트 변수 선언 부분으로서 유효한 FORTRAN 선언의 서브세트만 인식합니다.

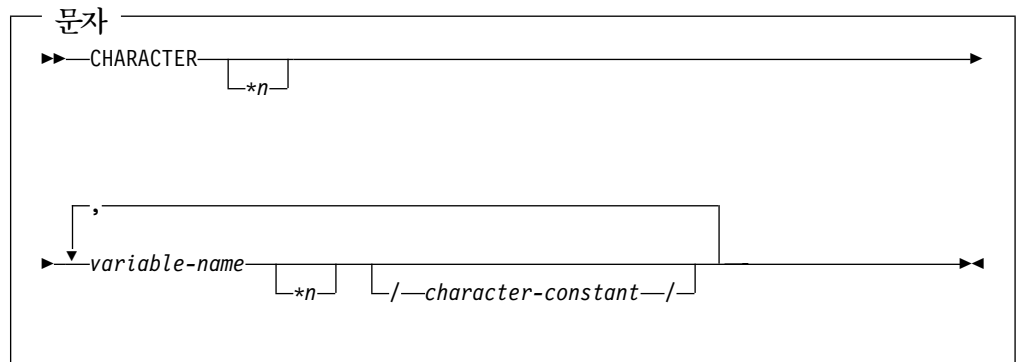
FORTRAN 어플리케이션에서 숫자 호스트 변수

다음 그림은 유효한 숫자 호스트 변수 선언 부분에 대한 구문을 표시합니다.



FORTRAN 어플리케이션에서 문자 호스트 변수

다음 그림에서는 유효한 문자 호스트 변수 선언 부분에 대한 구문을 보여줍니다.



주: n은 32766보다 크지 않은 상수이어야 합니다.

동일한 SQL 및 FORTRAN 자료 유형 판별

사전컴파일러는 다음 표에 있는 호스트 변수의 기본 SQLTYPE 및 SQLLEN을 판별합니다. 호스트 변수가 인디케이터 변수로 표시되면 SQLTYPE은 기본 SQLTYPE에 1을 더한 값이 됩니다.

표 12. 일반 SQL 자료 유형에 맵핑되는 FORTRAN 선언

FORTRAN 자료 유형	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
INTEGER*2	500	2	SMALLINT
INTEGER*4	496	4	INTEGER
REAL*4	480	4	FLOAT(단정밀도)
REAL*8	480	8	FLOAT(배정밀도)
CHARACTER*n	452	n	CHAR(n)

다음 표는 제공된 SQL 자료 유형과 같은 FORTRAN 자료 유형 판별에 사용할 수 있습니다.

표 13. 일반 FORTRAN 선언에 맵핑되는 SQL 자료 유형

SQL 자료 유형	동일한 FORTRAN	주석
SMALLINT	INTEGER*2	
INTEGER	INTEGER*4	
BIGINT	같은 것이 없습니다.	REAL*8 사용
DECIMAL(p,s) 또는 NUMERIC(p,s)	같은 것이 없습니다.	REAL*8 사용
FLOAT(단정밀도)	REAL*4	
FLOAT(배정밀도)	REAL*8	
CHAR(n)	CHARACTER*n	n은 1 - 32766 사이의 양의 정수입니다.
VARCHAR(n)	같은 것이 없습니다.	예측된 가장 큰 VARCHAR 값을 포함시키기에 충분히 큰 문자 호스트 변수를 사용하십시오.
GRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
VARGRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
DATE	CHARACTER*n	형식이 *USA, *JIS, *EUR 또는 *ISO인 경우 n은 최소한 10자이어야 합니다. 형식이 *YMD, *DMY 또는 *MDY인 경우 n은 최소한 8자이어야 합니다. 형식이 *JUL인 경우 n은 최소한 6자이어야 합니다.
TIME	CHARACTER*n	n은 최소한 6자여야 하며 초를 포함시키려면 n은 최소한 8자이어야 합니다.

표 13. 일반 FORTRAN 선언에 맵핑되는 SQL 자료 유형 (계속)

SQL 자료 유형	동일한 FORTRAN	주석
TIMESTAMP	CHARACTER*n	n은 최소한 19이어야 합니다. 전체 정밀도에 마이크로초를 포함시키려면 n은 26이어야 합니다. n이 26보다 작으면 마이크로초 부분에서 잘립니다.

자세한 내용은 『FORTRAN 변수 선언 및 사용에 대한 주』를 참조하십시오.

FORTRAN 변수 선언 및 사용에 대한 주

FORTRAN에서 소수점이 있는 자릿수의 스트링은 실수로 해석됩니다. SQL문에서 그러한 스트링은 10진 상수로 해석됩니다. 그러므로 SQL문에 실수(부동 소수점) 상수를 지정할 때에는 지수 표기를 사용하십시오.

FORTRAN에서 길이가 8바이트인 실수(부동 소수점) 상수는 지수 인디케이터로서 D를 사용합니다(예: 3.14159D+04). SQL문에서 8바이트 부동 소수점 상수는 E를 사용해야 합니다(예를 들어, 3.14159E+04).

FORTRAN 어플리케이션에서 인디케이터 변수 사용

인디케이터 변수는 2바이트 정수(INTEGER*2)입니다. 검색시 연관 호스트 변수에 널(null)이 할당되었는지 여부를 나타내기 위해 인디케이터 변수를 사용합니다. 음의 인디케이터 변수는 열에 할당할 때 널이 할당되어야 함을 나타내기 위해 사용됩니다.

자세한 정보는 SQL 참조서의 인디케이터 변수 주제를 참조하십시오.

인디케이터 변수는 호스트 변수와 동일한 방식으로 선언됩니다. 프로그래머는 적절한 방식으로 그 두 가지 선언 부분을 혼합할 수 있습니다.

예:

다음 명령문을 보십시오.

```
EXEC SQL FETCH CLS_CURSOR INTO :CLS_CD,
C                               :DAY :DAY_IND,
C                               :BGN :BGN_IND,
C                               :ENDCLS :ENDCLS_IND
```

이 경우 변수는 다음과 같이 선언될 수 있습니다.

```
EXEC SQL BEGIN DECLARE SECTION
CHARACTER*7 CLS_CD
INTEGER*2 DAY
CHARACTER*8 BGN, ENDCLS
INTEGER*2 DAY_IND, BGN_IND, ENDCLS_IND
EXEC SQL END DECLARE SECTION
```


색인

[가]

개념

지정 규칙, 호스트 언어로 SQL 사용 3

호스트 언어로 SQL, 사용

리턴 코드 취급 8

호스트 구조 7

호스트 변수 1

SQLCODE 8

SQLSTATE 8

SQLSTATE 8

경고

음수 SQLCODE에 대한 테스트 9

계속

C 18

COBOL 53

C++ 18

FORTRAN 352

iSeries용 ILE RPG 117

iSeries용 RPG 102

PL/I 82

구조 매개변수 전달

iSeries용 RPG 111

PL/I 97

규칙

지정 규칙 5, 6

할당 3

호스트 변수, 사용 5

호스트 언어로 SQL, 사용 1

그래픽 호스트 변수

C 25

COBOL 59

C++ 25

iSeries용 ILE RPG 127

[나]

날짜 지정 규칙

호스트 변수, 사용 6

널(null)

인디케이터 변수에 의해 설정 8

C에서 사용 19

C++에서 사용 19

널(null)값, SQL

REXX의 널(null)값과 대조됨 140

[다]

대시

COBOL 55

COBOL 호스트 변수 내의 55

대체 고려사항

삽입된 SQL을 사용하는 프로그램 실행

159

데이터베이스 파일 대체(OVRDBF) 명령

159, 160

iSeries용 RPG /COPY로 사용 106

동적 SQL

COBOL에서의 코딩 51

C에서의 코딩 15

C++에서의 코딩 15

FETCH, 복수 행

iSeries용 ILE RPG 132

FORTRAN에서 코딩 351

iSeries용 ILE RPG에 코딩 115

iSeries용 RPG에 코딩 100

PL/I에서의 코딩 80

디버그 행

COBOL 53

[라]

리스트

사전점파일러에서의 출력 147

리턴 코드

삽입된 SQL을 사용하는 프로그램 실행

160

취급

일반 8

[마]

매개변수 전달

차이

iSeries용 RPG 111

PL/I 97

메세지

오류 및 경고 메세지 분석 156

컴파일시 오류 및 경고 메세지 156

명령문 9, 163, 168, 176, 181, 187, 193

샘플 프로그램 161

이 있는 프로그램의 준비 및 실행 145

INSERT

지정 조작 3

SELECT INTO

열 값 3

SQL에서의 호스트 변수, 사용 1

UPDATE

지정 조작 3

WHENEVER 20, 54, 83, 354

예외 조건 처리 9

iSeries용 ILE RPG 118

iSeries용 RPG 103

WHENEVER SQLERROR 9

명령문 레이블

C 19

COBOL 54

C++에서 19

FORTRAN 프로그램의 요구사항 353

iSeries용 ILE RPG 요구사항 118

iSeries용 RPG 103

명령문명

DESCRIBE에서

REXX에서 136

명령(CL) 348

데이터베이스 파일 대체(OVRDBF) 106,

159, 160

모듈 표시(DSPMOD) 159

서비스 프로그램 표시

(DSPSRVPGM) 159

소스 실제 파일 작성(CRTSRCPF) 명령

148

프로그램 참조 표시(DSPPGMREF) 158

프로그램 표시(DSPPGM) 159

CRTSQLCBLI(SQL ILE COBOL 작

성) 329

iSeries용 SQL ILE C 작성

(CRTSQLCI) 329

SQL COBOL 작성(CRTSQLCBL) 329

SQL C++ 작성(CRTSQLCPPI) 329

명명(CL) (계속)

- SQL ILE/RPG 작성
(CRTSQLRPGI) 329
- SQL PL/I 작성(CRTSQLPLI) 329
- SQL RPG 작성(CRTSQLRPG) 329
- SQL 정보 인쇄(PRTSQLINF) 159

명명 규칙

- C 19
- COBOL 54
- C++ 19
- FORTRAN 353
- iSeries용 ILE RPG 118
- iSeries용 RPG 103
- PL/I 83
- REXX 140

모듈 표시(DSPMOD) 159

문자 호스트 변수

- C 21
- COBOL 58
- C++ 21
- FORTRAN 355
- iSeries용 ILE RPG 120, 127
- iSeries용 RPG 104, 107
- PL/I 84

문제점 취급 8

[바]

바인딩 157

발생 자료 구조

- iSeries용 ILE RPG 121
- iSeries용 RPG 105

번호

- 순서
 - COBOL 54
 - iSeries용 ILE RPG 118
 - iSeries용 RPG 103

변수 21, 46

- 널(null) 설정에 사용 8
- 인디케이터 6
- 호스트

REXX 141

호스트 구조와 함께 인디케이터 사용, 예 7

부동 소수점 소스 변수

COBOL 57

부동 소수점 숫자 5

분배된 자료 관리(DDM) 159

분산 자료 관리(DDM)

- 고려사항 159
- 삽입된 SQL을 사용하는 프로그램 실행 159

[사]

사전처리기

- SQL C 프로그램과 함께 사용 20
- SQL C++ 프로그램으로 사용 20
- SQL과 함께 147

사전컴파일러

- 경고 156
- 기본 처리 145
- 다른 사전처리기 147
- 레코드 번호 150

소스 레코드 150

소스 파일

- 여백 146
- CCSID 147
- DBCS 상수가 들어 있는 147

순서 번호 150

에서의 출력

리스트 147

샘플 149

임시 소스 파일 멤버 148

오류 156

입력 146

자세한 진단 146

진단 148

참조열 152

컴파일러에 전달된 매개변수 153

포함 파일

CCSID 147

표시

옵션 159

2차 입력 146

사전컴파일러 매개변수

리스트에 표시된 147

컴파일러에 전달된 매개변수 153

DATFMT 119, 126

DATSEP 119, 126

INCFILE 146

MARGINS 82, 146, 156

C 19

C++ 19

OBJ 148

OBJTYPE(*MODULE) 154

사전컴파일러 매개변수 (계속)

- OBJTYPE(*PGM) 154
- OBJTYPE(*SRVPGM) 154
- OPTION(*APOST) 54
- OPTION(*CNULQRD) 22, 26
- OPTION(*CVTDT) 125
- OPTION(*NOCNULQRD) 23, 26
- OPTION(*NOGEN) 153, 154
- OPTION(*NOSEQSRC) 118
- OPTION(*SEQSRC) 103
- OPTION(*QUOTE) 54
- OPTION(*SEQSRC) 118
- OPTION(*SOURCE) 146
- OPTION(*XREF) 146, 148
- OUTPUT 146
- PGM 148
- PRTFILE 147
- RDB
 - 사전컴파일러에 대한 효과 146
- TIMFMT 119, 126
- TIMSEP 119, 126
- *CVTDT 125
- *NOCVTDT 125, 126

사전컴파일러 명령

설명 153

CRTSQLCBL 153

CRTSQLCBLI 154

CRTSQLCI 19, 22, 23, 26, 154

CRTSQLCPPI 19, 22, 23, 26, 154

CRTSQLFTN 348

CRTSQLPLI 82, 153

CRTSQLRPG 153

CRTSQLRPGI 154

사전컴파일러 파일

QSQLTEMP 148

QSQLTEMP1 148

사전컴파일러 오류 메시지

리스트에 표시된 148

삼중 문자

C 20

C++ 20

삽입된 SQL

사전컴파일러 145

프로그램 실행 159

C 17

COBOL 52

C++ 17

FORTRAN 351

삽입된 SQL (계속)
 ILE RPG 116
 iSeries용 RPG 101
 PL/I 81
 새로운 릴리스
 고려사항 159
 샘플 프로그램
 보고서 197
 iSeries용 DB2 UDB 명령문, 사용 161
 SQL문
 COBOL 168
 ILE C 163
 ILE COBOL 168
 iSeries용 ILE RPG 프로그램 187
 iSeries용 RPG 181
 PL/I 176
 REXX 193
 샘플 프로그램에 의해 작성된 보고서 197
 샘플 프로그램으로 보고서 작성 197
 서브필드
 iSeries용 ILE RPG 120
 iSeries용 RPG 104
 서비스 프로그램 표시(DSPSRVPGM) 159
 설명자명
 REXX에서 136
 소스 실제 파일 작성(CRTSRCPF) 명령
 사전컴파일 사용 148
 소스 파일
 멤버, 임시
 사전컴파일러에서의 출력 148
 사전컴파일러에 대한 입력 146
 사전컴파일용 임시 148
 여백 146
 포함 파일 146
 CCSID 147
 COBOL에서의 복수 소스 54
 DBCS 상수가 들어 있는 147
 수행
 삽입된 SQL로 프로그램
 대체 고려사항 159
 리턴 코드 160
 명령어 159
 DDM 고려사항 159
 프로그램 159
 순서 번호
 COBOL 54
 iSeries용 ILE RPG 프로그램 118
 iSeries용 RPG 프로그램 103

숫자 지정 규칙
 호스트 변수, 사용 5
 숫자 호스트 변수
 C 21
 COBOL 55
 C++ 21
 FORTRAN 355
 iSeries용 ILE RPG 127
 iSeries용 RPG 107
 PL/I 84
 스트링 할당
 호스트 변수 사용 규칙 4
 시간 지정 규칙
 호스트 변수, 사용 6
 시간소인 지정 규칙
 호스트 변수, 사용 6

[아]
 액세스 계획 157
 어플리케이션
 바인딩 157
 어플리케이션 계획 157
 어플리케이션 프로그램
 컴파일링, ILE 154
 컴파일링, 비ILE 153
 SQLCA(SQL 통신 영역)
 C 13
 COBOL 49
 C++ 13
 FORTRAN 349
 iSeries용 ILE RPG 114
 iSeries용 RPG 100
 PL/I 79
 SQLDA
 C 15
 COBOL 51
 C++ 15
 FORTRAN 351
 iSeries용 ILE RPG 115
 iSeries용 RPG 100
 PL/I 80
 SQL문 코딩
 C 13, 49
 COBOL 49, 79
 C++ 13
 FORTRAN 349, 359
 iSeries용 ILE RPG 113, 133

어플리케이션 프로그램 (계속)
 SQL문 코딩 (계속)
 iSeries용 RPG 99, 112
 PL/I 79, 97
 어플리케이션 프로시저어
 SQL문 코딩
 REXX 135
 언어, 호스트
 개념과 규칙 1
 여백
 C 19
 COBOL 53
 C++ 19
 FORTRAN 353
 PL/I 82
 REXX 140
 예 7, 8
 변수 선언 부분 78
 사전컴파일러에서 출력, COBOL 149
 COBOL, UPDATE문 52
 iSeries용 RPG
 변수 선언 110
 SQL문에서의 호스트 변수 1
 예외 조건 9
 오류 리턴 코드, 취급
 일반 8
 외부 파일 설명
 호스트 구조 배열
 COBOL 73
 iSeries용 ILE RPG 126
 iSeries용 RPG 107
 C 42
 COBOL 73
 C++ 42
 iSeries용 ILE RPG 125
 iSeries용 RPG 106
 PL/I 93
 위치 지정자
 LOB
 C 30
 COBOL 62
 C++ 30
 iSeries용 ILE RPG 123
 PL/I 86
 인디케이터 구조 7
 인디케이터 배열
 C 36, 39
 COBOL 68, 72

인디케이터 배열 (계속)
 C++ 36, 39
 PL/I 90, 93
 인디케이터 변수
 널(null) 설정에 사용 8
 정의 6
 호스트 구조 7
 호스트 구조와 함께 사용, 예 7
 C 46
 COBOL 78
 C++ 46
 FORTRAN 357
 iSeries용 ILE RPG 131
 iSeries용 RPG 110
 PL/I 96
 REXX 144
 입시 소스 파일 멤버
 사전컴파일러에서의 출력 148

[자]

자료 유형
 동등한 COBOL 판별
 C 43
 COBOL 74
 C++ 43
 FORTRAN 356
 iSeries용 ILE RPG 127
 iSeries용 RPG 107
 PL/I 94
 REXX 142
 자료 항목
 iSeries용 ILE RPG 120
 iSeries용 RPG 104
 작은 따옴표
 C 46
 C++ 46
 전단어 2진 정수(INTEGER) 5
 정의
 바인딩 157
 액세스 계획 157
 인디케이터 구조 7
 인디케이터 변수 6
 호스트 구조 1
 호스트 변수 1
 주석
 C 18
 COBOL 53

주석 (계속)
 C++ 18
 FORTRAN 352
 iSeries용 ILE RPG 117
 iSeries용 RPG 102
 PL/I 82
 REXX 140
 지시문
 iSeries용 ILE RPG 프로그램 118
 지정 규칙
 날짜 6
 숫자 지정 5
 스트링 4
 시간 6
 시간소인 6
 호스트 변수
 사용 3
 집합
 C 21
 C++ 21

[차]

참조, 프로그램 158
 처리, 기본
 사전컴파일러 145
 취급
 예외 조건(WHENEVER문) 9
 오류 리턴 코드
 SQLCODE 및 SQLSTATE 8

[카]

컴파일 단계
 경고 156
 컴파일링
 경고 메시지 156
 어플리케이션 프로그램
 비ILE 153
 ILE 154
 오류 메시지 156
 컴파일시 경고 메시지 156
 C 프로그램 156
 COBOL 프로그램 156, 157
 C++ 프로그램 156
 PL/I 프로그램 156
 RPG 프로그램 156, 157
 컴파일시 오류 메시지 156

컴파일시 오류 메시지 (계속)
 C 프로그램 156
 COBOL 프로그램 156, 157
 C++ 프로그램 156
 PL/I 프로그램 156
 RPG 프로그램 156, 157
 컴파일시 옵션
 COBOL 54
 코드 포함
 FORTRAN 353
 iSeries용 RPG 102
 코드화 문자 세트 변환 ID(CCSID) 4
 코딩 예, SQL문
 COBOL 168
 ILE C 163
 ILE COBOL 168
 iSeries용 ILE RPG 프로그램 187
 iSeries용 RPG 181
 PL/I 176
 REXX 193
 REXX 어플리케이션 138
 코딩 요구사항
 C 프로그램
 계속 18
 널(null) 19
 명령문 레이블 19
 명명 규칙 19
 사전처리기 순서 20
 삼중 문자 20
 여백 19
 인디케이터 변수 46
 주석 18
 포함 코드 19
 호스트 변수 20
 WHENEVER문 20
 COBOL 프로그램
 계속 53
 디버그 행 53
 명령문 레이블 54
 명명 규칙 54
 복수 소스 프로그램 54
 여백 53
 인디케이터 변수 78
 주석 53
 컴파일시 옵션 54
 호스트 변수 54
 COBOL PROCESS문 54
 WHENEVER문 54

코딩 요구사항 (계속)

C++ 프로그램

- 계속 18
- 널(null) 19
- 명령문 레이블 19
- 명명 규칙 19
- 사전처리기 순서 20
- 삼중 문자 20
- 여백 19
- 주석 18
- 포함 코드 19
- 호스트 변수 20

WHENEVER문 20

FORTRAN 프로그램

- 계속 352
- 명령문 레이블 353
- 명명 규칙 353
- 여백 353
- 인디케이터 변수 357
- 주석 352
- 코드 포함 353
- 행 디버그 352
- 호스트 변수 354
- WHENEVER문 354

iSeries용 ILE RPG 프로그램

- 계속 117
- 명령문 레이블 118
- 명명 규칙 118
- 인디케이터 변수 131
- 주석 117
- 포함 코드 117
- 호스트 변수 118
- WHENEVER문 118

iSeries용 RPG 프로그램

- 계속 102
- 명령문 레이블 103
- 명명 규칙 103
- 인디케이터 변수 110
- 주석 102
- 코드 포함 102
- 호스트 변수 103
- WHENEVER문 103

PL/I 프로그램

- 계속 82
- 명명 규칙 83
- 여백 82
- 인디케이터 변수 96
- 주석 82

코딩 요구사항 (계속)

PL/I 프로그램 (계속)

- 포함 코드 82
- 호스트 변수 83
- WHENEVER문 83

콜론

- C 호스트 변수 20
- COBOL 호스트 변수 내의 55
- C++ 호스트 변수에서 20
- FORTRAN 호스트 변수에서 354
- iSeries용 ILE RPG 호스트 변수에서 119
- iSeries용 RPG 호스트 변수에서 103
- PL/I 호스트 변수 83

큰 따옴표

- C 46
- C++ 46

[타]

통합 언어 환경(ILE)

- 어플리케이션 컴파일링 154

[파]

파일 서술

외부

- C 42
- COBOL 73
- C++ 42
- iSeries용 ILE RPG 125
- iSeries용 RPG 106
- PL/I 93

호스트 구조 배열

- COBOL 73
- iSeries용 ILE RPG 126
- iSeries용 RPG 107

파일 참조 변수

LOB

- C 30
- COBOL 62
- C++ 30
- iSeries용 ILE RPG 124
- PL/I 86

포인터

- C 40
- C++ 40

포함 코드

- C 19

포함 코드 (계속)

- COBOL 53
- COBOL COPY문 53
- C++ 19
- iSeries용 ILE RPG 117
- PL/I 82

포함 파일

- 사전컴파일러에 대한 입력 146
- C 19
- CCSID 147
- COBOL 53
- C++ 19
- iSeries용 ILE RPG 117
- iSeries용 RPG 102
- PL/I 82

프로그램

삼입된 SQL로 실행

- 대체 고려사항 159
- 리턴 코드 160
- 명령어 159
- DDM 고려사항 159

샘플 161

샘플에 의해 작성된 보고서 197

어플리케이션 컴파일링

- 비ILE 153
- ILE 154

참조 158

SQL문

- COBOL 168
- ILE C 163
- ILE COBOL 168
- iSeries용 ILE RPG 프로그램 187
- iSeries용 RPG 181
- PL/I 176
- REXX 193

SQL문이 있는, 준비 및 실행 145

프로그램 참조 표시(DSPPGMREF) 명령 158

프로그램 표시(DSPPGM) 명령 159

프린터 파일 147

- CCSID 148

[하]

행 디버그

- FORTRAN 352

호스트 구조

- 널(null) 설정에 사용 8

호스트 구조 (계속)

배열 사용

- C 36
- COBOL 68, 73
- C++ 36
- iSeries용 ILE RPG 121
- iSeries용 RPG 105
- PL/I 91

인디케이터 배열

- C 36, 39
- COBOL 68, 72
- C++ 36, 39
- PL/I 90, 93

인디케이터 변수 사용, 예 7

정의 1

- C 32
- COBOL 64
- C++ 32
- iSeries용 ILE RPG 120
- iSeries용 RPG 104
- PL/I 88

호스트 구조 배열

배열 사용

- C 36
- COBOL 68
- C++ 36
- iSeries용 ILE RPG 121
- iSeries용 RPG 105
- PL/I 91

호스트 구조 인디케이터 배열

- C 36
- COBOL 68
- C++ 36
- PL/I 90

호스트 변수 21

그래픽

- C 25
- COBOL 59
- C++ 25
- iSeries용 ILE RPG 127

날짜/시간

- iSeries용 ILE RPG 127

문자

- C 21
- COBOL 58
- C++ 21
- FORTRAN 355
- iSeries용 ILE RPG 120, 127

호스트 변수 (계속)

문자 (계속)

- iSeries용 RPG 104, 107
- PL/I 84

부동 소수점

- COBOL 57

숫자

- C 21
- COBOL 55
- C++ 21
- FORTRAN 355
- iSeries용 ILE RPG 127
- iSeries용 RPG 107
- PL/I 84

스트링 할당, 규칙 4

외부 파일 설명

- C 42
- COBOL 73
- C++ 42
- iSeries용 ILE RPG 125
- iSeries용 RPG 106
- PL/I 93

정의 1

지정 규칙 3

BLOB

- C 29
- COBOL 61
- C++ 29
- iSeries용 ILE RPG 122
- PL/I 85

C 20

포인터 사용 40

CLOB

- C 29
- COBOL 61
- C++ 29
- iSeries용 ILE RPG 122
- PL/I 86

COBOL 54, 55

COBOL 프로그램에 대한 요구사항 54

C++ 20

포인터 사용 40

Datetime

- COBOL 63
- iSeries용 ILE RPG 119

DBCLOB

- C 29
- COBOL 61

호스트 변수 (계속)

DBCLOB (계속)

- C++ 29
- iSeries용 ILE RPG 122

FORTRAN 354

선언 355

iSeries용 ILE RPG 118

선언 119

iSeries용 ILE RPG 요구사항 118

iSeries용 RPG 103

선언 103

LOB

C 28

COBOL 60

C++ 28

iSeries용 ILE RPG 122

PL/I 85

PL/I 83

선언 83

PL/I 프로그램에 대한 요구사항 83

REXX 141

ROWID

C 32

COBOL 63

C++ 32

iSeries용 ILE RPG 125

PL/I 87

SQL문에서 일반 사용 1

SQL문, 사용

날짜, 시간 및 시간소인 지정에 대한 규칙 6

숫자 지정에 대한 규칙 5

SQL-varchar

C 21

C++ 21

varchar

C 21

C++ 21

호스트 언어

개념과 규칙 1

[숫자]

2배 전단어 2진 정수(BIGINT) 5

2진 반단어 정수(SMALLINT) 5

B

BEGIN DECLARE SECTION문

- C 20
 - COBOL 54
 - C++ 20
 - FORTRAN 354
 - iSeries용 ILE RPG 118
 - iSeries용 RPG 103
 - PL/I 83
- ### BLOB 호스트 변수
- C 29
 - COBOL 61
 - C++ 29
 - iSeries용 ILE RPG 122
 - PL/I 85

C

C 프로그램

- 계속 18
- 널(null) 19
- 동적 SQL 코딩 15
- 명령문 레이블 19
- 명명 규칙 19
- 사전처리기 순서 20
- 삼중 문자 20
- 여백 19
- 외부 파일 설명 42
- 위치 지정자
 - LOB 30
- 인디케이터 구조 46
- 인디케이터 변수 46
- 작은 따옴표 46
- 주석 18
- 컴파일러 매개변수 154
- 컴파일시 오류 및 경고 메시지 156
- 큰 따옴표 46
- 통합 요소 21
- 파일 참조 변수
 - LOB 30
- 포함 코드 19
- 호스트 구조
 - 배열, 선언 36
 - 선언 32
 - 인디케이터 구조 배열, 선언 39
 - 인디케이터 배열 36
- 호스트 변수 20

C 프로그램 (계속)

- 그래픽 25
 - 문자 21
 - 선언 21, 28
 - 숫자 21
 - 외부적으로 서술된 42
 - 포인터 사용 40
 - BLOB 29
 - CLOB 29
 - DBCLOB 29
 - LOB 28
 - ROWID 32
 - SQL-varchar 21
 - varchar 21
 - BEGIN/END DECLARE SECTION 20
 - INCLUDE문 19
 - SQL 자료 유형
 - 동등한 C를 판별 43
 - SQLCA, 선언 13
 - SQLCODE, 선언 13
 - SQLDA, 선언 15
 - SQLSTATE, 선언 13
 - SQL문 코딩 13, 49
 - typedef 41
 - WHENEVER문 20
 - #include 지시문 19
 - #pragma mapinc 지시문 42
- ### CCSID
- 사용에 대한 규칙 4
 - 소스 파일 147
 - 임시 소스 파일 148
 - 포함 파일 147
 - 프린터 파일 148
- ### CLOB 호스트 변수
- C 29
 - COBOL 61
 - C++ 29
 - iSeries용 ILE RPG 122
 - PL/I 86
- ### COBOL 프로그램 73
- 계속 53
 - 동적 SQL 코딩 51
 - 다버그 행 53
 - 명령문 레이블 54
 - 명명 규칙 54
 - 복수 소스 프로그램 54
 - 순서 번호 54
 - 여백 53

COBOL 프로그램 (계속)

- 외부 파일 설명 73
 - 위치 지정자
 - LOB 62
 - 인디케이터 구조 78
 - 인디케이터 변수 78
 - 주석 53
 - 컴파일러 매개변수 153
 - 컴파일시 오류 및 경고 메시지 157
 - 컴파일시 옵션 54
 - 파일 참조 변수
 - LOB 62
 - 포함 코드 53
 - 호스트 구조
 - 배열, 선언 68
 - 선언 64
 - 인디케이터 구조 배열, 선언 72
 - 인디케이터 배열 68
 - 호스트 변수 54
 - 그래픽 59
 - 문자 58
 - 부동 소수점 57
 - 선언 55, 60
 - 숫자 55
 - 외부적으로 서술된 73
 - BLOB 61
 - CLOB 61
 - DBCLOB 61
 - LOB 60
 - ROWID 63
 - BEGIN/END DECLARE SECTION 54
 - COBOL COPY문 53, 73
 - COBOL PROCESS문 54
 - Datetime 호스트 변수 63
 - FILLER 54
 - REDEFINES 77
 - SQL 168
 - SQL 자료 유형
 - 동등한 COBOL 판별 74
 - SQLCA, 선언 49
 - SQLCODE, 선언 49
 - SQLDA, 선언 51
 - SQLSTATE, 선언 49
 - SQL문 코딩 49, 79
 - SQL문이 있는 샘플 프로그램 168
 - WHENEVER문 54
- ### COPY문
- COBOL 53

COPY문 (계속)
 외부적으로 서술된 73

CRTSQCLB(LI(SQL ILE/COBOL 작성) 명령
 329

CRTSQCLC(iSeries용 SQL ILE C 작성) 명령
 329

CRTSQCLC(SQL ILE COBOL 작성) 명령
 329

CRTSQCLCPP(I(SQL C++ 작성) 명령 329

CRTSQCLC(SQL COBOL 작성) 명령 329

CRTSQCLFTN(SQL FORTRAN 작성) 명령
 348

CRTSQCLPLI(SQL PL/I 작성) 명령 329

CRTSQCLRPGI(SQL ILE/RPG 작성) 명령
 329

C++ 프로그램
 계속 18
 널(null) 19
 동적 SQL 코딩 15
 명령문 레이블 19
 명명 규칙 19
 사전처리기 순서 20
 삼중 문자 20
 여백 19
 외부 파일 설명 42
 위치 지정자
 LOB 30
 작은 따옴표 46
 주석 18
 컴파일러 매개변수 154
 컴파일시 오류 및 경고 메시지 156
 큰 따옴표 46
 파일 참조 변수
 LOB 30
 포함 코드 19
 호스트 구조
 배열, 선언 36
 선언 32
 인디케이터 구조 배열, 선언 39
 인디케이터 배열 36
 호스트 변수 20
 그래픽 25
 문자 21
 선언 21
 숫자 21
 외부적으로 서술된 42
 포인터 사용 40
 BLOB 29

C++ 프로그램 (계속)
 호스트 변수 (계속)
 CLOB 29
 DBCLOB 29
 LOB 28
 ROWID 32
 SQL-varchar 21
 varchar 21
 BEGIN/END DECLARE SECTION 20
 INCLUDE문 19
 SQL 자료 유형
 동등한 C++ 판별 43
 SQLCA, 선언 13
 SQLCODE, 선언 13
 SQLDA, 선언 15
 SQLSTATE, 선언 13
 SQL문 코딩 13
 typedef 41
 WHENEVER문 20
 #include 지시문 19
 #pragma mapinc 지시문 42

D

Datetime 호스트 변수
 COBOL 63
 iSeries용 ILE RPG 119

DATFMT
 iSeries용 ILE RPG 119, 126

DATSEP
 iSeries용 ILE RPG 119, 126

DBCLOB 호스트 변수
 C 29
 COBOL 61
 C++ 29
 iSeries용 ILE RPG 122

DBCS 상수
 계속
 C 18
 COBOL 53
 C++ 18
 FORTRAN 352
 iSeries용 ILE RPG 117
 iSeries용 RPG 102
 PL/I 82
 SQL 소스 내에 147

E

END DECLARE SECTION문
 C 20
 COBOL 54
 C++ 20
 FORTRAN 354
 iSeries용 ILE RPG 118
 iSeries용 RPG 103
 PL/I 83
 EXEC SQL REXX 명령 135, 138

F

FETCH문
 복수 행
 iSeries용 ILE RPG 121, 132
 iSeries용 RPG 105

FORTRAN 프로그램
 계속 352
 동적 SQL 코딩 351
 명령문 레이블 353
 명명 규칙 353
 여백 353
 인디케이터 변수 357
 주석 352
 컴파일시 옵션 354
 코드 포함 353
 행 디버그 352
 호스트 변수 354
 문자 355
 선언 355, 356
 숫자 355

BEGIN/END DECLARE
 SECTION 354

IMPLICIT문 354

PROCESS문 354

SQL 자료 유형
 동일한 FORTRAN 판별 356

SQLCA, 선언 349
 SQLCODE, 선언 349
 SQLCOD, 선언 349
 SQLSTATE, 선언 349
 SQLSTA, 선언 349
 SQL문 코딩 349, 359
 WHENEVER문 354

I

ILE C 프로그램
SQL문, 샘플 163
ILE COBOL 프로그램
SQL 168
SQL문이 있는 샘플 프로그램 168
ILE RPG 프로그램
SQLCA 대체 161
IMPLICIT문
FORTRAN 354
INCLUDE문 146
C 19
COBOL 53
C++ 19
iSeries용 ILE RPG 117
iSeries용 RPG 102
PL/I 82
INSERT문
블록
iSeries용 ILE RPG 121
iSeries용 RPG 105
열 값 3
iSeries용 DB2 UDB
C 프로그램 161
iSeries용 ILE RPG 프로그램
계속 117
동적 SQL 코딩 115
명령문 레이블 118
명명 규칙 118
문자 호스트 변수 120
발생 자료 구조 121
변수 선언 부분 131
순서 번호 118
외부 파일 설명 125
위치 지정자
LOB 123
인디케이터 구조 131
인디케이터 변수 131
주 및 사용 131
주석 117
컴파일러 매개변수 154
컴파일시 오류 및 경고 메시지 157
파일 참조 변수
LOB 124
포함 코드 117
호스트 구조
선언 120

iSeries용 ILE RPG 프로그램 (계속)
호스트 구조 배열
선언 121
호스트 변수 118
그래픽 127
날짜/시간 119, 127
문자 127
선언 119
숫자 127
외부적으로 서술된 125
BLOB 122
CLOB 122
DBCLOB 122
LOB 122
ROWID 125
SQL 자료 유형
동등한 RPG 판별 127
SQLCA 114
SQLCA 대체 114
SQLDA
예 132
SQLDA, 선언 115
SQL문
샘플 187
SQL문 코딩 113, 133
WHENEVER문 118
/COPY문 117, 125
iSeries용 RPG 프로그램 113
계속 102
구조 매개변수 전달 111
동적 SQL 코딩 100
명령문 레이블 103
명명 규칙 103
문자 호스트 변수 104
발생 자료 구조 105
순서 번호 103
외부 파일 설명 106
인디케이터 구조 110
인디케이터 변수 110
종료
LR 인디케이터 사용 111
RETRN문 사용 111
주석 102
컴파일러 매개변수 153
컴파일시 오류 및 경고 메시지 157
코드 포함 102
호스트 구조
배열, 선언 105

iSeries용 RPG 프로그램 (계속)
호스트 구조 (계속)
선언 104
호스트 변수 103
문자 107
선언 103
숫자 107
외부적으로 서술된 106
SQL 자료 유형
동등한 RPG 판별 107
SQLCA
배치 100
SQLDA 사용 100
SQL문
샘플 181
SQL문 코딩 99, 111
WHENEVER문 103
/COPY문 102, 106
iSeries용 SQL ILE C 작성(CRTSQLCI) 명령
329

L

LOB 위치 지정자
C 30
COBOL 62
C++ 30
iSeries용 ILE RPG 123
PL/I 86
LOB 파일 참조 변수
C 30
COBOL 62
C++ 30
iSeries용 ILE RPG 124
PL/I 86
LOB 호스트 변수
C 28
COBOL 60
C++ 28
iSeries용 ILE RPG 122
PL/I 85
LR 인디케이터
iSeries용 RPG 프로그램 종료 111

M

MARGINS 매개변수
C 19

MARGINS 매개변수 (계속)

C++ 19

N

NUL 종료자(NUL-terminator)

문자 호스트 변수

C 21

C++ 21

C 22

C++ 22

P

PL/I

호스트 변수

ROWID 87

PL/I 프로그램

계속 82

구조 매개변수 전달 97

동적 SQL 코딩 80

명명 규칙 83

여백 82

외부 파일 설명 93

위치 지정자

LOB 86

인디케이터 구조 96

인디케이터 변수 96

주석 82

컴파일러 매개변수 153

컴파일시 오류 및 경고 메시지 156

파일 참조 변수

LOB 86

포함 코드 82

호스트 구조

배열, 선언 91

선언 88

인디케이터 구조 배열, 선언 93

인디케이터 배열 90

호스트 변수 83

문자 84

선언 83, 85

숫자 84

외부적으로 서술된 93

BLOB 85

CLOB 86

LOB 85

BEGIN/END DECLARE SECTION 83

PL/I 프로그램 (계속)

INCLUDE문 82

SQL 자료 유형

동일한 PL/I 판별 94

SQLCA, 선언 79

SQLCODE, 선언 79

SQLDA, 선언 80

SQLSTATE, 선언 79

SQL문 코딩 79, 97

SQL문, 샘플 176

WHENEVER문 83

%INCLUDE 지시문 82, 93

PROCESS문

COBOL 54

FORTRAN 354

Q

QSQLTEMP 148

QSQLTEMP1 148

R

RETRN문

iSeries용 RPG 프로그램 종료 111

REXX

SQL문

샘플 193

SQL문 코딩 135, 144

REXX의 널(null) 스트링 140

REXX의 SIGNAL ON ERROR 141

REXX의 SIGNAL ON FAILURE 141

ROWID 호스트 변수

C 32

COBOL 63

C++ 32

iSeries용 ILE RPG 125

PL/I 87

RPG 99, 113

S

SELECT INTO문

열 값 3

SQL

명령문

호스트 변수 사용 1

COBOL 168

SQL (계속)

명령문 (계속)

ILE COBOL 168

iSeries용 ILE RPG 프로그램 187

iSeries용 RPG 181

PL/I 163, 176

REXX 193

호스트 언어로 사용, 개념과 규칙 1

SQL COBOL 작성(CRTSQLCBL) 명령 329

SQL C++ 작성(CRTSQLCPPI) 명령 329

SQL RPG 작성(CRTSQLRPG) 명령 329

SQL 자료 유형

동등한 COBOL 판별

C 43

COBOL 74

C++ 43

FORTRAN 356

iSeries용 ILE RPG 127

iSeries용 RPG 107

PL/I 94

REXX 142

SQL 정보 인쇄(PRTSQLINF) 159

SQL 패키지 작성(CRTSQLPKG) 명령 154

SQLCA(SQL 통신 영역)

C 13

COBOL 49

C++ 13

FORTRAN 349

iSeries용 ILE RPG 114

iSeries용 RPG 100

PL/I 79

REXX 135

SQLCA의 SQLERRD 필드 135

SQLCA의 SQLERRMC 필드 135

SQLCA의 SQLERRP 필드 135

SQLCA의 SQLWARN 필드 135

SQLCOD

FORTRAN 349

SQLCODE

정의 8

C 13

COBOL 49

C++ 13

FORTRAN 349

PL/I 79

REXX에서 135

SQLDA(SQL 설명자 영역)
 C 15
 COBOL 51
 C++ 15
 FORTRAN 351
 iSeries용 ILE RPG 115
 iSeries용 RPG 100
 PL/I 80
 REXX 136
 SQLDA의 SQL 필드
 REXX에서 136
 SQLDA의 SQLDATA 필드
 REXX에서 138
 SQLDA의 SQLIND 필드
 REXX에서 138
 SQLDA의 SQLLEN 필드
 REXX에서 137
 SQLDA의 SQLNAME 필드
 REXX에서 137
 SQLDA의 SQLPRECISION 필드 137
 SQLDA의 SQLSCALE 필드 137
 SQLDA의 SQLTYPE 필드
 REXX에서 137
 SQLERROR문
 WHENEVER 9
 SQLSTA
 FORTRAN 349
 SQLSTATE
 정의 8
 C 13
 COBOL 49
 C++ 13
 FORTRAN 349
 PL/I 79
 REXX에서 135
 SQL문 코딩
 REXX 어플리케이션에서의 135
 SQL문으로 프로그램 준비 145
 SQL-varchar 호스트 변수
 C 21
 C++ 21

T

TAG문
 iSeries용 ILE RPG 118
 iSeries용 RPG 103

TIMFMT
 iSeries용 ILE RPG 119, 126
 TIMSEP
 iSeries용 ILE RPG 119, 126
 typedef
 C 41
 C++ 41

U

UPDATE문
 지정 조작 3

V

varchar 호스트 변수
 C 21
 C++ 21

W

WHENEVER SQLERROR 9
 WHENEVER문
 예외 조건 처리 9
 C 20
 COBOL 54
 C++ 20
 FORTRAN 354
 iSeries용 ILE RPG 118
 iSeries용 RPG 103
 PL/I 83
 REXX, 대체 140

[특수 문자]

#include 지시문
 C 19
 C++ 19
 #pragma mapinc 지시문
 C 42
 C++ 42
 *APOST 54
 *CNULRQD 22
 *NOCNULRQD 23
 *NOCVTDT 126
 *NOSEQSRC
 iSeries용 ILE RPG 118
 iSeries용 RPG 103

*QUOTE 54
 *SEQSRC
 iSeries용 ILE RPG 118
 iSeries용 RPG 103
 -(붙임표)
 COBOL 호스트 변수 내의 55
 -(빼기)
 COBOL 55
 /COPY
 iSeries용 ILE RPG 117, 125
 iSeries용 RPG 102, 106
 :(콜론)
 C 호스트 변수 20
 COBOL 55
 C++ 호스트 변수 20
 FORTRAN 354
 iSeries용 ILE RPG 119
 iSeries용 RPG 103
 PL/I 83
 REXX 141
 %INCLUDE 지시문 93
 PL/I 82



Printed in U.S.A.