



IBM 시스템 - iSeries

데이터베이스

iSeries 삽입된 SQL 프로그래밍용 DB2 Universal  
Database

버전 5 릴리스 4







**IBM 시스템 - iSeries**

**데이터베이스**

**iSeries 삽입된 SQL 프로그래밍용 DB2 Universal  
Database**

버전 5 릴리스 4

주!

이 정보와 이 정보가 지원하는 제품을 사용하기 전에, 193 페이지의 『주의사항』의 정보를 읽으십시오.

**제 6 판(2006년 2월)**

이 개정판은 새 개정판에서 별도로 명시하지 않는 한 IBM i5/OS의 버전 5, 릴리스 4, 수정 0(제품 번호 5722-SS1) 및 모든 후속 릴리스와 수정에 적용됩니다. 이 버전은 모든 축약 명령어 세트 컴퓨터(RISC) 모델 및 CISC 모델에서도 실행되지 않습니다.

© Copyright International Business Machines Corporation 1998, 2006. All rights reserved.

# 목차

삽입된 SQL 프로그래밍 . . . . .	1	SQL을 사용하는 COBOL 어플리케이션에서 외부	
V5R4의 새로운 사항 . . . . .	1	파일 설명 사용 . . . . .	69
인쇄 가능한 PDF. . . . .	1	동등한 SQL 및 COBOL 자료 유형 판별 . . . . .	70
삽입된 SQL 사용을 위한 일반적인 개념 및 규칙 . . . . .	2	SQL을 사용하는 COBOL 어플리케이션에서 인디	
SQL을 사용하는 어플리케이션 작성. . . . .	2	케이터 변수 사용 . . . . .	73
SQL문에서 호스트 변수 사용. . . . .	3	PL/I 어플리케이션에서 SQL문 코딩 . . . . .	73
SQLCA를 사용하여 SQL 오류 리턴 코드 처리. . . . .	9	SQL을 사용하는 PL/I 어플리케이션에서 SQL 통	
SQL 진단 영역 사용 . . . . .	10	신 영역 정의. . . . .	74
WHENEVER문에 대한 예외 조건 처리. . . . .	13	SQL을 사용하는 PL/I 어플리케이션에서 SQL 설	
C 및 C++ 어플리케이션에서 SQL문 코딩 . . . . .	15	명자 영역 정의 . . . . .	75
SQL을 사용하는 C 및 C++ 어플리케이션에서		SQL을 사용하는 PL/I 어플리케이션에서 SQL문	
SQL 통신 영역 정의 . . . . .	15	삽입. . . . .	76
SQL을 사용하는 C 및 C++ 어플리케이션에서		SQL을 사용하는 PL/I 어플리케이션에서 호스트	
SQL 설명자 영역 정의 . . . . .	17	변수 사용. . . . .	77
SQL을 사용하는 C 및 C++ 어플리케이션에		SQL을 사용하는 PL/I 어플리케이션에서 호스트	
SQL문 삽입 . . . . .	19	구조 사용 . . . . .	82
SQL을 사용하는 C 및 C++ 어플리케이션에서 호		SQL을 사용하는 PL/I 어플리케이션에서 호스트	
스트 변수 사용 . . . . .	21	구조 배열 사용 . . . . .	84
SQL을 사용하는 C 및 C++ 어플리케이션에서 호		SQL을 사용하는 PL/I 어플리케이션에서 외부 파	
스트 구조 사용 . . . . .	33	일 설명 사용. . . . .	86
SQL을 사용하는 C 및 C++ 어플리케이션에서 호		동일한 SQL 및 PL/I 자료 유형 판별. . . . .	87
스트 구조 배열 사용 . . . . .	36	SQL을 사용하는 PL/I 어플리케이션에서 인디케이	
SQL을 사용하는 C 및 C++ 어플리케이션에서 포		터 변수 사용. . . . .	88
인터 자료 유형 사용 . . . . .	40	구조 매개변수 전달 기법에 따른 PL/I의 차이점	89
SQL을 사용하는 C 및 C++ 어플리케이션에서		RPG/400 어플리케이션에서 SQL문 코딩 . . . . .	89
typedef 사용. . . . .	41	SQL을 사용하는 RPG/400 어플리케이션에서	
SQL을 사용하는 C 및 C++ 어플리케이션에서		SQL 통신 영역 정의 . . . . .	90
ILE C 컴파일러 외부 파일 설명 . . . . .	42	SQL을 사용하는 RPG/400 어플리케이션에서	
동등한 SQL 및 C 또는 C++ 자료 유형 판별. . . . .	43	SQL 설명자 영역 정의 . . . . .	91
SQL을 사용하는 C 및 C++ 어플리케이션에서 인		SQL을 사용하는 RPG/400 어플리케이션에 SQL	
디케이터 변수 사용. . . . .	45	문 삽입. . . . .	92
COBOL 어플리케이션에서 SQL문 코딩 . . . . .	46	SQL을 사용하는 RPG/400 어플리케이션에서 호	
SQL을 사용하는 COBOL 어플리케이션에서		스트 변수 사용 . . . . .	93
SQL 통신 영역 정의 . . . . .	46	SQL을 사용하는 RPG/400 어플리케이션에서 호	
SQL을 사용하는 COBOL 어플리케이션에서		스트 구조 사용 . . . . .	94
SQL 설명자 영역 정의 . . . . .	48	SQL을 사용하는 RPG/400 어플리케이션에서 호	
SQL을 사용하는 COBOL 어플리케이션에 SQL		스트 구조 배열 사용 . . . . .	95
문 삽입. . . . .	49	SQL을 사용하는 RPG/400 어플리케이션에 외부	
SQL을 사용하는 COBOL 어플리케이션에서 호스		파일 설명 사용 . . . . .	95
트 변수 사용. . . . .	51	동등한 SQL 및 RPG/400 자료 유형 판별. . . . .	97
SQL을 사용하는 COBOL 어플리케이션에서 호스		SQL을 사용하는 RPG/400 어플리케이션에서 인	
트 구조 사용. . . . .	60	디케이터 변수 사용. . . . .	99

구조 매개변수 전달 기법에 따른 RPG/400의 차이점 . . . . .	100	SQL을 사용하는 어플리케이션에서 컴파일 오류 해석 . . . . .	146
SQL을 사용하는 호출된 RPG/400 프로그램의 올바른 종료. . . . .	100	SQL을 사용하는 어플리케이션 바인딩 . . . . .	147
ILE RPG 어플리케이션에서 SQL문 코딩 . . . . .	101	SQL 사전컴파일러 옵션 표시 . . . . .	149
SQL을 사용하는 ILE RPG 어플리케이션에서 SQL 통신 영역 정의. . . . .	101	삽입된 SQL이 있는 프로그램 실행 . . . . .	149
SQL을 사용하는 ILE RPG 어플리케이션에서 SQL 설명자 영역 정의 . . . . .	102	iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램 . . . . .	150
SQL을 사용하는 ILE RPG 어플리케이션에 SQL문 삽입. . . . .	104	예: ILE C 및 C++ 프로그램의 SQL문 . . . . .	152
SQL을 사용하는 ILE RPG 어플리케이션에서 호스트 변수 사용 . . . . .	107	예: COBOL 및 ILE COBOL 프로그램의 SQL문 . . . . .	158
SQL을 사용하는 ILE RPG 어플리케이션에서 호스트 구조 사용 . . . . .	113	예: PL/I 프로그램의 SQL문 . . . . .	167
SQL을 사용하는 ILE RPG 어플리케이션에서 호스트 구조 배열 사용 . . . . .	115	예: RPG/400 프로그램의 SQL문. . . . .	172
SQL을 사용하는 ILE RPG 어플리케이션에서 외부 파일 설명 사용 . . . . .	116	예: ILE RPG 프로그램의 SQL문 . . . . .	178
동등한 SQL 및 ILE RPG 자료 유형 판별 . . . . .	117	예: REXX 프로그램의 SQL문 . . . . .	184
SQL을 사용하는 ILE RPG 어플리케이션에서 인디케이터 변수 사용. . . . .	122	SQL을 사용하는 샘플 프로그램에 의해 작성된 보고서. . . . .	188
SQL을 사용하는 ILE RPG 어플리케이션의 복수 행 영역 페치에 대한 SQLDA 예 . . . . .	123	호스트 언어 사전컴파일러용 iSeries용 DB2 UDB CL 명령 설명 . . . . .	189
SQL을 사용하는 ILE RPG 어플리케이션의 동적 SQL 예 . . . . .	124	CRTSQLCBL(구조화 조회 언어 COBOL 작성) 명령 . . . . .	190
REXX 어플리케이션에서 SQL문 코딩 . . . . .	125	CRTSQLCBLI(SQL ILE COBOL 오브젝트 작성) 명령 . . . . .	190
REXX 어플리케이션에서 SQL 통신 영역 사용	126	CRTSQLCI(구조화 조회 언어 ILE C 오브젝트 작성) 명령 . . . . .	190
REXX 어플리케이션에서 SQL 설명자 영역 사용 . . . . .	127	CRTSQCPPI(구조화 조회 언어 C++ 오브젝트 작성) 명령 . . . . .	190
REXX 어플리케이션에서 SQL문 삽입 . . . . .	129	CRTSQPLI(구조화 조회 언어 PL/I 작성) 명령	190
SQL을 사용하는 REXX 어플리케이션에서 호스트 변수 사용 . . . . .	132	CRTSQLRPG(구조화 조회 언어 RPG 작성) 명령 . . . . .	191
SQL을 사용하는 REXX 어플리케이션에서 인디케이터 변수 사용 . . . . .	134	CRTSQLRPGI(SQL ILE RPG 오브젝트 작성) 명령 . . . . .	191
SQL문을 사용하는 프로그램 준비 및 실행 . . . . .	134	삽입된 SQL 프로그래밍 관련 정보 . . . . .	191
SQL 사전컴파일러의 기본 처리 . . . . .	135	코드 라이선스 및 면책사항 정보 . . . . .	192
비ILE SQL 사전컴파일러 명령 . . . . .	143	부록. 주의사항 . . . . .	193
ILE SQL 사전컴파일러 명령 . . . . .	144	프로그래밍 인터페이스 정보 . . . . .	195
사전컴파일러 명령을 사용하여 컴파일러 옵션 설정 . . . . .	145	상표 . . . . .	195
		조건 . . . . .	195

---

## 삽입된 SQL 프로그래밍

이 주제 컬렉션은 iSeries™용 DB2 Universal Database™ SQL문과 함수를 사용하는 호스트 언어로 데이터 베이스 어플리케이션을 작성하는 방법에 대해 설명합니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

---



### V5R4의 새로운 사항

이 주제에서는 V5R4 주제 컬렉션의 변경 내용을 중점적으로 설명합니다.

- | • RPG 자유 형식의 삽입된 SQL에 대한 지원이 104 페이지의 『SQL을 사용하는 ILE RPG 어플리케이션에 SQL문 삽입』 및 그 안의 일부 주제에 추가되었습니다.
- | • 106 페이지의 『SQL을 사용하는 ILE RPG 어플리케이션의 이름』에 대한 규칙이 갱신되었습니다.

### 새로운 사항 또는 변경된 사항을 확인하는 방법

기술적 사항이 변경된 위치를 확인하는 데 도움이 되도록 다음 정보가 사용됩니다.

- 새로운 정보 또는 변경된 정보가 시작되는 위치를 표시하는  이미지.
- 새로운 정보 또는 변경된 정보가 종료되는 위치를 표시하는  이미지.

이 릴리스에서 변경된 사항 또는 새로운 사항에 대한 다른 정보를 찾으려면 사용자 메모를 참조하십시오.

---

### 인쇄 가능한 PDF

이 정보를 PDF로 보고 인쇄하는 데 사용합니다.


이 문서의 PDF 버전을 보거나 다운로드하려면 삽입된 SQL 프로그래밍(약 1750KB)을 선택하십시오.

### PDF 파일 저장

보거나 인쇄하기 위해 워크스테이션에 PDF를 저장하려면 다음을 참조하십시오.

1. 브라우저에서 PDF를 마우스 오른쪽으로 클릭하십시오(위의 링크를 마우스 오른쪽으로 클릭).
2. PDF를 로컬로 저장하는 옵션을 클릭하십시오.
3. PDF를 저장할 디렉토리를 탐색하십시오.
4. 저장을 클릭하십시오.

## Adobe Reader 다운로드

- | 이러한 PDF를 보거나 인쇄하려면 시스템에 Adobe Reader를 설치해야 합니다. Adobe 웹 사이트
- | ([www.adobe.com/products/acrobat/readstep.html](http://www.adobe.com/products/acrobat/readstep.html))  에서 사본을 무료로 다운로드할 수 있습니다.

---

## 삽입된 SQL 사용을 위한 일반적인 개념 및 규칙

이 주제에서는 호스트 언어로 SQL문 사용에 해당되는 일반적인 일부 개념 및 규칙을 설명합니다.

### SQL을 사용하는 어플리케이션 작성

iSeries용 DB2® UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성할 수 있습니다.

삽입된 SQL을 사용하려면 DB2 Query Manager 및 SQL Development Kit이 설치되어 있어야 합니다. 또한 설치할 호스트에 언어에 대한 컴파일러가 설치되어 있어야 합니다.

#### 관련 개념

15 페이지의 『C 및 C++ 어플리케이션에서 SQL문 코딩』

이 주제에서는 C 또는 C++ 프로그램에 SQL문을 삽입하기 위한 고유의 어플리케이션과 코딩 요구사항에 대해 설명합니다.

46 페이지의 『COBOL 어플리케이션에서 SQL문 코딩』

이 주제에서는 COBOL 프로그램에 SQL문을 삽입할 때의 고유한 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.

73 페이지의 『PL/I 어플리케이션에서 SQL문 코딩』

이 주제에서는 iSeries PL/I 프로그램에 SQL문을 삽입하는 경우 고유한 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.

89 페이지의 『RPG/400 어플리케이션에서 SQL문 코딩』

RPG/400® 라이선스 프로그램은 RPG II 및 RPG III 프로그램을 모두 지원합니다.

101 페이지의 『ILE RPG 어플리케이션에서 SQL문 코딩』

이 주제에서는 ILE RPG 프로그램에 SQL문을 삽입할 때의 고유 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 변수에 대한 코딩 요구사항도 정의됩니다.

125 페이지의 『REXX 어플리케이션에서 SQL문 코딩』

REXX 프로시저는 사전처리할 필요가 없습니다. 실행 시에 REXX 인터프리터는 자신이 이해하지 못하는 명령문을 현재 사용 중인 명령 환경으로 전달하여 처리합니다.

134 페이지의 『SQL문을 사용하는 프로그램 준비 및 실행』

이 주제에서는 어플리케이션 프로그램을 준비하고 실행하기 위한 일부 타스크를 설명합니다.

#### 관련 정보

IBM Developer Kit for Java



## SQL문에서 호스트 변수 사용

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

호스트 변수는 SQL문에 지정된 프로그램 내의 필드로서 일반적으로 열 값의 소스나 목표를 말합니다. 호스트 변수와 열은 반드시 호환되는 자료 유형이어야 합니다. 호스트 변수는 DESCRIBE TABLE문을 제외하고는 표나 뷰와 같은 SQL 오브젝트 식별에 사용할 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 유연성을 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.

- **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.
```

- **(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 검색 중이라고 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.

- **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

- **SQL문의 다른 절에 있는 값으로서:**

- UPDATE문의 SET절
- INSERT문의 VALUES절
- CALL문

관련 정보  
SQL 참조

### SQL문에서 호스트 변수 지정 규칙

SQL 값은 FETCH, SELECT INTO, SET 및 VALUES INTO문 실행 시 호스트 변수에 할당됩니다. SQL 값은 INSERT, UPDATE 및 CALL문 실행 시 호스트 변수로부터 할당됩니다.

모든 지정 조작은 다음 규칙을 준수해야 합니다.

- 숫자와 스트링은 호환될 수 있습니다.
  - 숫자는 문자 또는 그래픽 스트링 열 또는 호스트 변수에 할당될 수 있습니다.
  - 문자 및 그래픽 스트링은 숫자 열 또는 숫자 호스트 변수에 할당될 수 있습니다.
- 모든 문자와 DBCS 그래픽 스트링은 CCSID 간에 변환이 지원되는 경우 UCS-2 및 UTF-16 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 CCSID 간에 변환이 지원되는 경우 지정 조작에 있어서 UCS-2 및 UTF-16 그래픽 문자열과 호환됩니다. CALL문의 경우에 변환이 지원되면 문자 및 DBCS 그래픽 매개변수가 UCS-2 및 UTF-16 매개변수와 호환될 수 있습니다.
- 2진 스트링은 2진 스트링과만 호환됩니다.
- 널(null)은 연관된 인디케이터 변수가 없는 호스트 변수에 할당될 수 없습니다.

- 다른 유형의 날짜/시간 값은 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 있습니다. 날짜 열의 삽입 또는 갱신 값은 날짜 또는 날짜의 문자열 표시이어야 합니다. DBCS 개방 또는 DBCS 선택 변수는 외부 서술 파일의 정의를 포함하여 호스트 언어로 선언된 변수입니다. 작업 CCSID가 혼합 자료를 표시하거나 DECLARE VARIABLE문이 사용되고 MIXED CCSID 또는 FOR MIXED DATA절이 지정되면 DBCS 개방 변수도 선언됩니다.

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.

#### 관련 정보

DECLARE VARIABLE

### SQL문에서 호스트 변수 스트링 지정 규칙:

이 주제에서는 문자 스트링 지정에 대한 규칙을 소개합니다.

규칙은 다음과 같습니다.

- 문자 또는 그래픽 스트링이 열에 할당되면 스트링 길이 값이 열의 길이 속성보다 클 수 없습니다. (후미 공백은 대개 스트링의 길이에 포함됩니다. 그러나 스트링 할당의 경우 후미 공백은 스트링의 길이에 포함되지 않습니다.)
- 2진 스트링이 열에 할당되면 스트링 길이 값이 열의 길이 속성보다 클 수 없습니다. (16진 0은 대개 스트링의 길이에 포함됩니다. 그러나 스트링 할당의 경우 16진 0은 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 SQLWARN0 및 SQLWARN1(SQL 통신 영역 (SQLCA) 내의)이 W로 설정됩니다.
- 결과 열의 값이 고정 길이 문자 또는 그래픽 호스트 변수에 할당되거나 호스트 변수 값이 고정 길이 문자 또는 그래픽 결과 열에 할당되고 스트링 길이 값이 목표의 길이 속성보다 작으면 스트링의 오른쪽이 필요한 수만큼 공백으로 채워집니다.
- 결과 열의 값이 고정 길이 2진 호스트 변수로 할당되거나 호스트 변수 값이 고정 길이 2진 결과 열에 할당되고 스트링 길이 값이 대상의 길이 속성보다 작으면 스트링의 오른쪽이 필요한 수만큼 16진 0으로 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.

- 소스의 CCSID가 목표의 CCSID와 일치하면 변환 없이 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.

관련 정보

국제화

## 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. 8자릿수 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.
- 십진, 숫자, 정수가 십진, 숫자 또는 정수 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.
- 정수 또는 부동 소수점 숫자가 십진 또는 숫자 열이나 호스트 변수에 지정되는 경우 이 숫자는 먼저 임시 십진 또는 숫자로 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 CRTSQL<sub>xxx</sub> 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 6바이트인 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 CRTSQL<sub>xxx</sub> 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 안 됩니다.
- \*HMS, \*ISO, \*EUR 또는 \*JIS 형식이 사용되면 호스트 변수의 길이는 초가 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 19바이트인 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 어플리케이션의 인디케이터 변수

인디케이터 변수는 해당되는 연관된 호스트 변수가 널값으로 지정되었는지 여부를 나타내는데 사용되는 반단어 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 매핑 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 리턴합니다.

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.

DISTINCT 술부는 널값이 있을 경우 비교를 수행하는 데 사용될 수 있습니다.

관련 정보

술부

### 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 각 값은 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 사용되는 인디케이터 변수:

인디케이터 변수를 사용하여 열 내의 널(null)을 설정할 수 있습니다.

UPDATE 또는 INSERT문 처리 시, 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.
```

### SQLCA를 사용하여 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.

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.

관련 정보

SQL 참조

SQL 메시지 및 코드

## **SQL 진단 영역 사용**

SQL 진단 영역을 사용하여 프로그램에서 실행되는 SQL문에 대해 리턴된 정보를 보존합니다. SQLCA를 통해 어플리케이션 프로그래머로서 사용자가 사용할 수 있는 모든 정보가 들어 있습니다.

연결 정보를 포함하여 SQL문에 대한 추가 정보를 제공하는 데 사용할 수 있는 추가 값이 있습니다. 둘 이상의 조건은 단일 SQL문에서 리턴될 수 있습니다. SQL 진단 영역의 정보는 다음 SQL문이 실행될 때까지 이전 SQL문에 대해 사용할 수 있습니다.

진단 영역의 정보에 액세스하려면 GET DIAGNOSTICS문을 사용하십시오. 이 명령문에서는 이전에 실행한 SQL문에 대한 복수 정보를 한 번에 요구할 수 있습니다. 각 항목은 호스트 변수에서 리턴됩니다. 또한 사용할 수 있는 모든 진단 정보가 들어 있는 스트링을 얻도록 요구할 수 있습니다. GET DIAGNOSTICS문을 실행하면 진단 영역은 지워지지 않습니다.

관련 정보

GET DIAGNOSTICS

## **SQL 진단 영역을 사용할 수 있도록 어플리케이션 갱신**

SQL 진단 영역은 SQLCA(SQL 통신 영역)에 비해 몇 가지 중요한 이점을 제공하므로 SQLCA 대신 SQL 진단 영역을 사용할 수 있도록 어플리케이션 변경을 고려할 수 있습니다.



가장 좋은 이유 중 하나는 SQLCA의 SQLERRM 필드가 70바이트의 길이라는 것입니다. 이러한 이유로 오류 정보를 호출 어플리케이션에 리턴하기에는 충분하지 않습니다. SQL 진단 영역을 고려하는 추가 이유는 복수 행 조작 및 긴 열과 오브젝트명 때문입니다. 136바이트 SQLCA로 제한 시 단순 경고를 보고하기도 어렵습니다. 가끔씩 리턴된 토큰은 SQLCA의 제한사항에 맞추기 위해 절단됩니다.

현재 어플리케이션에는 다음을 사용하여 SQLCA 정의가 포함됩니다.

```
EXEC SQL INCLUDE SQLCA; /* Existing SQLCA */
```

SQL 진단 영역을 사용하기 위해 변환할 때 어플리케이션은 먼저 다음과 같은 독립형 SQLSTATE 변수를 선언합니다.

```
char SQLSTATE[6]; /* Stand-alone sqlstate */
```

그러면 독립형 SQLCODE 변수는 다음과 같습니다.

```
long int SQLCODE; /* Stand-alone sqlcode */
```

독립형 SQLSTATE 변수를 검사하면 SQL문의 완료 상태가 확인됩니다. 현재 SQL문을 완료할 때 어플리케이션이 진단을 검색하면 어플리케이션은 다음과 같이 SQL GET DIAGNOSTICS문을 실행합니다.

```
char hv1[256];  
long int hv2;
```

```
EXEC SQL GET DIAGNOSTICS :hv1 = COMMAND_FUNCTION, :hv2 = COMMAND_FUNCTION_CODE;
```

## iSeries 서버 프로그래밍 모델

iSeries ILE(Integrated Language Environment)<sup>®</sup>에서 SQL 진단 영역은 스텝드 및 활성 그룹 범위에 있습니다. 이는 스텝드가 SQL문을 실행하는 활성 그룹마다 별도의 진단 영역이 활성화될 수 있도록 존재함을 의미합니다.

### SQL 진단 영역 사용에 대한 추가 주의사항

어플리케이션 프로그램에서 SQLCA는 내재적 또는 독립형 SQLSTATE 변수로 대체되며, 이 변수는 해당 프로그램에서 선언되어야 합니다.

SQL 진단 영역에 있는 복수 조건 영역을 사용하면 가장 심각한 오류 또는 경고가 첫 번째 진단 영역에서 리턴됩니다. 첫 번째 진단 영역에 SQLSTATE 변수에서도 리턴되는 SQLSTATE의 정보가 들어 있는 것을 제외하면 복수 조건에 대한 특정 순서는 없습니다.

SQLCA를 사용하면 어플리케이션 프로그램은 SQL문의 실행 결과를 통신하는 데 사용되는 SQLCA용 기억장치를 제공합니다. SQL 진단 영역을 사용하면 데이터베이스 관리자는 진단에 필요한 기억장치를 관리하며 GET DIAGNOSTICS문이 진단 영역의 내용을 검색하기 위해 제공됩니다.

SQLCA가 어플리케이션 프로그램에서 계속 지원된다는 점에 유의하십시오. 또한 GET DIAGNOSTICS문은 SQLCA를 사용하는 어플리케이션 프로그램에서 사용할 수 있습니다.

### 예: SQL 루틴 예외

이 어플리케이션 예에서 저장 프로시저는 입력 값이 범위 밖에 있을 때 오류를 신호로 보냅니다.

```

EXEC SQL CREATE PROCEDURE check_input (IN p1 INT)
LANGUAGE SQL READS SQL DATA
test: BEGIN
  IF p1 < 0 THEN
    SIGNAL SQLSTATE VALUE '99999' SET MESSAGE_TEXT = 'Bad input value'; END IF
END test;

```

호출 어플리케이션은 실패가 있는지 검사하며 SQL 진단 영역에서 다음과 같이 실패 정보를 검색합니다.

```

char SQLSTATE[6]; /* Stand-alone sqlstate */
long int SQLCODE; /* Stand-alone sqlcode */

```

```

long int hv1;
char hv2[6];
char hv3[256];

```

```

hv1 = -1;
EXEC SQL CALL check_input(:hv1);

```

```

if (strncmp(SQLSTATE, "99999", 5) == 0)
{
  EXEC SQL GET DIAGNOSTICS CONDITION 1
    :hv2 = RETURNED_SQLSTATE,
    :hv3 = MESSAGE_TEXT;
}
else
{
}

```

### 예: SQL 진단 영역으로부터 항목 로깅

이 예의 어플리케이션에서는 보안상의 이유로 모든 오류를 기록해야 합니다. 기록부는 시스템의 이상 유무를 모니터링하거나 데이터베이스의 부적절한 사용이 있는지 모니터링하는 데 사용될 수 있습니다.

발생하는 각 SQL 오류의 경우 항목이 기록부에 배치됩니다. 항목에는 오류가 발생한 시간, 어플리케이션을 사용했던 사용자, 실행된 SQL문의 유형, 리턴된 SQLSTATE 값 및 메세지 번호와 해당 완료 메세지 텍스트 등이 포함됩니다.

```

char stmt_command[256];
long int error_count;
long int condition_number;
char auth_id[256];
char error_state[6];
char msgid[128];
char msgtext[1024];

```

```

EXEC SQL WHENEVER SQLERROR GOTO error;

```

(application code)

```

error:
EXEC SQL GET DIAGNOSTICS :stmt_command = COMMAND_FUNCTION,
                        :error_count = NUMBER;

```

```

for (condition_number=1; i<=error_count; ++condition_number)
{

```

```

EXEC SQL GET DIAGNOSTICS CONDITION :condition_number
      :auth_id = DB2_AUTHORIZATION_ID,
      :error_state = RETURNED_SQLSTATE,
      :msgid = DB2_MESSAGE_ID,
      :msgtext = DB2_MESSAGE_TEXT;

EXEC SQL INSERT INTO error_log VALUES(CURRENT_TIMESTAMP,
      :stmt_command,
      :condition_number,
      :auth_id,
      :error_state,
      :msgid,
      :msgtext);
}

```

관련 정보

GET DIAGNOSTICS

## WHENEVER문에 대한 예외 조건 처리

WHENEVER문은 SQL이 SQLSTATE와 SQLCODE를 체크하여 프로그램을 계속해서 처리하도록 하거나 SQL 문을 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.

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

SQL문 (SQLCODE < 0) (SUBSTR(SQLSTATE,1,2) > '02')의 결과로 오류 코드가 리턴되는 경우 수행할 작업을 표시하도록 SQLERROR를 지정하십시오.

## NOT FOUND

다음과 같은 이유로 SQLCODE +100 및 SQLSTATE '02000'이 리턴되는 경우 수행할 작업을 표시하도록 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 TO문은 다음과 같습니다.

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

예를 들어 커서를 사용해 행을 검색 중이면 FETCH문 발행 시 SQL이 결국 다른 행을 찾을 수 없게 됩니다. 이 상황에 대처하기 위해 WHENEVER NOT FOUND GO TO...문을 지정하여 커서를 올바르게 닫기 위해 CLOSE문을 발행하는 프로그램 내의 특정 위치로 SQL을 분기시키십시오.

주: WHENEVER문은 다른 WHENEVER가 발견될 때까지 모든 후속 소스 SQL문에 영향을 줍니다

다시 말하면 두 개의 WHENEVER문 사이에서 작성된 모든 SQL문(또는 만약 하나밖에 없을 경우 첫 번째 다음)은 프로그램이 지정하는 경로에 관계없이 첫 번째 WHENEVER문에 의해 관리됩니다.

이로 인해 WHENEVER문은 영향을 받게 되는 첫 번째 SQL문 앞에 와야 합니다. WHENEVER가 SQL문 다음에 오는 경우, 분기는 SQL문에 의해 설정된 SQLCODE와 SQLSTATE의 값에 기초하지 않습니다. 그러나 사용자의 프로그램이 SQLCODE 또는 SQLSTATE를 직접 체크할 경우 이 체크는 SQL문 수행 후에 실행되어야 합니다.

WHENEVER문은 서브루틴 옵션에 CALL을 제공하지 않습니다. 이런 이유로 사용자는 각 SQL문이 수행된 후에 SQLCODE 또는 SQLSTATE 값을 검토하고 WHENEVER문을 사용하기보다는 서브루틴을 호출하려고 할 수도 있을 것입니다.

### 관련 개념

125 페이지의 『REXX 어플리케이션에서 SQL문 코딩』

REXX 프로시듀어는 사전처리할 필요가 없습니다. 실행 시에 REXX 인터프리터는 자신이 이해하지 못하는 명령문을 현재 사용 중인 명령 환경으로 전달하여 처리합니다.

---

## C 및 C++ 어플리케이션에서 SQL문 코딩

이 주제에서는 C 또는 C++ 프로그램에 SQL문을 삽입하기 위한 고유의 어플리케이션과 코딩 요구사항에 대해 설명합니다.

C 프로그램은 iSeries용 ILE C 프로그램을 말합니다. C++ 프로그램은 ILE C++ 프로그램을 말합니다. 또한 이 주제에서는 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다. 자세한 내용은 다음 절을 참조하십시오.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

### 관련 개념

2 페이지의 『SQL을 사용하는 어플리케이션 작성』

iSeries용 DB2 UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성할 수 있습니다.

146 페이지의 『SQL을 사용하는 어플리케이션 프로그램의 컴파일 시 오류 및 경고 메시지』

다음은 컴파일 처리 시 오류 또는 경고 메시지를 발생시킬 수도 있는 조건에 대한 설명입니다.

### 관련 참조

150 페이지의 『iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램』

이 주제에는 iSeries용 DB2 UDB 시스템에서 지원되는 각 언어로 SQL문을 코딩하는 방법을 나타내는 샘플 어플리케이션이 들어 있습니다.

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

C 또는 C++ 프로그램은 SQLCA를 사용하여 삽입된 SQL문의 리턴 상태를 확인하거나 SQL 진단 영역을 사용하여 리턴 상태를 확인할 수 있습니다.

SQLCA를 사용할 경우, 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. 직접 코딩

| 하는 경우 다음 명령문을 사용하여 SQLCA를 초기화하십시오.

```
| struct sqlca sqlca = {0x0000000000000000};
```

SQL INCLUDE문을 사용할 때에는 다음의 표준 선언 부분이 포함되어야 합니다.

```
EXEC SQL INCLUDE SQLCA ;
```

- 1 표준 선언에는 구조 정의와 sqlca라는 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 = {0x0000000000000000};
```

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. 이로 인해 C나 C++ 프로그램으로부터 sqlerrmc 값을 인쇄하는 경우 예상하지 못한 결과가 나타나기도 합니다

#### 관련 개념

10 페이지의 『SQL 진단 영역 사용』

SQL 진단 영역을 사용하여 프로그램에서 실행되는 SQL문에 대해 리턴된 정보를 보존합니다. SQLCA를 통해 어플리케이션 프로그래머로서 사용자가 사용할 수 있는 모든 정보가 들어 있습니다.

#### 관련 정보

SQL 통신 영역

GET DIAGNOSTICS

## SQL을 사용하는 C 및 C++ 어플리케이션에서 SQL 설명자 영역 정의

| SQL 설명자 영역에는 두 가지 유형이 있습니다. 하나는 ALLOCATE DESCRIPTOR문으로 정의됩니다. 다른 하나는 SQL 설명자 영역(SQLDA) 구조를 사용하여 정의됩니다. 여기서는 SQLDA 형식에 대해서만 설명합니다.

| 다음 명령문은 SQLDA를 사용할 수 있습니다.

- EXECUTE...USING DESCRIPTOR *descriptor-name*
- FETCH...USING DESCRIPTOR *descriptor-name*
- OPEN...USING DESCRIPTOR *descriptor-name*
- DESCRIBE *statement-name* INTO *descriptor-name*
- DESCRIBE INPUT *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. 다음은 SQLDA를 필요로 하는 명령문 리스트입니다. SQLDA는 직접 또는 SQL INCLUDE문을 사용해 C나 C++ 프로그램에서 코드화될 수 있습니다. SQL INCLUDE문 사용 시 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 sqldaid[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;
    } 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 새 값으로 설정합니다.

#### **GETSQLDALONGLEN(daptr,n)**

daptr이 지정한 SQLDA의 n번째 항목의 길이 속성을 리턴합니다. SQLDA가 두 배가 되고 n번째 SQLVAR 항목이 LOB 자료 유형인 경우에만 이를 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 포인터 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 고급 프로그래밍 기술입니다. 동적 SQL을 사용하면 프로그램 수행 중 사용자 프로그램이 SQL 문을 개발 및 수행할 수 있습니다. 동적으로 수행하는 가변 SELECT 리스트(조회된 한 부분으로 리턴될 자료의 리스트)가 있는 SELECT문은 SQL 설명자 영역(SQLDA)을 필요로 합니다. 왜냐하면 사용자가 SELECT의 결과를 받아들이기 위해 지정된 변수의 수와 유형을 미리 알 수 없기 때문입니다.

#### 관련 정보

동적 SQL 어플리케이션

SQL 설명자 영역

## SQL을 사용하는 C 및 C++ 어플리케이션에 SQL문 삽입

SQL문은 실행 가능한 명령문이 표시되는 모든 C 또는 C++ 프로그램에서 작성할 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 ;
```

## 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 SQL문에는 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'<AABCCDDEEFFGGHH>
<IIJJKK>';
```

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

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

## SQL을 사용하는 C 및 C++ 어플리케이션에 코드 포함

다음 SQL문을 소스 코드에 삽입하여 SQL문, C 또는 C++문을 포함할 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

CRTSQLCI 또는 CRTSQLCPPI 명령의 MARGINS 매개변수로 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.

### 관련 개념

189 페이지의 『호스트 언어 사전컴파일러용 iSeries용 DB2 UDB CL 명령 설명』

iSeries용 DB2 UDB는 다음 프로그래밍 언어에서 코딩된 사전컴파일러 프로그램에 대한 명령을 제공합니다.

## 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 DSN으로 시작하는 호스트 변수 이름이나 외부 항목 이름을 사용하지 마십시오. 이러한 이름들은 데이터베이스 관리자 예약어입니다. 호스트 변수명의 길이는 128자로 제한됩니다.

- 1 대문자 또는 소문자 조합으로 SQL 이름을 사용할 경우 예상치 못한 결과가 발생할 수 있습니다.

## SQL을 사용하는 C 및 C++ 어플리케이션의 NULL 및 NUL

C, C++ 및 SQL 모두는 단어 널(null)을 사용하지만 다른 의미를 갖습니다.

C와 C++ 언어에는 널 문자(NUL), 널 포인터(NULL) 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 널은 모든 비 널과 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 처음으로 사용하기 전에 명시적으로 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 SECTION 문 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 존재하더라도 프로그램 내에서는 고유해야 합니다.

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

호스트 변수는 통합 요소가 될 수 없습니다.

호스트 변수는 이름에 계속 표시 문자를 포함할 수 없습니다.

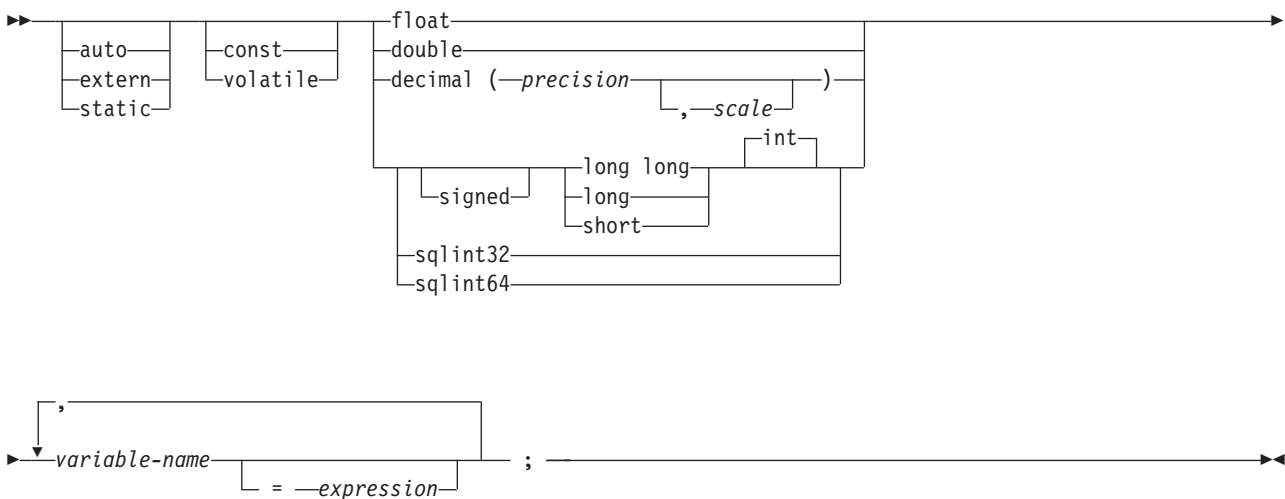
### SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 변수 선언

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

### SQL을 사용하는 C 및 C++ 어플리케이션의 숫자 호스트 변수:

이 주제에는 유효한 숫자 호스트 변수 선언 부분에 대한 구문이 포함됩니다.

#### 숫자



#### 주:

1. precision 및 scale은 정수 상수이어야 합니다. precision의 범위는 1 - 63 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 세 가지의 유효 양식이 있습니다.

세 가지 양식은 다음과 같습니다.

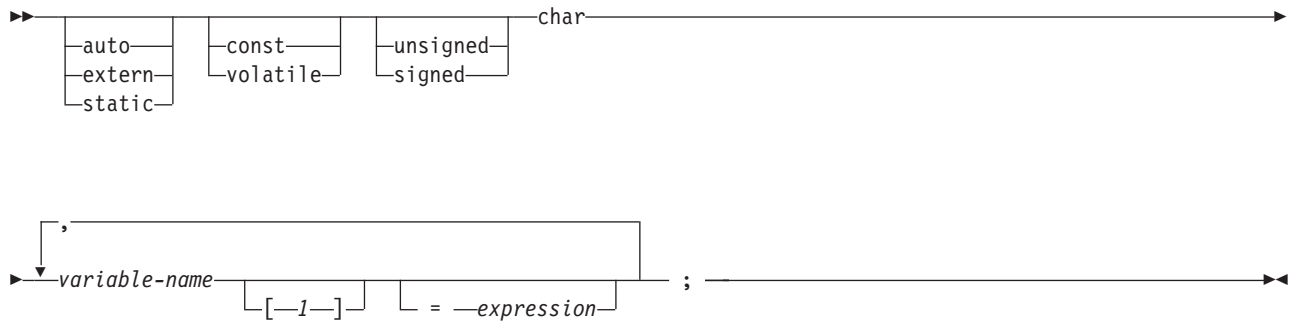
- 단일 문자 양식
- NUL 종료 문자 양식

- VARCHAR 구조화 양식

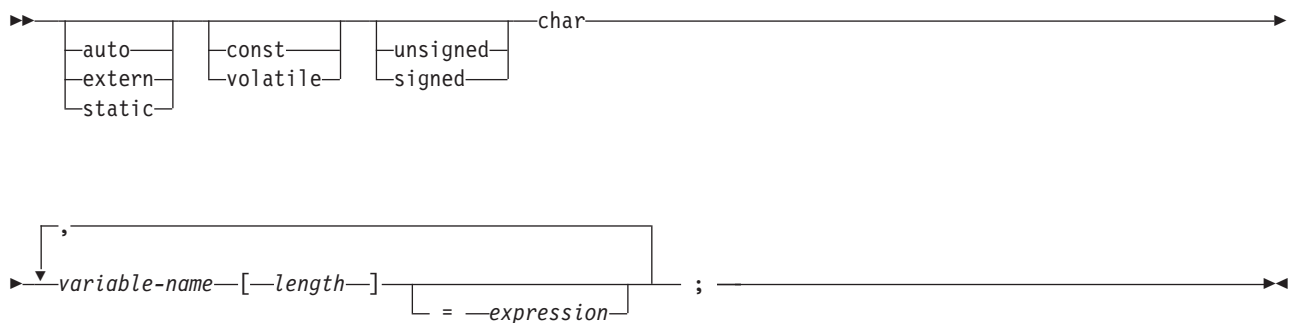
또한, SQL VARCHAR 선언은 varchar 호스트 변수를 정의하는데 사용될 수 있습니다.

모든 문자 유형은 부호없이 처리됩니다.

### 단일 문자 양식



### NUL 종료 문자 양식



주:

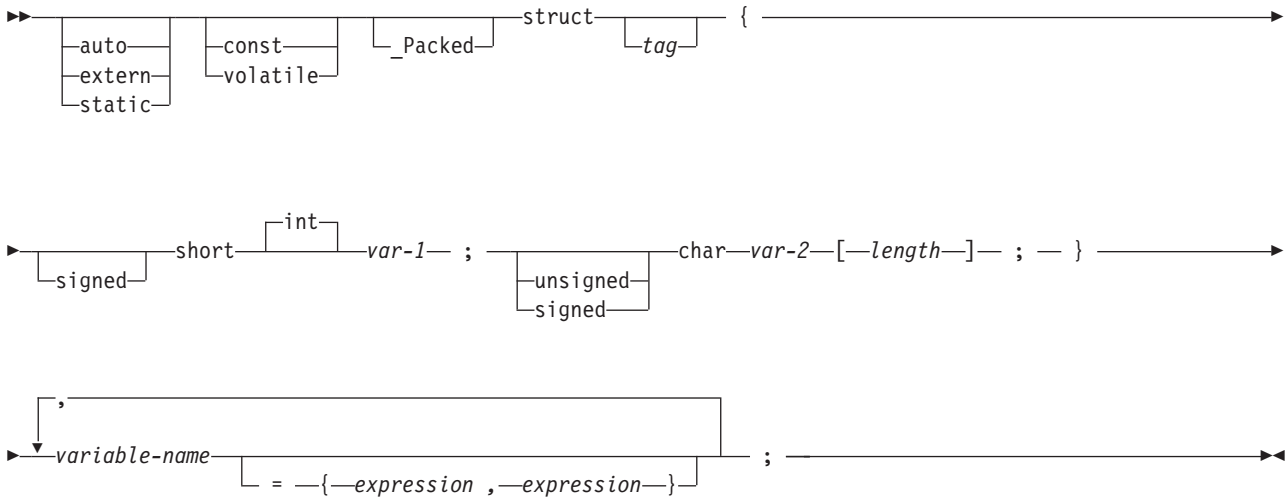
1. length는 1보다 크고 32 741 이하의 정수 상수여야 합니다.
2. \*CNULRQD 옵션이 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정될 경우 입력 호스트 변수에는 NUL 종료자가 포함되어야 합니다. 출력 호스트 변수는 공백 사이에 있으며 마지막 문자는 NUL 종료자입니다. 출력 호스트 변수가 너무 작아서 자료 및 NUL 종료자를 포함할 수 없다면 다음과 같은 경우가 발생합니다.
  - 자료가 절단됩니다.
  - 마지막 문자가 NUL 종료자입니다.
  - SQLWARN1이 'W'에 설정됩니다.
3. \*CNULRQD 옵션이 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정될 경우 입력 변수에 NUL 종료자를 포함시킬 필요가 없습니다.

다음은 출력 호스트 변수에 적용되는 사항입니다.

- 단일 호스트 변수가 자료 및 NUL 종료자를 포함할 만큼 충분히 크다면 다음과 같은 경우가 발생합니다.

- 자료가 리턴되지만 공백 사이에 들어가지는 않습니다.
- NUL 종료자가 자료 바로 뒤에 옵니다.
- 만일 호스트 변수가 자료를 포함할 만큼 충분히 크지만 NUL 종료자는 포함할 수 없다면 다음과 같은 경우가 발생합니다.
  - 자료가 리턴됩니다.
  - NUL 종료자가 리턴되지 않습니다.
  - SQLWARN1이 'n'에 설정됩니다.
- 만일 호스트 변수가 자료를 포함할 만큼 충분히 크지 않다면 다음과 같은 경우가 발생합니다.
  - 자료가 절단됩니다.
  - NUL 종료자가 리턴되지 않습니다.
  - SQLWARN1이 'W'에 설정됩니다.

## VARCHAR 구조화 양식



주:

1. *length*는 0보다 크고 32 740 이하의 정수 상수여야 합니다.
2. *var-1*과 *var-2*는 단순 변수 참조여야 하며 정수와 문자 호스트 변수로 개별적으로 사용될 수 없습니다.
3. `struct tag`는 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):

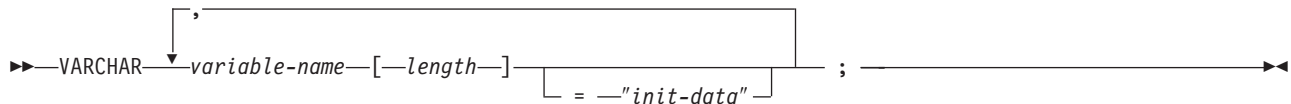
```
EXEC SQL BEGIN DECLARE SECTION;
/* valid declaration of host variable vstring */

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 32 740 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"};

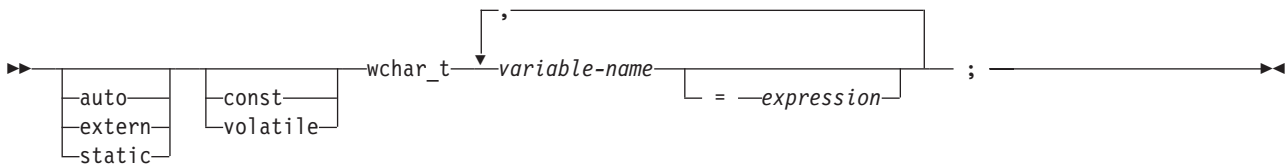
```

**SQL을 사용하는 C 및 C++ 어플리케이션의 그래픽 호스트 변수:**

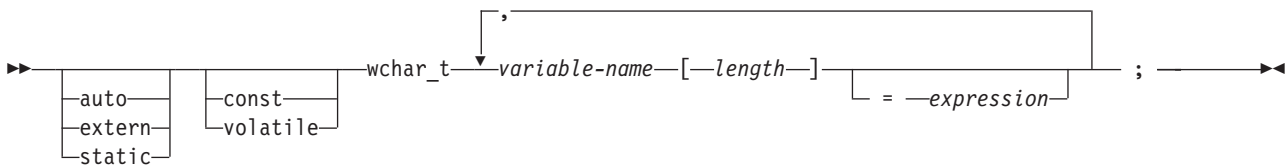
그래픽 호스트 변수에는 세 가지의 유효 양식이 있습니다.

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

**단일 그래픽 양식**



**NUL 종료 그래픽 양식**



주:

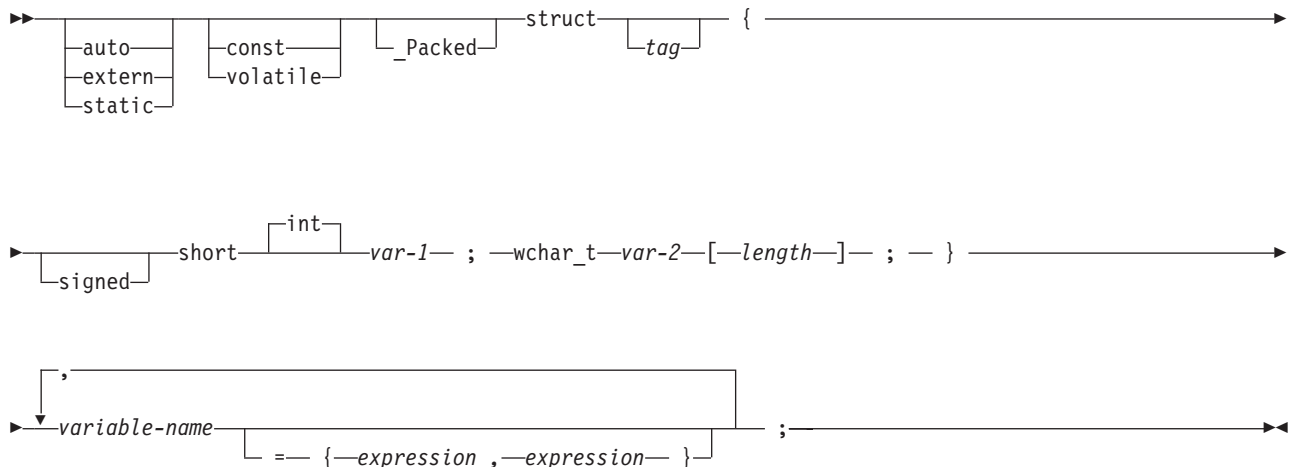
1. `length`는 1보다 크고 16371 이하의 정수 상수이어야 합니다.
2. \*CNULRQD 옵션을 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정할 경우 입력 호스트 변수에는 그래픽 NUL 종료자(/0/0)가 포함되어야 합니다. 출력 호스트 변수는 DBCS 공백 사이에 놓이며 최종 문자는 그래픽 NUL 종료자입니다. 출력 호스트 변수가 너무 작아서 자료 및 NUL 종료자를 포함할 수 없다면 다음과 같은 경우가 발생합니다.
  - 자료가 절단됩니다.
  - 최종 문자로 그래픽 NUL 종료자가 사용됩니다.
  - SQLWARN1이 'W'에 설정됩니다.

\*NOCNULRQD 옵션을 CRTSQLCI 또는 CRTSQLCPPI 명령에 지정할 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 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.

## VARGRAPHIC 구조화 양식



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;

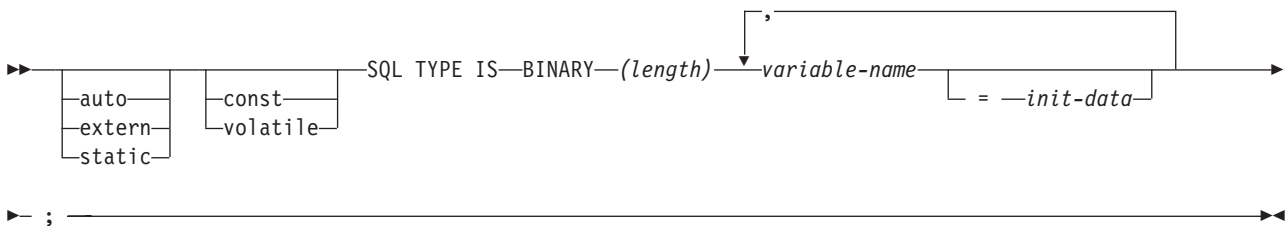
/* invalid declaration of host variable wstring */

struct VARGRAPH wstring;
```

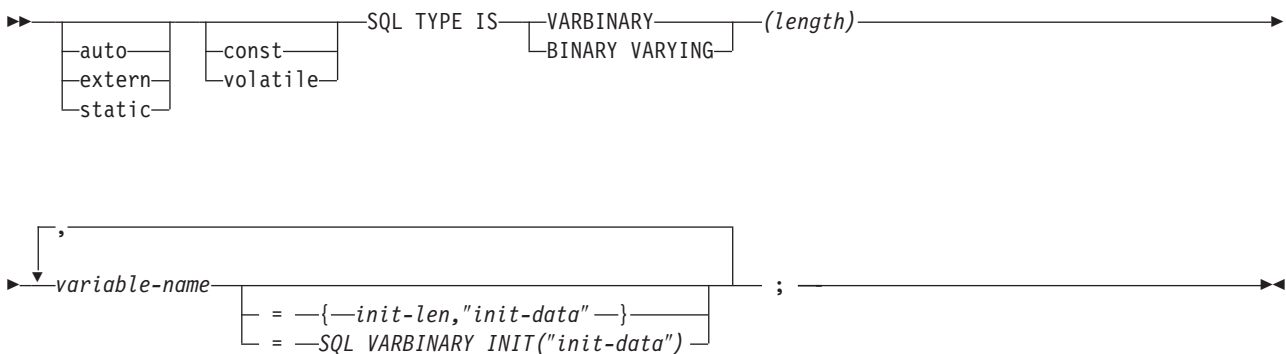
## SQL을 사용하는 C 및 C++ 어플리케이션의 2진 호스트 변수:

C 및 C++에는 SQL 2진 자료 유형에 해당되는 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.

## BINARY



## VARBINARY



## 주:

1. BINARY 호스트 변수의 경우, 길이의 범위는 1 - 32766입니다.
2. VARBINARY 및 BINARY VARYING 호스트 변수의 경우, 길이의 범위는 1 - 32740입니다.
3. SQL TYPE IS, BINARY, VARBINARY 및 BINARY VARYING은 대소문자를 혼합하여 사용할 수 있습니다.

## BINARY 예

다음 선언을 사용할 경우:

```
SQL TYPE IS BINARY(4) myBinField;
```

다음과 같은 코드가 생성됩니다.

```
unsigned char myBinField[4];
```

## VARBINARY 예

다음 선언을 사용할 경우:

```
SQL TYPE IS VARBINARY(12) myVarBinField;
```

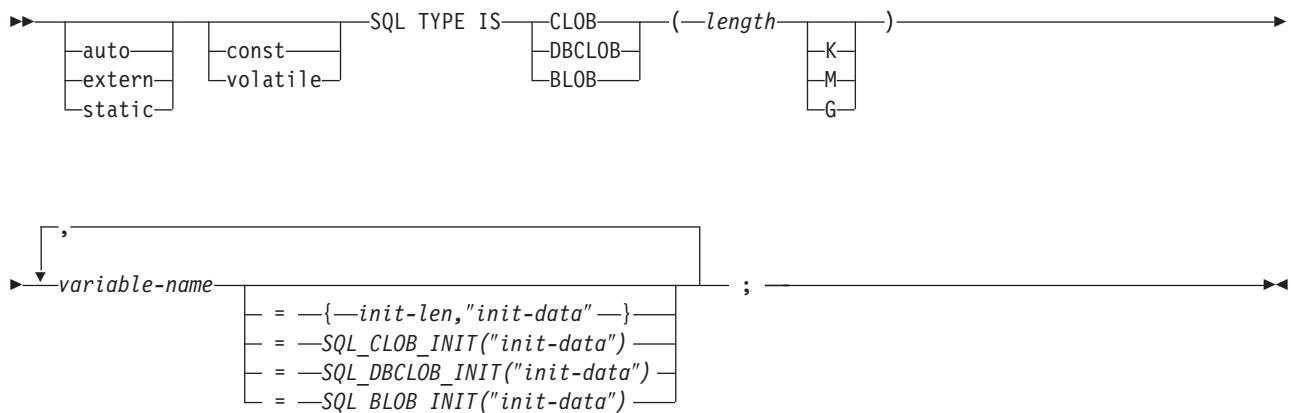
다음과 같은 구조가 생성됩니다.

```
_Packed struct myVarBinField_t {  
    short length;  
    char data[12]; }  
myVarBinField;
```

## 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.

## LOB 호스트 변수



주:

1. K는 *length*에 1024를 곱합니다. M은 *length*에 1 048 576을 곱합니다. G는 *length*에 1 073 741 824를 곱합니다.
2. BLOB 및 CLOB의 경우  $1 \leq length \leq 2\,147\,483\,647$
3. DBCLOB의 경우  $1 \leq length \leq 1\,073\,741\,823$
4. SQL TYPE IS, BLOB, CLOB, DBCLOB, K, M, G는 대소문자를 혼합하여 사용할 수 있습니다.

5. 초기화 스트링에 허용되는 최대 길이는 32 766바이트입니다.
6. 초기화 길이인 *init-len*은 숫자 상수(즉, K, M 또는 G를 포함할 수 없음)여야 합니다.
7. 선언 내에서 LOB가 초기화되지 않은 경우 사전컴파일러 생성 코드 내에서 어떤 초기화도 수행되지 않습니다.
8. 사전컴파일러는 호스트 변수 유형에 대해 캐스트하는 데 사용될 수 있는 구조 태그를 생성합니다.
9. LOB 호스트 변수에 대한 포인터는 다른 호스트 변수 유형에 대한 포인터와 같은 규칙 및 제한사항을 적용하여 선언할 수 있습니다.
10. LOB 호스트 변수에 대한 CCSID 처리는 다른 문자 및 그래픽 호스트 변수 유형에 대한 처리와 같습니다.
11. DBCLOB가 초기화되면 사용자가 스트링 앞에 접두부로 'L'을 붙여야 합니다(와이드 문자 스트링을 나타냄).

## 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;
```

## BLOB 예

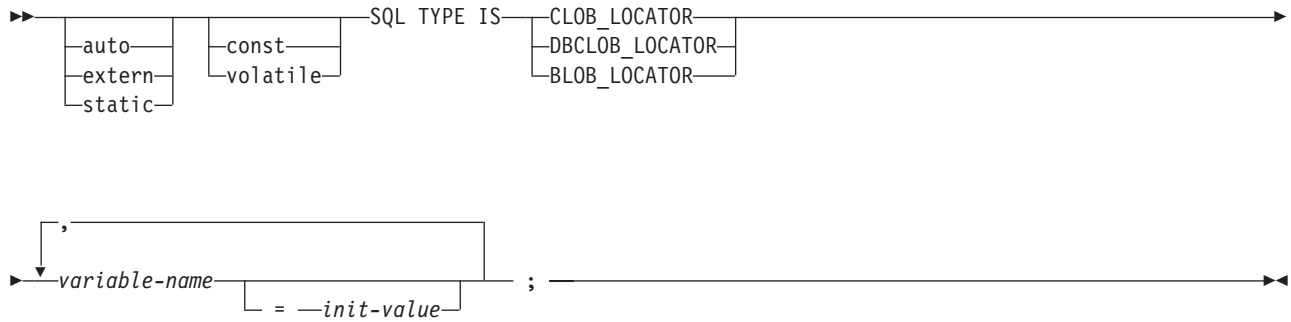
다음 선언을 사용할 경우:

```
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");
```

## LOB 위치 지정자



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:

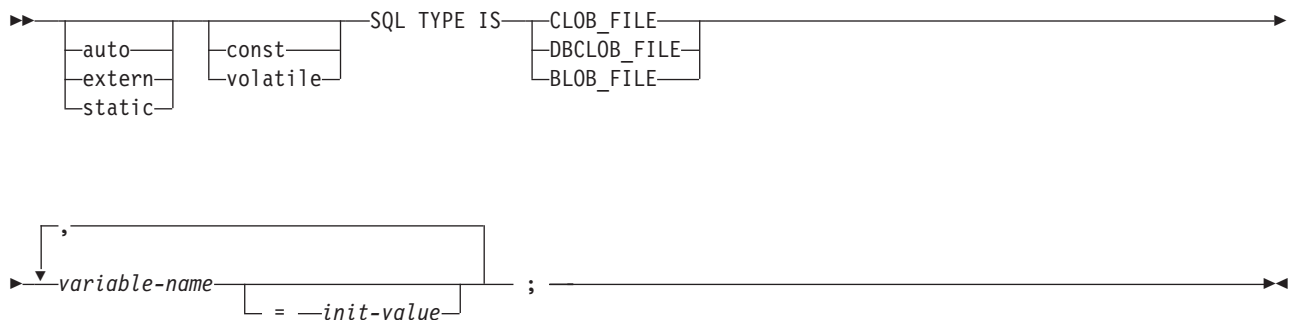
```
static SQL TYPE IS CLOB_LOCATOR my_locator;
```

다음과 같이 생성됩니다.

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

BLOB 및 DBCLOB 위치 지정자의 구문은 비슷합니다.

## LOB 파일 참조 변수



주:

1. `SQL TYPE IS`, `BLOB_FILE`, `CLOB_FILE`, `DBCLOB_FILE`은 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 다음 상수를 사용하여 file\_options 변수를 설정할 수 있습니다.

- SQL\_FILE\_READ (2)
- SQL\_FILE\_CREATE (8)
- SQL\_FILE\_OVERWRITE (16)
- SQL\_FILE\_APPEND (32)

관련 정보

LOB 파일 참조 변수

## 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.

## ROWID

```
▶▶ SQL TYPE IS ROWID variable-name ;
```

주: 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.

```
_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.

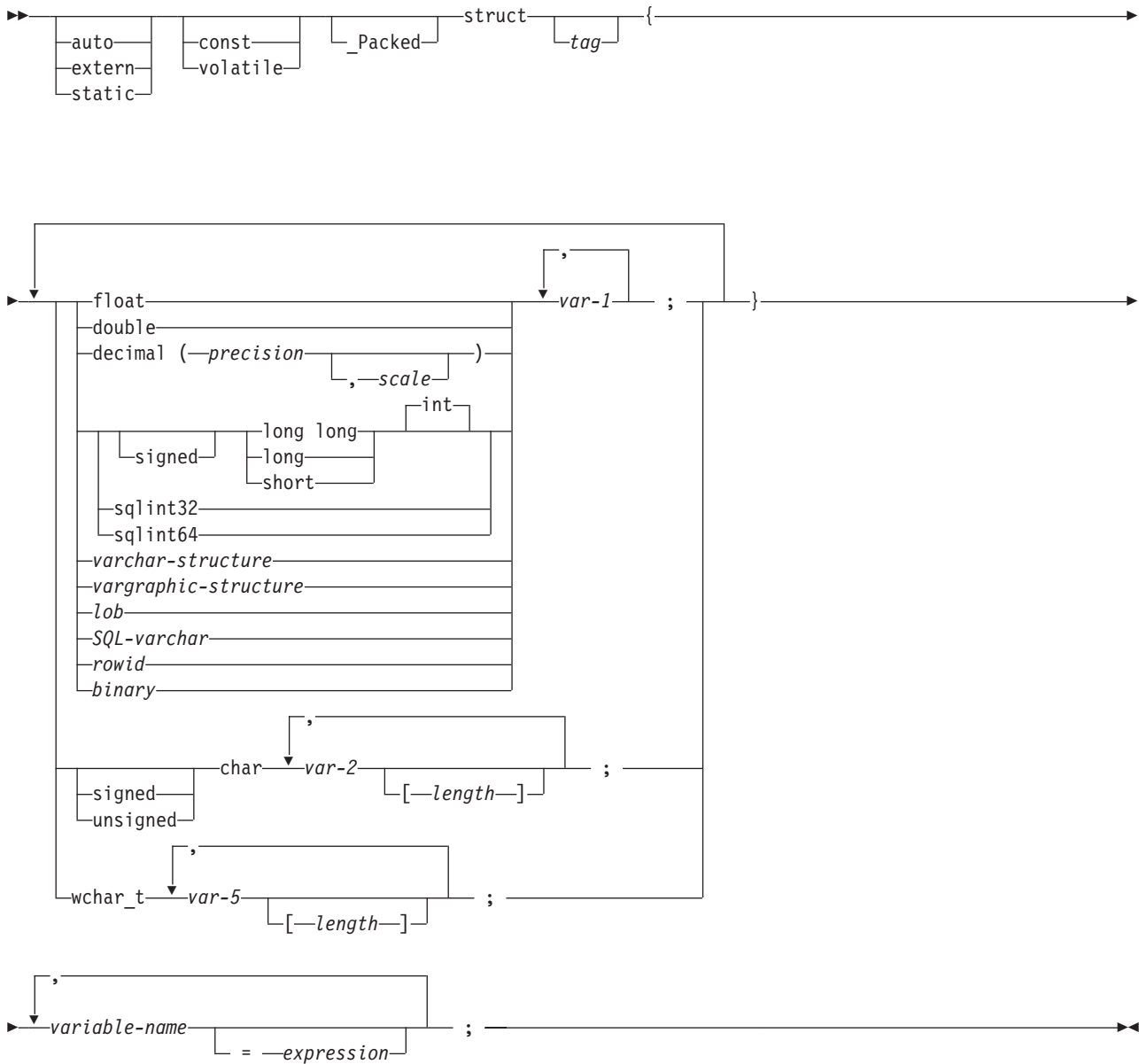
```
struct { char empno[7];
        struct { short int firstname_len;
                char firstname_text[12];
                } firstname;
        char midint,
        struct { short int lastname_len;
                char lastname_text[15];
                } lastname;
        char workdept[4];
    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

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

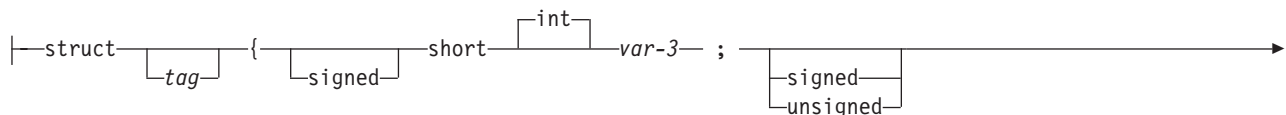
## SQL을 사용하는 C 및 C++ 어플리케이션의 호스트 구조 선언

다음 그림은 호스트 구조 선언에 대한 유효한 구문을 보여줍니다.

### 호스트 구조



### varchar-structure:





▶char-var-4-[length]- ; -}

## 호스트 구조(계속)

### vargraphic-structure:

|struct-tag-{signed-short-int-var-6 ; wchar\_t-var-7-[length]- ; -}

### lob:

|SQL TYPE IS CLOB (-length) |  
DBCLOB		K
BLOB		M
		G
CLOB\_LOCATOR		
DBCLOB\_LOCATOR		
BLOB\_LOCATOR		
CLOB\_FILE		
DBCLOB\_FILE		
BLOB\_FILE		

### SQL-varchar:

|VARCHAR-variable-name-[length]-

### rowid:

|SQL TYPE IS ROWID-

### binary:

|SQL TYPE IS BINARY (-length)-  
|VARBINARY |  
|BINARY VARYING |

### 주:

1. 숫자, 문자, 그래픽, LOB, ROWID 및 2진 호스트 변수의 선언에 대한 자세한 내용은 숫자, 문자, 그래픽, LOB, ROWID 및 2진 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.

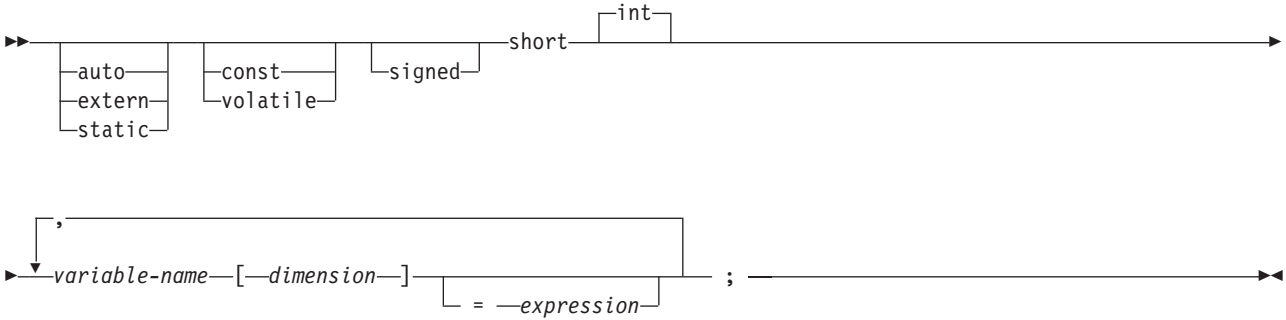
```
#pragma pack(1)
struct {
    short myshort;
    long mylong;
    char mychar[5];
} a_st;
#pragma pack()
```

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

## SQL을 사용하는 C 및 C++ 어플리케이션의 호스트 구조 인디케이터 배열

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

### 호스트 구조 인디케이터 배열



주: dimension은 1-32767 사이의 정수 상수이어야 합니다.

## SQL을 사용하는 C 및 C++ 어플리케이션에서 호스트 구조 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_array`의 모든 멤버가 유효한 변수 선언이어야 합니다.
- `_Packed` 속성은 `struct` 태그에 지정되어야 합니다.
- `b_array`는 `c1_var` 및 `c2_var` 멤버를 포함하는 호스트 구조의 배열 이름입니다.
- `b_array`는 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.

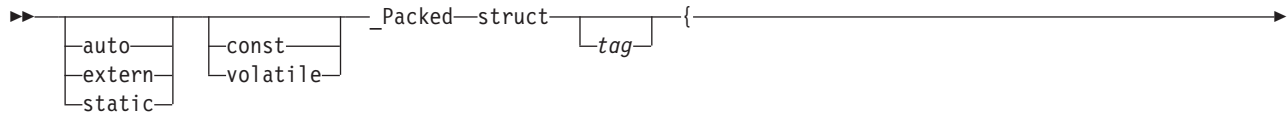
```

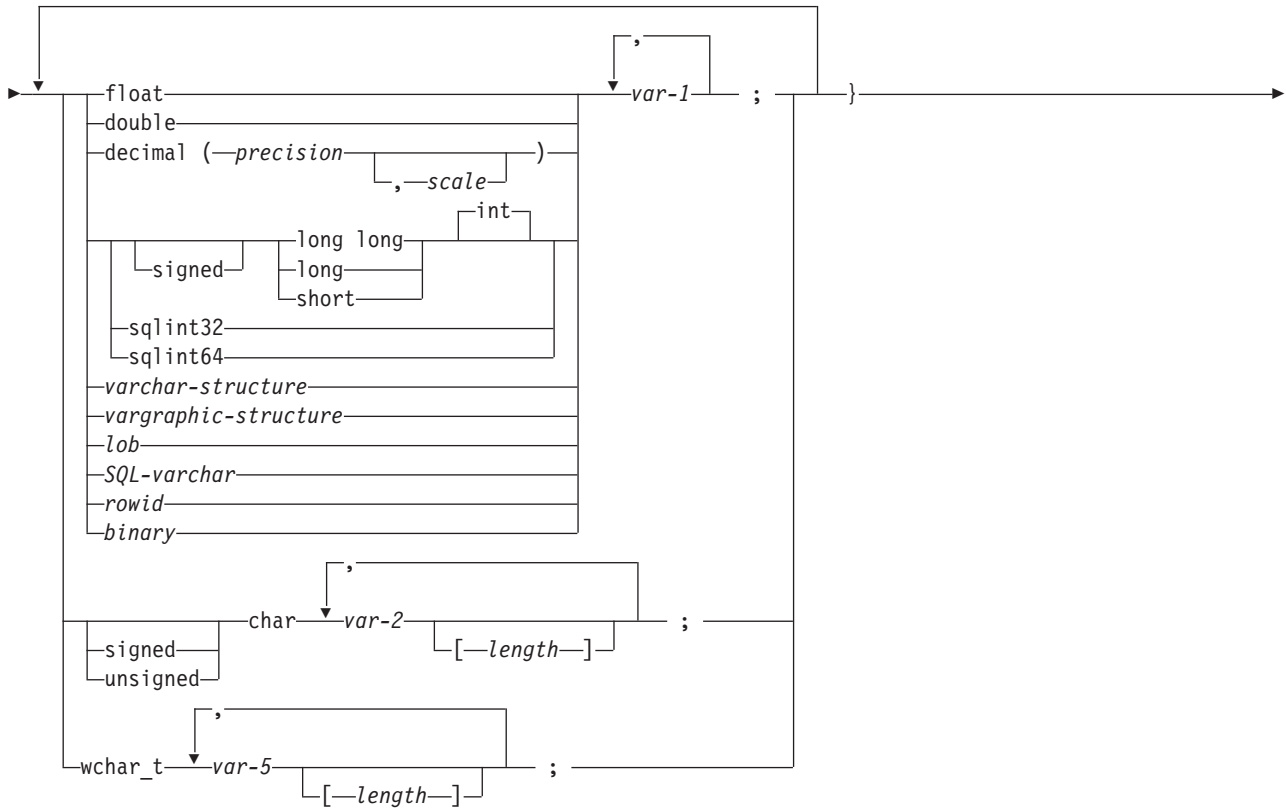
_Packed struct {char first_initial;
                char middle_initial;
                _Packed struct {short lastname_len;
                char lastname_data[15];
                                } lastname;
                double total_salary;
            } employee_rec[10];
struct { short inds[4];
        } 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;...

```

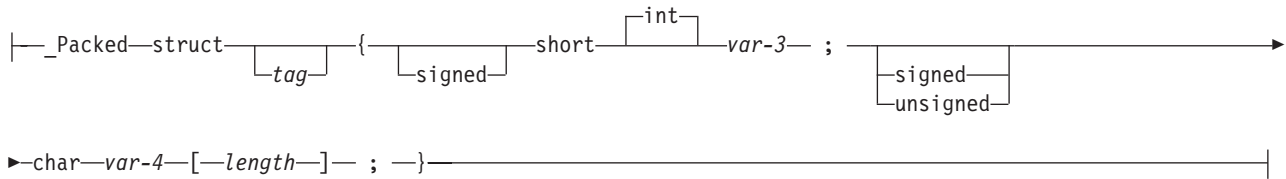
### SQL을 사용하는 C 및 C++ 어플리케이션의 호스트 구조 배열

다음 그림은 호스트 구조 배열 선언에 대한 유효한 구문을 보여줍니다.

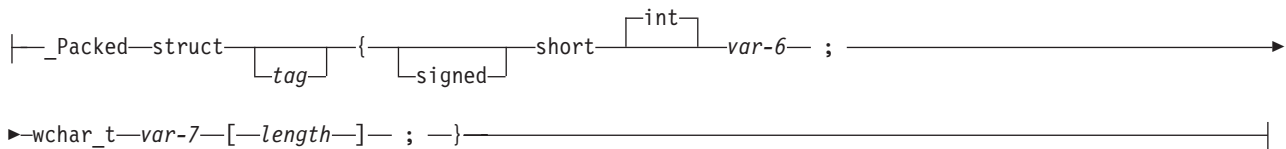




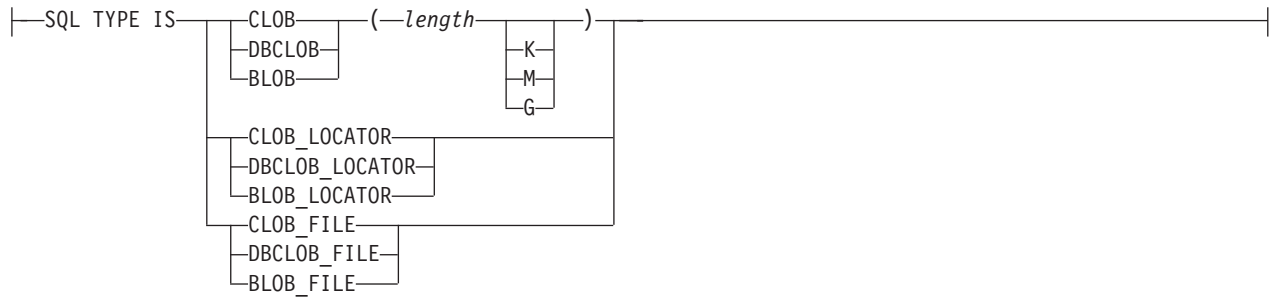
**varchar-structure:**



**vargraphic-structure:**



**lob:**



**SQL-varchar:**



**rowid:**



**binary:**



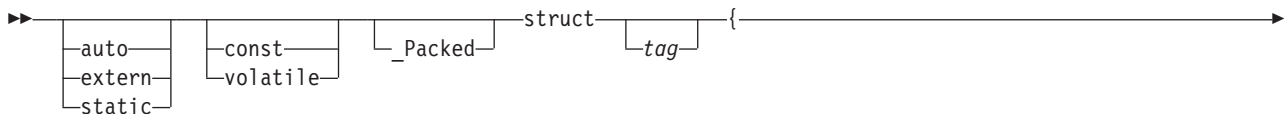
**주:**

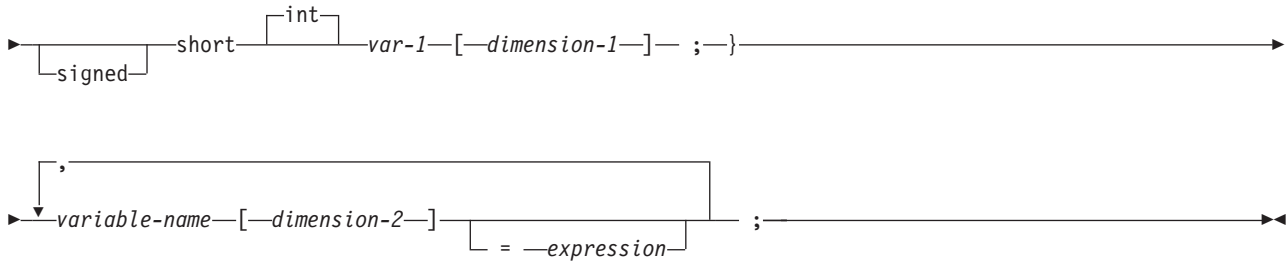
- 숫자, 문자, 그래픽, LOB, ROWID 및 2진 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트, 그래픽 호스트 변수, LOB 호스트 변수, ROWID 호스트 변수 및 2진 호스트 변수 아래에 있는 주를 참조하십시오.
- struct tag는 다른 자료 영역을 정의하는 데 사용될 수 있으나 호스트 변수로는 사용될 수 없습니다.
- dimension은 1-32767 사이의 정수 상수이어야 합니다.
- \_Packed는 C++에서 사용될 수 없습니다. 대신, 선언 이전에 #pragma pack(1)를 지정하고 선언 후 #pragma pack()를 지정하십시오.
- sqlint32 또는 sqlint64를 사용 중이면 sqlsystem.h 헤더 파일이 포함되어야 합니다.

**SQL을 사용하는 C 및 C++ 어플리케이션의 호스트 구조 배열 인디케이터 구조**

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

**호스트 구조 배열 인디케이터 구조**





주:

1. struct tag는 기타 자료 영역을 정의하는 데 사용될 수 있으나 호스트 변수로서 사용될 수는 없습니다.
2. dimension 1과 dimension 2는 1-32767 사이의 정수 상수이어야 합니다.
3. \_Packed는 C++에서 사용될 수 없습니다. 대신, 선언 이전에 #pragma pack(1)를 지정하고 선언 후 #pragma pack()를 지정하십시오.

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

또한 다음의 제한사항과 함께 지원되는 C와 C++ 자료 유형에 대한 포인터인 호스트 변수를 선언할 수 있습니다.

- 호스트 변수가 포인터로서 선언되는 경우 호스트 변수는 그 앞에 별표(\*)가 붙은 호스트 변수이어야 합니다. 다음 예는 모두 유효합니다.

```
short *mynum;           /* Ptr to an integer           */
long **mynumptr;      /* Ptr to a ptr to a long integer */
char *mychar;         /* Ptr to a single character     */
char(*mychara)[20];   /* Ptr to a char array of 20 bytes */
struct {              /* Ptr to a variable char array of 30 */
    short mylen;      /* bytes.                          */
} *myvarchar;
char mydata[30];
```

주: 괄호는 NUL 종료 문자 배열에 포인터를 선언할 때만 허용되며 이 경우에는 반드시 괄호가 필요합니다. 괄호를 사용하지 않으면 원하는 배열에 대한 포인터 대신 포인터들의 배열을 선언하게 됩니다. 예를 들면 다음과 같습니다.

```
char (*a)[10];         /* pointer to a null-terminated char array */
char *a[10];          /* pointer to an array of pointers          */
```

- 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++ 어플리케이션에서 typedef 사용

typedef 선언을 사용하여 short, float 및 double과 같은 C 유형 지정자 대신에 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]; /* use for blocked insert or fetch */
```

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

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 typedef문이 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 없습니다. typedef 구조가 들어 있는 구조는 SQL문에서는 사용될 수 없습니다. typedef를 사용하여 선언된 구조는 사용될 수 있습니다.

iSeries용 DB2 UDB SQL 프로그래밍 주제 컬렉션의 iSeries용 DB2 UDB 샘플 표에 설명된 샘플 표 DEPARTMENT 정의를 검색하려면 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. GRAPHIC 또는 VARGRAPHIC 열에 UTF-16 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UTF-16 CCSID를 갖게 됩니다.



존, 2진(비 존 소수 자릿수 필드) 및 선택적으로 십진수가 iSeries용 ILE C 내의 문자 필드에 맵핑되어도 SQL에서는 이러한 필드를 숫자로 처리합니다. 확장 프로그램 모델(EPM) 루틴을 사용하여 존 및 팩 10진 자료를 변환하기 위해 이들 필드를 조작할 수 있습니다. 자세한 정보는 ILE C for iSeries Language

Reference  주제를 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 C 또는 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	1바이트에는 p, 2바이트에는 s	DECIMAL(p,s)
float	480	4	FLOAT(단정밀도)
double	480	8	FLOAT(배정밀도)
단일 문자 양식	452	1	CHAR(1)
NUL 종료 문자 양식	460	length	VARCHAR(길이 - 1)
VARCHAR 구조화 양식	448	length	VARCHAR(길이)
단일 그래픽 양식	468	1	GRAPHIC(1)
NUL 종료 단일 그래픽 양식	400	length	VARGRAPHIC(길이 - 1)
VARGRAPHIC 구조화 양식	464	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 - 63의 양수이고, s는 0 - 63의 양수입니다.
NUMERIC(p,s) 또는 0이 아닌 스케일 2진	같은 것이 없습니다.	DECIMAL 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 사용하십시오.

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

SQL 자료 유형	C 또는 C++ 자료 유형	주
VARCHAR(n)	NUL 종료 문자 양식	NUL 종료자를 수용하려면 최소한 $n+1$ 을 허용하십시오. 자료가 문자 NUL(#0)을 사용할 수 있는 경우 VARCHAR 구조화 양식 또는 SQL VARCHAR을 사용하십시오.  $n$ 은 양의 정수입니다. $n$ 의 최대값은 32740입니다.
	VARCHAR 구조화 양식	$n$ 의 최대값은 32740입니다. 또한 SQL VARCHAR 양식도 사용됩니다.
CLOB	없음	C 또는 C++로 CLOB를 선언하려면 SQL TYPE IS를 사용하십시오.
GRAPHIC(1)	단일 그래픽 양식	
GRAPHIC(n)	같은 것이 없습니다.	
VARGRAPHIC(n)	NUL 종료 그래픽 양식	$n > 1$ 인 경우 NUL 종료 그래픽 양식을 사용하십시오.
	VARGRAPHIC 구조화 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.
DBCLOB	없음	C 또는 C++로 DBCLOB를 선언하려면 SQL TYPE IS를 사용하십시오.
BINARY	없음	C 또는 C++로 BINARY를 선언하려면 SQL TYPE IS를 사용하십시오.
VARBINARY	없음	C 또는 C++로 VARBINARY를 선언하려면 SQL TYPE IS를 사용하십시오.
BLOB	없음	C 또는 C++로 BLOB를 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.

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

SQL 자료 유형	C 또는 C++ 자료 유형	주
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.
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 포함시키려면 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.

## C 및 C++ 변수 선언 및 사용에 대한 주

작은 따옴표와 큰 따옴표는 C, C++ 및 SQL에서 서로 다른 의미가 있습니다.

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 구분하는데 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. 검색 시 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.

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 :C1sCd,
                                :Bgn :BgnInd,
                                :End :EndInd;
```

이 경우 다음과 같이 변수를 선언할 수 있습니다.

```
EXEC SQL BEGIN DECLARE SECTION; char C1sCd[8];
char Bgn[9];
char End[9];
short Day, DayInd, BgnInd, EndInd;
EXEC SQL END DECLARE SECTION;
```

관련 참조

변수 참조

---

## COBOL 어플리케이션에서 SQL문 코딩

이 주제에서는 COBOL 프로그램에 SQL문을 삽입할 때의 고유한 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.

iSeries 시스템은 둘 이상의 COBOL 컴파일러를 지원합니다. DB2 UDB 조회 관리자 및 SQL 개발 킷 라이선스 프로그램만이 iSeries용 COBOL 및 iSeries용 COBOL 언어를 지원합니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

관련 개념

2 페이지의 『SQL을 사용하는 어플리케이션 작성』

iSeries용 DB2 UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성할 수 있습니다.

146 페이지의 『SQL을 사용하는 어플리케이션 프로그램의 컴파일 시 오류 및 경고 메시지』

다음은 컴파일 처리 시 오류 또는 경고 메시지를 발생시킬 수도 있는 조건에 대한 설명입니다.

관련 참조

150 페이지의 『iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램』

이 주제에는 iSeries용 DB2 UDB 시스템에서 지원되는 각 언어로 SQL문을 코딩하는 방법을 나타내는 샘플 어플리케이션이 들어 있습니다.

## SQL을 사용하는 COBOL 어플리케이션에서 SQL 통신 영역 정의

COBOL 프로그램은 SQLCA를 사용하여 삽입된 SQL문의 리턴 상태를 확인하거나 SQL 진단 영역을 사용하여 리턴 상태를 확인할 수 있습니다.

SQLCA 대신 SQL 진단 영역을 사용하려면 SQLCA = \*NO 옵션과 함께 SET OPTION SQL문을 사용하십시오.

SQLCA를 사용할 경우, SQL문이 들어 있는 COBOL 프로그램에는 반드시 다음 중 하나 또는 모두가 포함되어 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. 직접 코딩하는 경우 반드시 초기화해야 합니다. 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 사용 시 SQL COBOL 사전컴파일러는 SQLCA에 대한 COBOL 소스 명령문을 포함합니다.

```
01 SQLCA.  
  05 SQLCAID      PIC X(8).  
  VALUE X"0000000000000000".  
  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 GLOBAL 절을 사용하여 SQLCA를 선언합니다. 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.

관련 개념

10 페이지의 『SQL 진단 영역 사용』

SQL 진단 영역을 사용하여 프로그램에서 실행되는 SQL문에 대해 리턴된 정보를 보존합니다. SQLCA를 통해 어플리케이션 프로그래머로서 사용자가 사용할 수 있는 모든 정보가 들어 있습니다.

관련 정보

SQL 통신 영역

## SQL을 사용하는 COBOL 어플리케이션에서 SQL 설명자 영역 정의

SQL 설명자 영역에는 두 가지 유형이 있습니다. 하나는 ALLOCATE DESCRIPTOR문으로 정의됩니다. 다른 하나는 SQLDA 구조를 사용하여 정의됩니다. 여기서는 SQLDA 형식에 대해서만 설명합니다.

다음 명령문은 SQLDA를 사용할 수 있습니다.

- 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 INPUT *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 사용 시 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 SQLLIND    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 놓일 수 있습니다. iSeries용 ILE COBOL의 경우 SQLDA는 GLOBAL 절을 사용하여 선언됩니다.

동적 SQL은 고급 프로그래밍 기술입니다. 동적 SQL을 사용하면 프로그램 수행 중 사용자 프로그램이 SQL 문을 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.

#### 관련 정보

동적 SQL 어플리케이션

SQL 설명자 영역

## SQL을 사용하는 COBOL 어플리케이션에 SQL문 삽입

SQL문은 다음 표에서처럼 COBOL 프로그램 섹션에서 코딩될 수 있습니다.

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

## 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 SQL문에는 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 어플리케이션에 코드 포함

SQL문 또는 COBOL 호스트 변수 선언문은 명령문이 삽입될 소스 코드에 다음 SQL문을 삽입하여 포함시킬 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

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

### SQL을 사용하는 COBOL 어플리케이션의 순서 번호

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

### SQL을 사용하는 COBOL 어플리케이션의 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 어플리케이션의 COBOL 컴파일 시간 옵션

COBOL PROCESS문을 사용하여 COBOL 컴파일러에 대한 컴파일 시 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 \*APOST 및 \*QUOTE를 CRTSQLCBL 및 CRTSQLCBLI 명령의 OPTION 매개변수에 지정해야 합니다.

### SQL을 사용하는 COBOL 어플리케이션에서 명령문 레이블

PROCEDURE DIVISION의 실행 가능한 SQL문은 단락명 앞에 올 수 있습니다.

### SQL을 사용하는 COBOL 어플리케이션의 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 사용되는 모든 호스트 변수는 처음으로 사용하기 전에 명시적으로 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 SECTION 이 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.

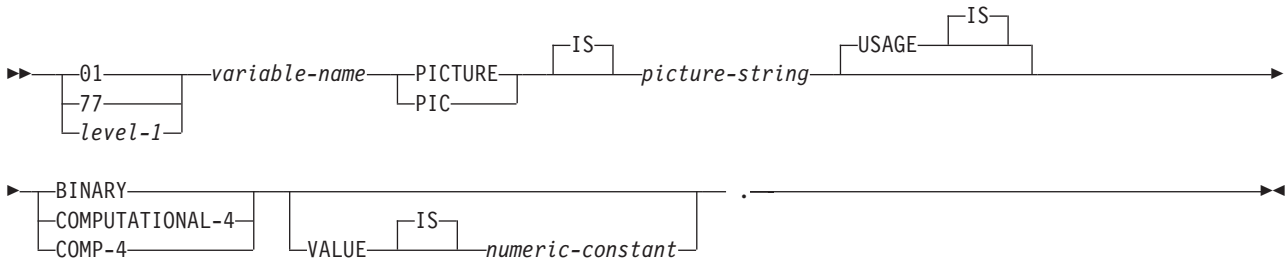
### SQL을 사용하는 COBOL 어플리케이션에서 호스트 변수 선언

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

### SQL을 사용하는 COBOL 어플리케이션의 숫자 호스트 변수:

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

### BIGINT, INTEGER 및 SMALLINT

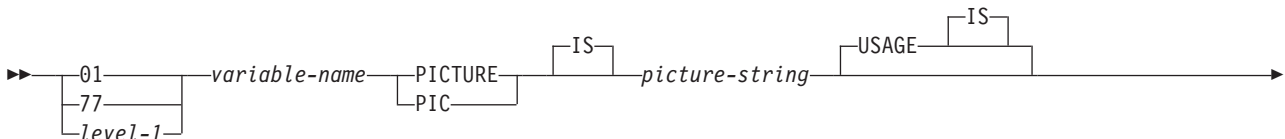


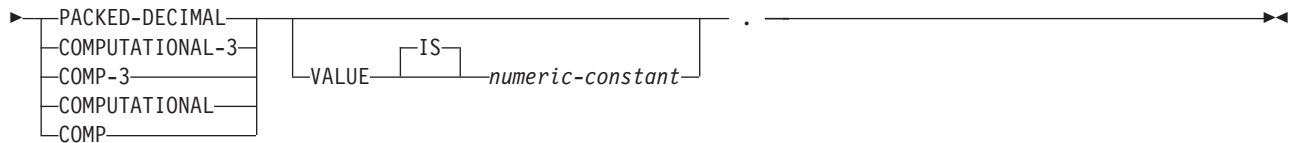
주:

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 이하여야 합니다.
2. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

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

### DECIMAL



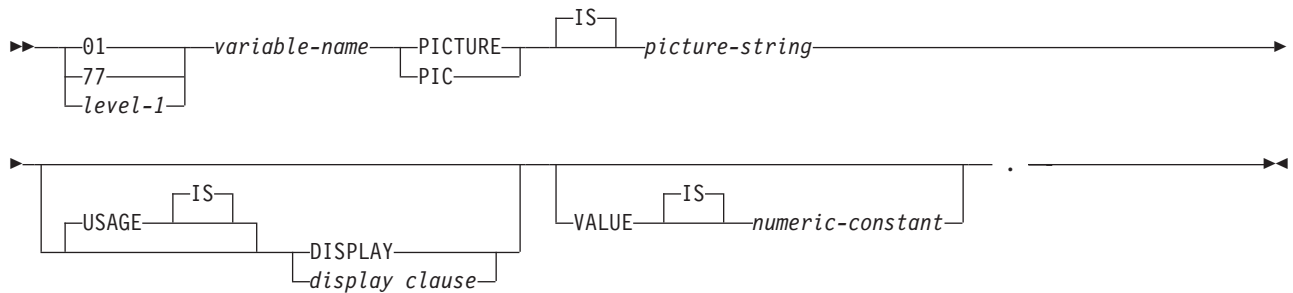


주:

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 63 이하여야 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 63 이하여야 합니다.
3. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

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

### 숫자



### display



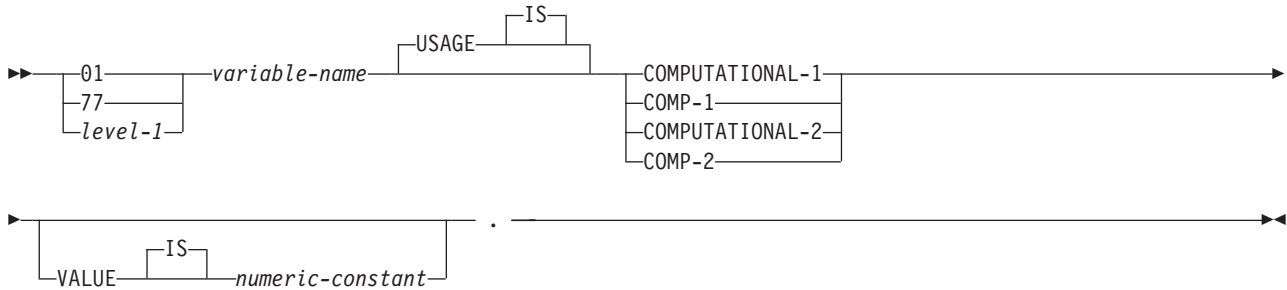
주:

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 이하여야 합니다.
2. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

### SQL을 사용하는 COBOL 어플리케이션의 부동 소수점 호스트 변수:

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

## 부동 소수점



주:

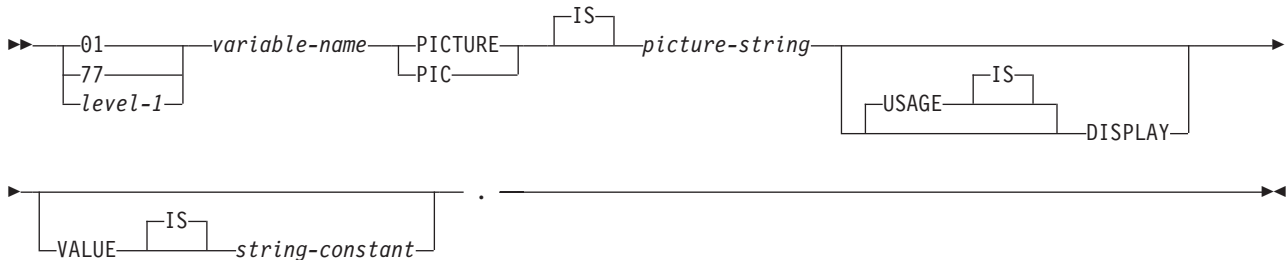
1. COMPUTATIONAL-1과 COMP-1은 동등합니다. COMPUTATIONAL-2와 COMP-2는 동등합니다.
2. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

## SQL을 사용하는 COBOL 어플리케이션의 문자 호스트 변수:

문자 호스트 변수에는 두 가지 유효한 양식이 있습니다.

- 고정 길이 스트링
- 가변 길이 스트링

### 고정 길이 문자 스트링

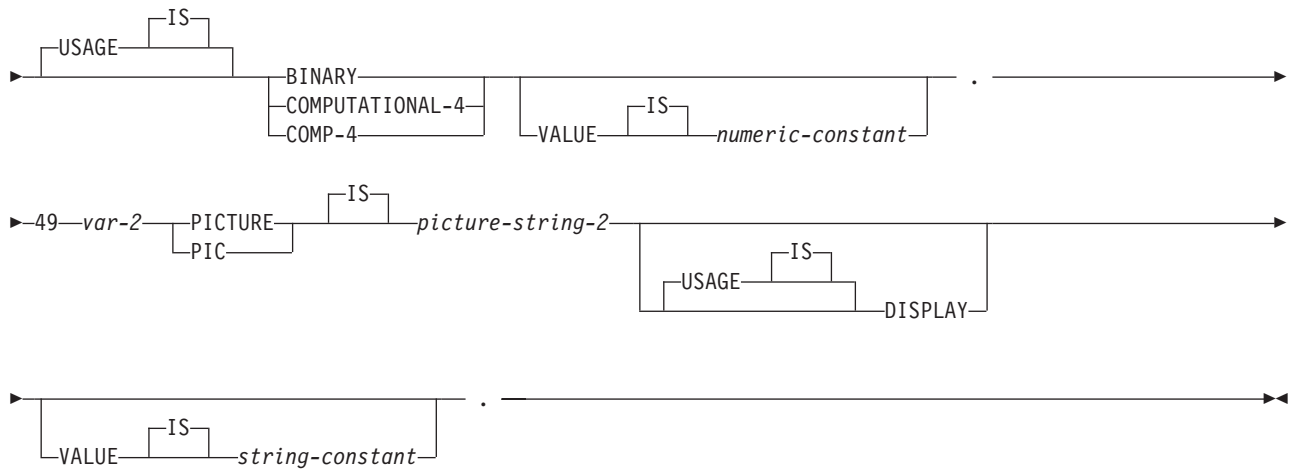


주:

1. 이러한 양식과 연관된 픽처 스트링은 X(m)(또는 XXX...X, X의 m 인스턴스)여야 하며  $1 \leq m \leq 32766$ 입니다.
2. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

### 가변 길이 문자 스트링





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 인스턴스)여야 하고  $1 \leq m \leq 32\,740$ 입니다.
3. *var-1*과 *var-2*는 호스트 변수로 사용될 수 없습니다.
4. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

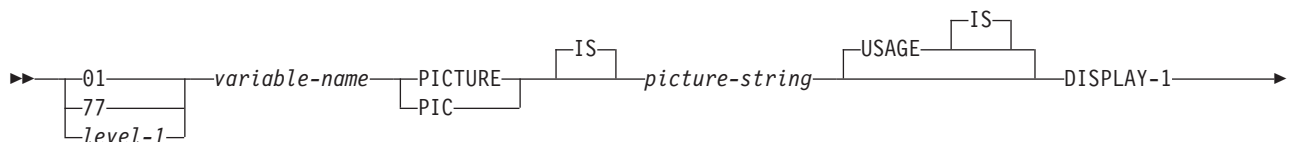
### SQL을 사용하는 COBOL 어플리케이션의 그래픽 호스트 변수:

그래픽 호스트 변수는 iSeries용 ILE COBOL에서만 지원됩니다.

문자 호스트 변수에는 다음의 두 가지 유효한 양식이 있습니다.

- 고정 길이 그래픽 스트링
- 가변 길이 그래픽 스트링

### 고정 길이 그래픽 스트링

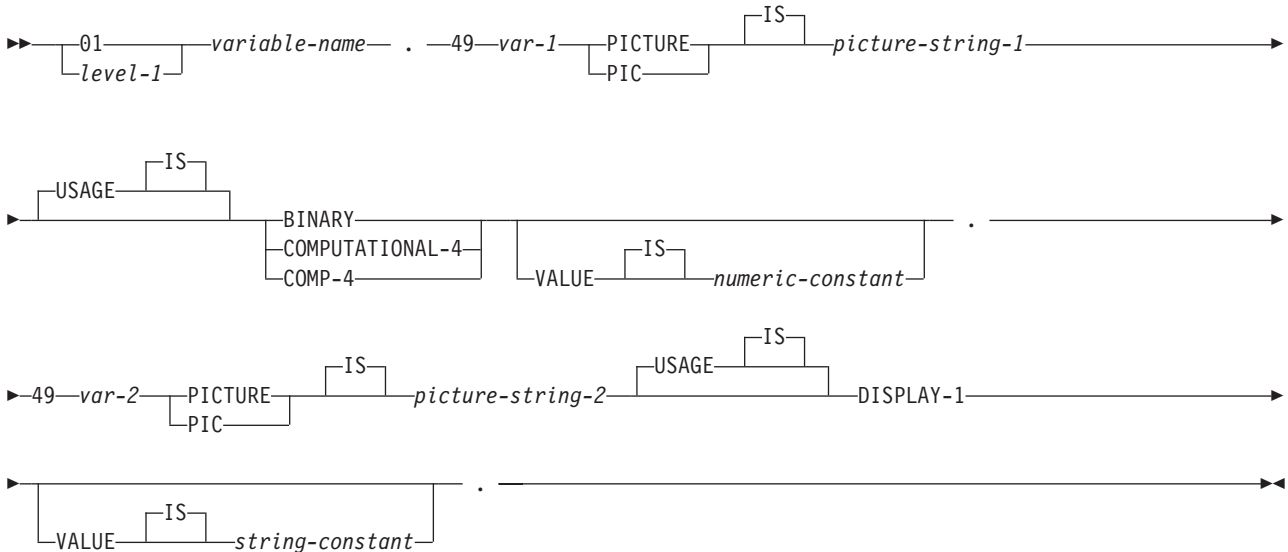




주:

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

### 가변 길이 그래픽 스트링



주:

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

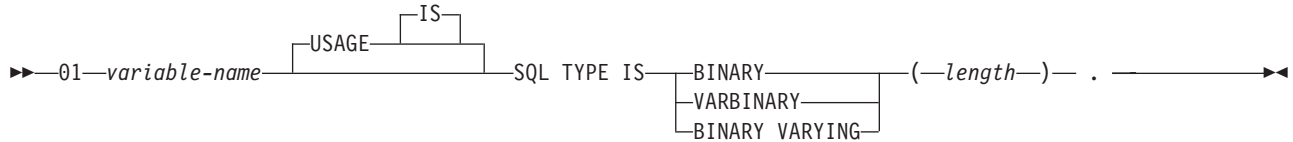
iSeries의 COBOL이 최대 지정된 정밀도까지 값만 인식한다고 해도 데이터베이스 관리자는 S9(m) 변수의 전체 크기를 사용한다는 점에 유의하십시오. COBOL문이 수행 중이고 가변 길이 스트링의 최대 길이를 효과적으로 지정된 정밀도로 제한하는 경우 이는 자료 절단 오류를 발생시킬 수 있습니다.

2. 이러한 양식과 연결된 *picture-string-2*는 G(m), GG...G(G의 m 인스턴스), N(m) 또는 NN...N(N의 m 인스턴스)이고  $1 \leq m \leq 16\ 370$ 이어야 합니다.
3. *var-1*과 *var-2*는 호스트 변수로 사용될 수 없습니다.
4. level-1은 COBOL 레벨이 2-48 사이임을 나타냅니다.

### SQL을 사용하는 COBOL 어플리케이션의 2진 호스트 변수:

COBOL에는 SQL 2진 자료 유형에 해당되는 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.

## BINARY 및 VARBINARY



주:

1. BINARY 호스트 변수의 경우, 길이의 범위는 1 - 32766입니다.
2. VARBINARY 호스트 변수의 경우, 길이의 범위는 1 - 32740입니다.
3. SQL TYPE IS, BINARY, VARBINARY 및 BINARY VARYING은 대소문자를 혼합하여 사용할 수 있습니다.

## BINARY 예

다음 선언을 사용할 경우:

```
01 MY-BINARY SQL TYPE IS BINARY(200).
```

다음과 같은 코드가 생성됩니다.

```
01 MY-BINARY PIC X(200).
```

## VARBINARY 예

다음 선언을 사용할 경우:

```
01 MY-VARBINARY SQL TYPE IS VARBINARY(250).
```

다음과 같은 구조가 생성됩니다.

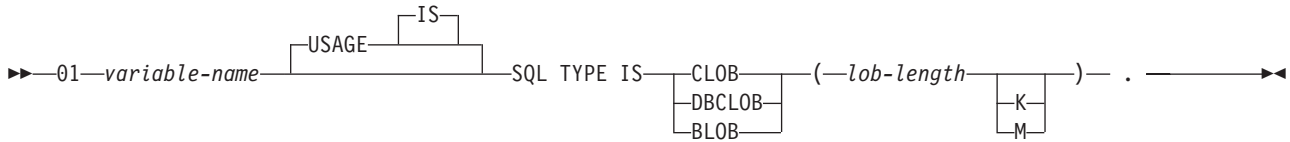
```
01 MY-VARBINARY.  
  49 MY-VARBINARY-LENGTH PIC 9(5) BINARY.  
  49 MY-VARBINARY-DATA PIC X(250).
```

## SQL을 사용하는 COBOL 어플리케이션의 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.

## LOB 호스트 변수



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 혼용하여 사용할 수 있습니다.

## 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.
```

## BLOB 예

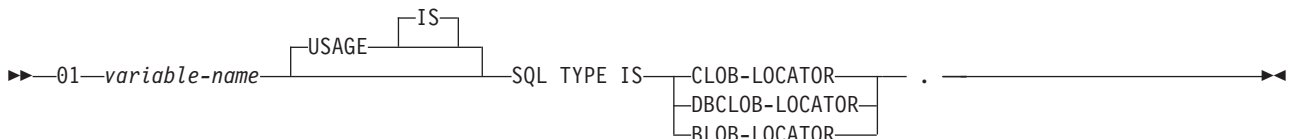
다음 선언을 사용할 경우:

```
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).
```

## LOB 위치 지정자





주:

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

CLOB 및 DBCLOB 위치 지정자의 구문은 비슷합니다.

## BLOB 위치 지정자 예

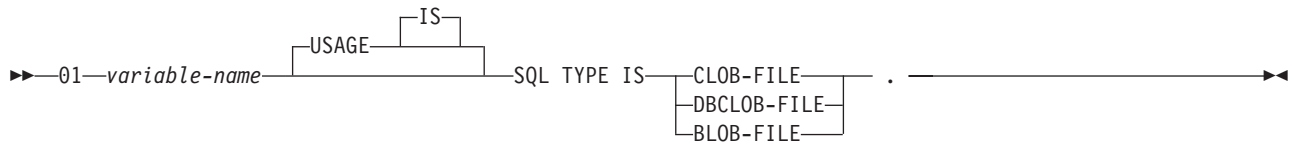
다음 선언을 사용할 경우:

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

다음과 같이 생성됩니다.

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

## LOB 파일 참조 변수



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.

- SQL\_FILE\_READ (2)
- SQL\_FILE\_CREATE (8)
- SQL\_FILE\_OVERWRITE (16)
- SQL\_FILE\_APPEND (32)

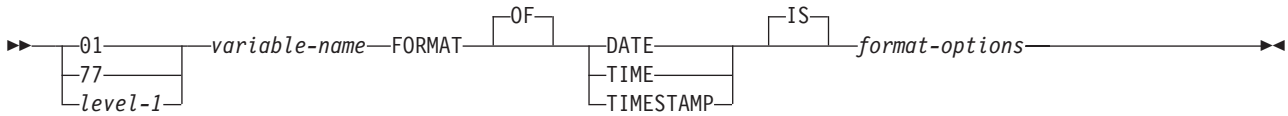
## 관련 정보

LOB 파일 참조 변수


### SQL을 사용하는 COBOL 어플리케이션의 Datetime 호스트 변수:

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

#### Datetime 호스트 변수



주:

1. *level-1*은 COBOL 레벨이 2-48 사이임을 나타냅니다.
2. *format-options*는 COBOL 컴파일러에서 지원되는 유효한 datetime 옵션을 나타냅니다. 자세한 정보는 V5R1 보충 매뉴얼  웹 사이트에 있는 ILE COBOL 참조 매뉴얼을 참조하십시오.

### SQL을 사용하는 COBOL 어플리케이션의 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



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 필드명을 사용하십시오. 예를 들어, C1 OF B 또는 C1 IN B가 아닌 B.C1을 지정하십시오. 그러나 지침은 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.

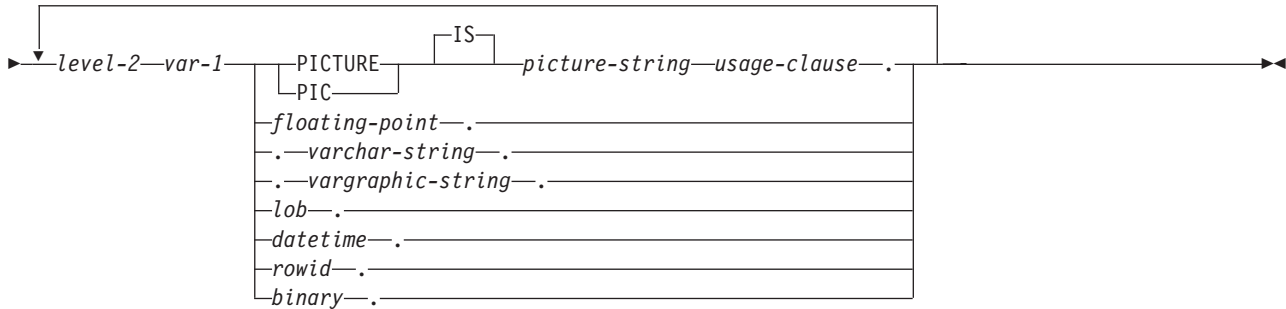
```
01 PEMPL.
  10 EMPNO          PIC X(6).
  10 FIRSTNME.
    49 FIRSTNME-LEN PIC S9(4) USAGE BINARY.
    49 FIRSTNME-TEXT PIC X(12).
  10 MIDINIT        PIC X(1).
  10 LASTNAME.
    49 LASTNAME-LEN PIC S9(4) USAGE BINARY.
    49 LASTNAME-TEXT PIC X(15).
  10 WORKDEPT       PIC X(3).
...
MOVE "000220" 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.

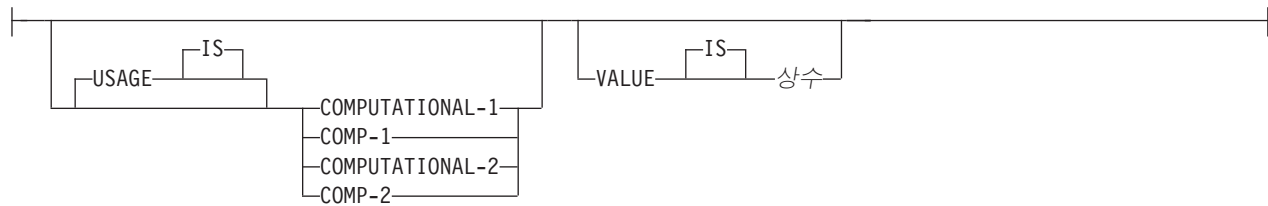
## SQL을 사용하는 COBOL 어플리케이션의 호스트 구조

다음 그림은 올바른 호스트 구조에 대한 구문을 보여줍니다.

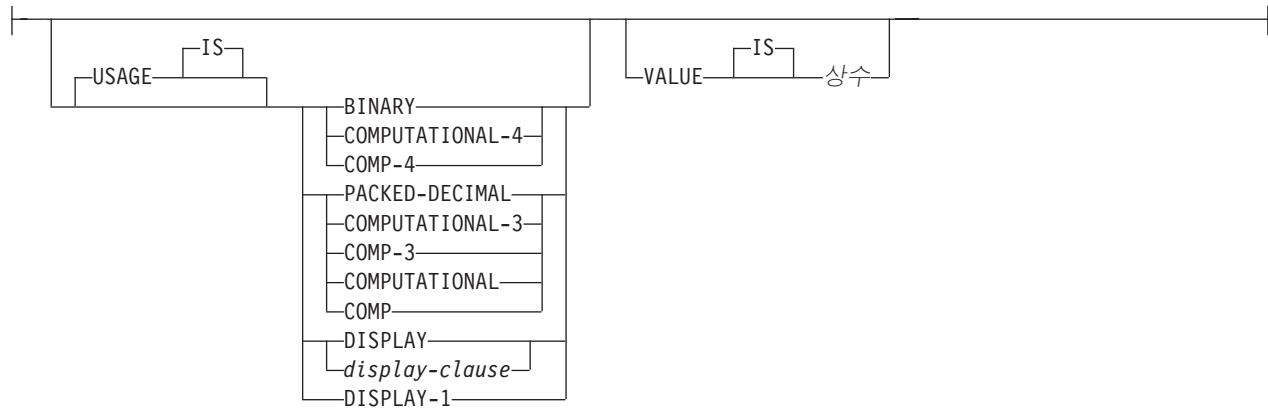
▶▶—level-1—variable-name—.—————▶▶



### floating-point:



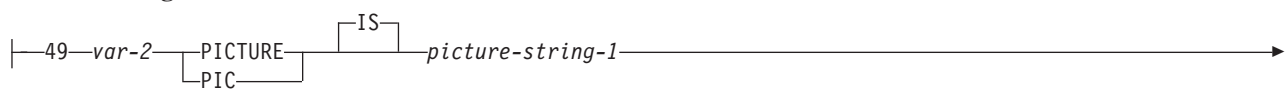
### usage-clause:

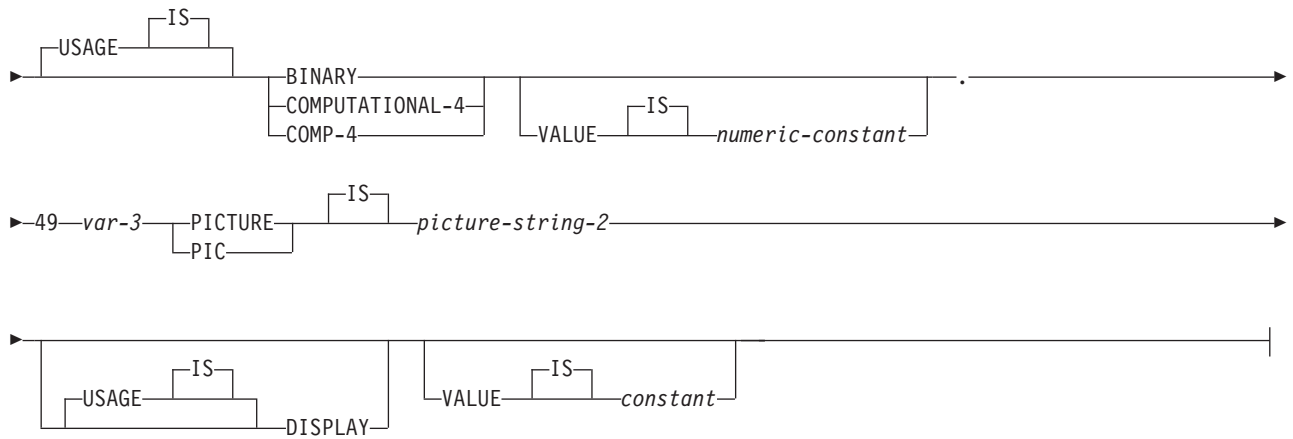


### display-clause:

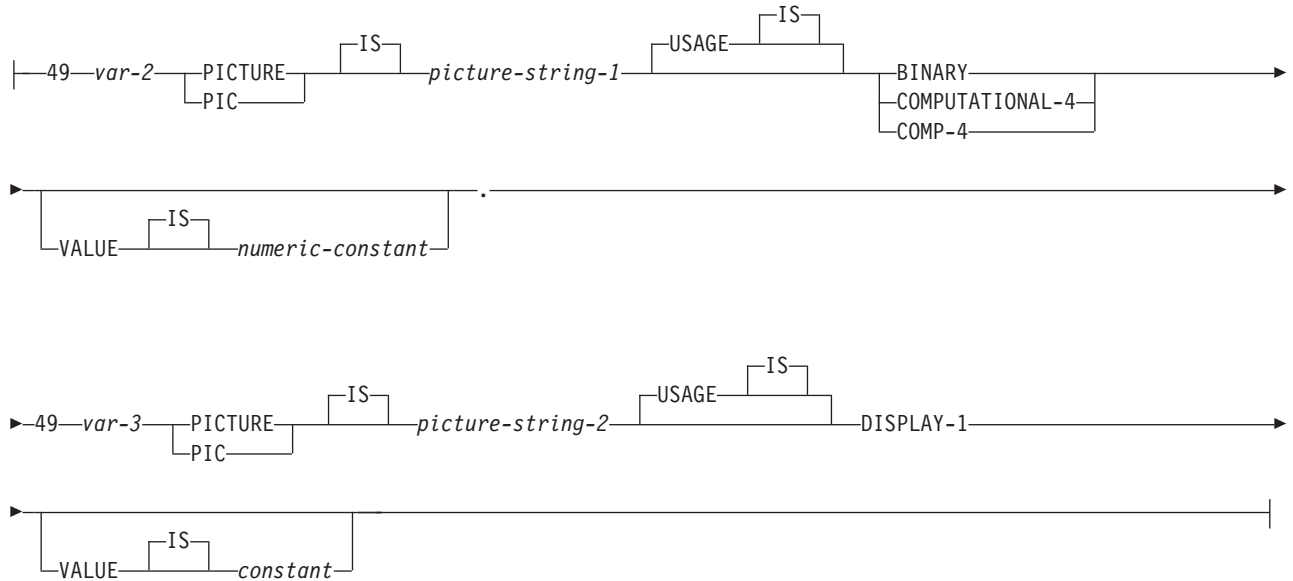


### varchar-string:

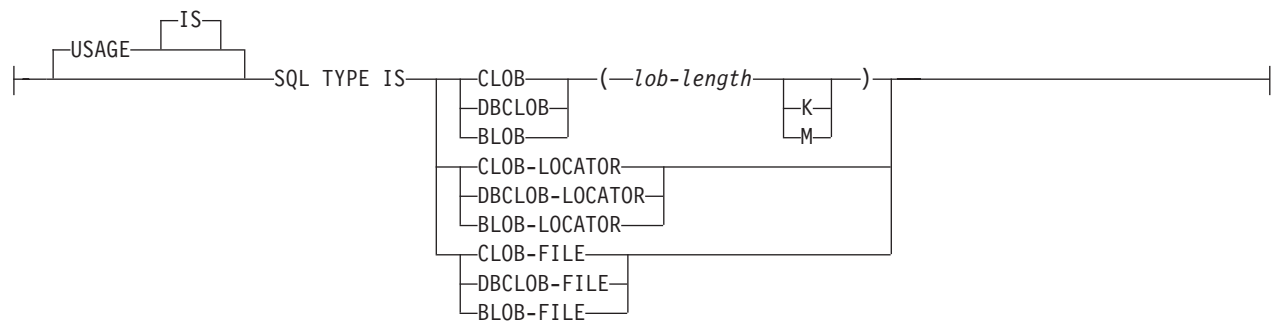




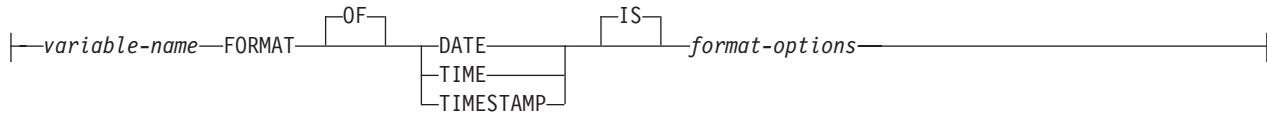
**vargraphic-string:**



**lob:**



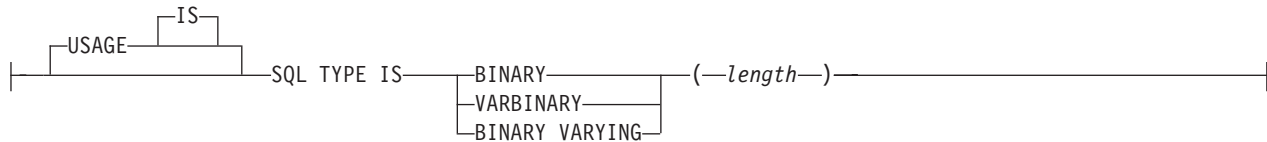
**datetime:**




**rowid:**



**binary:**



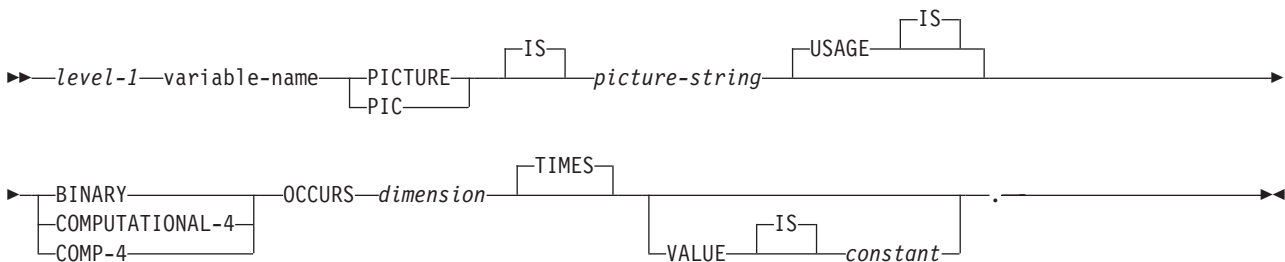
**주:**

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, ROWID 및 2진 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트 변수, 그래픽 호스트 변수, LOB 호스트 변수, ROWID 및 2진 호스트 변수 아래에 있는 주를 참조하십시오.
5. *format-options*는 COBOL 컴파일러에서 지원되는 유효한 datetime 옵션을 나타냅니다. 자세한 정보는 V5R1 보충 매뉴얼  웹 사이트에 있는 ILE COBOL 참조 매뉴얼을 참조하십시오.

**SQL을 사용하는 COBOL 어플리케이션의 호스트 구조 인디케이터 배열**

다음 그림은 유효한 인디케이터 배열 선언에 대한 구문을 보여줍니다.

**호스트 구조 인디케이터 배열**



**주:**

1. `dimension`은 1-32767 사이의 정수이어야 합니다.
2. 레벨 1은 2-48 사이의 정수이어야 합니다.

3. BINARY, COMPUTATIONAL 및 COMP-4는 동등합니다. COMPUTATIONAL-4 및 COMP-4는 ISO/ANSI COBOL에서 지원되지 않는 IBM 확장자이므로 이식가능한 어플리케이션은 BINARY로 작성되어야 합니다. 이 유형과 연결된 픽처 스트링에는 양식 S9(i)(또는 S9...9, 9의 i 인스턴스가 있어야 합니다. i는 4 이하여야 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.
```

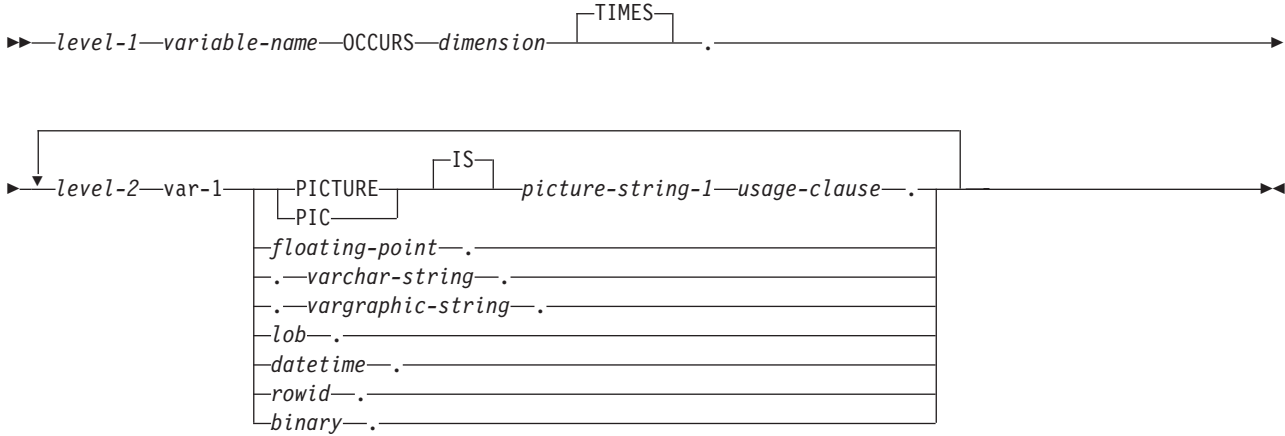
```
    02 B-ARRAY OCCURS 10 TIMES.
        03 C1-VAR PIC X(20).
        03 C2-VAR PIC S9(4).
```

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

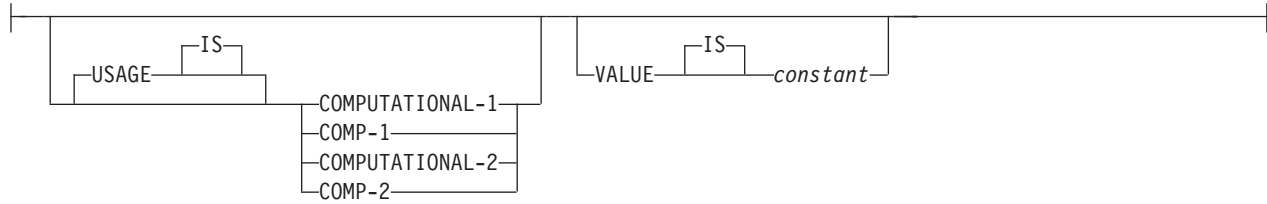
```
01 TABLE-1.
    02 DEPT OCCURS 10 TIMES.
    05 DEPTNO PIC X(3).
    05 DEPTNAME.
        49 DEPTNAME-LEN PIC S9(4) BINARY.
        49 DEPTNAME-TEXT PIC X(29).
    05 MGRNO PIC X(6).
    05 ADMRDEPT PIC X(3).
01 TABLE-2.
    02 IND-ARRAY OCCURS 10 TIMES.
        05 INDS PIC S9(4) BINARY OCCURS 4 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 어플리케이션의 호스트 구조 배열

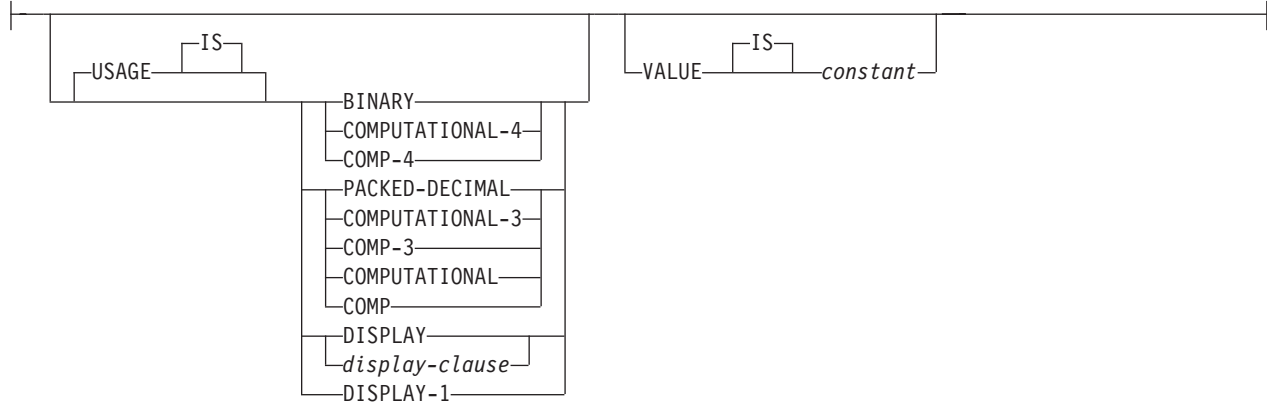
다음 그림은 유효한 호스트 구조 배열 선언 부분에 대한 구문을 표시합니다.



### floating-point:



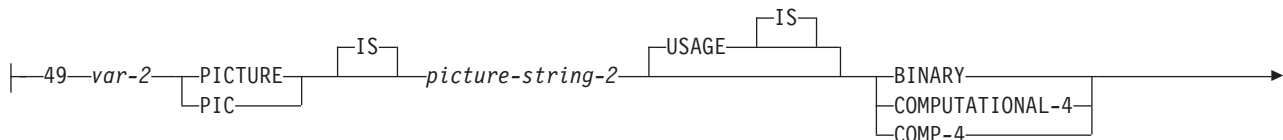
### usage-clause:



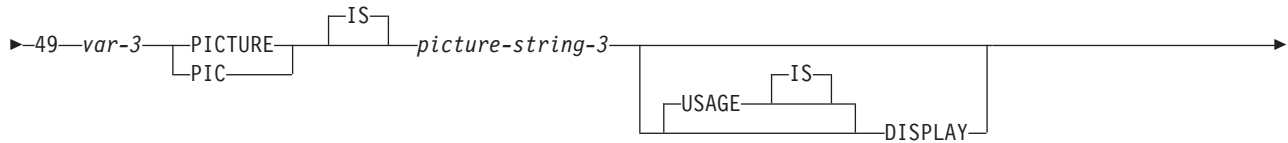
### display-clause:



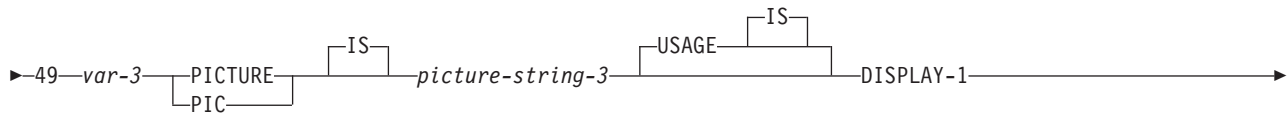
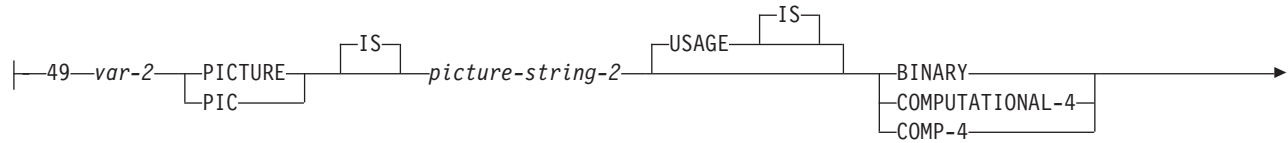
### vvarchar-string:



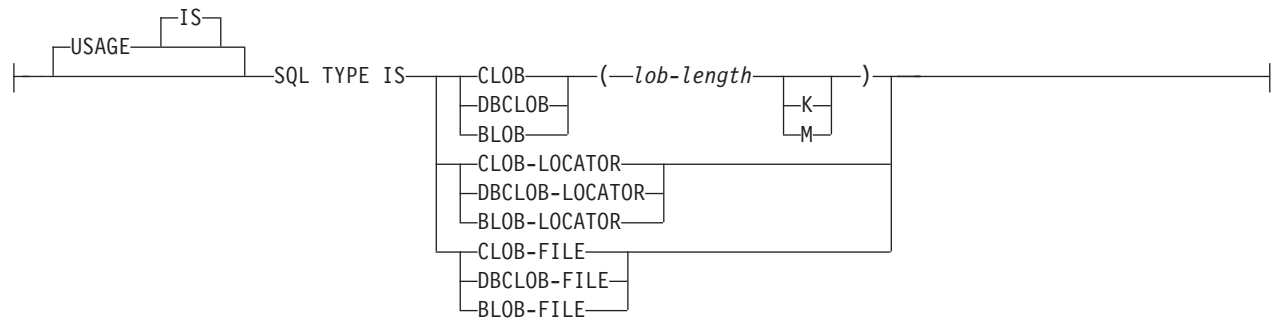




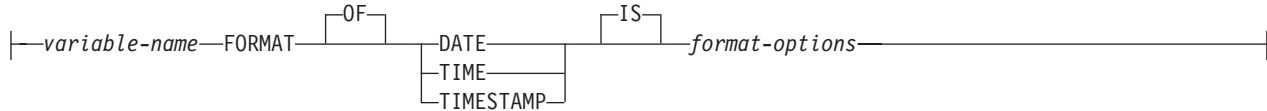
**vargraphic-string:**



**lob:**



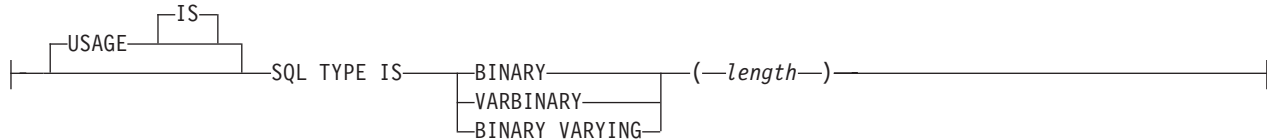
**datetime:**




**rowid:**



**binary:**

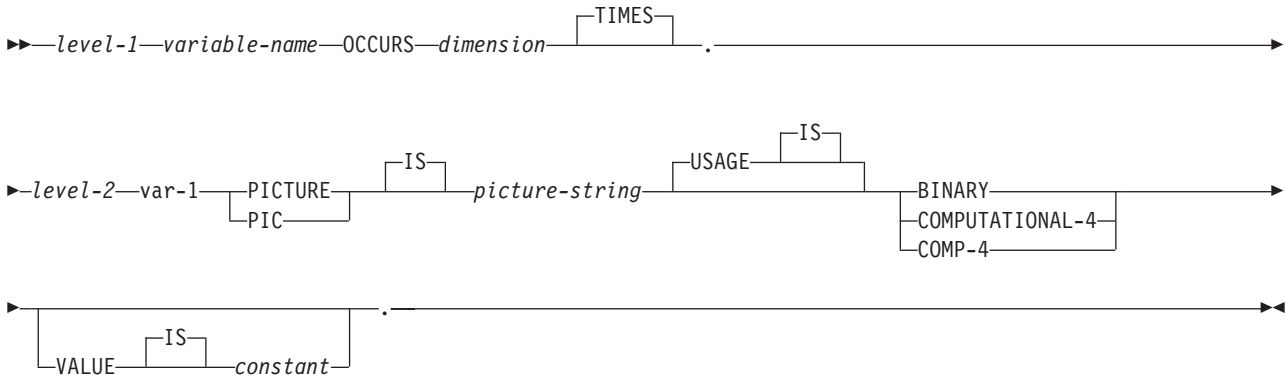


**주:**

1. level-1은 COBOL 레벨이 2-47 사이임을 나타냅니다.
2. level-2는 COBOL 레벨이 3-48(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, ROWID 및 2진 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트 변수, 그래픽 호스트 변수, LOB, ROWID 및 2진 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 V5R1 보충 매뉴얼  웹 사이트에 있는 ILE COBOL 참조 매뉴얼을 참조하십시오.

**SQL을 사용하는 COBOL 어플리케이션의 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(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 이하여야 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 SQL 프로그래밍 개념 주제 컬렉션의 iSeries용 DB2 UDB 샘플 표에 설명된 샘플 표 DEPARTMENT 정의를 검색하려면 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 V5R1 보충 매뉴얼  웹 사이트의 COBOL/400® 사용자 안내서 및 ILE COBOL 참조서 매뉴얼을 참조하십시오.

## SQL을 사용하는 COBOL 어플리케이션에서 호스트 구조 배열에 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 10행을 호스트 구조 배열에 배치하는 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 변수로 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. GRAPHIC 또는 VARGRAPHIC 열에 UTF-16 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UTF-16 CCSID를 갖게 됩니다.

### 동등한 SQL 및 COBOL 자료 유형 판별

사전컴파일러는 이 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	1바이트에서 i+d, 2바이트에서 d	DECIMAL(i+d,d)
S9(i)V9(d) DISPLAY SIGN LEADING SEPARATE	504	1바이트에서 i+d, 2바이트에서 d	정확히 같은 것은 없습니다. DECIMAL(i+d,d) 또는 NUMERIC(i+d,d)을 사용합니다.
S9(i)V9(d)DISPLAY	488	1바이트에서 i+d, 2바이트에서 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
S9(i) BINARY 또는 S9(i) COMP-4 , i는 10 - 18 iSeries용 COBOL에 대해 지원되지 않습니다.	492	8	BIGINT

표 3. 일반 SQL 자료 유형에 맵핑되는 COBOL 선언 (계속)

COBOL 자료 유형	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
S9(i)V9(d) BINARY 또는 S9(i)V9(d) COMP-4, 여기서 $i+d \leq 4$	500	1바이트에서 $i+d$ , 2바이트에서 $d$	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, 여기서 $4 < i+d \leq 9$	496	1바이트에서 $i+d$ , 2바이트에서 $d$	정확히 같은 것은 없습니다. DECIMAL( $i+d,d$ ) 또는 NUMERIC( $i+d,d$ )을 사용합니다.
COMP-1 iSeries용 COBOL에 대해 지원되지 않습니다.	480	4	FLOAT(단정밀도)
COMP-2 iSeries용 COBOL에 대해 지원되지 않습니다.	480	8	FLOAT(배정밀도)
고정 길이 문자 자료	452	m	CHAR(m)
가변 길이 문자 자료	448	m	VARCHAR(m)
고정 길이 그래픽 자료 iSeries용 COBOL에 대해 지원되지 않습니다.	468	m	GRAPHIC(m)
가변 길이 그래픽 자료 iSeries용 COBOL에 대해 지원되지 않습니다.	464	m	VARGRAPHIC(m)
DATE iSeries용 COBOL에 대해 지원되지 않습니다.	384		DATE
TIME iSeries용 COBOL에 대해 지원되지 않습니다.	388		TIME
TIMESTAMP iSeries용 COBOL에 대해 지원되지 않습니다.	392	26	TIMESTAMP

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

표 4. 일반 COBOL 선언 부분에 맵핑되는 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.
BIGINT	iSeries용 ILE COBOL의 S9(m) COMP-4. iSeries용 COBOL에 대해 지원되지 않습니다.	m은 10 - 18
DECIMAL(p,s)	$p < 64$ : S9(p-s)V9(s) PACKED-DECIMAL이 나 S9(p-s)V9(s) COMP 또는 S9(p-s)V9(s) COMP-3, $p > 63$ :인 경우는 지원되지 않습니다.	$p$ 는 정밀도이고 $s$ 는 스케일입니다 $0 \leq s \leq p \leq 63$ . $s=0$ 인 경우 S9(p) 또는 S9(p)V를 사용하고 $s=p$ 인 경우는 SV9(s)를 사용합니다.
NUMERIC(p,s)	$p < 19$ : S9(p-s)V9(s) DISPLAY, $p > 18$ 인 경우 는 지원되지 않습니다.	$p$ 는 정밀도이고 $s$ 는 스케일입니다 $0 \leq s \leq p \leq 18$ . $s=0$ 인 경우 S9(p) 또는 S9(p)V를 사용하고 $s=p$ 인 경우는 SV9(s)를 사용합니다.
FLOAT(단정밀도)	iSeries용 ILE COBOL의 COMP-1. iSeries용 COBOL에 대해 지원되지 않습니다.	

표 4. 일반 COBOL 선언 부분에 맵핑되는 SQL 자료 유형 (계속)

SQL 자료 유형	COBOL 자료 유형	주
FLOAT(배정밀도)	iSeries용 ILE COBOL의 COMP-2. iSeries용 COBOL에 대해 지원되지 않습니다.	
CHAR(n)	고정 길이 문자 스트링	32766≥n≥1
VARCHAR(n)	가변 길이 문자 스트링	32740≥n≥1
CLOB	없음	iSeries용 ILE COBOL에 대해 CLOB을 선언하려면 SQL TYPE IS를 사용하십시오. iSeries용 COBOL에 대해 지원되지 않습니다.
GRAPHIC(n)	iSeries용 ILE COBOL의 고정 길이 그래픽 스트링. iSeries용 COBOL에 대해 지원되지 않습니다.	16383≥n≥1
VARGRAPHIC(n)	iSeries용 ILE COBOL의 가변 길이 그래픽 스트링. iSeries용 COBOL에 대해 지원되지 않습니다.	16370≥n≥1
DBCLOB	없음	iSeries용 ILE COBOL에 대해 DBCLOB을 선언하려면 SQL TYPE IS를 사용하십시오.
BINARY	없음	BINARY를 선언하려면 SQL TYPE IS를 사용하십시오.
VARBINARY	없음	VARBINARY를 선언하려면 SQL TYPE IS를 사용하십시오.
BLOB	없음	BLOB를 선언하려면 SQL TYPE IS를 사용하십시오.
DATE	iSeries용 ILE COBOL의 고정 길이 문자 스트링 또는 DATE.	형식이 *USA, *JIS, *EUR 또는 *ISO이면 최소한 10자를 허용합니다. 형식이 *YMD, *DMY 또는 *MDY이면 최소한 8자를 허용합니다. 형식이 *JUL이면 최소한 6자를 허용합니다.
TIME	iSeries용 ILE COBOL의 고정 길이 문자 스트링 또는 TIME.	최소한 6자를 허용합니다. 초를 포함할 경우에는 8자를 허용합니다.
TIMESTAMP	iSeries용 ILE COBOL의 고정 길이 문자 스트링 또는 TIMESTAMP.	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.

## 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, INTEGER 및 BIGINT 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 수행 시 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 1-4)입니다.

또한 인디케이터 구조(반단어 정수 변수의 배열로 정의됨)를 지정하여 호스트 구조를 지원할 수도 있습니다. 검색 시 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.

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 선언할 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.
```

관련 참조

변수 참조

---

## PL/I 어플리케이션에서 SQL문 코딩

이 주제에서는 iSeries PL/I 프로그램에 SQL문을 삽입하는 경우 고유한 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

#### 관련 개념

2 페이지의 『SQL을 사용하는 어플리케이션 작성』

iSeries용 DB2 UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성할 수 있습니다.

146 페이지의 『SQL을 사용하는 어플리케이션 프로그램의 컴파일 시 오류 및 경고 메시지』  
다음은 컴파일 처리 시 오류 또는 경고 메시지를 발생시킬 수도 있는 조건에 대한 설명입니다.

#### 관련 참조

150 페이지의 『iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램』

이 주제에는 iSeries용 DB2 UDB 시스템에서 지원되는 각 언어로 SQL문을 코딩하는 방법을 나타내는 샘플 어플리케이션이 들어 있습니다.

## 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 INCLUDE 문을 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.

관련 정보

SQL 통신 영역

## SQL을 사용하는 PL/I 어플리케이션에서 SQL 설명자 영역 정의

- | SQL 설명자 영역에는 두 가지 유형이 있습니다. 하나는 ALLOCATE DESCRIPTOR문으로 정의됩니다. 다른 하나는 SQLDA 구조를 사용하여 정의됩니다. 여기서는 SQLDA 형식에 대해서만 설명합니다.
- | 다음 명령문은 SQLDA를 사용할 수 있습니다.

- 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 INPUT *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 사용 시 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 고급 프로그래밍 기술입니다. 동적 SQL을 사용하면 프로그램 수행 중 사용자 프로그램이 SQL 문을 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.

관련 정보

동적 SQL 어플리케이션

SQL 설명자 영역

## 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.

### 예: SQL을 사용하는 PL/I 어플리케이션에서 SQL문 삽입

PL/I 프로그램에서 코딩된 UPDATE문은 다음과 같습니다.

```

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

```

## SQL을 사용하는 PL/I 어플리케이션의 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 SQL문에는 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 어플리케이션에 코드 포함

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

EXEC SQL INCLUDE member-name ;

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

### SQL을 사용하는 PL/I 어플리케이션의 여백

CRTSQLPLI 명령의 MARGINS 매개변수에 의해 지정된 여백 내에 SQL문을 작성하십시오. EXEC SQL이 지정된 여백에서 시작하지 않으면 SQL 컴파일러는 SQL문을 인식하지 못합니다.

#### 관련 개념

189 페이지의 『호스트 언어 사전컴파일러용 iSeries용 DB2 UDB CL 명령 설명』

iSeries용 DB2 UDB는 다음 프로그래밍 언어에서 코딩된 사전컴파일러 프로그램에 대한 명령을 제공합니다.

### SQL을 사용하는 PL/I 어플리케이션에서 이름

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

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

### SQL을 사용하는 PL/I 어플리케이션의 명령문 레이블

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

### SQL을 사용하는 PL/I 어플리케이션의 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 SECTION 문 사이에 있어야 합니다.

SQL문 내의 모든 호스트 변수 앞에는 콜론(:)이 와야 합니다.

호스트 변수명은 호스트 변수가 다른 블록이나 프로시저에 존재하더라도 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.

## SQL을 사용하는 PL/I 어플리케이션에서 호스트 변수 선언

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

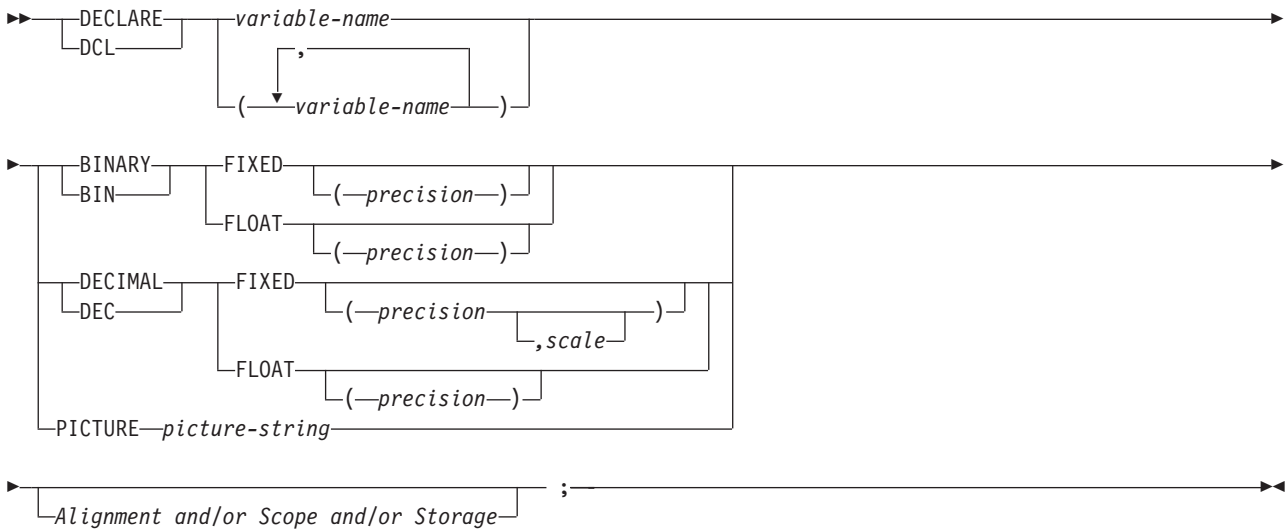
변수의 이름과 자료 속성만이 사전컴파일러에 의해 사용됩니다. 정렬, 범위 및 기억영역 속성은 무시됩니다. 정렬, 범위 및 기억영역이 무시되더라도 이들을 사용할 때는 몇 가지 제한사항이 있으므로 무시될 경우 사전컴파일러에 의해 생성된 PL/I 소스 코드 컴파일 시 문제점이 발생할 수 있습니다. 제한사항은 다음과 같습니다.

- 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.

### 숫자



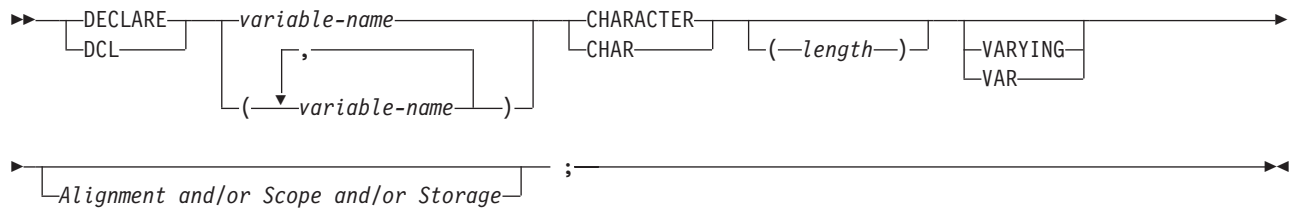
### 주:

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. R이 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.

### 문자



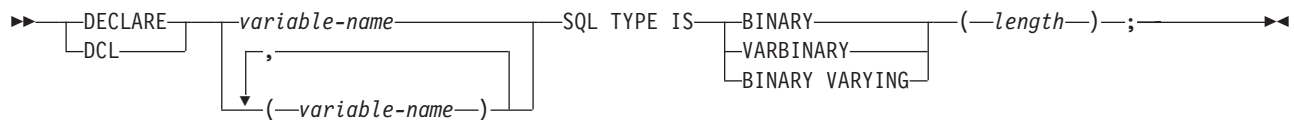
주:

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

### SQL을 사용하는 PL/I 어플리케이션의 2진 호스트 변수:

PL/I에는 SQL 2진 자료 유형에 해당되는 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.

### BINARY 및 VARBINARY



주:

1. BINARY 호스트 변수의 경우, 길이의 범위는 1 - 32766입니다.
2. VARBINARY 및 BINARY VARYING 호스트 변수의 경우, 길이의 범위는 1 - 32740입니다.
3. SQL TYPE IS, BINARY, VARBINARY, BINARY VARYING은 대소문자를 혼합하여 사용할 수 있습니다.

### BINARY 예

다음 선언을 사용할 경우:

```
DCL MY_BINARY SQL TYPE IS BINARY(100);
```

다음과 같은 코드가 생성됩니다.

```
DCL MY_BINARY CHARACTER(100);
```

### VARBINARY 예

다음 선언을 사용할 경우:

```
DCL MY_VARBINARY SQL TYPE IS VARBINARY(250);
```

다음과 같은 코드가 생성됩니다.

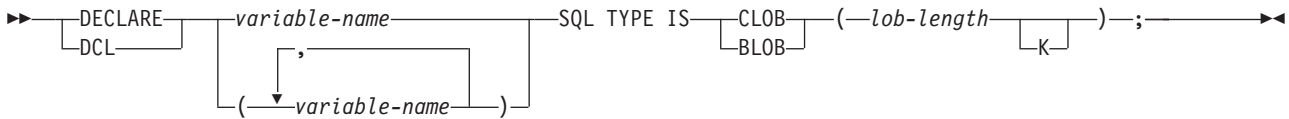
```
DCL MY_VARBINARY CHARACTER(250) VARYING;
```

## 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.

### LOB



주:

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

### 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);
```

### BLOB 예

다음 선언을 사용할 경우:

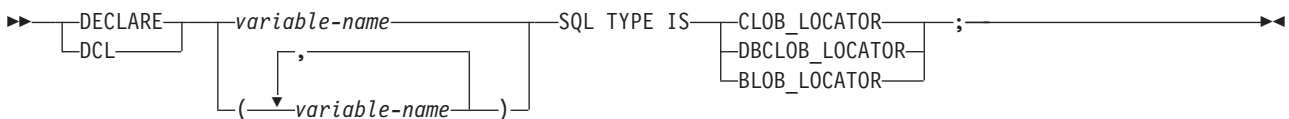
```
DCL MY_BLOB SQL TYPE IS BLOB(16384);
```

다음과 같은 구조가 생성됩니다.

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

다음 그림은 유효한 LOB 위치 지정자에 대한 구문을 보여줍니다.

### LOB 위치 지정자



주: SQL TYPE IS, BLOB\_LOCATOR, CLOB\_LOCATOR, DBCLOB\_LOCATOR는 대소문자를 혼용하여 사용할 수 있습니다.

## CLOB 위치 지정자 예

다음 선언을 사용할 경우:

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

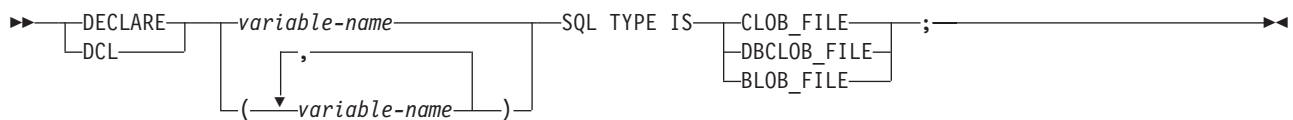
다음과 같이 생성됩니다.

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

BLOB 및 DBCLOB 위치 지정자의 구문은 비슷합니다.

다음 그림은 유효한 LOB 파일 참조 변수에 대한 구문을 보여줍니다.

## LOB 파일 참조 변수



주: SQL TYPE IS, BLOB\_FILE, CLOB\_FILE 및 DBCLOB\_FILE은 대소문자를 혼용하여 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)

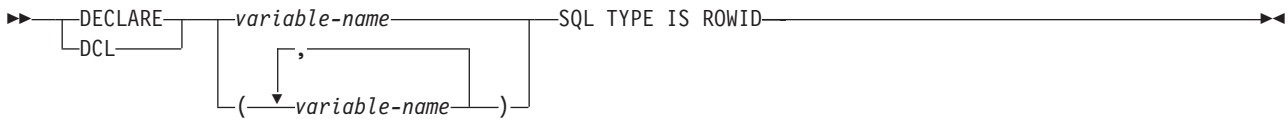
관련 정보

LOB 파일 참조 변수

## 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.

### ROWID



주: SQL TYPE IS ROWID는 대소문자를 혼합하여 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 세트라고 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.



```

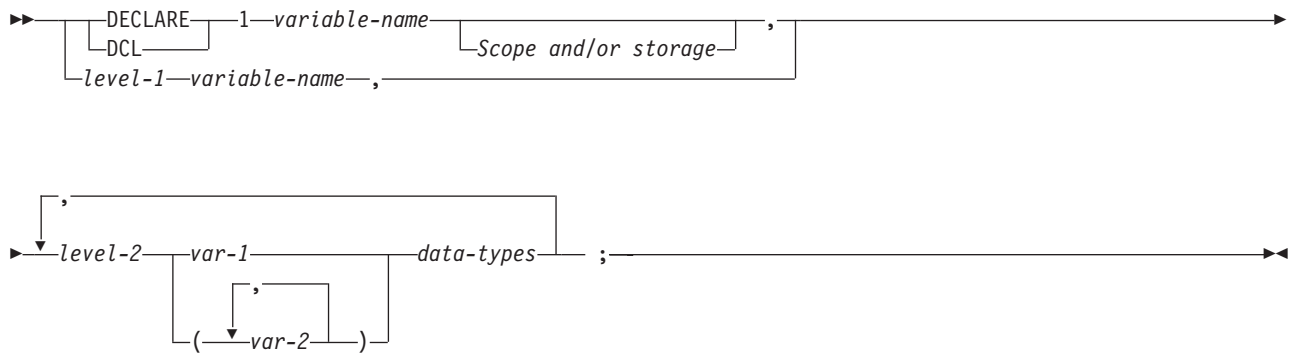
DCL 1 PEMPL,
      5 EMPNO    CHAR(6),
      5 FIRSTNME CHAR(12) VAR,
      5 MIDINIT  CHAR(1),
      5 LASTNAME CHAR(15) VAR,
      5 WORKDEPT CHAR(3);
...
EMPID = '000220';
...
      EXEC SQL
      SELECT *
      INTO :PEMPL
      FROM CORPDATA.EMPLOYEE
      WHERE EMPNO = :EMPID;

```

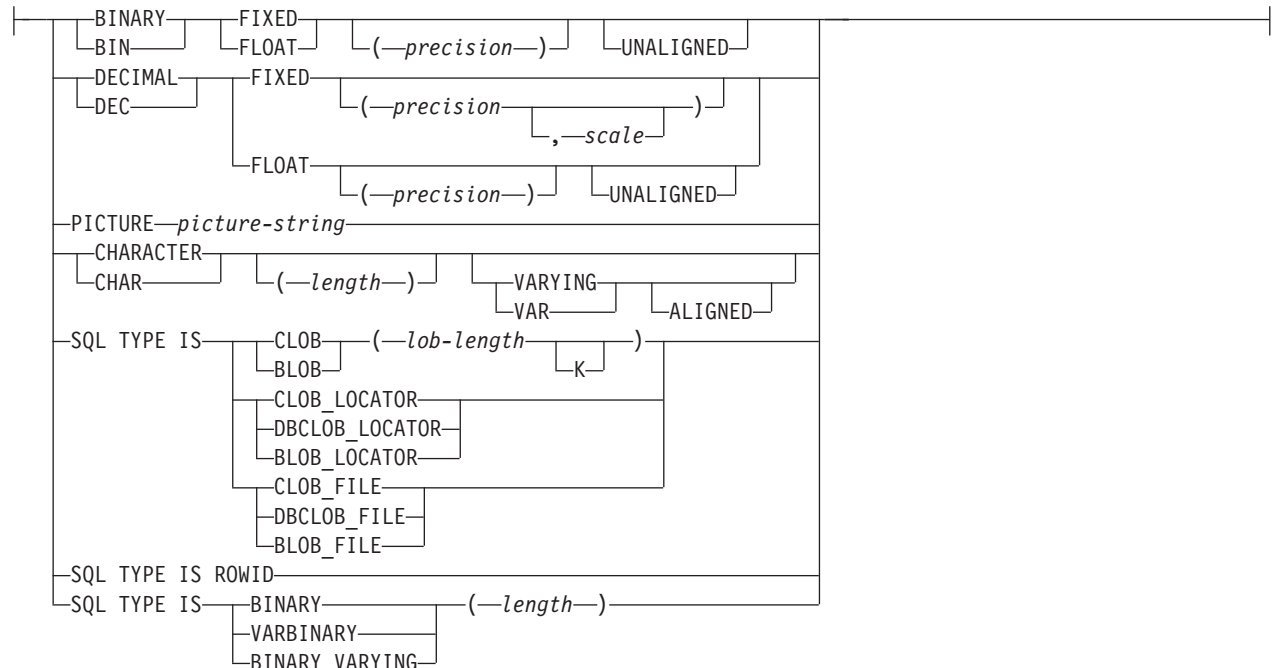
### SQL을 사용하는 PL/I 어플리케이션의 호스트 구조

이 그림은 유효한 호스트 구조 선언에 대한 구문을 보여줍니다.

#### 호스트 구조



#### data-types:



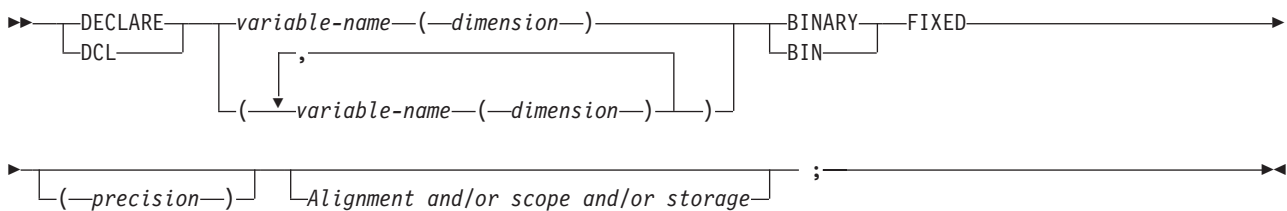
주:

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, ROWID 및 2진 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트 변수, LOB 호스트 변수, ROWID 호스트 변수 및 2진 호스트 변수 아래에 있는 주를 참조하십시오.

## SQL을 사용하는 PL/I 어플리케이션의 호스트 구조 인디케이터 배열

이 그림은 유효한 인디케이터 배열에 대한 구문을 보여줍니다.

### 호스트 구조 인디케이터 배열



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 DEPTNO 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;

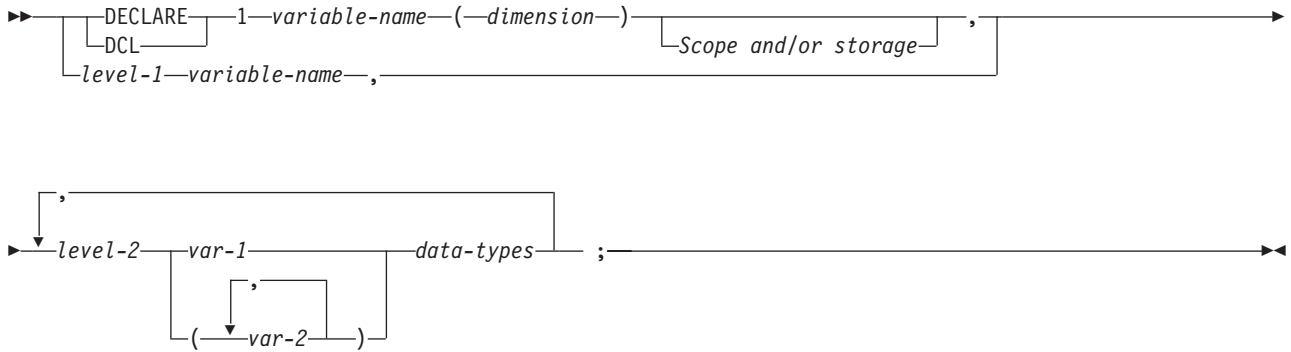
EXEC SQL
FETCH C1 FOR 10 ROWS INTO :DEPT :IND_ARRAY;

```

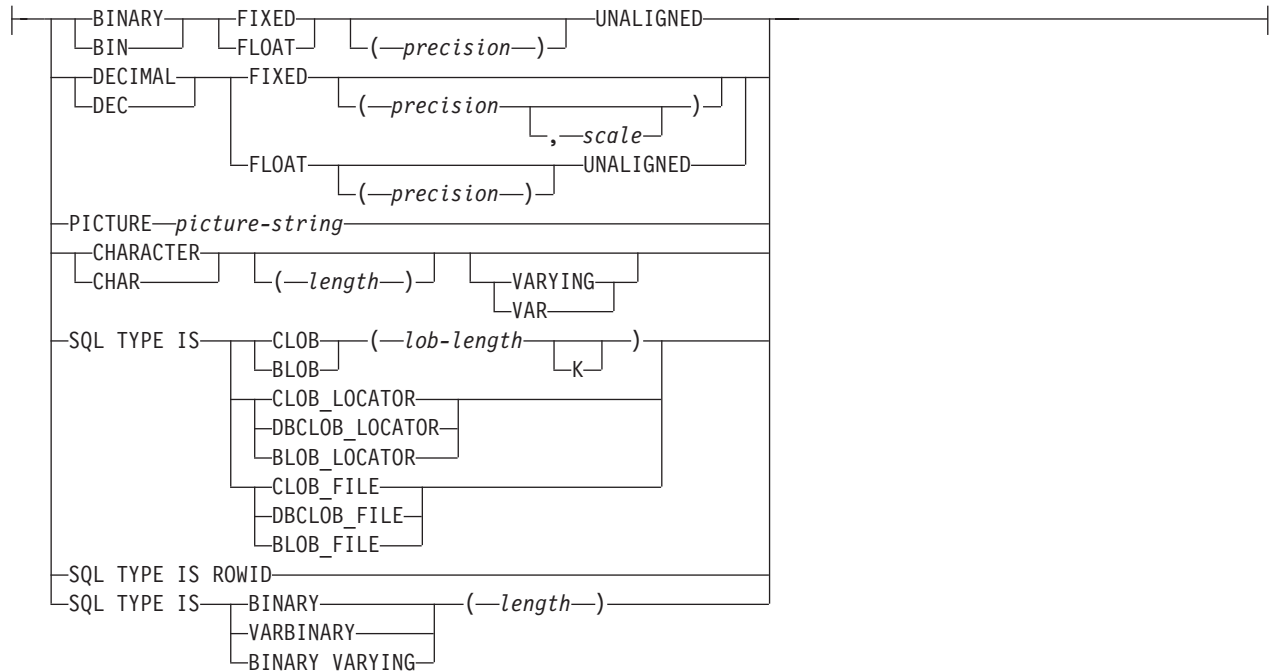
### SQL을 사용하는 PL/I 어플리케이션의 호스트 구조 배열

다음의 구문 다이어그램은 유효한 구조 배열 선언에 대한 구문을 보여줍니다.

#### 호스트 구조 배열



#### data-types:



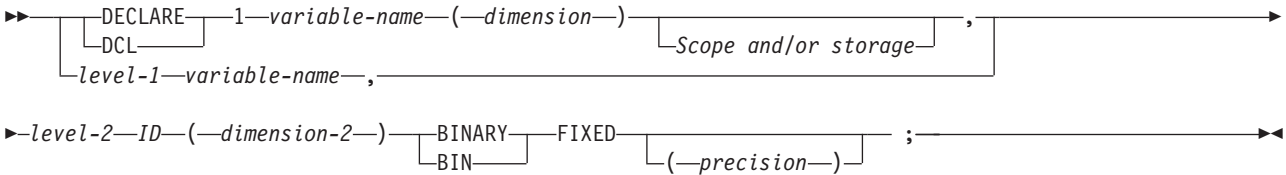
#### 주:

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

- level-2는 2-255 사이의 정수 상수이어야 합니다.
- 숫자, 문자, LOB, ROWID 및 2진 호스트 변수의 선언에 대한 자세한 내용은 숫자 호스트 변수, 문자 호스트 변수, LOB 호스트 변수, ROWID 호스트 변수 및 2진 호스트 변수 아래에 있는 주를 참조하십시오.
- dimension은 1-32767 사이의 정수 상수이어야 합니다.

### SQL을 사용하는 PL/I 어플리케이션의 호스트 구조 배열 인디케이터:

다음 그림은 유효한 호스트 구조 배열에 대한 구문 다이어그램을 보여줍니다.



주:

- level-1은 중간 레벨 구조가 있음을 나타냅니다.
- level-1은 1-254 사이의 정수 상수이어야 합니다.
- level-2는 2-255 사이의 정수 상수이어야 합니다.
- 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 SQL 프로그래밍 주제 컬렉션의 iSeries용 DB2 UDB 샘플 표에 설명된 샘플 표 DEPARTMENT의 정의를 포함시키려면 다음을 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. GRAPHIC 또는 VARGRAPHIC 열에 UTF-16 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UTF-16 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	1바이트에는 p, 2바이트에는 s	DECIMAL(p,s)
BIN FLOAT(p), p는 1 - 24	480	4	FLOAT(단정밀도)
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	1바이트에는 p, 2바이트에는 s	NUMERIC(p,s)
PICTURE 형상 문자(선행 부호 분리)	504	1바이트에는 p, 2바이트에는 s	정확히 같은 것은 없습니다. NUMERIC(p,s)을 사용합니다.
CHAR(n)	452	n	CHAR(n)
CHAR(n) VARYING	448	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 동등	주
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) VARYING	n은 1 - 32740 사이의 양의 정수입니다.
CLOB	없음	CLOB를 선언하려면 SQL TYPE IS를 사 용하십시오.
GRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
VARGRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
DBCLOB	지원되지 않습니다.	지원되지 않습니다.
BINARY	없음	BINARY를 선언하려면 SQL TYPE IS를 사용하십시오.
VARBINARY	없음	VARBINARY를 선언하려면 SQL TYPE IS를 사용하십시오.
BLOB	없음	BLOB를 선언하려면 SQL TYPE IS를 사 용하십시오.
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 경 우 n은 최소한 6자이어야 합니다.
TIME	CHAR(n)	n은 최소한 6자여야 하며 초를 포함시키려 면 n은 최소한 8자이어야 합니다.
TIMESTAMP	CHAR(n)	n은 최소한 19이어야 합니다. 전체 정밀도에 마이크로초를 포함시키려면 n은 26이어야 하고 n이 26 미만이면 마이크로초 부분 에서 잘립니다.
DATALINK	지원되지 않습니다.	지원되지 않습니다.
ROWID	없음	ROWID를 선언하려면 SQL TYPE IS를 사 용하십시오.

## 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. 검색 시 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.

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;
```

관련 참조

변수 참조

## 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 64바이트를 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.

관련 정보

데이터베이스 응용프로그램 설계 추가 정보: 구조화된 매개변수 전달 기술 사용

---


## RPG/400 어플리케이션에서 SQL문 코딩

RPG/400 라이선스 프로그램은 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 주제에서는 RPG/400 프로그램에 SQL문을 삽입할 때 고유한 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 변수에 대한 요구사항도 정의됩니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

RPG를 사용한 프로그래밍에 대한 자세한 정보는 V5R1 보충 매뉴얼  웹 사이트에 있는 RPG/400 사용자 안내서 및 RPG/400 참조서 매뉴얼을 참조하십시오.

### 관련 개념

2 페이지의 『SQL을 사용하는 어플리케이션 작성』

iSeries용 DB2 UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성할 수 있습니다.

146 페이지의 『SQL을 사용하는 어플리케이션 프로그램의 컴파일 시 오류 및 경고 메시지』

다음은 컴파일 처리 시 오류 또는 경고 메시지를 발생시킬 수도 있는 조건에 대한 설명입니다.

### 관련 참조

150 페이지의 『iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램』

이 주제에는 iSeries용 DB2 UDB 시스템에서 지원되는 각 언어로 SQL문을 코딩하는 방법을 나타내는 샘플 어플리케이션이 들어 있습니다.

## SQL을 사용하는 RPG/400 어플리케이션에서 SQL 통신 영역 정의

SQL 사전컴파일러는 첫 번째 연산 스펙 이전의 RPG/400 프로그램 입력 스펙에 SQLCA를 자동으로 배치합니다.

소스 프로그램에 INCLUDE SQLCA를 코딩해서는 안됩니다. 소스 프로그램이 INCLUDE SQLCA를 지정하면 명령문이 허용되기는 하지만 중복됩니다. RPG/400에 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: RPG/400의 변수 이름은 6자로 제한됩니다. 표준 SQLCA 이름의 길이가 6자로 변경되었습니다. RPG/400에서는 배열을 확장 스펙에 정의하지 않고 자료 구조에 정의하는 방법이 없습니다. SQLERR은 요소명으로 사용된 SQLER1에서 SQLER6까지의 문자로 정의됩니다.

관련 정보

SQL 통신 영역

## SQL을 사용하는 RPG/400 어플리케이션에서 SQL 설명자 영역 정의

| SQL 설명자 영역에는 두 가지 유형이 있습니다. 하나는 ALLOCATE DESCRIPTOR문으로 정의됩니다. 다른 하나는 SQLDA 구조를 사용하여 정의됩니다. 여기서는 SQLDA 형식에 대해서만 설명합니다.

| 다음 명령문은 SQLDA를 사용할 수 있습니다.

- 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 INPUT *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 고급 프로그래밍 기술입니다. 동적 SQL을 사용하면 프로그램 수행 중 사용자 프로그램이 SQL 문을 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.

SQLDA는 RPG/400에서 지원하지 않는 포인터 변수를 사용하므로, INCLUDE SQLDA문을 RPG/400 프로그램에서 지정할 수 없습니다. SQLDA는 C, COBOL, PL/I 또는 ILE RPG 프로그램에 의해 설정된 후에 사용하기 위해 RPG 프로그램으로 전달되어야 합니다.

관련 정보

동적 SQL 어플리케이션

SQL 설명자 영역

## SQL을 사용하는 RPG/400 어플리케이션에 SQL문 삽입

RPG/400 프로그램에서 코딩된 SQL문은 연산 섹션에 위치해야 합니다.

이 경우 C가 위치 6에 있어야 합니다. SQL문은 상세 계산, 합계 또는 RPG/400 서브루틴에 위치할 수 있습니다. SQL문은 RPG/400문의 논리를 기반으로 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.

### 예: SQL을 사용하는 RPG/400 어플리케이션에 SQL문 삽입

RPG/400 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 RPG/400 어플리케이션의 주석

SQL 주석(--)을 제외하고 RPG/400 주석은 EXEC 및 SQL 키워드 사이를 제외하고 공백이 허용되는 모든 SQL문 내에 포함될 수 있습니다.

RPG/400 주석을 SQL문 내에 삽입하려면 위치 7에 별표(\*)를 배치하십시오.

### SQL을 사용하는 RPG/400 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 위치 시킴으로 복수 행에 걸쳐 계속될 수 있습니다. 이 SQL문에는 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>'<br>C/END-EXEC
```

## SQL을 사용하는 RPG/400 어플리케이션에 코드 포함

SQL문 및 RPG/400 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 SQL문 또는 RPG/400 스펙을 포함시킬 수 있습니다.

## SQL을 사용하는 RPG/400 어플리케이션의 순서 번호

SQL 사전컴파일러로 생성된 소스문의 순서 번호는 CRTSQLRPG 명령에 있는 OPTION 매개변수의 \*NOSEQSRC/\*SEQSRC 키워드를 기준으로 지정됩니다.

\*NOSEQSRC 지정 시 입력 소스 멤버에서 순서 번호가 사용됩니다. \*SEQSRC에 대해 순서 번호는 000001에서 시작하며 1씩 증가합니다.

## SQL을 사용하는 RPG/400 어플리케이션의 이름

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

‘SQ’, ‘SQL’, ‘DI’ 또는 ‘DSN’으로 시작하는 호스트 변수명이나 외부 항목명을 사용하지 마십시오. 이러한 이름들은 데이터베이스 관리자 예약어입니다.

## SQL을 사용하는 RPG/400 어플리케이션의 명령문 레이블

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

## SQL을 사용하는 RPG/400 어플리케이션의 WHENEVER문

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

## SQL을 사용하는 RPG/400 어플리케이션에서 호스트 변수 사용

SQL문에서 사용된 모든 호스트 변수는 명시적으로 선언되어야 합니다. LOB, ROWID 및 2진 호스트 변수는 RPG/400에서 지원되지 않습니다.

RPG/400에 삽입된 SQL은 호스트 변수를 식별하는 데 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.

## SQL을 사용하는 RPG/400 어플리케이션에서 호스트 변수 선언

SQL RPG/400 사전컴파일러는 RPG/400 선언의 서브세트만 유효한 호스트 변수 선언으로 인식합니다.

Most variables defined in RPG/400에 정의된 대부분의 변수는 SQL문에 사용할 수 있습니다. 지원되지 않는 일부 변수에 대한 리스트는 다음과 같습니다.

- 인디케이터 필드 이름(\*INxx)

- 표
- UPDATE
- UDAY
- UMONTH
- UYEAR
- 선행 참조 필드
- 명명된 상수

호스트 변수로 사용되는 필드는 RPG/400의 CALL/PARM 함수를 사용하여 SQL에 전달됩니다. PARM의 결과 필드에 사용될 수 없는 필드는 호스트 변수로서 사용할 수 없습니다.

### SQL을 사용하는 RPG/400 어플리케이션에서 호스트 구조 사용

자료 구조에 서브필드가 있을 경우 RPG/400 자료 구조명을 호스트 구조명으로 사용할 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 RPG/400 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 RPG/400 어플리케이션에서 호스트 구조 배열 사용

호스트 구조 배열은 발생 자료 구조로 정의됩니다. 발생 자료 구조는 복수 행을 페치할 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 RPG/400 어플리케이션에 외부 파일 설명 사용

SQL 사전컴파일러는 ILE RPG 컴파일러와 거의 같은 방식으로 RPG/400 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 RPG/400 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                            KRENAMEDEPTREC
IDEPTREC
I          DEPTNAME                          DEPTN
I          ADMRDEPT                          ADMRD
```

주: RPG/400 명령문을 사용하여 파일에 대한 I/O 조작을 수행하는 경우에만 사용자의 RPG 프로그램에서 파일의 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 RPG/400의 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. GRAPHIC 또는 VARGRAPHIC 열에 UTF-16 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UTF-16 CCSID를 갖게 됩니다.

## SQL을 사용하는 RPG/400 어플리케이션의 호스트 구조 배열에 대한 외부 파일 설명 고려사항 that use SQL

필드 이름 변경을 포함한 외부 서술 파일의 필드 정의는 SQL 사전컴파일러에 의해 인식됩니다.

자료 구조의 외부 정의 형식을 사용하여 호스트 변수로 사용될 열명의 사본을 확보할 수 있습니다.

다음 예에서 DEPARTMENT 표는 RPG/400 프로그램에 포함되고, 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 RPG/400 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 RPG/400 선언

RPG/400 자료 유형	43열	52열	기타 RPG/400 코딩	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
자료 구조 서브필드	공백	공백	길이 = n 여기서 n ≤ 256	452	n	CHAR(n)
자료 구조(서브필드 없음)	n/a	n/a	길이 = n 여기서 n ≤ 9999	452	n	CHAR(n)
입력 필드	공백	공백	길이 = n 여기서 n ≤ 256	452	n	CHAR(n)
연산 결과 필드	n/a	공백	길이 = n 여기서 n ≤ 256	452	n	CHAR(n)
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	1바이트에는 p, 2바이트에는 s	DECIMAL(p,s) , p = n*2-1이고 s = 52열
입력 필드	P	0-9	길이 = n , n= 1-16	484	1바이트에는 p, 2바이트에는 s	DECIMAL(p,s) , p = n*2-1이고 s = 52열
입력 필드	공백	0-9	길이 = n , n= 1-30	484	1바이트에는 p, 2바이트에는 s	DECIMAL(p,s), p = n이고 s = 52열
입력 필드	B	0-4이면 n = 2; 0-9이면 n = 4	길이 = 2 또는 4	484	1바이트에는 p, 2바이트에는 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	1바이트에는 p, 2바이트에는 s	DECIMAL(p,s), p = n이고 s = 52열

표 7. 일반 SQL 자료 유형에 맵핑되는 RPG/400 선언 (계속)

RPG/400 자료 유형	43열	52열	기타 RPG/400 코딩	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
자료 구조 서브필드	공백	0-9	길이 = n , n= 1-30	488	1바이트에는 p, 2바이트에는 s	NUMERIC(p,s), p = n이고 s = 52열

다음 표의 정보를 사용하여 제공된 SQL 자료 유형과 동일한 RPG/400 자료 유형을 판별할 수 있습니다.

표 8. 일반 RPG/400 선언 부분에 맵핑되는 SQL 자료 유형

SQL 자료 유형	RPG/400 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.
DECIMAL	자료 구조의 서브필드. 43열은 P, 서브필드 스펙의 53 열은 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 52 열은 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.
CLOB	지원되지 않습니다.	지원되지 않습니다.
GRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
VARGRAPHIC(n)	지원되지 않습니다.	지원되지 않습니다.
DBCLOB	지원되지 않습니다.	지원되지 않습니다.
BINARY	지원되지 않습니다.	지원되지 않습니다.
VARBINARY	지원되지 않습니다.	지원되지 않습니다.
BLOB	지원되지 않습니다.	지원되지 않습니다.



표 8. 일반 RPG/400 선언 부분에 맵핑되는 SQL 자료 유형 (계속)

SQL 자료 유형	RPG/400 자료 유형	주
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.

## SQL을 사용하는 RPG/400 어플리케이션의 지정 규칙

RPG/400은 정밀도와 스케일을 모든 숫자 유형과 연관시킵니다.

RPG/400은 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 수행 시 자료는 내포된 소수점으로 배열됩니다.

## SQL을 사용하는 RPG/400 어플리케이션에서 인디케이터 변수 사용

인디케이터 변수는 2바이트 정수입니다.

97 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. 검색 시 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.

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

관련 참조

변수 참조

## 예: SQL을 사용하는 RPG/400 어플리케이션에 인디케이터 변수 사용

다음 예는 RPG에서 인디케이터 변수 선언을 표시합니다.

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 RPG/400의 차이점

SQL RPG/400 사전컴파일러는 가능한 경우 구조 매개변수 전달 기법을 사용하려고 합니다.

사전컴파일러는 다음 조건에 모두 참일 경우 각 호스트 변수가 별도의 매개변수인 코드를 생성합니다.

- 명령문에서 참조된 호스트 변수의 자료 길이는 9935보다 큽니다. SQL이 구조의 64바이트를 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 큽니다. 이것은 서브필드가 없는 자료 구조가 호스트 변수로서 사용된 경우이며 길이가 256을 초과합니다. 서브필드의 길이는 256보다 크게 정의될 수 없습니다.

### 관련 정보

데이터베이스 응용프로그램 설계 추가 정보: 구조화된 매개변수 전달 기술 사용

## SQL을 사용하는 호출된 RPG/400 프로그램의 올바른 종료

SQL 수행 시 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 수행 시 사용되는 정적 기억영역을 할당합니다.

SQL을 포함하는 RPG/400 프로그램이 역시 SQL을 포함하는 다른 프로그램에서 호출되면 RPG/400 프로그램은 최종 레코드(LR) 인디케이터를 On으로 설정해서는 안됩니다. LR 인디케이터를 On으로 설정하면 다음에 RPG/400 프로그램을 실행할 때 정적 기억장치가 다시 초기화됩니다. 정적 기억영역이 다시 초기설정되면 내부 SQLDA가 재설정되므로 성능이 떨어지게 됩니다.

SQL문을 포함하는 프로그램에서 호출된 SQL문을 포함하는 RPG/400 프로그램은 다음 두 방법 중 하나로 종료해야 합니다.

- RETRN문에 의해
- RT 인디케이터를 on으로 설정하여



이는 내부 SQLDA를 다시 사용할 수 있게 하며 총 수행 시간을 감소시킵니다.

---

## ILE RPG 어플리케이션에서 SQL문 코딩

이 주제에서는 ILE RPG 프로그램에 SQL문을 삽입할 때의 고유 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 변수에 대한 코딩 요구사항도 정의됩니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

ILE RPG를 사용한 프로그래밍에 대한 자세한 정보는 ILE RPG 프로그래머 안내서  주제 및 ILE RPG 참조서  주제를 참조하십시오.

### 관련 개념

2 페이지의 『SQL을 사용하는 어플리케이션 작성』

iSeries용 DB2 UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성할 수 있습니다.

146 페이지의 『SQL을 사용하는 어플리케이션 프로그램의 컴파일 시 오류 및 경고 메시지』

다음은 컴파일 처리 시 오류 또는 경고 메시지를 발생시킬 수도 있는 조건에 대한 설명입니다.

### 관련 참조

178 페이지의 『예: ILE RPG 프로그램의 SQL문』

이 샘플 프로그램은 ILE 프로그래밍 언어로 작성되었습니다.

## SQL을 사용하는 ILE RPG 어플리케이션에서 SQL 통신 영역 정의

SQL 사전컴파일러는 SET OPTION SQLCA = \*NO문이 있는 경우를 제외하고 첫 번째 연산 스펙 이전의 ILE RPG 프로그램의 정의 스펙에 SQLCA를 자동으로 배치합니다.

소스 프로그램에 INCLUDE SQLCA를 코딩해서는 안됩니다. 소스 프로그램이 INCLUDE SQLCA를 지정하면 명령문이 허용되기는 하지만 중복됩니다. ILE RPG의 SQLCA 소스문은 다음과 같습니다.

```

D*      SQL Communications area
D SQLCA      DS
D  SQLCAID      8A  INZ(X'0000000000000000')
D  SQLAID      8A  OVERLAY(SQLCAID)
D  SQLCABC     10I  0
D  SQLABC     9B  0 OVERLAY(SQLCABC)
D  SQLCODE     10I  0
D  SQLCOD     9B  0 OVERLAY(SQLCODE)
D  SQLERRML    5I  0
D  SQLERL     4B  0 OVERLAY(SQLERRML)
D  SQLERRMC   70A
D  SQLERM     70A  OVERLAY(SQLERRMC)
D  SQLERRP     8A
D  SQLERP     8A  OVERLAY(SQLERRP)
D  SQLERR     24A
D  SQLER1     9B  0 OVERLAY(SQLERR:*NEXT)
D  SQLER2     9B  0 OVERLAY(SQLERR:*NEXT)
D  SQLER3     9B  0 OVERLAY(SQLERR:*NEXT)
D  SQLER4     9B  0 OVERLAY(SQLERR:*NEXT)
D  SQLER5     9B  0 OVERLAY(SQLERR:*NEXT)
D  SQLER6     9B  0 OVERLAY(SQLERR:*NEXT)
D  SQLERRD    10I  0 DIM(6)  OVERLAY(SQLERR)
D  SQLWRN     11A
D  SQLWN0     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWN1     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWN2     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWN3     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWN4     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWN5     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWN6     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWN7     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWN8     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWN9     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWNA     1A  OVERLAY(SQLWRN:*NEXT)
D  SQLWARN     1A  DIM(11) OVERLAY(SQLWRN)
D  SQLSTATE    5A
D  SQLSTT     5A  OVERLAY(SQLSTATE)
D* End of SQLCA

```

SET OPTION SQLCA = \*NO문이 있으면 SQL 사전컴파일러는 정의 스펙에 SQLCODE 및 SQLSTATESQLCA 변수를 자동으로 위치시킵니다. SQLCA가 포함되지 않으면 다음과 같이 정의됩니다.

```

D SQLCODE      S      10I  0
D SQLSTATE     S      5A

```

### 관련 정보

#### SQL 통신 영역

## SQL을 사용하는 ILE RPG 어플리케이션에서 SQL 설명자 영역 정의

- | SQL 설명자 영역에는 두 가지 유형이 있습니다. 하나는 ALLOCATE DESCRIPTOR문으로 정의됩니다. 다른 하나는 SQLDA 구조를 사용하여 정의됩니다. 여기서는 SQLDA 형식에 대해서만 설명합니다.
- | 다음 명령문은 SQLDA를 사용할 수 있습니다.

- 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 INPUT *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 가  
| 질 수 있습니다.

동적 SQL은 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 열의 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.

- | ILE RPG 프로그램에서 INCLUDE SQLDA문을 지정할 수는 있으나 자유 형식은 허용되지 않습니다. 명령  
| 문의 형식은 다음과 같습니다.

```
C/EXEC SQL INCLUDE SQLDA
C/END-EXEC
```

INCLUDE SQLDA는 다음과 같이 자료 구조를 생성합니다.

```
D*      SQL Descriptor area
D SQLDA      DS
D  SQLDAID      1      8A
D  SQLDABC      9      12B 0
D  SQLN        13      14B 0
D  SQLD        15      16B 0
D  SQL_VAR      80A    DIM(SQL_NUM)
D              17      18B 0
D              19      20B 0
D              21      32A
D              33      48*
D              49      64*
D              65      66B 0
D              67      96A
D*
D SQLVAR      DS
D  SQLTYPE      1      2B 0
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 및 참조하는 데 사용됩니다.

SQLDA의 필드 설명을 설정하기 위해 프로그램은 SQLVAR의 서브필드에서 필드 설명을 설정한 후 SQLVAR을 to SQL\_VAR(n)으로 할당합니다. 여기서 n은 SQLDA의 필드 수입니다. 이것은 모든 필드 설명이 설정될 때까지 반복됩니다.

SQLDA 필드 설명이 참조될 때 사용자는 SQLVAR(n)을 SQL\_VAR로 설정합니다. 여기서 n은 처리되는 필드 설명의 수입니다.

관련 정보

동적 SQL 어플리케이션

SQL 설명자 영역

## SQL을 사용하는 ILE RPG 어플리케이션에 SQL문 삽입

ILE RPG 프로그램에서 코딩된 SQL문은 연산 구역에 위치해야 합니다.

이 경우 C가 위치 6에 있어야 합니다. SQL문은 자세한 계산, 합계 또는 RPG 서브루틴에 위치할 수 있습니다. SQL문은 RPG문의 논리를 기반으로 수행됩니다.

SQL문에는 대문자와 소문자 모두 사용될 수 있습니다.

### 고정 양식 RPG

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.

ILE RPG 프로그램에서 코딩된 UPDATE문은 다음과 같이 코딩될 수 있습니다.

```
C/EXEC SQL UPDATE DEPARTMENT
C+           SET MANAGER = :MGRNUM
C+           WHERE DEPTNO = :INTDEP
C/END-EXEC
```

### 자유 양식 RPG

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: 자유 양식으로 코딩된 UPDATE문은 다음과 같이 코딩될 수 있습니다.

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

### | SQL을 사용하는 ILE RPG 어플리케이션의 주석

SQL이 공백 문자를 허용하는 경우 SQL 주석(--) 이외에도 ILE RPG 주석을 SQL문에 포함할 수 있습니다.

#### 고정 양식 RPG

SQL문 내에 ILE RPG 주석을 삽입하려면 위치 7에 별표(\*)를 배치하십시오.

### | 자유 양식 RPG

| 공백이 허용되는 경우 언제든지 EXEC와 SQL 키워드 사이를 제외하고 삽입된 SQL문 내의 위치 8에서 80  
| 사이에 브라켓이 있는 주석(\*...\*)이 허용됩니다. 주석은 여러 행에 걸쳐 있을 수 있습니다. 한 행으로 된 주  
| 석(//)도 사용할 수 있습니다.

#### SQL을 사용하는 ILE RPG 어플리케이션에서 SQL문의 연속

| 이 주제에서는 두 가지 유형의 RPG 형식으로 된 SQL문의 연속에 대해 소개합니다.

#### 고정 양식 RPG

SQL문을 포함시키기 위한 추가 레코드가 필요할 때에는 9-80열이 사용될 수 있습니다. 7열은 반드시 +(더하  
기 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 8열  
에 DBCS 시작(SO) 제어 문자를 위치시킴으로써 복수 행에 걸쳐 계속됩니다.

이 예에서 SQL문에는 G'<AABBCCDDEEFFGGHHIIJJKK>'의 유효한 그래픽 상수가 있습니다.

```
C/EXEC SQL SELECT * FROM GRAPHTAB WHERE GRAPHCOL = G'<AABBCCDDEE>
C+<FFGGHHIIJJKK>'
C/END-EXEC
```

### | 자유 양식 RPG

| SQL문은 하나 이상의 행에 포함될 수 있습니다. 여러 행에 걸쳐 SQL문을 계속하려면 공백이 허용되는 곳마  
| 다 SQL문을 분할할 수 있습니다. 더하기 부호(+)를 사용하여 스트링 상수의 연속을 표시할 수 있습니다. 리터  
| 럽은 다음 행에서 첫 번째 비공백 문자로 계속할 수 있습니다.

#### SQL을 사용하는 ILE RPG 어플리케이션에 코드 포함

SQL문 및 RPG 스펙은 다음 SQL문을 사용하여 포함시킬 수 있습니다.

```
C/EXEC SQL INCLUDE member-name C/END-EXEC
```

RPG 지시문은 RPG 프리프로세서 옵션 매개변수(RPGPPOPT)의 값에 따라 SQL 사전 컴파일러에서 처리됩  
니다.

관련 참조

『SQL을 사용하는 ILE RPG 어플리케이션에서 주석 사용』

RPG 지시문은 RPG 프리프로세서 옵션 매개변수(RPGPPOPT)의 값에 따라 SQL 사전 컴파일러에서 처리됩니다. RPG 프리프로세서가 사용되면 SQL 사전컴파일러가 확장 사전처리 소스를 사용하여 실행됩니다.

## SQL을 사용하는 ILE RPG 어플리케이션에서 주석 사용

RPG 지시문은 RPG 프리프로세서 옵션 매개변수(RPGPPOPT)의 값에 따라 SQL 사전 컴파일러에서 처리됩니다. RPG 프리프로세서가 사용되면 SQL 사전컴파일러가 확장 사전처리 소스를 사용하여 실행됩니다.

- 값이 \*NONE이면 RPG 소스를 사전처리하도록 RPG 프리프로세서가 호출되지 않습니다. SQL 사전컴파일러에서 처리되는 유일한 지시문은 /COPY입니다. 내포된 /COPY문은 처리되지 않습니다. 기타 모든 지시문은 RPG 컴파일러가 호출될 때까지 무시됩니다. 이는 조건부 논리 블록 내의 모든 RPG 및 SQL문이 SQL 사전컴파일러에서 무조건적으로 처리됨을 의미합니다.
- 값이 \*LVL1이면 RPG 소스를 사전처리하도록 RPG 프리프로세서가 호출됩니다. 모든 /COPY문이 확장되고 내포 /COPY문까지 확장된 후 조건부 컴파일 지시문이 처리됩니다.
- 값이 \*LVL2이면 RPG 소스를 사전처리하도록 RPG 프리프로세서가 호출됩니다. 모든 /COPY 및 /INCLUDE문이 확장된 후 조건부 컴파일 지시문이 처리됩니다.
- \*LVL1 또는 \*LVL2가 사용되면 RPG 프리프로세서가 생성한 확장 소스가 매우 커져서 /COPY 및 /INCLUDE문의 확장으로 인한 자원 제한에 도달할 가능성이 있습니다. 이러한 경우가 발생하면 소스를 더 작게 나누거나, RPG 프리프로세서를 사용하지 말아야 합니다.

### 관련 참조

105 페이지의 『SQL을 사용하는 ILE RPG 어플리케이션에 코드 포함』  
SQL문 및 RPG 스펙은 다음 SQL문을 사용하여 포함시킬 수 있습니다.

## SQL을 사용하는 ILE RPG 어플리케이션의 순서 번호

SQL 사전컴파일러로 생성된 소스문의 순서 번호는 CRTSQLRPGI 명령에 있는 OPTION 매개변수의 \*NOSEQSRC/\*SEQSRC 키워드를 기준으로 지정됩니다.

\*NOSEQSRC 지정 시 입력 소스 멤버에서 순서 번호가 사용됩니다. \*SEQSRC에 대해 순서 번호는 000001에서 시작하며 1씩 증가합니다.

## SQL을 사용하는 ILE RPG 어플리케이션의 이름

유효한 모든 ILE RPG 변수명은 호스트 변수로 사용할 수 있으며, 다음과 같은 제한사항이 있습니다.

- 문자 SQ, SQL, RDI 또는 DSN으로 시작하는 호스트 변수명이나 외부 항목명을 사용하지 마십시오. 이러한 이름들은 데이터베이스 관리자 예약어입니다.
- 호스트 변수명의 길이는 64자로 제한됩니다.
- 호스트 변수명은 프로그램 내에서 고유해야 합니다. 한 가지 예외는 독립형 필드, 매개변수 또는 둘 다가 다른 독립형 필드, 매개변수 또는 둘 다와 동일하게 정의된 경우 중복 이름이 허용됩니다.
- 호스트 변수가 중복된 이름이고, 이전 항목에서 언급한 예외 범주에 속하지 않지만 같은 유형일 경우 사전 컴파일러는 일반 심각도 35 대신 심각도 11 오류로서 SQL0314를 발행합니다. 이러한 심각도 11 오류를 무시하고 싶은 경우 CRTSQLRPGI 명령의 GENLVL 매개변수 값을 11 이상으로 변경하십시오.



## SQL을 사용하는 ILE RPG 어플리케이션의 명령문 레이블

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

## SQL을 사용하는 ILE RPG 어플리케이션의 WHENEVER문

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

## SQL을 사용하는 ILE RPG 어플리케이션에서 호스트 변수 사용

SQL문에서 사용된 모든 호스트 변수는 명시적으로 선언되어야 합니다.

ILE RPG에 삽입된 SQL은 호스트 변수를 식별하는 데 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 존재하더라도 프로그램 내에서는 고유해야 합니다. 그러나 자료 구조에 QUALIFIED 키워드가 있으면 해당 자료 구조의 서브필드에는 다른 자료 구조의 서브필드 또는 독립형 변수와 동일한 이름이 있을 수 있습니다. QUALIFIED 키워드가 있는 자료 구조의 서브필드는 서브필드명을 규정하는 자료 구조명을 사용하여 참조되어야 합니다.

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

호스트 변수가 정의되지 않거나 사용할 수 없음을 나타내는 오류가 발행되는 경우 사전컴파일러 리스팅에서 상호 참조를 찾아 사전컴파일러가 변수를 정의하는 방법을 참조하십시오. 리스팅에서 상호 참조를 생성하려면 OPTIONS 매개변수에 지정된 \*XREF를 사용하여 사전컴파일 명령을 실행하십시오.

## SQL을 사용하는 ILE RPG 어플리케이션에서 호스트 변수 선언

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

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

- 부호 없는 정수
- 포인터
- 표
- UDATE
- UDAY
- UMONTH
- UYEAR
- 선행 참조 필드
- 명명된 상수
- 복수 차원 배열

- %SIZE 또는 %ELEM의 분석을 필요로 하는 정의
- 상수가 OCCURS 또는 DIM에서 사용되지 않을 경우에 상수의 분석을 필요로 하는 정의

호스트 변수로 사용되는 필드는 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 명령어나 SET OPTION문의 DATFMT, DATSEP, TIMFMT 및 TIMSEP 매개변수에 의해 지정된 형식과 분리문자를 사용함으로써 선언됩니다. 사용자 선언 호스트 변수 형식에서 사전컴파일 지정 형식으로의 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 사전컴파일러에서의 DATFMT는 IBM SQL 형식인 \*ISO, \*USA, \*EUR 또는 \*JIS 중 하나로 지정되어야 합니다.

그래픽 호스트 변수는 이 중 하나가 지정된 경우 RPG CCSID 값을 사용합니다. SQL DECLARE VARIABLE 문은 CCSID가 RPG에 정의된 호스트 변수 또는 UCS-2나 UTF-16으로 정의되는 호스트 변수의 CCSID를 변경하는 데 사용할 수 없습니다.

사전컴파일러는 RPG 논리(인디케이터) 변수를 길이 1의 문자로 생성합니다. 이 유형은 SQL이 문자 호스트 변수를 허용하는 위치에 관계없이 사용될 수 있습니다. 사전컴파일러는 SQL 인디케이터 변수로 사용될 수 없습니다. 1 또는 0의 값만 사용자에게 할당되는지를 확인해야 합니다.

사전컴파일러는 EXTNAME(filename : fmtname)을 지원하지만, EXTNAME(filename : fmtname : fieldtype)은 지원하지 않습니다. 여기서 fieldtype은 \*ALL, \*INPUT, \*OUTPUT 또는 \*KEY입니다.

사전 컴파일러는 LIKERECE(intrecname)를 지원하지만, 선택적인 두 번째 매개변수는 지원하지 않습니다.

명명되지 않은 서브필드가 있는 경우 사전컴파일러를 통해 서브필드가 있는 자료 구조를 블록화된 페치 및 블록화된 삽입 명령문에 사용할 수 없습니다. 서브필드가 들어 있는 자료 구조를 사용하는 기타 모든 SQL문의 경우 명명된 서브필드만 사용됩니다.

PREFIX 키워드에 쉼표를 포함하는 접두부가 있으면 사전컴파일러는 외부 서술 파일을 인식하지 않게 됩니다.

### **SQL을 사용하는 ILE RPG 어플리케이션에서 2진 호스트 변수 선언:**

ILE RPG에는 SQL 2진 자료 유형에 해당하는 변수가 없습니다.

이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQLTYPE 키워드를 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버의 ILE RPG 언어 선언으로 이 선언을 대체합니다. 2진 선언은 독립적이거나 자료 구조 내에 있습니다.

## BINARY 예

다음 선언을 사용할 경우:

```
D MYBINARY S SQLTYPE(BINARY:50)
```

다음과 같은 코드가 생성됩니다.

```
D MYBINARY S 50A
```

## VARBINARY 예

다음 선언을 사용할 경우:

```
D MYVARBINARY S SQLTYPE(VARBINARY:100)
```

다음과 같은 코드가 생성됩니다.

```
D MYVARBINARY S 100A VARYING
```

주:

1. BINARY 호스트 변수의 경우, 길이의 범위는 1 - 32766입니다.
2. VARBINARY 호스트 변수의 경우, 길이의 범위는 1 - 32740입니다.
3. BINARY 및 VARBINARY 호스트 변수는 호스트 구조에서 선언할 수 있습니다.
4. SQLTYPE, BINARY 및 VARBINARY는 대소문자를 혼합하여 사용할 수 있습니다.
5. SQLTYPE은 위치 44-80열 사이에 있어야 합니다.
6. BINARY 또는 VARBINARY가 독립형 호스트 변수로서 선언되면 24열은 S자를 포함하며 25열은 공백이어야 합니다.
7. 24열의 독립형 필드 인디케이터 S는 BINARY 또는 VARBINARY 호스트 변수가 호스트 구조에서 선언되면 생략됩니다.

## SQL을 사용하는 ILE RPG 어플리케이션에서 LOB 호스트 변수 선언:

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 ILE RPG 언어 구조로 대체합니다. LOB 선언은 독립적이거나 자료 구조 내에 있습니다.

## SQL을 사용하는 ILE RPG 어플리케이션의 LOB 호스트 변수:

다음은 ILE RPG에 있는 LOB 호스트 변수의 예입니다.

## 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
```

## 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
```

주:

1. BLOB 및 CLOB의 경우  $1 \leq \text{lob-length} \leq 65\ 531$
2. DBCLOB의 경우  $1 \leq \text{lob-length} \leq 16\ 383$
3. LOB 호스트 변수는 호스트 구조에서 선언할 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 24는 'S'자를 포함해야 하고 위치 25는 공백이어야 합니다.
9. 위치 24의 독립형 필드 인디케이터 S는 LOB이 호스트 구조에서 선언되면 생략됩니다.
10. LOB 호스트 변수는 초기화될 수 없습니다.

### SQL을 사용하는 ILE RPG 애플리케이션의 LOB 위치 지정자:

다음은 ILE RPG의 LOB 위치 지정자 예입니다.

## 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.

### Notes:

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

### SQL을 사용하는 ILE RPG 애플리케이션의 LOB 파일 참조 변수:

다음은 ILE RPG에 있는 LOB 파일 참조 변수의 예입니다.

## 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.

### Notes:

1. LOB 파일 참조 변수는 호스트 구조에서 선언할 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.

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

관련 정보

LOB 파일 참조 변수

### SQL을 사용하는 ILE RPG 어플리케이션에서 호스트 ROWID 선언:

ILE RPG에는 SQL 자료 유형 ROWID에 해당하는 변수가 없습니다.

이러한 자료 유형에 대해 사용될 수 있는 호스트 변수를 작성하려면 SQLTYPE 키워드를 사용하십시오. SQL 사전컴파일러는 출력 소스 멤버의 ILE RPG 언어 선언으로 이 선언을 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 선언할 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 ILE RPG 어플리케이션에서 호스트 구조 사용

자료 구조에 서브필드가 있을 경우 ILE RPG 자료 구조명을 호스트 구조명으로 사용할 수 있습니다. SQL문에 자료 구조명을 사용하면 서브필드명의 리스트가 자료 구조를 구성한다는 의미가 됩니다.

자료 구조에 명명되지 않은 서브필드가 포함되면, 자료 구조 이름은 SQL문에서 호스트 변수로 사용될 수 없습니다. 명명된 서브필드는 호스트 변수로 사용될 수 있습니다.

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

```
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.

```
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+ FROM CORPDATA.EMPLOYEEC+ WHERE EMPNO = :EMPNO C/END-EXEC
```

SQL문을 작성할 때, QUALIFIED 자료 구조에 있지 않은 서브필드에 대한 참조를 규정할 수 있습니다. 뒤에 마침표와 서브필드명이 있는 자료 구조명을 사용하십시오. 예를 들어 PEMPL.MIDINT는 MIDINT만 지정하는 것과 동일합니다. 자료 구조에 QUALIFIED 키워드가 있으면 서브필드는 서브필드명을 규정하는 자료 구조명을 사용하여 참조되어야 합니다.

이 예에서는 다음과 같이 동일한 서브필드명이 포함된 두 개의 자료 구조(하나는 QUALIFIED이고 다른 하나는 QUALIFIED가 아님)가 있습니다.

```
Dfststruct      DS
D sub1          4B 0
D sub2          9B 0
D sub3          20I 0
D sub4          9B 0

Dsecstruct      DS          QUALIFIED
D sub1          4A
D sub2          12A
D sub3          20I 0
D myvar         5A
D sub5          20A

D myvar         S          10I 0
```

호스트 변수로서의 *secstruct.sub1* 참조는 길이가 4인 문자 변수입니다.

호스트 변수로서의 *sub2*에는 작은 정수의 SQL 자료 유형이 있습니다. QUALIFIED가 아닌 자료 구조에서 해당 속성을 선택합니다.

*myvar*에 대한 호스트 변수 참조는 독립형 선언을 사용하여 데이터 유형 정수를 선택합니다. *secstruct.myvar*을 사용하는 경우, QUALIFIED 구조에 문자 변수가 사용됩니다.

*sub5*는 QUALIFIED 자료 구조에 없기 때문에 *secstruct*로 규정해야 참조할 수 있습니다.

사전 컴파일러는 LIKEDS 키워드를 사용하여 정의된 호스트 구조를 인식하지 않습니다. 그러나 호스트 변수의 SQL 구문은 SQL문에 있는 규정의 단일 레벨만 사용할 수 있습니다. 이는 자료 구조 DS에 서브필드 S2가 있는 자료 구조와 같은 방식으로 정의된 서브필드 S1이 있는 경우에 SQL문이 완전 규정 호스트 변수 DS.S1.S2를 사용하여 S2로 참조할 수 없음을 의미합니다. 호스트 변수 참조로 S1.S2를 사용하는 경우, 사전컴파일러는 DS.S1.S2로 변수를 인식하게 됩니다. 다음과 같은 추가 제한사항이 적용됩니다.

- 최상위 레벨 구조인 DS를 배열할 수 없습니다.
- S1.S2는 고유해야 합니다. 즉, 서브필드 S1.S2가 있는 구조 S1 또는 서브필드 DS3.S0.S1.S2가 있는 구조 DS3과 같이 S1.S2로 끝나는 프로그램에 다른 유효한 이름이 없어야 합니다.

## 예

```
D CustomerInfo      DS                QUALIFIED
D   Name            20A
D   Address         50A

D ProductInfo      DS                QUALIFIED
D   Number          5A
D   Description     20A
D   Cost            9P 2

D SalesTransaction...
D                   DS                QUALIFIED
D   Buyer           LIKEDS(CustomerInfo)
D   Seller          LIKEDS(CustomerInfo)
D   NumProducts    10I 0
D   Product        LIKEDS(ProductInfo)
D                   DIM(10)

C/EXEC SQL
C+ SELECT * INTO :CustomerInfo.Name, :Buyer.Name FROM MYTABLE
C/END-EXEC
```

*CustomerInfo.Name*은 QUALIFIED 구조의 변수에 대한 참조로 인식됩니다. *Buyer.Name*은 *SalesTransaction.Buyer.Name*으로 정의됩니다.

SQL 구문에서는 하나의 규정 레벨만 허용되므로 SQL문의 *SalesTransaction.Buyer.Name*을 사용할 수 없습니다. COST가 차원화 배열에 있으므로 SQL문에서는 *Product.Cost*를 사용할 수 없습니다.

*SalesTransaction*과 같이 정의된 *SalesTransaction2*가 있으면 구조로 된 서브필드는 SQL문에서 사용될 수 없습니다. SQL은 하나의 규정 레벨만 지원하므로 *Buyer.Name*에 대한 참조는 분명하지 않습니다.



## SQL을 사용하는 ILE RPG 어플리케이션에서 호스트 구조 배열 사용

호스트 구조 배열은 발생 자료 구조로 정의됩니다. 두 유형의 자료 구조는 모두 복수 행 처리 시 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 분리자가 CRTSQLRPGI 명령(또는 SET OPTION문)에서의 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 DEPARTMENT로 불리는 호스트 구조 및 블록화된 FETCH문을 사용합니다.

```
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
```

블록화된 FETCH 및 블록화된 INSERT는 DIM 키워드가 있는 자료 구조를 허용하는 유일한 SQL문입니다. *MyStructure(index).MySubfield*와 같은 아래 첨자가 있는 호스트 변수 참조는 SQL에서 지원하지 않습니다.

### 예

```
Dfststruct       DS              DIM(10)  QUALIFIED
D sub1            4B 0
D sub2            9B 0
D sub3            20I 0
D sub4            9B 0

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

## SQL을 사용하는 ILE RPG 어플리케이션에서 외부 파일 설명 사용

필드 이름 변경을 포함한 외부 서술 파일의 필드 정의는 SQL 사전컴파일러에 의해 인식됩니다. 자료 구조의 외부 정의 형식을 사용하여 호스트 변수로 사용될 열 이름의 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 검색 시 형식과 분리문자가 무시되고 사전컴파일러는 변수 선언이 문자 형식의 날짜/시간 호스트 변수인 것으로 가정합니다. \*CVTDT는 ILE RPG 사전컴파일러에 대한 호환성 옵션입니다.

GRAPHIC 또는 VARGRAPHIC 열에 UCS-2 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UCS-2 CCSID를 갖게 됩니다. GRAPHIC 또는 VARGRAPHIC 열에 UTF-16 CCSID가 있으면 생성된 호스트 변수는 그 변수에 할당된 UTF-16 CCSID를 갖게 됩니다.

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

```
FDEPARTMENTIP E DISK RENAME(ORIGREC:DEPTREC)
```

주: ILE RPG문을 사용하여 파일에 I/O 조작을 수행하는 경우에만 ILE RPG 프로그램의 파일에 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.

```
DTDEPT E DS EXTNAME(DEPARTMENT)
D DEPTN E EXTFLD(DEPTNAME)
D ADMRD E EXTFLD(ADMRDEPT)
```

## SQL을 사용하는 ILE RPG 어플리케이션의 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 ILE RPG 프로그램에 포함되고, 호스트 구조 배열을 선언하는 데 사용됩니다. 그런 후 블록 FETCH문은 호스트 구조 배열 내로 10개의 행을 검색하는 데 사용됩니다.

```
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 ROWSC+ INTO :DEPARTMENT C/END-EXEC

```

## 동등한 SQL 및 ILE RPG 자료 유형 판별

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

표9. 일반 SQL 자료 유형에 맵핑되는 ILE RPG 선언

RPG 자료 유형	RPG 코딩	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
자료 구조(서브필드 없음)	길이 = n 여기서 n ≤ 32766.	452	n	CHAR(n)
존 자료	<ul style="list-style-type: none"> <li>자료 유형이 S 또는 공백인 서브필드로 정의 스펙에 정의됨</li> <li>자료 유형이 S인 정의 스펙에 정의됨</li> <li>자료 유형이 S 또는 공백인 입력 스펙에 정의됨</li> </ul>	488	1바이트에는 p, 2바이트에는 s	NUMERIC(p, s). 여기서 p는 자릿수 번호이고 s는 소수 자리 번호임
팩 자료	<ul style="list-style-type: none"> <li>공백이 아닌 소수 자리(위치 69 - 70)가 있는 정의 스펙에 정의됨</li> <li>자료 유형이 P인 정의 스펙 서브필드에 정의됨</li> <li>자료 유형이 P 또는 공백인 정의 스펙에 정의됨</li> <li>자료 유형이 P인 입력 스펙에 정의됨</li> </ul>	484	1바이트에는 p, 2바이트에는 s	DECIMAL(p, s). 여기서 p는 자릿수 번호이고 s는 소수 자리 번호임
제로 소수 자릿수가 있는 22바이트진	<ul style="list-style-type: none"> <li>시작 및 끝 위치가 있고 자료 유형이 B이며 바이트 길이가 2인 서브필드로 정의 스펙에 정의됨</li> <li>자료 유형이 B이고 자릿수가 1 - 4인 정의 스펙에 정의됨</li> <li>자료 유형이 B이고 바이트 길이가 2인 입력 스펙에 정의됨</li> </ul>	500	2	SMALLINT

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

RPG 자료 유형	RPG 코딩	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
제로 소수 자릿수가 있는 42바이트진	<ul style="list-style-type: none"> <li>• 시작 및 끝 위치가 있고 자료 유형이 B이며 바이트 길이가 4인 서브필드로 정의 스펙에 정의됨</li> <li>• 자료 유형이 B이고 자릿수가 5 - 9인 정의 스펙에 정의됨</li> <li>• 자료 유형이 B이고 바이트 길이가 4인 입력 스펙에 정의됨</li> </ul>	496	4	INTEGER
2바이트 정수	<ul style="list-style-type: none"> <li>• 시작 및 끝 위치가 있고 자료 유형이 I이며 바이트 길이가 2인 서브필드로 정의 스펙에 정의됨</li> <li>• 자료 유형이 I이고 자릿수가 5인 정의 스펙에 정의됨</li> <li>• 자료 유형이 I이고 바이트 길이가 2인 입력 스펙에 정의됨</li> </ul>	500	2	SMALLINT
4바이트 정수	<ul style="list-style-type: none"> <li>• 시작 및 끝 위치가 있고 자료 유형이 I이며 바이트 길이가 4인 서브필드로 정의 스펙에 정의됨</li> <li>• 자료 유형이 I이고 자릿수가 10인 정의 스펙에 정의됨</li> <li>• 자료 유형이 I이고 바이트 길이가 4인 입력 스펙에 정의됨</li> </ul>	496	4	INTEGER
8바이트 정수	<ul style="list-style-type: none"> <li>• 시작 및 끝 위치가 있고 자료 유형이 I이며 바이트 길이가 8인 서브필드로 정의 스펙에 정의됨</li> <li>• 자료 유형이 I이고 자릿수가 20인 정의 스펙에 정의됨</li> <li>• 자료 유형이 I이고 바이트 길이가 8인 입력 스펙에 정의됨</li> </ul>	492	8	BIGINT
short float	자료 유형 = F, 길이 = 4	480	4	FLOAT(단정밀도)
long float	자료 유형 = F, 길이 = 8	480	8	FLOAT(배정밀도)
문자	자료 유형 = A 또는 공백, 소수 자리 공백, 1 - 32766의 길이	452	n	CHAR (n). n은 길이임

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

RPG 자료 유형	RPG 코딩	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
254보다 큰 길이를 변환하는 문자	자료 유형 = A 또는 공백, 소수 자리 공백, 정의 스펙의 VARYING 키워드 또는 입력 스펙의 형식 *VAR	448	n	VARCHAR (n). n은 길이임
1 - 254의 길이를 변환하는 문자	자료 유형 = A 또는 공백, 소수 자리 공백, 정의 스펙의 VARYING 키워드 또는 입력 스펙의 형식 *VAR	456	n	VARCHAR (n). n은 길이임
그래픽	<ul style="list-style-type: none"> <li>• 시작 및 끝 위치가 있고 자료 유형이 G이며 바이트 길이가 b인 서브필드로 정의 스펙에 정의됨</li> <li>• 자료 유형이 G이고 길이가 n인 정의 스펙에 정의됨</li> <li>• 자료 유형이 G이고 바이트 길이가 b인 입력 스펙에 정의됨</li> </ul>	468	m	GRAPHIC(m). m = n 또는 m = b/2
그래픽 변환	<ul style="list-style-type: none"> <li>• 시작 및 끝 위치가 있고 자료 유형이 G이고 바이트 길이가 b이며 VARYING 키워드가 있는 서브필드로 정의 스펙에 정의됨</li> <li>• 자료 유형이 G이고 길이가 n이며 VARYING 키워드가 있는 정의 스펙에 정의됨</li> <li>• 자료 유형이 G이고 바이트 길이가 b이며 형식이 *VAR인 입력 스펙에 정의됨</li> </ul>	464	m	VARGRAPHIC(m). m = n 또는 m = (b-2)/2
UCS-2	<ul style="list-style-type: none"> <li>• 시작 및 끝 위치가 있고 자료 유형이 C이며 바이트 길이가 b인 서브필드로 정의 스펙에 정의됨</li> <li>• 자료 유형이 C이고 길이가 n인 정의 스펙에 정의됨</li> <li>• 자료 유형이 C이고 바이트 길이가 b인 입력 스펙에 정의됨</li> </ul>	468	m	CCSID가 13488인 GRAPHIC(m). 여기서 m = n 또는 m = b/2

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

RPG 자료 유형	RPG 코딩	SQLTYPE 호스트 변수	SQLLEN 호스트 변수	SQL 자료 유형
UCS-2 변환	<ul style="list-style-type: none"> <li>• 시작 및 끝 위치가 있고 자료 유형이 C이고 바이트 길이가 b이며 VARYING 키워드가 있는 서브필드로 정의 스펙에 정의됨</li> <li>• 자료 유형이 C이고 길이가 n이며 VARYING 키워드가 있는 정의 스펙에 정의됨</li> <li>• 자료 유형이 C이고 바이트 길이가 b이며 형식이 *VAR인 입력 스펙에 정의됨</li> </ul>	464	m	CCSID가 13488인 VARGRAPHIC(m). 여기서 m = n 또는 m = b/2
날짜	<ul style="list-style-type: none"> <li>• DATFMT 키워드에서 자료 유형이 D, 형식이 f이며 분리자가 s인 정의 스펙에 정의됨</li> <li>• 자료 유형이 D이고 위치 31 - 34의 형식, 위치 35의 분리자가 있는 입력 스펙에 정의됨</li> </ul>	384	n	DATE DATFMT(f) DATSEP(s) <sup>1</sup>
시간	<ul style="list-style-type: none"> <li>• TIMFMT 키워드에서 자료 유형이 T, 형식이 f 및 분리자가 s인 정의 스펙에 정의됨</li> <li>• 자료 유형이 T이고 위치 31 - 34의 형식, 위치 35의 분리자가 있는 입력 스펙에 정의됨</li> </ul>	388	n	TIME TIMFMT(f) TIMSEP(s) <sup>1</sup>
시간소인	자료 유형 Z	392	n	TIMESTAMP

<sup>1</sup>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 형식으로서의 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	

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

SQL 자료 유형	RPG 자료 유형	주
INTEGER	정의 스펙. 위치 40에서 I, 길이는 위치 42에서 10과 0  또는  정의 스펙. 위치 40에서 B, 길이는 위치 42에서 ≤ 9 및 ≥ 5 및 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	
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.
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.
BINARY	지원되지 않습니다.	BINARY를 선언하려면 SQLTYPE 키워드를 사용하십시오.

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

SQL 자료 유형	RPG 자료 유형	주
VARBINARY	지원되지 않습니다.	BINARY를 선언하려면 SQLTYPE 키워드를 사용하십시오.
BLOB	지원되지 않습니다.	BLOB를 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. 마이크로초를 포함하려면 최소한 26이 되어야 합니다. 길이가 25 이하일 경우 마이크로초 부분에서 잘립니다.
DATALINK	지원되지 않습니다.	
ROWID	지원되지 않습니다.	ROWID를 선언하려면 SQLTYPE 키워드를 사용하십시오.

## ILE RPG 변수 선언 및 사용에 대한 주

ILE RPG는 정밀도와 스케일을 모든 숫자 유형과 연관시킵니다.

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 수행 시 자료는 내포된 소수점으로 배열됩니다.

## SQL을 사용하는 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.



검색 시 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.

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

관련 참조

변수 참조

## 예: SQL을 사용하는 ILE RPG 어플리케이션에서 인디케이터 변수 사용

다음은 ILE RPG에서 인디케이터 변수를 선언하는 예입니다.

다음 명령문을 보십시오.

```
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.

```
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
```

## SQL을 사용하는 ILE RPG 어플리케이션의 복수 행 영역 페치에 대한 SQLDA 예

다음은 ILE RPG의 복수 행 영역 페치에 대한 SQLDA 예입니다.

```
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           sqllen = 3
C                eval           sql_var(1) = sqlvar
```

```

C*
C* setup the second entry in the sqlda
C*
C          eval      sqltype = 453
C          eval      sqllen  = 29
C          eval      sql_var(2) = sqlvar
...
C*
C* setup the forth entry in the sqlda
C*
C          eval      sqltype = 453
C          eval      sqllen  = 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 :SQLDAC+          INTO :DEPARTMENT:IND_ARRAY   C/END-EXEC

```

## SQL을 사용하는 ILE RPG 어플리케이션의 동적 SQL 예

다음은 ILE RPG에서 동적 SQL 사용 예입니다.

```

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

```

---

## REXX 어플리케이션에서 SQL문 코딩


REXX 프로시듀어는 사전처리할 필요가 없습니다. 실행 시에 REXX 인터프리터는 자신이 이해하지 못하는 명령문을 현재 사용 중인 명령 환경으로 전달하여 처리합니다.

명령 환경은 알 수 없는 모든 명령문들을 데이터베이스 관리자에 송신하기 위해 다음 두 가지 방식에 의해 \*EXECSQL로 변경될 수 있습니다.

1. STREXPRC CL 명령의 CMDENV 매개변수
2. ADDRESS REXX 명령의 주소 위치 매개변수

STREXPRC CL 명령 또는 ADDRESS REXX 명령에 대한 자세한 정보는 REXX/400 프로그래머 안내서



주제 및 REXX/400 참조서  주제를 참조하십시오.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

### 관련 개념

2 페이지의 『SQL을 사용하는 어플리케이션 작성』

iSeries용 DB2 UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성할 수 있습니다.

146 페이지의 『SQL을 사용하는 어플리케이션 프로그램의 컴파일 시 오류 및 경고 메시지』  
다음은 컴파일 처리 시 오류 또는 경고 메시지를 발생시킬 수도 있는 조건에 대한 설명입니다.

#### 관련 참조

13 페이지의 『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.

184 페이지의 『예: REXX 프로그램의 SQL문』

이 샘플 프로그램은 REXX 프로그래밍 언어로 작성되었습니다.

## REXX 어플리케이션에서 SQL 통신 영역 사용

SQL 통신 영역(SQLCA)을 구성하는 필드는 SQL/REXX 인터페이스에 의해 자동으로 포함됩니다.

INCLUDE SQLCA문은 필요없으며 허용되지도 않습니다. SQLCA의 SQLCODE 및 SQLSTATE 필드에는 SQL 리턴 코드가 들어 있습니다. 이 값은 각각의 SQL문이 실행된 후에 데이터베이스 관리자에 의해 설정됩니다. 어플리케이션은 SQLCODE 값을 체크하여 최종 SQL문이 제대로 수행되었는지를 판별합니다.

SQL/REXX 인터페이스는 일반적인 SQL 사용과 똑같은 방법으로 SQLCA를 사용합니다. 그러나 SQL/REXX 인터페이스는 연속 자료 영역보다는 분리된 변수에 있는 SQLCA의 필드를 유지보수합니다. SQL/REXX 인터페이스가 SQLCA를 위해 유지보수하는 변수는 다음과 같이 정의됩니다.

### SQLCODE

1차 SQL 리턴 코드

### SQLERRMC

오류 및 경고 메시지 토큰

### SQLERRP

제품 코드 및 오류가 있을 경우 오류를 리턴한 모듈명

### SQLERRD.n

진단 정보가 들어 있는 6개의 변수( $n$ 은 1 - 6 사이의 숫자).

### SQLWARN.n

경고 플래그가 들어 있는 11개의 변수( $n$ 은 0 - 10 사이의 숫자).

### SQLSTATE

대체 SQL 리턴 코드

관련 정보

SQL 통신 영역

## REXX 어플리케이션에서 SQL 설명자 영역 사용

| SQL 설명자 영역에는 두 가지 유형이 있습니다. 하나는 ALLOCATE DESCRIPTOR문으로 정의됩니다. 다른 하나는 SQL 설명자 영역(SQLDA) 구조를 사용하여 정의됩니다. 여기서는 SQLDA 형식에 대해서만 설명합니다. 할당된 설명자는 REXX에서 지원되지 않습니다.

| 다음 명령문은 SQLDA를 사용할 수 있습니다.

- 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 세트로 구성되며 여기서 어간 이름은 해당 SQL문의 *descriptor-name*입니다. 이는 단순 어간이어야 합니다. 즉, 어간 자체에 마침표가 들어 있어서는 안됩니다. 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.

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 DESCRIPTOR문 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

**908/909**  
VARBINARY 호스트 변수

**912/913**  
BINARY 호스트 변수

**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.

관련 정보

SQL 설명자 영역

## **REXX 어플리케이션에서 SQL문 삽입**

SQL문은 REXX 명령이 위치할 수 있는 곳이면 어디에서나 위치할 수 있습니다.

REXX 프로시저어의 각 SQL문은 반드시 EXECSQL(대소문자의 조합으로)로 시작하고 뒤에 다음 중 하나가 와야 합니다.

- 작은 따옴표 또는 큰 따옴표로 묶인 SQL문 또는
- 명령문이 들어 있는 REXX 변수. SQL문이 들어 있을 때 콜론은 REXX에 선행해서는 안됩니다.

예를 들면 다음과 같습니다.

```
EXECSQL "COMMIT"
```

는 다음과 같습니다.

```
rexxvar = "COMMIT"  
EXECSQL rexxvar
```

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 명령어를 허용합니다. 예를 들면 다음과 같습니다.

'EXECSQL COMMIT'

SQL/REXX 인터페이스는 다음 SQL문을 지원합니다.

ALTER SEQUENCE	EXECUTE
ALTER TABLE	EXECUTE IMMEDIATE
CALL <sup>2</sup>	FETCH <sup>1</sup>
CLOSE	GRANT
COMMENT ON	INSERT <sup>1</sup>
COMMIT	LABEL ON
CREATE ALIAS	LOCK TABLE
CREATE DISTINCT TYPE	OPEN
CREATE FUNCTION	PREPARE
CREATE INDEX	REFRESH
CREATE PROCEDURE	RELEASE SAVEPOINT
CREATE SCHEMA	RENAME
CREATE SEQUENCE	REVOKE
CREATE TABLE	ROLLBACK
CREATE TRIGGER	SAVEPOINT
CREATE VIEW	SET ENCRYPTION PASSWORD
DECLARE CURSOR <sup>2</sup>	SET OPTION <sup>3</sup>
DECLARE GLOBAL TEMPORARY TABLE	SET PATH
DELETE <sup>2</sup>	SET SCHEMA
DESCRIBE	SET TRANSACTION
DESCRIBE TABLE	SET variable <sup>2</sup>
DROP	UPDATE <sup>2</sup>
	VALUES INTO <sup>2</sup>

다음 SQL문은 SQL/REXX 인터페이스에 의해 지원되지 않습니다.

ALLOCATE DESCRIPTOR	GET DIAGNOSTICS
BEGIN DECLARE SECTION	HOLD LOCATOR
CONNECT	INCLUDE
DEALLOCATE DESCRIPTOR	RELEASE
DECLARE PROCEDURE	SELECT INTO
DECLARE STATEMENT	SET CONNECTION
DECLARE VARIABLE	SET CURRENT DEGREE
DESCRIBE INPUT	SET DESCRIPTOR
DISCONNECT	SET RESULT SETS
END DECLARE SECTION	SET SESSION AUTHORIZATION
FREE LOCATOR	SIGNAL
GET DESCRIPTOR	WHENEVER <sup>1</sup>

123

---

1. 이 명령문의 블록화 양식은 지원되지 않습니다.

2. 이들 명령문은 호스트 변수가 들어 있는 경우 직접 실행될 수 없으며 PREPARE 오브젝트인 경우에만 EXECUTE입니다.

3. SET OPTION문은 SQL문 실행에 사용되는 일부 처리 옵션을 변경하기 위해 REXX 프로시저에서 사용됩니다. 이 옵션에는 확약 제어 레벨 및 날짜 형식이 들어 있습니다. SET OPTION문에 대한 자세한 정보는 SQL 참조서 주제를 참조하십시오.



## SQL을 사용하는 REXX 어플리케이션의 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 쉼표나 연결 연산자에 의해 분리된 여러 개의 스트링으로 분할될 수 있습니다.

## SQL을 사용하는 REXX 어플리케이션에 코드 포함

다른 호스트 언어와는 달리 외부에서 정의된 명령문을 포함하기 위한 지원은 제공되지 않습니다.

## SQL을 사용하는 REXX 어플리케이션의 여백

SQL/REXX 인터페이스를 위한 특별한 여백 규칙은 없습니다.

## SQL을 사용하는 REXX 어플리케이션의 이름

마침표(.)로 끝나는 유효한 REXX 이름은 호스트 변수에 사용될 수 있습니다. 이름은 64개 이하의 문자이어야 합니다.

변수 이름은 'SQL', 'RDI', 'DSN', 'RXSQL' 또는 'QRW'로 시작해서는 안됩니다.

## SQL을 사용하는 REXX 어플리케이션의 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 어플리케이션의 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. 그러나 몇 가지 대체 항목 중 하나를 사용할 수 있습니다.

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, ROWID 및 2진 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.

## 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 변수절'에서 사용된 호스트 변수)은 아래 표에 따라 변수 내용에서 수행 시 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. REXX에서 호스트 변수의 자료 유형 판별

호스트 변수 내용	가정된 자료 유형	SQL 유형 코드	SQL 유형 설명
소수점과 지수 모두가 없는 숫자 앞에 덧셈 또는 뺄셈 부호가 있는 정수 부호를 가질 수 있습니다.		496/497	INTEGER
소수점을 포함하고 있으나 지수는 없는 숫자	팩 십진수	484/485	DECIMAL(m,n)

또는 소수점 또는 지수를 포함하고 있지 않으며  
2147483647 이상이거나 -2147483647 이하인 숫자.

앞에 덧셈 또는 뺄셈 부호를 가질 수 있습니다. *m*은 숫자에서 총 자릿수입니다. *n*은 소수점의 왼쪽에 있는 자릿수입니다(있는 경우).

표 11. REXX에서 호스트 변수의 자료 유형 판별 (계속)

호스트 변수 내용	가정된 자료 유형	SQL 유형 코드	SQL 유형 설명
과학 또는 공학에서의 표기법으로 된 숫자(즉, 'E' 또는 'e', 선택적인 덧셈이나 뺄셈 부호 및 일련의 숫자가 바로 뒤에 오는). 앞에 덧셈 또는 뺄셈 부호를 가질 수 있습니다.	부동 소수점	480/481	DOUBLE PRECISION
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: <sup>1</sup>  • G, g, N 또는 n으로 시작되는 스트링. 그 다음에는 작은 따옴표나 큰 따옴표 및 SO(x'OE')가 옵니다. 이는 각각 2개의 문자 길이를 갖는 n의 그래픽 문자가 뒤에 옵니다. 스트링은 한글/한자 끝 제어 문자(X'OF') 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'OE')가 붙는 스트링. 이는 각각 2개의 문자 길이를 갖는 n의 그래픽 문자가 뒤에 옵니다. 스트링은 한글/한자 끝 제어 문자(X'OF') 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)
정의되지 않은 변수	값이 할당되지 않은 변수	없음	유효하지 않은 자료가 검출되었습니다.

주: 선행 작은 따옴표 바로 다음에 오는 바이트는 X'OE' SO이고 후미 작은 따옴표 바로 앞에 오는 바이트는 X'OF' SI입니다.

### SQL을 사용하는 REXX 어플리케이션에서 출력 호스트 변수의 형식

출력 호스트 변수(즉, FETCH문의 'INTO 호스트 변수' 절에 사용되는 호스트 변수)의 자료 유형을 판별할 필요가 없습니다

출력 값은 다음과 같이 호스트 변수에 할당됩니다.

- 문자값은 선행 및 후미 작은 따옴표없이 할당됩니다.

- 그래픽 값은 선행 G나 작은 따옴표없이, 후미 작은 따옴표없이 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 대문자로 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.

예를 들면 다음과 같습니다.

```
stringvar = '100'
```

은 REXX가 변수 *stringvar*를 문자 스트링 100(작은 따옴표 포함 안함)으로 변환하도록 합니다. 즉, SQL/REXX 인터페이스에 의해 숫자 100으로 평가되며 이 상태에서 SQL로 전달됩니다.

한편,

```
stringvar = ""100""
```

은 REXX가 변수 *stringvar*를 문자 스트링 '100'(작은 따옴표 포함)으로 변환하도록 합니다. 이는 SQL/REXX 인터페이스에 의해 스트링 100으로 평가되며 이 상태에서 SQL로 전달됩니다.

## SQL을 사용하는 REXX 어플리케이션에서 인디케이터 변수 사용

인디케이터 변수는 정수입니다.

검색 시, 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.

관련 참조

변수 참조

---

## SQL문을 사용하는 프로그램 준비 및 실행

이 주제에서는 어플리케이션 프로그램을 준비하고 실행하기 위한 일부 작업을 설명합니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

관련 개념

2 페이지의 『SQL을 사용하는 어플리케이션 작성』

iSeries용 DB2 UDB SQL문과 함수를 사용하는 호스트 언어로 데이터베이스 어플리케이션을 작성할 수 있습니다.

## SQL 사전컴파일러의 기본 처리

삽입된 SQL문을 수행하기 전에 그것이 들어 있는 어플리케이션 프로그램을 사전컴파일 및 컴파일해야 합니다.

주: REXX 프로시저어 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 프로시저어는 발생하는 모든 오류를 정정할 때 도움을 주는 출력 리스트에 오류 메시지를 제공합니다.
- 데이터베이스의 설명을 사용하여 SQL문의 유효성을 확인합니다. 사전컴파일 시 SQL문은 유효한 표명, 뷰명 및 열명에 대해 체크됩니다. 지정된 표나 뷰가 존재하지 않거나 사전컴파일이나 컴파일 시 표나 뷰에 대해 권한이 없으면 유효성 검사는 수행 시에 이루어집니다. 수행 시에 표나 뷰가 존재하지 않으면 오류가 발생합니다.

### Notes:

1. 대체는 외부 정의를 검색할 때 처리됩니다.
  2. 사용자가 SQL문의 유효성을 검사하기 위해서는 SQL문에서 참조된 표나 뷰에 대한 일부 권한(최소한 \*OBJPR)이 필요합니다. SQL문을 처리하는 데 필요한 실제 권한은 수행 시 체크됩니다.
  3. RDB 매개변수가 CRTSQLxxx 명령에 지정될 때 사전컴파일러는 표 및 뷰 설명을 구하기 위하여 지정된 관계형 데이터베이스에 액세스합니다.
- 호스트 언어로 컴파일할 각 SQL문을 준비합니다. 대부분의 SQL문에 대해 SQL 사전컴파일러는 주석 및 CALL문을 SQL 인터페이스 모듈 중 하나에 삽입합니다. 일부 SQL문(예를 들어 DECLARE문)의 경우 SQL 사전컴파일러는 주석을 제외하고는 호스트 언어를 생성하지 않습니다.
  - 사전컴파일된 각 SQL문에 관한 정보를 작성합니다. 정보는 바인드 처리 시 사용 가능한 임시 소스 파일 멤버에 내부적으로 저장됩니다.

사전컴파일 시 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)

관련 정보

데이터베이스 프로그래밍  
데이터베이스 파일 관리  
SQL 참조

## SQL 사전컴파일러에 대한 입력

어플리케이션 프로그램 작성 명령문과 삽입된 SQL문이 SQL 사전컴파일러에 대한 1차 입력입니다.

PL/I, C 및 C++ 프로그램에서 SQL문은 CRTSQLPLI, CRTSQLCI 및 CRTSQLCPPI 명령의 MARGINS 매개변수에서 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 CRTSQLxxx의 INCFILE 매개변수에 의해 지정된 파일의 2차 입력을 얻는 데 사용될 수 있습니다. 이 명령에서 xxx는 호스트 언어 인디케이터를 말합니다. iSeries용 COBOL 언어의 경우 CBL, iSeries용 ILE COBOL 언어의 경우 CBL I, iSeries PL/I 언어의 경우 PL I, iSeries용 ILE C 언어의 경우 CI, RPG/400 언어의 경우 RPG, ILE RPG 언어의 경우 RPGI, ILE C++/400 언어의 경우 CPPI입니다. SQL INCLUDE문을 사용하면 멤버 끝에 도달할 때까지 입력이 지정된 멤버에서 읽힙니다. 포함된 멤버에는 다른 사전컴파일러 INCLUDE문이 들어 있지 않아도 되지만 어플리케이션 프로그램과 SQL문 모두 포함시킬 수 있습니다.

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.

CRTSQLRPGI의 RPG 프리프로세서 옵션(RPGPPORT) 매개변수에는 RPG 프리프로세서를 호출하는 옵션이 두 개 있습니다.\*LVL1 또는 \*LVL2 가 지정된 경우, RPG 컴파일러는 SQL 사전컴파일러가 실행되기 전에 소스 멤버를 사전처리하기 위해 호출될 수 있습니다. SQL 소스 멤버를 사전처리하면 SQL 사전컴파일 전에 복수 컴파일러를 처리할 수 있습니다. 사전 처리된 소스는 QTEMP의 QSQLPRE 파일에 놓이게 됩니다. 이 소스는 SQL 사전컴파일러의 입력으로 사용됩니다. SQL 사전컴파일러에서 사용하는 CCSID(코드화 문자 세트 ID)는 QSQLPRE의 CCSID입니다.

관련 정보

SET OPTION

CRTSQLRPGI 명령

## SQL 사전컴파일러의 소스 파일 CCSID

SQL 사전컴파일러는 소스 파일의 CCSID를 사용하여 소스 레코드를 읽습니다.

SQL INCLUDE문 처리 시 포함 소스는 필요한 경우 소스 원본 파일의 CCSID로 변환됩니다. 포함 소스가 소스 원본 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 있을 것으로 예상한다는 의미입니다.

소스 파일 CCSID가 65535이면 SQL은 가변 문자가 CCSID 37을 가지는 것처럼 처리합니다. 이는 SQL이 '5F'X에서 not sign(¬)을 찾는다는 의미입니다.

## SQL 사전컴파일러에서 출력

이 섹션에서는 사전컴파일러가 제공하는 여러 종류의 출력에 대해 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.

- 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 사전컴파일러로 작성한 임시 소스 파일 멤버:

사전컴파일러가 처리한 소스문은 출력 소스 파일에 기록됩니다.

사전컴파일러 변경 소스 코드에서 SQL문은 주석으로 변환되며, SQL 런타임을 호출합니다. SQL에서 처리되는 포함이 확장됩니다.

출력 소스 파일이 TOSRCFILE 매개변수에서 CRTSQLxxx 명령에 지정됩니다. C 및 C++ 언어 이외의 경우 디폴트 파일은 QTEMP 라이브러리의 QSQLTEMP(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 QSQLTxxxxxx로 생성됩니다. 여기서 xxxxxx는 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 컴파일이 실패합니다.

사전컴파일러의 결과로 SQL에서 생성된 소스 멤버는 또 다른 사전컴파일 단계에 대한 입력 멤버로 편집되고 다시 작성되어서는 안 됩니다. 처음 사전컴파일 시 소스 멤버로 저장된 추가 SQL 정보로 인해 두 번째 사전컴파일러가 잘못 작동합니다. 일단 이 정보가 소스 멤버에 접속되면 멤버가 작성될 때까지 멤버로 남아 있습니다.

SQL 사전컴파일러는 CRTSRCPF 명령을 사용하여 출력 소스 파일을 작성합니다. 이 명령에 대한 디폴트가 변경될 경우 결과는 예측할 수 없습니다. SQL 사전컴파일러 대신 사용자가 소스 파일을 작성하는 경우에 역시 파일 속성이 달라질 수 있습니다. 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 V5R4M0 060210          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.....V5R4M0
INCLUDE file.....*SRCFILE
Commit.....*CHG
Allow copy of data.....*YES
Close SQL cursor.....*ENDPGM
Allow blocking.....*READ
Delay PREPARE.....*NO
Generation level.....10
Printer file.....*LIBL/QSYSVRT
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
SQL rules.....*DB2
User profile.....*NAMING
Dynamic User Profile.....*USER
Sort Sequence.....*JOB
Language ID.....*JOB
IBM SQL flagging.....*NOFLAG
ANS flagging.....*NONE
Text.....*SRCBRTXT
Source file CCSID.....65535
Job CCSID.....65535
Decimal result options:
  Maximum precision.....31
  Maximum scale.....31
  Minimum divide scale....0
Compiler options.....*NONE
(2) Source member changed on 06/06/00 10:16:44

```

- 1 SQL 사전컴파일러가 호출되었을 때 지정한 옵션 리스트
- 2 소스 멤버가 마지막으로 변경된 날짜

그림 2. 샘플 COBOL 사전컴파일러 출력 형식

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

- 1 사전검파일러가 소스 레코드를 읽을 때 할당하는 레코드 번호. 레코드 번호는 오류 메시지와 SQL 런타임(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.

```

5722ST1 V5R4M0 060210      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
49 *****
50 * Declare cursor *
51 *****
52 EXEC SQL
53 DECLARE CURS CURSOR FOR
54 SELECT WORKDEPT, AVG(EDLEVEL), AVG(SALARY)
55 FROM CORPDATA.EMPLOYEE
56 GROUP BY WORKDEPT
57 END-EXEC.
58 *****
59 * Open cursor *
60 *****
61 EXEC SQL
62 OPEN CURS
63 END-EXEC.
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.
***** END OF SOURCE *****

```

CROSS REFERENCE

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 (6)
EDLEVEL	55	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
MIDINIT	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  
 \* \* \* \* \* E N D O F L I S T I N G \* \* \* \* \*

- 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
 기호명이 유효한 호스트 변수를 참조하는 경우 data-type 6 또는 data-structure 7도 표시됩니다

## 비ILE SQL 사전컴파일러 명령

DB2 UDB 조회 관리자 및 SQL 개발 킷에는 CRTSQLCBL(iSeries용 COBOL의 경우), CRTSQLPLI(iSeries PL/I의 경우) 및 CRTSQLRPG(RPG/400의 일부인 RPG III의 경우) 호스트 언어를 위한 비ILE 사전컴파일러 명령이 포함됩니다.

일부 옵션은 특정 언어에만 적용됩니다. 예를 들어 옵션 \*APOST와 \*QUOTE는 COBOL에만 고유합니다. 그 옵션들은 다른 언어에 대한 명령에는 포함되지 않습니다.

### 관련 개념

189 페이지의 『호스트 언어 사전컴파일러용 iSeries용 DB2 UDB CL 명령 설명』

iSeries용 DB2 UDB는 다음 프로그래밍 언어에서 코딩된 사전컴파일러 프로그램에 대한 명령을 제공합니다.

## 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 CRTCBLPGM 명령에 전달됩니다.
- RPG 및 COBOL의 경우 SAAFLAG(\*FLAG)가 CRTxxxPGM 명령에 전달됩니다.
- RPG 및 COBOL의 경우 CRTSQLxxx 명령의 SRTSEQ 및 LANGID 매개변수가 CRTxxxPGM 명령에 지정됩니다.
- RPG 및 COBOL의 경우 CVTOPT(\*DATETIME \*VARCHAR)는 항상 CRTxxxPGM 명령에 지정됩니다.
- COBOL 및 RPG의 경우 CRTSQLxxx 명령의 TGTRLS 매개변수 값은 CRTxxxPGM 명령에 지정됩니다. TGTRLS는 CRTPLIPGM 명령에 지정되지 않습니다. 프로그램은 CRTSQLPLI 명령의 TGTRLS 매개변수에 지정된 레벨로 저장되거나 복원될 수 있습니다.
- PL/I의 경우 MARGINS는 임시 소스 파일에 설정됩니다.
- 모든 언어의 경우 CRTSQLxxx 명령의 REPLACE 매개변수는 CRTxxxPGM 명령에 지정됩니다.

패키지가 사전컴파일 프로세스의 일부로 작성될 경우 CRTSQLxxx 명령의 REPLACE 매개변수 값이 CRTSQLPKG 명령에 지정됩니다.

- 모든 언어의 경우 USRPRF(\*USER) 또는 USRPRF(\*NAMING)가 있는 시스템명(\*SYS)이 지정되면 USRPRF(\*USER)가 CRTxxxPGM 명령에 지정됩니다. USRPRF(\*OWNER) 또는 USRPRF(\*NAMING)가 있는 SQL명(\*SQL)이 지정되면 USRPRF(\*OWNER)가 CRTxxxPGM 명령에 지정됩니다.

CRTxxxPGM 명령의 모든 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 별도의 단계로 수행하면 CRTSQLPKG 명령을 사용하여 분산 프로그램을 위한 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, CRTSQLCPPI, CRTSQLCBLI 및 CRTSQLRPGI ILE 사전컴파일러 명령이 있습니다.

iSeries용 ILE C, iSeries용 ILE C++, iSeries용 ILE COBOL 및 ILE RPG 등 각 호스트 언어에는 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.

### 관련 개념

189 페이지의 『호스트 언어 사전컴파일러용 iSeries용 DB2 UDB CL 명령 설명』

iSeries용 DB2 UDB는 다음 프로그래밍 언어에서 코딩된 사전컴파일러 프로그램에 대한 명령을 제공합니다.

## SQL을 사용하는 ILE 어플리케이션 프로그램 컴파일링

SQL 사전컴파일러는 \*NOGEN이 지정되지 않을 경우 CRTSQLxxx 명령의 사전컴파일을 완료한 다음 호스트 언어 컴파일러를 자동으로 호출합니다.

\*MODULE 옵션이 지정된 경우 SQL 사전컴파일러는 CRTxxxMOD 명령을 발행하여 모듈을 작성합니다.  
\*PGM 옵션이 지정된 경우 SQL 사전컴파일러는 CRTBNDxxx 명령을 발행하여 프로그램을 작성합니다.  
\*SRVPGM 옵션이 지정된 경우 SQL 사전컴파일러는 CRTxxxMOD 명령을 발행하여 모듈을 작성하고, CRTSRVPGM(서비스 프로그램 작성) 명령을 발행하여 서비스 프로그램을 작성합니다. CRTSQLCPPI 명령만 \*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 명령에 지정될 경우 CRTxxxMOD 명령이나 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 프로세스의 일부로 작성될 경우 CRTSQLxxx 명령의 REPLACE 매개변수 값이 CRTSQLPKG 명령에 지정됩니다.

- OBJTYPE이 \*PGM 또는 \*SRVPGM이고 USRPRF(\*USER) 또는 USRPRF(\*NAMING)가 있는 시스템 명(\*SYS)이 지정되면 USRPRF(\*USER)가 CRTBNDxxx 또는 CRTSRVPGM 명령에 지정됩니다.

OBJTYPE이 \*PGM 또는 \*SRVPGM이고 USRPRF(\*OWNER) 또는 USRPRF(\*NAMING)이 있는 SQL명(\*SQL)이 지정되면 USRPRF(\*OWNER)가 CRTBNDxxx 또는 CRTSRVPGM 명령에 지정됩니다.

- C와 C++의 경우 MARGINS는 임시 소스 파일에 설정됩니다.

사전컴파일러에서 LOB 호스트 변수의 총 길이를 약 15M로 계산한 경우 TERASPACE(\*YES \*TSIFC) 옵션이 CRTCMOD, CRTBNDC 또는 CRTCPMOD 명령에 지정됩니다.

- COBOL의 경우 \*QUOTE 또는 \*APOST가 CRTBNDCBL 또는 CRTCBMOD 명령에 전달됩니다.
- RPG 및 COBOL의 경우 CRTSQLxxx 명령의 SRTSEQ 및 LANGID 매개변수가 CRTxxxMOD 및 CRTBNDxxx 명령에 지정됩니다.
- COBOL의 경우, CVTOPT(\*VARCHAR \*DATETIME \*PICGRAPHIC \*FLOAT)는 항상 CRTCBMOD 및 CRTBNDCBL 명령에 지정됩니다. OPTION(\*NOCVTDT)이 지정되면(제공된 명령 디폴트), 추가 옵션 \*DATE \*TIME \*TIMESTAMP도 CVTOPT에 지정됩니다.
- RPG의 경우 OPTION(\*CVTDT)이 지정되면 CVTOPT(\*DATETIME)가 CRTRPGMOD 및 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 프로그래밍을 멤버명으로 사용하면 사전컴파일러는 출력 소스 파일(TOSRCFILE 매개 변수)에 소스 멤버를 작성합니다. 이제 명시적으로 호스트 언어 컴파일러를 호출하고 출력 소스 파일에서 소스 멤버를 지정 및 디폴트를 변경할 수 있습니다. 사전컴파일과 컴파일을 별도의 단계로 수행하면 CRTSQLPKG 명령을 사용하여 분산 프로그램을 위한 SQL 패키지를 작성할 수 있습니다.

프로그램이나 서비스 프로그램이 나중에 작성되면 USRPRF 매개변수가 CRTBNDxxx, 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.

## 사전컴파일러 명령을 사용하여 컴파일러 옵션 설정

COMPILEOPT 스트링은 컴파일러 명령에 추가 매개변수를 사용할 수 있게 하는 SET OPTION문 및 사전컴파일러 명령에서 사용할 수 있습니다.

COMPILEOPT 스트링은 사전컴파일러에 의해 빌드된 컴파일러 명령에 추가됩니다. 이를 통해 사전컴파일 후 컴파일하는 두 단계의 프로세스 없이 컴파일러 매개변수를 지정할 수 있습니다. SQL 사전컴파일러가 전달하는 COMPILEOPT 스트링에 매개변수를 지정하지 마십시오. 그렇게 하면 중복 매개변수 오류가 표시되며 컴파일러 명령이 실패합니다. 나중에 SQL 사전컴파일러에서 컴파일러에 추가 매개변수를 전달할 수 있습니다. 이 때 COMPILEOPT 스트링을 변경해야 하는 중복 매개변수 오류가 발생할 수 있습니다.

"INCDIR(" 스트링이 COMPILEOPT 스트링에 포함되는 경우 사전컴파일러는 SRCSTMF 매개변수를 사용하여 컴파일러를 호출합니다.

```
EXEC SQL SET OPTION COMPILEOPT = 'OPTION(*SHOWINC *EXPMAC)
      INCDIR('/QSYS.LIB/MYLIB.LIB/MYFILE.MBR '');
```

## SQL을 사용하는 어플리케이션에서 컴파일 오류 해석

컴파일 오류가 발생할 경우 다음 정보를 사용하여 이러한 오류를 해석합니다.

**경고:** 사용자가 사전컴파일과 컴파일 단계를 분리하고 소스 프로그램이 외부적으로 설명된 파일을 참조한다면 참조된 파일이 사전컴파일과 컴파일 사이에서 변경되어서는 안됩니다. 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
- RPG에서의 외부 서술 파일 및 외부 서술 자료

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.

## SQL을 사용하는 어플리케이션 프로그램의 컴파일 시 오류 및 경고 메시지

다음은 컴파일 처리 시 오류 또는 경고 메시지를 발생시킬 수도 있는 조건에 대한 설명입니다.

### 관련 개념

15 페이지의 『C 및 C++ 어플리케이션에서 SQL문 코딩』

이 주제에서는 C 또는 C++ 프로그램에 SQL문을 삽입하기 위한 고유의 어플리케이션과 코딩 요구사항에 대해 설명합니다.

46 페이지의 『COBOL 어플리케이션에서 SQL문 코딩』

이 주제에서는 COBOL 프로그램에 SQL문을 삽입할 때의 고유의 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.

73 페이지의 『PL/I 어플리케이션에서 SQL문 코딩』

이 주제에서는 iSeries PL/I 프로그램에 SQL문을 삽입하는 경우 고유의 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.



89 페이지의 『RPG/400 어플리케이션에서 SQL문 코딩』

RPG/400 라이선스 프로그램은 RPG II 및 RPG III 프로그램을 모두 지원합니다.

101 페이지의 『ILE RPG 어플리케이션에서 SQL문 코딩』

이 주제에서는 ILE RPG 프로그램에 SQL문을 삽입할 때의 고유 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 변수에 대한 코딩 요구사항도 정의됩니다.

125 페이지의 『REXX 어플리케이션에서 SQL문 코딩』

REXX 프로시듀어는 사전처리할 필요가 없습니다. 실행 시에 REXX 인터프리터는 자신이 이해하지 못하는 명령문을 현재 사용 중인 명령 환경으로 전달하여 처리합니다.

### **PL/I, C 또는 C++ 컴파일 시 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 컴파일 시 오류 및 경고 메시지:**

EXEC SQL이 12열 앞에서 시작하면 SQL 사전컴파일러는 명령문을 SQL문으로 인식하지 않습니다. 결과적으로 그것은 컴파일러에 대한 것처럼 전달됩니다.

### **RPG 컴파일 시 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 정보는 언어 섹션에서 구체적인 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 작성 시 액세스 계획을 바인딩하고 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 수행 시 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 처리 시 SQL문은 유효한 표, 뷰 및 열명에 대해 체크됩니다. 지정된 표나 뷰가 사전컴파일이나 컴파일 시 존재하지 않으면 수행 시에 유효성 검사가 수행됩니다. 표나 뷰가 수행 시 존재하지 않으면 다음의 SQLCODE가 리턴됩니다.
2. 프로그램이 처리하려는 자료에 액세스하는 데 필요한 색인을 선택합니다. 색인, 표 크기 및 다른 인자 선택 시 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 변경하고 CRTSQLxxx 명령을 재발행해야 합니다.

### SQL을 사용하는 어플리케이션의 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 프로그램의 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 컴파일되면 DSPMOD(모듈 표시), DSPPGM(프로그램 표시) 또는 DSPSRVPGM(서비스 프로그램 표시) 명령으로 SQL 사전컴파일러에서 지정된 일부 옵션을 결정할 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 SQL이 있는 프로그램 실행

사전컴파일과 컴파일이 제대로 수행된 후 삽입된 SQL문으로 호스트 언어 프로그램을 실행하는 것은 호스트 프로그램의 실행과 동일합니다.

다음 CALL문을 실행하십시오.

```
CALL pgm-name
```

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.

관련 정보

CL 프로그래밍

## 삽입된 SQL이 있는 프로그램 실행: i5/OS™ DDM 고려사항

SQL은 분산 자료 관리(DDM) 파일을 통해 리모트 파일 액세스를 지원하지 않습니다. SQL은 DRDA®(분산 관계형 데이터베이스 구조™)를 통해 리모트 액세스를 지원합니다.

## 삽입된 SQL이 있는 프로그램 실행: 대체 고려사항

대체(OVRDBF 명령으로 지정)를 사용하여 다른 표나 뷰에 대한 참조를 지정하거나 프로그램 또는 SQL 패키지 지의 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) 매개변수 지정
2. SBMRMTCMD(리모트 명령 제출) 명령을 사용하여 명령을 어플리케이션 서버로 전송

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.

관련 정보

데이터베이스 프로그래밍

데이터베이스 파일 관리

### 삽입된 SQL이 있는 프로그램 실행: SQL 리턴 코드

각 SQL문이 완료되면 데이터베이스 관리자가 SQL 리턴 코드를 전송합니다.

관련 정보

SQL 메시지 및 코드

## iSeries용 DB2 UDB 명령문을 사용하는 샘플 프로그램

이 주제에는 iSeries용 DB2 UDB 시스템에서 지원되는 각 언어로 SQL문을 코딩하는 방법을 나타내는 샘플 어플리케이션이 들어 있습니다.

이 샘플 어플리케이션은 수당에 근거한 임금 인상을 보여줍니다.

각 샘플 프로그램은 동일한 보고서를 작성하며 이 보고서는 이 주제의 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 변수 나 호스트 구조로 사용될 수 있습니다.

주:

1. RPG/400에서는 6자가 넘는 외부 서술 구조의 필드명을 변경해야 합니다.
2. REXX는 외부 정의 검색을 지원하지 않습니다.

- 2 SQL INCLUDE SQLCA문은 PL/I, C 및 COBOL 프로그램에 SQLCA를 포함시키는데 사용됩니

다. 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 SALARY 열을 갱신하며, 이 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 EXECUTE문을 사용합니다.
- 5 이 SQL COMMIT문은 SQL UPDATE 문에 의해 수행된 변경 내용을 요약합니다. 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 내용에 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 기대하

지만, 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 갱신 중에 오류가 발생하는 경우에 표를 원래 상태로 복원시킵니다.

#### 관련 개념

15 페이지의 『C 및 C++ 어플리케이션에서 SQL문 코딩』

이 주제에서는 C 또는 C++ 프로그램에 SQL문을 삽입하기 위한 고유의 어플리케이션과 코딩 요구사항에 대해 설명합니다.

46 페이지의 『COBOL 어플리케이션에서 SQL문 코딩』

이 주제에서는 COBOL 프로그램에 SQL문을 삽입할 때의 고유한 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.

73 페이지의 『PL/I 어플리케이션에서 SQL문 코딩』

이 주제에서는 iSeries PL/I 프로그램에 SQL문을 삽입하는 경우 고유한 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 구조와 호스트 변수에 대한 요구사항도 정의합니다.

89 페이지의 『RPG/400 어플리케이션에서 SQL문 코딩』

RPG/400 라이선스 프로그램은 RPG II 및 RPG III 프로그램을 모두 지원합니다.

#### 예: ILE C 및 C++ 프로그램의 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: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

```

| 5722ST1 V5R4M0 060210          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.....v5r4m0
| INCLUDE file.....*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/QSYSVRT
| 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
| SQL rules.....*DB2
| 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
| Decimal result options:
|   Maximum precision.....31
|   Maximum scale.....31
|   Minimum divide scale....0
| Compiler options.....*NONE
| Source member changed on 06/06/00 17:15:17

```

그림 3. SQL문을 사용하는 샘플 C 프로그램

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	
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??(??);	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??(??);	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	



Record	*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8	SEQNBR	Last change
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 W	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	
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\\n	10400	
105	BY PROJECT");	10500	
106	fprintf(qprint, "\\n\\nPROJECT W	10600	
107	NUMBER OF TOTAL");	10700	
108	fprintf(qprint, "\\n\\nNUMBER PROJECT NAME W	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	

```

5722ST1 V5R4M0 060210          Create SQL ILE C Object          CEX          08/06/02 15:52:26 Page 4
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8  SEQNBR Last change
| 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,"Wn%6s %-36s %6d %9.2f",         13100
| 132              rpt2.projno,rpt2.project_name,rpt2.employee_count,
| 133              rpt2.total_proj_cost);                       13200
| 134          }                                               13300
| 135      while (SQLCODE==0);                                  13400
| 136                                     13500
| 137      done2:                                             13600
| 138      EXEC SQL                                           13700
| 139          CLOSE C2;                                       13800
| 140      goto finished;                                     13900
| 141                                     14000
| 142      /* Error occurred while updating table. Inform user and rollback */ 14100
| 143      /* changes. */                                     14200
| 144      update_error:                                       14300
| 145      13 EXEC SQL WHENEVER SQLERROR CONTINUE;            14400
| 146      fprintf(qprint,"*** ERROR Occurred while updating table. SQLCODE="
| 147          "%5dWn",SQLCODE);                               14500
| 148      14 EXEC SQL                                         14600
| 149          ROLLBACK;                                       14700
| 150      goto finished;                                     14800
| 151                                     14900
| 152      /* Error occurred while generating reports. Inform user and exit. */ 15000
| 153      report_error:                                       15100
| 154      fprintf(qprint,"*** ERROR Occurred while generating reports. "
| 155          "SQLCODE=%5dWn",SQLCODE);                       15200
| 156      goto finished;                                     15300
| 157                                     15400
| 158      /* All done */                                     15500
| 159      finished:                                         15600
| 160          fclose(qprint);                                 15700
| 161          exit(0);                                       15800
| 162                                     15900
| 163      }                                               16000
|                                     16100
|                                     16200
|                                     16300
| * * * * * E N D O F S O U R C E * * * * *

```

## 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 86
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 86
total_proj_cost	41	FLOAT(53) IN rpt2
update_error	****	LABEL 50
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

CROSS REFERENCE

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
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

\*\*\*\*\* END OF LISTING \*\*\*\*\*

## 예: COBOL 및 ILE COBOL 프로그램의 SQL문

이 샘플 프로그램은 COBOL 프로그래밍 언어로 작성되었습니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

```

| 5722ST1 V5R4M0 060210      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.....v5r4m0
| INCLUDE file.....*SRCFILE
| Commit.....*CHG
| Allow copy of data.....*YES
| Close SQL cursor.....*ENDPGM
| Allow blocking.....*READ
| Delay PREPARE.....*NO
| Generation level.....10
| Printer file.....*LIBL/QSYSVRT
| 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
| Created object type.....*PGM
| SQL rules.....*DB2
| 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
| Decimal result options:
|   Maximum precision.....31
|   Maximum scale.....31
|   Minimum divide scale....0
| Compiler options.....*NONE
| Source member changed on 07/01/96 09:44:58

```

그림 4. Sample COBOL program using SQL statements

```

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 IDENTIFICATION DIVISION.
17
18 PROGRAM-ID. CBLEX.
19 ENVIRONMENT DIVISION.
20 CONFIGURATION SECTION.
21 SOURCE-COMPUTER. IBM-AS400.
22 OBJECT-COMPUTER. IBM-AS400.
23 INPUT-OUTPUT SECTION.
24
25 FILE-CONTROL.
26 SELECT PRINTFILE ASSIGN TO PRINTER-QPRINT
27 ORGANIZATION IS SEQUENTIAL.
28
29 DATA DIVISION.
30
31 FILE SECTION.
32
33 FD PRINTFILE
34 BLOCK CONTAINS 1 RECORDS
35 LABEL RECORDS ARE OMITTED.
36 01 PRINT-RECORD PIC X(132).
37
38 WORKING-STORAGE SECTION.
39 77 WORK-DAYS PIC S9(4) BINARY VALUE 253.
40 77 RAISE-DATE PIC X(11) VALUE "1982-06-01".
41 77 PERCENTAGE PIC S999V99 PACKED-DECIMAL.
42 77 COMMISSION PIC S99999V99 PACKED-DECIMAL VALUE 2000.00.
43
44 *****
45 * Structure for report 1. *
46 *****
47
48 1 01 RPT1.
49 COPY DDS-PROJECT OF CORPDATA-PROJECT.
50 05 EMPNO PIC X(6).
51 05 NAME PIC X(30).
52 05 SALARY PIC S9(6)V99 PACKED-DECIMAL.
53
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 01 RPT1-DATA.
101 05 PROJNO PIC X(6).
102 05 FILLER PIC XXX VALUE SPACES.
103 05 EMPNO PIC X(6).
104 05 FILLER PIC X(4) VALUE SPACES.
105 05 NAME PIC X(30).
106 05 FILLER PIC X(3) VALUE SPACES.
107 05 SALARY PIC ZZZZ9.99.
108 05 FILLER PIC X(96) VALUE SPACES.
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 ZZZ9.
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

```

```

5722ST1 V5R4M0 060210      Create SQL COBOL Program      CBLEX      08/06/02 11:09:13      Page 4
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8      SEQNBR Last change
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 and 9 are located on Part 5 of this figure.



```

5722ST1 V5R4M0 060210      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
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

```

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 occurred 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 occurred 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.

```

\* \* \* \* \* E N D O F S O U R C E \* \* \* \* \*

```

5722ST1 V5R4M0 060210      Create SQL COBOL Program      CBLEX      08/06/02 11:09:13  Page  7
CROSS REFERENCE
Data Names      Define      Reference
ACTNO           168      SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
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.EMPPROJACT
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 EMPPROJACT
                166 169 171 216
EMPNO         ****      COLUMN IN EMPLOYEE
                169 216
EMPNO         168      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
EMPPROJACT    ****      TABLE
                166 169 210 215 216 218
EMPPROJACT    ****      TABLE IN CORPDATA
                168 213
EMPTIME       168      DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJACT
EMPTIME       ****      COLUMN
                212
EMSTDATE      168      DATE(10) COLUMN IN CORPDATA.EMPPROJACT
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
MIDINIT       134      CHARACTER(1) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
NAME          52      CHARACTER(30) IN RPT1
                251
NAME          105      CHARACTER(30) IN RPT1-DATA

```

```

5722ST1 V5R4M0 060210      Create SQL COBOL Program      CBLEX      08/06/02 11:09:13  Page  8
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
                * * * * * E N D O F L I S T I N G * * * * *

```

## 예: PL/I 프로그램의 SQL문

이 샘플 프로그램은 PL/I 프로그래밍 언어로 작성되었습니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.

```
| 5722ST1 V5R4M0 060210 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.....V5R4M0
| INCLUDE file.....*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
| SQL rules.....*DB2
| 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
| Decimal result options:
|   Maximum precision.....31
|   Maximum scale.....31
|   Minimum divide scale...0
| Compiler options.....*NONE
| Source member changed on 07/01/96 12:53:08
```

그림 5. SQL문을 사용하는 샘플 PL/I 프로그램

```

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
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

```

Record	*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8	SEQNBR	Last change
5722ST1 V5R4M0 060210	Create SQL PL/I Program	PLIEX	08/06/02 12:53:36 Page 3
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	
81	/* Fetch and write the rows to SYSPRINT */	8100	
82	8 EXEC SQL WHENEVER NOT FOUND GO TO DONE1;	8200	
83		8300	
84	DO UNTIL (SQLCODE ^= 0);	8400	
85	9 EXEC SQL	8500	
86	FETCH C1 INTO :RPT1.PROJNO, :rpt1.EMPNO, :RPT1.NAME,	8600	
87	:RPT1.SALARY;	8700	
88	PUT FILE(SYSPRINT)	8800	
89	EDIT(RPT1.PROJNO,RPT1.EMPNO,RPT1.NAME,RPT1.SALARY)	8900	
90	(SKIP,COL(1),A,COL(10),A,COL(20),A,COL(54),F(8,2));	9000	
91	END;	9100	
92		9200	
93	DONE1:	9300	
94	10 EXEC SQL	9400	
95	CLOSE C1;	9500	
96		9600	
97	/* For all projects ending at a date later than '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	PUT FILE(SYSPRINT) EDIT('ACCUMULATED STATISTICS BY PROJECT')	10400	
105	(SKIP(3),COL(22),A);	10500	
106	PUT FILE(SYSPRINT)	10600	
107	EDIT('PROJECT','NUMBER OF','TOTAL')	10700	
108	(SKIP(2),COL(1),A,COL(48),A,COL(63),A);	10800	
109	PUT FILE(SYSPRINT)	10900	
110	EDIT('NUMBER','PROJECT NAME','EMPLOYEES','COST')	11000	
111	(SKIP,COL(1),A,COL(10),A,COL(48),A,COL(63),A,SKIP);	11100	
112		11200	

```

5722ST1 V5R4M0 060210          Create SQL PL/I Program          PLIEX          08/06/02 12:53:36 Page 4
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8  SEQNBR Last change
113 11 EXEC SQL 11300
114     DECLARE C2 CURSOR FOR 11400
115         SELECT EMPPROJACT.PROJNO, PROJNAME, COUNT(*), 11500
116             SUM( (DAYS(EMENDATE) - DAYS(EMSTDATE)) * EMPTIME * 11600
117                 DECIMAL(( SALARY / :WORK_DAYS ),8,2) ) 11700
118         FROM CORPDATA/EMPPROJACT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE 11800
119         WHERE EMPPROJACT.PROJNO=PROJECT.PROJNO AND 11900
120             EMPPROJACT.EMPNO =EMPLOYEE.EMPNO AND 12000
121             PRENDATE > :RAISE_DATE 12100
122         GROUP BY EMPPROJACT.PROJNO, PROJNAME 12200
123         ORDER BY 1; 12300
124     EXEC SQL 12400
125         OPEN C2; 12500
126 12600
127     /* Fetch and write the rows to SYSPRINT */ 12700
128     EXEC SQL WHENEVER NOT FOUND GO TO DONE2; 12800
129 12900
130     DO UNTIL (SQLCODE ^= 0); 13000
131 12 EXEC SQL 13100
132     FETCH C2 INTO :RPT2; 13200
133     PUT FILE(SYSPRINT) 13300
134     EDIT(RPT2.PROJNO,RPT2.PROJECT_NAME,EMPLOYEE_COUNT, 13400
135         TOTL_PROJ_COST) 13500
136     (SKIP,COL(1),A,COL(10),A,COL(50),F(4),COL(62),F(8,2)); 13600
137     END; 13700
138 13800
139     DONE2: 13900
140     EXEC SQL 14000
141         CLOSE C2; 14100
142     GO TO FINISHED; 14200
143 14300
144     /* Error occurred while updating table. Inform user and rollback */ 14400
145     /* changes. */ 14500
146     UPDATE_ERROR: 14600
147 13 EXEC SQL WHENEVER SQLERROR CONTINUE; 14700
148     PUT FILE(SYSPRINT) EDIT('*** ERROR Occurred while updating table.'|| 14800
149         ' SQLCODE=',SQLCODE)(A,F(5)); 14900
150 14 EXEC SQL 15000
151     ROLLBACK; 15100
152     GO TO FINISHED; 15200
153 15300
154     /* Error occurred while generating reports. Inform user and exit. */ 15400
155     REPORT_ERROR: 15500
156     PUT FILE(SYSPRINT) EDIT('*** ERROR Occurred while generating '|| 15600
157         'reports. SQLCODE=',SQLCODE)(A,F(5)); 15700
158     GO TO FINISHED; 15800
159 15900
160     /* All done */ 16000
161     FINISHED: 16100
162     CLOSE FILE(SYSPRINT); 16200
163     RETURN; 16300
164 16400
165     END PLIEX; 16500

```

\*\*\*\*\* END OF SOURCE \*\*\*\*\*



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

```

5722ST1 V5R4M0 060210      Create SQL PL/I Program      PLIEX      08/06/02 12:53:36  Page 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 *****

```

## 예: RPG/400 프로그램의 SQL문

이 샘플 프로그램은 RPG 프로그래밍 언어로 작성되었습니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 동의합니다.

```

| 5722ST1 V5R4M0 060210      Create SQL RPG Program      RPGEX      08/06/02 12:55:22  Page 1
| Source type.....RPG
| Program name.....CORPDATA/RPGEX
| Source file.....CORPDATA/SRC
| Member.....RPGEX
| To source file.....QTEMP/QSQLTEMP
| Options.....*SRC      *XREF
| Target release.....V5R4M0
| INCLUDE file.....*SRCFILE
| Commit.....*CHG
| Allow copy of data.....*YES
| Close SQL cursor.....*ENDPGM
| Allow blocking.....*READ
| Delay PREPARE.....*NO
| Generation level.....10
| Printer file.....*LIBL/QSYSPT
| 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
| SQL rules.....*DB2
| 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
| Decimal result options:
|   Maximum precision.....31
|   Maximum scale.....31
|   Minimum divide scale....0
| Compiler options.....*NONE
| Source member changed on 07/01/96 17:06:17

```

그림 6. SQL문을 사용하는 샘플 RPG/400 프로그램

Record	1	2	3	4	5	6	7	8	SEQNBR	Last change
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	
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	MOVE '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 RPTERR							5800	
60	C/END-EXEC								5900	

```

5722ST1 V5R4M0 060210          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
|
| 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||', '||FIRSTNME, 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 DOUNE0 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
| 105 11 C/EXEC SQL 10400
| 106 C+ DECLARE C2 CURSOR FOR 10500
| 107 C+ SELECT EMPPROJECT.PROJNO, PROJNAME, COUNT(*), 10600
| 108 C+ SUM((DAYS(EMENDATE) - DAYS(EMSTDATE)) * EMPTIME * 10700
| 109 C+ DECIMAL((SALARY/:WRKDAY),8,2)) 10800
| 110 C+ FROM CORPDATA/EMPPROJECT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE 10900
| 111 C+ WHERE EMPPROJECT.PROJNO = PROJECT.PROJNO AND 11000
| 112 C+ EMPPROJECT.EMPNO = EMPLOYEE.EMPNO AND 11100
| 113 C+ PRENDATE > :RDATE 11200
| 114 C+ GROUP BY EMPPROJECT.PROJNO, PROJNAME 11300
| 115 C+ ORDER BY 1 11400
| 116 C/END-EXEC 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

```

```

5722ST1 V5R4M0 060210 Create SQL RPG Program      RPGEX      08/06/02 12:55:22 Page 4
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 occurred 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
146 C/END-EXEC                                         14500
147 C          RETRN                                    14600
148 C*                                                  14700
149 C* Error occurred 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'          15800
159 O          64 'TED BY EMPLOYEE RAISES'          15900
160 O          E 01          RECA                      16000
161 O          7 'PROJECT'                          16100
162 O          17 'EMPLOYEE'                        16200
163 O          32 'EMPLOYEE NAME'                  16300
164 O          60 'SALARY'                          16400
165 O          E 01          RECB                      16500
166 O          PROJNO      6                        16600
167 O          EMPNO      15                        16700
168 O          NAME      50                        16800
169 O          SALARYL     61                        16900
170 O          E 22          RECC                      17000
171 O          42 'ACCUMULATED STATISTIC'          17100
172 O          54 'S BY PROJECT'                   17200
173 O          E 01          RECC                      17300
174 O          7 'PROJECT'                          17400
175 O          56 'NUMBER OF'                      17500
176 O          67 'TOTAL'                          17600
177 O          E 02          RECC                      17700
178 O          6 'NUMBER'                          17800
179 O          21 'PROJECT NAME'                   17900
180 O          56 'EMPLOYEES'                      18000
181 O          66 'COST'                            18100
182 O          E 01          RECD                      18200
183 O          PRJNUM      6                        18300
184 O          PNAME      45                        18400
185 O          EMPCNTL    54                        18500
186 O          PRCOSTL    70                        18600
187 O          E 01          RECE                      18700
188 O          28 '*** ERROR Occurred while'        18800
189 O          52 ' updating table. SQLCODE'        18900
190 O          53 '='                                19000
191 O          SQLCODL     62                        19100
192 O          E 01          RECF                      19200
193 O          28 '*** ERROR Occurred while'        19300
194 O          52 ' generating reports. SQL'        19400
195 O          57 'CODE='                            19500
196 O          SQLCODL     67                        19600
          * * * * * E N D   O F   S O U R C E   * * * * *

```

## CROSS REFERENCE

## Data Names

## Define 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 68 105 105 105
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 68 105 105 105 105
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

```

5722ST1 V5R4M0 060210      Create SQL RPG Program      RPGEX      08/06/02 12:55:22  Page 6
PRENDATE      105      DATE(10) COLUMN IN CORPDATA.PROJECT
PRJNUM        24      CHARACTER(6) IN RPT2
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
***** END OF LISTING *****

```

## 예: ILE RPG 프로그램의 SQL문

이 샘플 프로그램은 ILE 프로그래밍 언어로 작성되었습니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 간주합니다.



```

5722ST1 V5R4M0 060210      Create SQL ILE RPG Object      RPGLEEX      08/06/02 16:03:02  Page  1
| Source type.....RPG
| Object name.....CORPDATA/RPGLEEX
| Source file.....CORPDATA/SRC
| Member.....*OBJ
| To source file.....QTEMP/QSQLTEMP1
| Options.....*XREF
| RPG preprocessor options..*NONE
| Listing option.....*PRINT
| Target release.....V5R4M0
| INCLUDE file.....*SRCFILE
| Commit.....*CHG
| Allow copy of data.....*YES
| Close SQL cursor.....*ENDMOD
| Allow blocking.....*READ
| Delay PREPARE.....*NO
| Generation level.....10
| Printer file.....*LIBL/QSYSPT
| 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
| SQL rules.....*DB2
| 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
| Decimal result options:
|   Maximum precision.....31
|   Maximum scale.....31
|   Minimum divide scale...0
| Compiler options.....*NONE
| Source member changed on 07/01/96 15:55:32

```

그림 7. SQL문을 사용하는 샘플 ILE RPG 프로그램

Record \*...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change Comments

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
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								3600
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								5600
57	C*	selected projects.								5700
58	C*									5800

12000

```

5722ST1 V5R4M0 060210      Create SQL ILE RPG Object      RPLGLEEX      08/06/02 16:03:02 Page 3
Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8 SEQNBR Last change Comments
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,                      6300
64      C+   LASTNAME||', '||FIRSTNME, SALARY                               6400
65      C+   FROM CORPDATA/EMPPROJACT, CORPDATA/EMPLOYEE                    6500
66      C+   WHERE EMPPROJACT.EMPNO = EMPLOYEE.EMPNO AND                    6600
67      C+   COMM >= :COMMI                                                  6700
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      9000
91      C* (i.e. those projects potentially affected by the salary raises)  9100
92      C* generate a report containing the project number, project name,    9200
93      C* the count of employees participating in the project and the      9300
94      C* total salary cost of the project.                                9400
95      C*                                                                    9500
96      C* Write out the header for report 2.                                9600
97      C*                                                                    9700
98      C          EXCEPT RECC                                            9800
99      C/EXEC SQL                                                            9900
100     11 C+  DECLARE C2 CURSOR FOR                                         10000
101     C+   SELECT EMPPROJACT.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/EMPPROJACT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE  10400
105     C+   WHERE EMPPROJACT.PROJNO = PROJECT.PROJNO AND                    10500
106     C+   EMPPROJACT.EMPNO = EMPLOYEE.EMPNO AND                          10600
107     C+   PRENDATE > :RDATE                                               10700
108     C+   GROUP BY EMPPROJACT.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                                                            12000
121     12 C+  FETCH C2 INTO :RPT2                                           12100
122     C/END-EXEC                                                            12200
123     C          EXCEPT RECD                                            12300

```

```

5722ST1 V5R4M0 060210 Create SQL ILE RPG Object      RPGLEEX      08/06/02 16:03:02 Page 4
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 occurred 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 occurred 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
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
***** END OF SOURCE *****

```

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

```

5722ST1 V5R4M0 060210 Create SQL ILE RPG Object      RPGLEEX          08/06/02 16:03:02 Page 6
PRENDATE          ****      COLUMN
                    99
PRENDATE          99      DATE(10) COLUMN IN CORPDATA.PROJECT
PRJNUM            18      CHARACTER(6) DBCS-open IN RPT2
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 *****

```

## 관련 개념

101 페이지의 『ILE RPG 어플리케이션에서 SQL문 코딩』

이 주제에서는 ILE RPG 프로그램에 SQL문을 삽입할 때의 고유 어플리케이션 및 코딩 요구사항에 대해 설명합니다. 호스트 변수에 대한 코딩 요구사항도 정의됩니다.

## 예: REXX 프로그램의 SQL문

이 샘플 프로그램은 REXX 프로그래밍 언어로 작성되었습니다.

주: 해당 코드 예제를 사용하는 것은 192 페이지의 『코드 라이선스 및 면책사항 정보』의 조건에 동의한 것으로 동의합니다.

```

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 3SIGNAL 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 4EXECSQL,
45 'EXECUTE S1 USING :PERCENTAGE,',
46 ':COMMISSION '
47 /* Commit changes */
48 5EXECSQL,
49 'COMMIT'
50 ERRLOC = 'REPORT_ERROR'
51

```

그림 8. SQL문을 사용하는 REXX 프로시저의 샘플

```

Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
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, EMPPROJECT.EMPNO, ',
66 ' LASTNAME||', '||FIRSTNAME, SALARY ',
67 'FROM CORPDATA/EMPPROJECT, CORPDATA/EMPLOYEE ',
68 'WHERE EMPPROJECT.EMPNO = EMPLOYEE.EMPNO AND ',
69 ' COMM >= ? ',
70 'ORDER BY PROJNO, EMPNO '
71 EXECSQL,
72 'PREPARE S2 FROM :SELECT_STMT'
73 6EXECSQL,
74 'DECLARE C1 CURSOR FOR S2'
75 7EXECSQL,
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 9EXECSQL,
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 8IF 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 10EXECSQL,
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

```



```

Record *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
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 EMPPROJECT.PROJNO, PROJNAME, COUNT(*),           ',
126                  '      SUM( (DAYS(EMENDATE) - DAYS(EMSTDATE)) * EMPTIME *           ',
127                  '      DECIMAL(( SALARY / ? ),8,2) )                               ',
128                  'FROM CORPDATA/EMPPROJECT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE',
129                  'WHERE EMPPROJECT.PROJNO = PROJECT.PROJNO AND                       ',
130                  '      EMPPROJECT.EMPNO = EMPLOYEE.EMPNO AND                       ',
131                  '      PRENDATE > ?                                                         ',
132                  'GROUP BY EMPPROJECT.PROJNO, PROJNAME                               ',
133                  'ORDER BY 1                                                         '
134      EXECSQL,
135          'PREPARE S3 FROM :SELECT_STMT'
136      11EXECSQL,
137          'DECLARE C2 CURSOR FOR S3'
138      EXECSQL,
139          'OPEN C2 USING :WORK_DAYS, :RAISE_DATE'
140
141      /* Handle the FETCH errors and warnings inline */
142      SIGNAL OFF ERROR
143
144      /* Fetch all of the rows */
145      DO UNTIL (SQLCODE <> 0)
146          12EXECSQL,
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 13SIGNAL 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 14EXECSQL,
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
***** END OF SOURCE *****

```

## 관련 개념

125 페이지의 『REXX 어플리케이션에서 SQL문 코딩』

REXX 프로시더는 사전처리할 필요가 없습니다. 실행 시에 REXX 인터프리터는 자신이 이해하지 못하는 명령문을 현재 사용 중인 명령 환경으로 전달하여 처리합니다.

## SQL을 사용하는 샘플 프로그램에 의해 작성된 보고서

이 보고서는 각 샘플 프로그램에 의해 작성됩니다.

### 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	LUCCHESI, 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

## 호스트 언어 사전컴파일러용 iSeries용 DB2 UDB CL 명령 설명

iSeries용 DB2 UDB는 다음 프로그래밍 언어에서 코딩된 사전컴파일러 프로그램에 대한 명령을 제공합니다.

### 관련 개념

143 페이지의 『비ILE SQL 사전컴파일러 명령』

DB2 UDB 조회 관리자 및 SQL 개발 킷에는 CRTSQLCBL(iSeries용 COBOL의 경우), CRTSQLPLI(iSeries PL/I의 경우) 및 CRTSQLRPG(RPG/400의 일부인 RPG III의 경우) 호스트 언어를 위한 비ILE 사전컴파일러 명령이 포함됩니다.

### 관련 참조

144 페이지의 『ILE SQL 사전컴파일러 명령』

DB2 UDB 조회 관리자 및 SQL 개발 킷에는 CRTSQLCI, CRTSQLCPPI, CRTSQLCBLI 및 CRTSQLRPGI ILE 사전컴파일러 명령이 있습니다.

## **CRTSQLCBL(구조화 조회 언어 COBOL 작성) 명령**

CRTSQLCBL(구조화 조회 언어 COBOL 작성) 명령은 구조화 조회 언어(SQL) 사전컴파일러를 호출합니다.

SQL문을 포함하는 COBOL 소스를 사전컴파일하고 임시 소스 멤버를 만든 후 선택적으로 COBOL 컴파일러를 호출하여 프로그램을 컴파일합니다.

관련 정보

CRTSQLCBL(구조화 조회 언어 COBOL 작성) 명령

## **CRTSQLCBLI(SQL ILE COBOL 오브젝트 작성) 명령**

CRTSQLCBLI(구조화 조회 언어 ILE COBOL 오브젝트 작성) 명령은 SQL문이 들어 있는 COBOL 소스문을 사전컴파일하는 구조화 조회 언어(SQL) 사전컴파일러를 호출하고, 임시 소스 멤버를 작성하며, 모듈, 프로그램 또는 서비스 프로그램을 작성하기 위하여 ILE COBOL 컴파일러를 선택적으로 호출합니다.

관련 정보

CRTSQLCBLI(구조화 조회 언어 ILE COBOL 오브젝트 작성) 명령

## **CRTSQLCI(구조화 조회 언어 ILE C 오브젝트 작성) 명령**

CRTSQLCI(구조화 조회 언어 ILE C 오브젝트 작성) 명령은 SQL문을 포함하는 C 소스를 사전컴파일하는 구조화 조회 언어(SQL) 사전컴파일러를 호출하고, 임시 소스 멤버를 작성하며, 선택적으로 ILE C 컴파일러를 호출하여 모듈, 프로그램 또는 서비스 프로그램을 작성합니다.

관련 정보

CRTSQLCI(구조화 조회 언어 ILE C 오브젝트 작성) 명령

## **CRTSQLCPPI(구조화 조회 언어 C++ 오브젝트 작성) 명령**

CRTSQLCPPI(구조화 조회 언어 C++ 오브젝트 작성) 명령은 SQL문을 포함하는 C++ 소스를 사전컴파일하는 구조화 조회 언어(SQL) 사전 컴파일러를 호출하여 임시 소스 멤버를 작성한 후 선택적으로 C++ 컴파일러를 호출하여 모듈을 작성합니다.

관련 정보

CRTSQLCPPI(구조화 조회 언어 C++ 오브젝트 작성) 명령

## **CRTSQLPLI(구조화 조회 언어 PL/I 작성) 명령**

CRTSQLPLI(구조화 조회 언어 PL/I 작성) 명령은 SQL문이 들어 있는 PL/I 소스를 사전컴파일하는 SQL(구조화 조회 언어) 사전컴파일러를 호출하여 임시 소스 멤버를 작성한 후 선택적으로 PL/I 컴파일러를 호출하여 프로그램을 컴파일합니다.

관련 정보

CRTSQLPLI(구조화 조회 언어 PL/I 작성) 명령

## CRTSQLRPG(구조화 조회 언어 RPG 작성) 명령

CRTSQLRPG(구조화 조회 언어 RPG 작성) 명령은 SQL문이 들어 있는 RPG 소스를 사전컴파일하는 SQL(구조화 조회 언어) 사전컴파일러를 호출하여 임시 소스 멤버를 작성한 후 선택적으로 RPG 컴파일러를 호출하여 프로그램을 컴파일합니다.

관련 정보

CRTSQLRPG(구조화 조회 언어 RPG 작성) 명령

## CRTSQLRPGI(SQL ILE RPG 오브젝트 작성) 명령

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.

관련 정보











CRTSQLRPGI(구조화 조회 언어 ILE RPG 오브젝트 작성) 명령

---

## 삽입된 SQL 프로그래밍 관련 정보

다음은 삽입된 SQL 프로그래밍 주제와 관련된 제품 매뉴얼 및 Information Center 주제 리스트입니다. PDF 로 보거나 인쇄할 수 있습니다.

### 매뉴얼

- V5R1 보충 매뉴얼  웹 사이트에 있는 *COBOL/400* 사용자 안내서 매뉴얼
- V5R1 보충 매뉴얼  웹 사이트에 있는 *COBOL/400* 참조 매뉴얼
- V5R1 보충 매뉴얼  웹 사이트에 있는 *RPG/400* 사용자 안내서 매뉴얼
- V5R1 보충 매뉴얼  웹 사이트에 있는 *RPG/400* 참조 매뉴얼
- V5R1 보충 매뉴얼  웹 사이트에 있는 *ILE RPG* 프로그래머 안내서 매뉴얼
- V5R1 보충 매뉴얼  웹 사이트에 있는 *ILE RPG* 참조 매뉴얼
- V5R1 보충 매뉴얼  웹 사이트에 있는 *ILE COBOL* 프로그래머 안내서 매뉴얼
- V5R1 보충 매뉴얼  웹 사이트에 있는 *ILE COBOL* 참조 매뉴얼
- V5R1 보충 매뉴얼  웹 사이트에 있는 *REXX/400* 프로그래머 안내서 매뉴얼
- V5R1 보충 매뉴얼  웹 사이트에 있는 *REXX/400* 참조 매뉴얼
- SQL 참조 PDF(13 343KB)

## 기타 정보

이러한 관련 주제를 보거나 다운로드할 수 있습니다.


- SQL 프로그래밍
- 데이터베이스 성능 및 조회 최적화
- SQL 호출 레벨 인터페이스

## PDF 파일 저장

보거나 인쇄하기 위해 워크스테이션에 PDF를 저장하려면 다음을 참조하십시오.

1. 브라우저에서 PDF를 마우스 오른쪽으로 클릭하십시오(위의 링크를 마우스 오른쪽으로 클릭).
2. PDF를 로컬로 저장하는 옵션을 클릭하십시오.
3. PDF를 저장할 디렉토리를 탐색하십시오.
4. 저장을 클릭하십시오.

## Adobe Reader 다운로드

- 1 이러한 PDF를 보거나 인쇄하려면 시스템에 Adobe Reader를 설치해야 합니다. Adobe 웹 사이트
- 1 ([www.adobe.com/products/acrobat/readstep.html](http://www.adobe.com/products/acrobat/readstep.html))  에서 사본을 무료로 다운로드할 수 있습니다.

---

## 코드 라이선스 및 면책사항 정보

IBM은 사용자의 특정 요구에 맞게 유사한 기능을 생성할 수 있도록 모든 프로그래밍 코드 예제를 사용할 수 있는 비독점적인 저작권 라이선스를 부여합니다.

- 1 강행 법규에 규정된 보증 조항의 적용을 제외하고, IBM은 해당 프로그램 또는 기술 지원에 대한 상품성, 특정
- 1 목적에의 적합성 및 타인의 권리 비침해에 대한 묵시적 보증을 포함한다(단, 이에 한하지 않음) 일체의 묵시적
- 1 또는 명시적인 보증이나 주장도 제공하지 않습니다.
  
- 1 IBM, IBM 프로그램 개발자 또는 공급자는, 손해 발생의 가능성을 통지 받은 경우를 포함한 어떠한 경우에도
- 1 다음에 대하여 책임 지지 않습니다.
- 1 1. 데이터의 손실 또는 손상
- 1 2. 직접적인, 특별한, 우연에 의한 또는 간접적인 손상 또는 이에 따른 경제적 손실 또는
- 1 3. 기대했던 이익, 사업, 수익, 영업권 또는 비용 절감이 실현되지 못함으로 인해 발생하는 손해
  
- 1 일부 관할권에서는 부수적 또는 결과적 손해의 제외사항이나 제한사항을 허용하지 않으므로, 상기 제외사항이
- 1 나 제한사항이 귀하에게 적용되지 않을 수도 있습니다.

---

## 부록. 주의사항

이 정보는 미국에서 제공되는 제품과 서비스용으로 작성된 것입니다.

IBM은 다른 국가에서 이 책에 기술된 제품, 서비스 또는 기능을 제공하지 않을 수도 있습니다. 현재 사용할 수 있는 제품 및 서비스에 대한 정보는 한국 IBM 담당자에게 문의하십시오. 이 책에서 IBM 제품, 프로그램 또는 서비스를 언급했다고 해서 해당 IBM 제품, 프로그램 또는 서비스만을 사용할 수 있다는 것을 의미하지는 않습니다. IBM의 지적 재산권을 침해하지 않는 한, 기능상으로 동등한 제품, 프로그램 또는 서비스를 대신 사용할 수도 있습니다. 그러나 비IBM 제품, 프로그램 또는 서비스의 운영에 대한 평가 및 검증은 사용자의 책임입니다.

IBM은 이 책에서 다루고 있는 특정 내용에 대해 특허를 보유하고 있거나 현재 특허 출원 중일 수 있습니다. 이 책을 제공한다고 해서 특허에 대한 라이선스까지 부여하는 것은 아닙니다. 라이선스에 대한 의문사항은 다음으로 문의하십시오.

135-270

서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩

한국 아이.비.엠 주식회사

고객만족센터

전화번호: 080-023-8080

2바이트(DBCS) 정보에 관한 라이선스 문의는 한국 IBM 고객만족센터에 문의하거나 다음 주소로 서면 문의하시기 바랍니다.

IBM World Trade Asia Corporation

Licensing

2-31 Roppongi 3-chome, Minato-ku

Tokyo 106, Japan

다음 단락은 현지법과 상충하는 영국이나 기타 국가에서는 적용되지 않습니다. IBM은 타인의 권리 비침해, 상품성 및 특정 목적에의 적합성에 대한 묵시적 보증을 포함하여(단, 이에 한하지 않음) 묵시적이든 명시적이든 일체의 보증없이 이 책을 『현상태대로』 제공합니다. 일부 국가에서는 특정 거래에서 명시적 또는 묵시적 보증의 면책사항을 허용하지 않으므로, 이 사항이 적용되지 않을 수도 있습니다.

이 정보에는 기술적으로 부정확한 내용이나 인쇄상의 오류가 있을 수 있습니다. 이 정보는 주기적으로 변경되며, 이 변경사항은 최신판에 통합됩니다. IBM은 이 책에서 설명한 제품 및/또는 프로그램을 사전 통지없이 언제든지 개선 및/또는 변경할 수 있습니다.

이 정보에서 비IBM의 웹 사이트는 단지 편의상 제공된 것으로, 어떤 방식으로든 이들 웹 사이트를 옹호하고자 하는 것은 아닙니다. 해당 웹 사이트의 자료는 본 IBM 제품 자료의 일부가 아니므로 해당 웹 사이트 사용으로 인한 위험은 사용자 본인이 감수해야 합니다.

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

(1) 독립적으로 작성된 프로그램과 기타 프로그램(본 프로그램 포함) 간의 정보 교환 및 (2) 교환된 정보의 상호 이용을 목적으로 정보를 원하는 프로그램 라이선스 사용자는 다음 주소로 문의하십시오.

135-270

서울특별시 강남구 도곡동 467-12, 군인공제회관빌딩  
한국 아이.비.엠 주식회사  
고객만족센터

이러한 정보는 해당 조항 및 조건에 따라(예를 들면, 사용료 지불 포함) 사용할 수 있습니다.

이 책에 기술된 라이선스가 있는 프로그램 및 사용 가능한 모든 라이선스가 있는 자료는 IBM이 IBM 기본 계약, IBM 프로그램 라이선스 계약(IPLA), 시스템 코드용 IBM 라이선스 계약 또는 이와 동등한 계약에 따라 제공한 것입니다.

본 문서에 포함된 모든 성능 데이터는 제한된 환경에서 산출된 것입니다. 따라서 다른 운영 환경에서 얻어진 결과는 상당히 다를 수 있습니다. 일부 성능은 개발 레벨 상태의 시스템에서 측정되었을 수 있으므로 이러한 측정치가 일반적으로 사용되고 있는 시스템에서도 동일하게 나타날 것이라고는 보증할 수 없습니다. 또한, 일부 성능은 추정을 통해 추측되었을 수도 있으므로 실제 결과는 다를 수 있습니다. 이 문서의 사용자는 해당 데이터를 사용자의 특정 환경에서 검증해야 합니다.

비IBM 제품에 관한 정보는 해당 제품의 공급업체, 공개 자료 또는 기타 범용 소스로부터 얻은 것입니다. IBM에서는 이러한 제품들을 테스트하지 않았으므로, 비IBM 제품과 관련된 성능의 정확성, 호환성 또는 기타 청구에 대해서는 확신할 수 없습니다. 비IBM 제품의 성능에 대한 의문사항은 해당 제품의 공급업체에 문의하십시오.

IBM의 향후 방향 또는 의도에 관한 모든 언급은 별도의 통지없이 변경될 수 있습니다.

이 정보에는 일상의 비즈니스 운영에서 사용되는 자료 및 보고서에 대한 예제가 들어 있습니다. 이들 예제에는 개념을 가능한 완벽하게 설명하기 위해 개인, 회사, 상표 및 제품의 이름이 사용될 수 있습니다. 이들 이름은 모두 가공의 것이며 실제 기업의 이름 및 주소와 유사하더라도 이는 전적으로 우연입니다.

저작권 라이선스:

이 정보에는 여러 운영 플랫폼에서의 프로그래밍 기법을 보여주는 원시 언어로 된 샘플 응용프로그램이 들어 있습니다. 귀하는 이러한 샘플 프로그램의 작성 기준이 된 운영 플랫폼의 응용프로그램 프로그래밍 인터페이스(API)에 부합하는 응용프로그램을 개발, 사용, 판매 또는 배포할 목적으로 추가 비용없이 이들 샘플 프로그램을 어떠한 형태로든 복사, 수정 및 배포할 수 있습니다. 이러한 샘플 프로그램은 모든 조건하에서 완전히 테스트된 것은 아닙니다. 그러므로 IBM은 이 프로그램들의 신뢰성, 서비스 및 기능을 보장할 수 없습니다.

이러한 샘플 프로그램 또는 파생 제품의 각 사본이나 그 일부에는 반드시 다음과 같은 저작권 표시가 포함되어야 합니다.



© (귀하의 회사명) (연도). 이 코드의 일부는 IBM Corp. 샘플 프로그램에서 파생됩니다. © Copyright IBM Corp. Copyright IBM Corp. \_연도\_. All rights reserved.All rights reserved.

이 정보를 소프트웨어로 보는 경우에는 사진과 컬러 삽화가 제대로 나타나지 않을 수도 있습니다.

---

## 프로그래밍 인터페이스 정보

삽입된 SQL 프로그래밍 서적은 사용자가 IBM i5/OS의 서비스를 확보하기 위해 프로그램을 작성할 수 있도록 프로그래밍 인터페이스를 제공합니다.

---

## 상표

다음 용어는 미국 또는 기타 국가에서 사용되는 IBM Corporation의 상표입니다.

- | COBOL/400
- | DB2
- | Distributed Relational Database Architecture
- | DRDA
- | e(로고)server
- | eServer
- | i5/OS
- | IBM
- | IBM(로고)
- | 통합 언어 환경
- | iSeries
- | RPG/400

기타 회사, 제품 및 서비스 이름은 해당 회사의 상표 또는 서비스표입니다.

---

## 조건

다음 조건에 따라 본 발행물을 사용할 수 있습니다.

**개인적 사용:** 귀하는 모든 소유권 사항을 표시하는 것을 조건으로 본 발행물을 개인적, 비상업적 용도로 복제할 수 있습니다. 귀하는 IBM의 명시적 동의없이 본 발행물 또는 그 일부를 배포 또는 게시하거나 이에 대한 2차적 저작물을 만들 수 없습니다.

**상업적 사용:** 귀하는 모든 소유권 사항을 표시하는 것을 조건으로 본 발행물을 귀하 사업장 내에서만 복제, 배포 및 게시할 수 있습니다. 귀하의 사업장 외에서는 IBM의 명시적 동의없이 본 발행물의 2차적 저작물을 만들거나 본 발행물 또는 그 일부를 복제, 배포 또는 게시할 수 없습니다.

본 허가에서 명시적으로 부여된 경우를 제외하고, 본 발행물이나 본 발행물에 포함된 정보, 데이터, 소프트웨어 또는 기타 지적 재산권에 대해서는 어떠한 허가나 라이선스 또는 권리도 명시적 또는 묵시적으로 부여되지 않습니다.

IBM은 본 발행물의 사용이 IBM의 이익을 해친다고 판단하거나 위에서 언급된 지시사항이 준수되지 않는다고 판단하는 경우 언제든지 부여한 허가를 철회할 수 있습니다.

귀하는 미국 수출법 및 관련 규정을 포함하여 모든 적용 가능한 법률 및 규정을 철저히 준수하는 것을 조건으로 본 정보를 다운로드, 송신 또는 재송신할 수 있습니다.

IBM은 본 발행물의 내용에 대해 어떠한 보증도 하지 않습니다. IBM은 상품성 및 특정 목적에의 적합성에 대한 보증을 포함하여 명시적이든 묵시적이든 일체의 보증없이 "현상태대로" 본 발행물을 제공합니다.



**IBM**