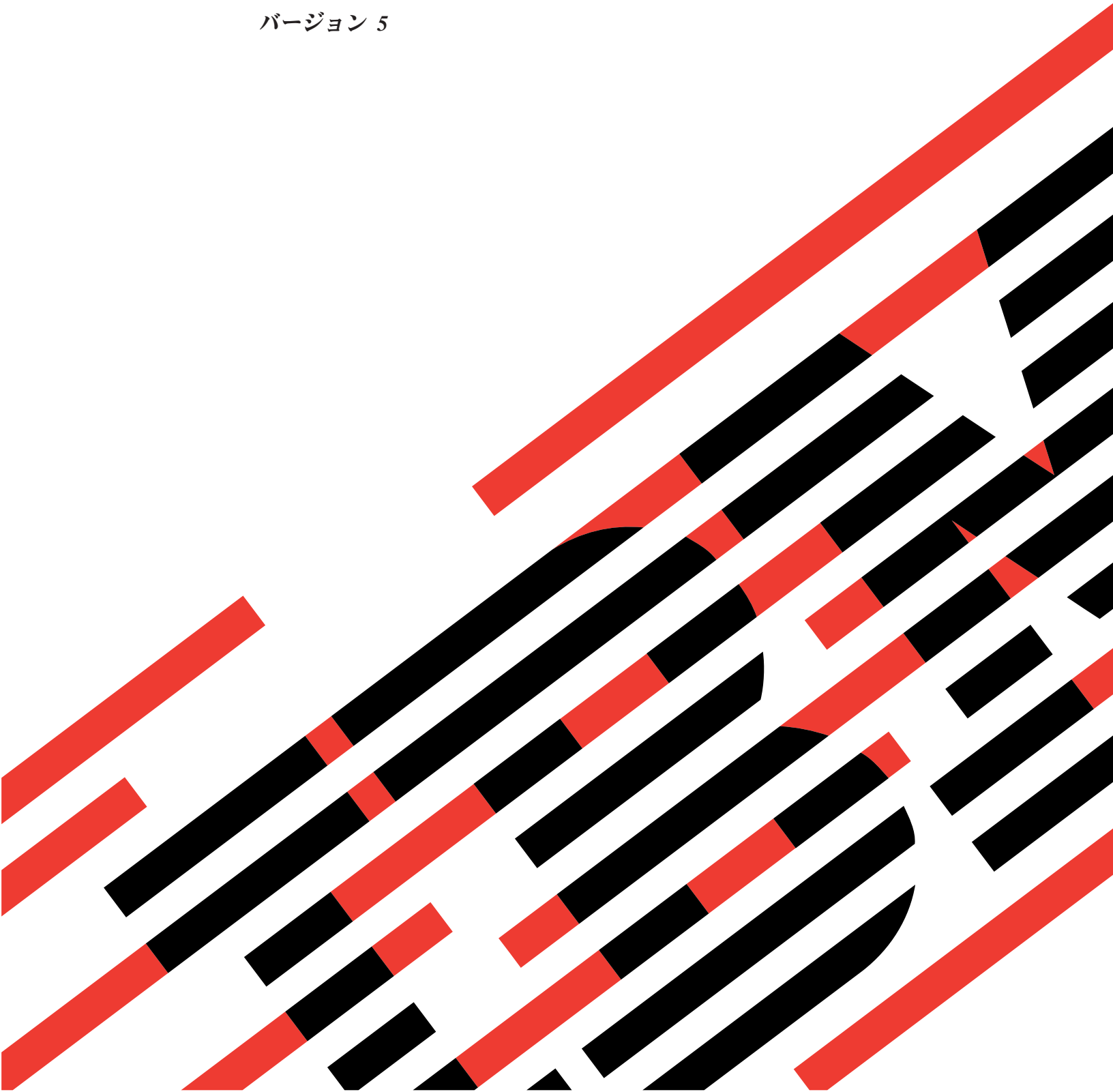
IBM

eserver

iSeries

DB2 Universal Database for iSeries ホスト言語での SQL プログラミング

バージョン 5



IBM

@server

iSeries

DB2 Universal Database for iSeries ホスト言語での SQL プログラミング

バージョン 5

目次

DB2 UDB for iSeries ホスト言語での SQL プログラミングの概要

本書は、プログラマーおよびデータベース管理者向けに、 DB2 UDB for iSeries SQL ステートメントと機能を使用したホスト言語でのデータベース・アプリケーション作成について説明したものです。

DB2 UDB for iSeries アプリケーション・プログラミング環境で SQL インプリメンテーションを行うための指針と例に関する詳細については、Information Center の**データベース**のカテゴリー内の以下の資料を参照してください。

- SQL 解説書
- SQL プログラミング概念
- データベース・パフォーマンスおよび Query 最適化
- SQL 呼び出しレベル・インターフェース (ODBC)

本書の詳細については、以下のセクションを参照してください。

- 『ホスト言語での SQL プログラミングの対象読者』
- 『ホスト言語での SQL プログラミングにおける SQL ステートメントの例に関する前提事項』
- ix ページの『ホスト言語での SQL プログラミングにおける構文の解釈』
- x ページの『ホスト言語での SQL プログラミング、バージョン 5 リリース 2 の新機能』

ホスト言語での SQL プログラミングの対象読者

本書が対象とする読者層は、COBOL for iSeries、ILE COBOL for iSeries、iSeries PL/I、ILE C for iSeries、ILE C++、REXX、RPG III (RPG for iSeries の一部)、または ILE RPG for iSeries 言語について精通していて、これらの言語を用いてプログラミングを行うことができ、基本的なデータベース・アプリケーションについて理解することができるアプリケーション・プログラマーおよびデータベース管理者の方々です。

ホスト言語での SQL プログラミングにおける SQL ステートメントの例に関する前提事項

本書で示されている SQL ステートメントの例は、iSeries Information Center の「SQL プログラミング 概念」の付録 A『DB2 UDB for iSeries 版 サンプル・テーブル』に基づいており、以下の事項を前提としています。

- これらの例は、対話式 SQL 環境で使用されるか、あるいは ILE C または COBOL で作成されています。 COBOL プログラム内での SQL ステートメントの区切りには、EXEC SQL および END-EXEC が使用されています。 COBOL プログラムの中で SQL ステートメントを使用する方法は、 47 ページの『第 3 章 COBOL アプリケーションでの SQL ステートメントのコーディング方法』で

説明されています。ILE C プログラムの中で SQL ステートメントを使用する方法は、 13 ページの『第 2 章 C および C++ アプリケーションでの SQL ステートメントのコーディング方法』で説明されています。

- 各 SQL ステートメントの例は、ステートメントの文節ごとに行を変えて、数行にまたがって示されています。
- SQL のキーワードは太字で示されています。
- サンプル・テーブルに記載されているテーブル名は、コレクション CORPDATA を使用します。これらのサンプル・テーブルにないテーブル名は、ユーザーが作成するコレクションを使用しなければなりません。これらのテーブルの定義および作成の方法ついては、「SQL プログラミング 概念」の付録 A『DB2 UDB for iSeries サンプル・テーブル』を参照してください。
- 計算対象の列は、括弧 () と大括弧 [] で囲まれています。
- SQL の命名規則が使用されています。
- APOST および APOSTSQL プリコンパイラー・オプションは、COBOL での省略時オプションではありませんが想定されています。 SQL およびホスト言語ステートメント内の文字ストリング・リテラルは、アポストロフィ (') によって区切られています。
- 特に断りがない限り、*HEX の分類順序が使用されています。
- どの例においても、通常、SQL ステートメントの構文全体は示されていません。本書に記載されているステートメントの完全な説明と構文については、「SQL 解説書」を参照してください。

上記の前提と異なる例が提示されている場合は、必ずその旨が記述されています。

本書はアプリケーション・プログラマーを対象としているため、ほとんどの例はアプリケーション・プログラムの中で書かれているものとして示されています。ただし、若干の変更を加えれば、対話式 SQL を使用して対話式で実行することができる例も多数あります。対話式 SQL を使用する場合の SQL ステートメントの構文は、同じステートメントをプログラムに組み込む場合の形式と若干異なります。

プログラム例の使用についての情報は、『コードについての特記事項』を参照してください。

コードについての特記事項

本書には、プログラムの例が含まれています。

IBM® は、お客様に、このプログラムをサンプルとして使用することができる非独占的な使用権を許諾します。お客様は、このサンプル・コードから、お客様独自の特別のニーズに合わせた類似のプログラムを作成することができます。

このサンプル・コードは、例として示す目的でのみ、IBM により提供されます。このサンプル・プログラムは、あらゆる条件下における完全なテストを経ていません。従って IBM は、これらのサンプル・プログラムについて信頼性、利便性もしくは機能性があることをほのめかしたり、保証することはできません。

ここに含まれるすべてのプログラムは、現存するままの状態で提供され、いかなる保証条件も適用されません。不侵害、商品性、特定目的適合性に関する黙示の保証の適用も一切ありません。

ホスト言語での SQL プログラミングにおける構文の解釈

本書では、以下のように定義された構造を使用して構文を記述します。

- 構文図は矢印に従って左から右へ、上から下へと見ます。

  ▸▸── 記号はステートメントの始まりを示します。

  ──▸ 記号は、ステートメントの構文が次の行へ続くことを示します。

  ▸── 記号は、ステートメントが前の行から続いていることを示します。

  ──▸◂ 記号はステートメントの終わりを示します。

  完結したステートメント以外の構文単位の図は、 ▸── 記号で始まり、 ──▸ 記号で終わります。

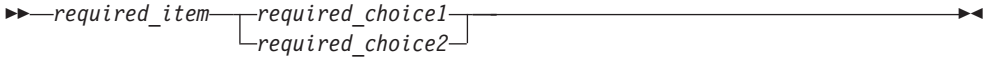
- required_item (必須項目) は水平線 (主線) 上に示されています。

- optional_item (任意選択項目) は主線より下に示されています。

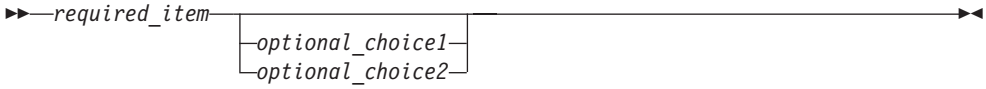
  optional_item が主線より上に示されている場合、その項目はステートメントの実行に対しては何の効果も持たず、読みやすさのためだけに使用されています。

- 2 つ以上の項目から選択できる場合には、それらは上下に重ねて (スタックされて) 示されています。

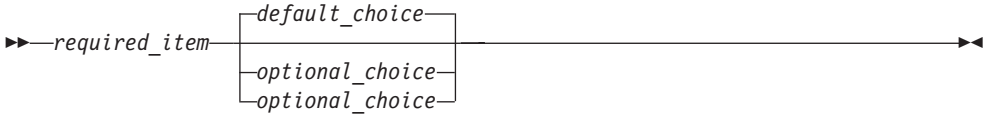
  項目の 1 つを選択しなくてはならない 場合は、スタックの内の 1 つの項目が主線上に示されます。



  項目の 1 つを選択することが任意の場合は、スタック全体が主線より下に示されています。



  項目の 1 つが省略時値である場合には、それが主線より上に示され、残りの選択項目は主線より下に示されています。



- 主線より上の左に戻る矢印は、反復可能な項目であることを示します。

反復矢印にコンマが含まれている場合は、反復される各項目をコンマで区切らなくてはなりません。

スタックより上にある反復矢印は、スタック内の項目を反復できることを示します。

- キーワードは英大文字で示されます (たとえば、FROM)。それらは示されているとおり正確に入力しなければなりません。変数はすべて小文字で示されています (たとえば、*column-name*)。変数はユーザー提供の名前または値を表します。
- 句読記号、括弧、算術演算子、あるいは他にそのような記号が示されている場合には、それらを構文の一部として入力しなくてはなりません。

ホスト言語での SQL プログラミング、バージョン 5 リリース 2 の新機能

C、C++、COBOL、ILE COBOL、ILE RPG、および PL/I における新規ホスト変数タイプ ROWID。以下を参照してください。

- 31 ページの『SQL を使用する C および C++ アプリケーションでの ROWID ホスト変数』
- 62 ページの『SQL を使用する COBOL アプリケーションでの ROWID ホスト変数』
- 86 ページの『SQL を使用する PL/I アプリケーションでの ROWID ホスト変数』
- 122 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの ROWID 変数』

C および C++ における SQL VARCHAR タイプ。詳細は、22 ページの『SQL を使用する C および C++ アプリケーションでのホスト変数の文字』を参照してください。

第 1 章 ホスト言語での SQL の使用における一般的な概念と規則

この章では、ホスト言語での SQL ステートメントの使用における一般的な概念と規則を、次の事項に分けて説明します。

- SQL ステートメントでのホスト変数の使用
- SQL のエラー・コードおよび戻りコードの処理
- WHENEVER ステートメントによる例外条件の処理

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

SQL を使用するアプリケーションの作成

DB2 UDB for iSeries SQL ステートメントおよび機能を使用したホスト言語でのデータベース・アプリケーションを作成できます。各ホスト言語についての、アプリケーション要件およびコーディング要件の詳細は、以下を参照してください。

- 13 ページの『第 2 章 C および C++ アプリケーションでの SQL ステートメントのコーディング方法』
- 47 ページの『第 3 章 COBOL アプリケーションでの SQL ステートメントのコーディング方法』
- 77 ページの『第 4 章 PL/I アプリケーションでの SQL ステートメントのコーディング方法』
- 97 ページの『第 5 章 RPG for iSeries アプリケーションでの SQL ステートメントのコーディング方法』
- 111 ページの『第 6 章 ILE RPG for iSeries アプリケーションでの SQL ステートメントのコーディング方法』
- 133 ページの『第 7 章 REXX アプリケーションでの SQL ステートメントのコーディング方法』
- 143 ページの『第 8 章 SQL ステートメントを含むプログラムの準備と実行』

注: ホスト言語として Java™ を使用する場合の詳細については、「IBM Developer Kit for Java」を参照してください。

SQL ステートメントでのホスト変数の使用

ユーザー・プログラムがデータを検索する場合、その値は、ユーザー・プログラムによって定義され、SELECT INTO ステートメントおよび FETCH ステートメントの INTO 文節で指定されたデータ項目に入れられます。このようなデータ項目を**ホスト変数**と呼びます。

ホスト変数は、ユーザー・プログラムのフィールドで、通常、列の取り出し元または受け入れ先として SQL ステートメントの中で指定されるものです。ホスト変数

と列は、データ・タイプに互換性がなければなりません。ホスト変数は、テーブルやビューなどの SQL オブジェクトの識別には使用できません。ただし、DESCRIBE TABLE ステートメントでは別です。

ホスト構造は、選択された一連の値 (たとえば、ある行の一連の列の値) の取り出し元または受け入れ先として使用されるホスト変数のグループです。**ホスト構造配列**は、複数行用 FETCH ステートメントおよびブロック化 INSERT ステートメントで使用されるホスト構造の配列です。

注: SQL ステートメントの中でリテラル値の代わりにホスト変数を使用すると、アプリケーション・プログラムがテーブルまたはビューのいくつかの異なる行を処理する際の柔軟性を高めることができます。

たとえば、WHERE 文節の中で実際の部門番号をコーディングする代わりに、その時処理を必要としている部門番号にセットされたホスト変数を使用することができます。

一般に、ホスト変数は、SQL ステートメントの中で次のように使用されます。

1. **WHERE 文節の中で:** 探索条件の述部の中の値を指定するため、または式の中のリテラル値を書き換えるために、ホスト変数を使用することができます。たとえば、社員番号を入れるための EMPID という名前のフィールドが定義されているとすれば、社員番号が 000110 である社員の名前を、次のようにして検索することができます。

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

2. **列の値の受け入れ値として (INTO 文節で指定):** 検索した行の列の値を入れるプログラム・データ域を指定するために、ホスト変数を使用することができます。INTO 文節に、SQL から返された列の値を入れる 1 つ以上のホスト変数の名前を指定します。たとえば、CORPDATA.EMPLOYEE テーブルの行から、*EMPNO*、*LASTNAME*、および *WORKDEPT* の各列の値を検索するものとします。この場合、各列の値を入れるホスト変数をユーザー・プログラムで定義し、それらのホスト変数の名前を INTO 文節に指定します。以下に、例を示します。

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

   この例では、ホスト変数 CBLEMPNO は EMPNO の値を受け入れ、CBLNAME は LASTNAME の値を受け入れ、CBLDEPT は WORKDEPT の値を受け入れます。

3. **SELECT 文節の値として:** SELECT 文節に項目のリストを指定する場合、指定できるのは、テーブルやビューの列名に限定されているわけではありません。ユーザー・プログラムは、ホスト変数の値およびリテラル定数を混在させて一連の列の値を返すことができます。以下に、例を示します。

```
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.
```

結果は次のようになります。

PROCESS	PERSON-NAME	EMP-SAL (給与)	EMP-RAISE (昇給額)	EMP-TTL (合計)
A	LUTZ	29840	4476	34316

4. **SQL ステートメントの他の文節の値として:**

   UPDATE ステートメントの SET 文節

   INSERT ステートメントの VALUES 文節

   CALL ステートメント

   これらのステートメントの詳細については、「SQL 解説書」を参照してください。

ホスト変数の使用法の詳細については、次のセクションを参照してください。

- 『SQL ステートメントでのホスト変数の割り当て規則』
- 6 ページの『SQL を使用するアプリケーションでの標識変数』

SQL ステートメントでのホスト変数の割り当て規則

SQL 値は、FETCH、SELECT INTO、SET、および VALUES INTO ステートメントの実行時に、ホスト変数にセットされます (すなわち割り当てられます)。SQL 値は、INSERT、UPDATE、および CALL ステートメントの実行時に、ホスト変数からセットされます (すなわち割り当てられます)。すべての割り当て操作は次の規則に従って行われます。

- 数字と文字列の間には互換性はありません。

  文字列の列または文字列のホスト変数に数値を割り当てることはできません。

  数値列または数値のホスト変数に文字列を割り当てることはできません。
- すべての文字および DBCS 漢字ストリングは、各 CCSID 間で変換がサポートされている場合は、UCS-2 グラフィック列と互換性があります。 CCSID に互換性があれば、すべての漢字ストリングは互換性があります。すべての数値は互換性があります。必要があれば、SQL により変換が行われます。すべての文字および DBCS 漢字ストリングは、各 CCSID 間で変換がサポートされている場合は、割り当て操作の UCS-2 グラフィック列と互換性があります。 CALL ステートメントに関しては、変換がサポートされている場合は、DBCS グラフィック・パラメーターは UCS-2 パラメーターと互換性があります。
- 関連する標識変数をもっていないホスト変数にヌル値を割り当てることはできません。
- タイプが異なる日付 / 時刻値の間には互換性がありません。日付は、日付または日付の文字表現とだけ互換性があります。時刻は、時刻または時刻の文字表現と

だけ互換性があります。タイム・スタンプは、タイム・スタンプまたはタイム・スタンプの文字表現とだけ互換性があります。

日付は、日付列、文字列、DBCS-open または DBCS-either 列または変数、または文字変数[1] にだけ割り当てることができます。日付列の挿入値または更新値は、日付または日付の文字表現でなければなりません。

時刻は、時刻列、文字列、DBCS-open または DBCS-either 列または変数、または文字変数にだけ割り当てることができます。時刻列の挿入値または更新値は、時刻または時刻の文字表現でなければなりません。

タイム・スタンプは、タイム・スタンプ列、文字列、DBCS-open または DBCS-either 列または変数、または文字変数にだけ割り当てることができます。タイム・スタンプ列の挿入値または更新値は、タイム・スタンプまたはタイム・スタンプの文字表現でなければなりません。

SQL ステートメントでのホスト変数の文字列割り当て規則

文字ストリングの割り当てに関する規則は次のとおりです。

- 文字列を列に割り当てる場合は、その文字列の長さはその列の長さ属性より小さくなければなりません。(末尾ブランクは、通常文字列の長さに含まれます。ただし、文字列割り当ての場合、末尾ブランクは文字列の長さに含まれません。)
- MIXED 文字結果列を MIXED 列に割り当てる場合は、MIXED 文字結果列の値は有効な MIXED 文字列でなければなりません。
- 結果列の値をホスト変数に割り当てる場合、結果列の文字列値の方がホスト変数の長さ属性より長いと、文字列の右側の部分が余分の桁数だけ切り捨てられます。その場合には、SQLWARN0 および SQLWARN1 (SQLCA の中の) が W にセットされます。
- 結果列の値を固定長ホスト変数に割り当てる場合、またはホスト変数の値を固定長の CHAR 結果列に割り当てる場合に、文字列値の長さが受け入れ先の長さ属性より小さいと、文字列の右側の部分に余分な桁数だけブランクが埋め込まれます。
- MIXED 文字結果列の値を入れたホスト変数の長さが文字列の長さより短かったために、その文字列の末尾が切り捨てられる場合も、文字列の終わりのシフトイン文字は残されます。したがって、結果は有効な MIXED 文字ストリングのまま残っています。

SQL ステートメントでのホスト変数の CCSID 規則

ある文字値またはグラフィック値を別の文字値またはグラフィック値に割り当てる時、CCSID を考慮する必要があります。これには、ホスト変数の割り当てが含まれます。データベース・マネージャーは、SBCS データ、DBCS データ、MIXED データ、およびグラフィック・データを変換するとき、共通のシステム・サービスを使用します。

CCSID に関する規則は次のとおりです。

---

1. DBCS-open または DBCS-either 変数とは、外部記述ファイルの定義を組み入れることによってホスト言語で宣言された変換です。DBCS-open 変数は、そのジョブの CCSID が MIXED (混合) データを示している場合や、DECLARE VARIABLE ステートメントが使用され、MIXED CCSID または FOR MIXED DATA 文節が指定されている場合にも、宣言されます。「SQL 解説書」の『変数の宣言』を参照してください。

- 取り出し元の CCSID が受け入れ先の CCSID と一致するときは、変換することなく値が割り当てられます。
- 取り出し元または受け入れ先のサブタイプが BIT のときは、変換することなく値が割り当てられます。
- 値がヌルかヌル・ストリングのときは、変換することなく値が割り当てられます。
- 特定の CCSID 相互間で変換することが定義されていないときは、値は割り当てられず、エラー・メッセージが出されます。
- 変換が定義されていて、変換が必要な場合は、取り出し元の値が受け入れ先の CCSID に変換されてから割り当てが行われます。

CCSID の詳細については、Information Center の『グローバリゼーション』を参照してください。

SQL ステートメントでのホスト変数の数値割り当て規則

数値の割り当てに関する規則は次のとおりです。

- **数値を浮動小数点数に変換するとき、その数値の整数部分が変更されることがあります。**単精度の浮動小数点フィールドには 7 桁の 10 進数しか入れることができません。7 桁を超える数値の整数部分は、丸めによって変更されます。倍精度の浮動小数点フィールドには 16 桁の 10 進数しか入れることはできません。 16 桁を超える数値の整数部分は、丸めによって変更されます。
- **数値の整数部分が切り捨てられることはありません。**必要ならば、ステートメント内の小数部が切り捨てられます。変換後の数値が受け入れ先のホスト変数または列に収まらない場合には、負の SQLCODE が返されます。
- **10 進数、数字、または 2 進数**が 10 進数、数字、または 2 進数の列またはホスト変数に割り当てられた場合には、必要ならば、その数値は受け入れ先の精度および位取りに変換されます。その際に、必要数の先行ゼロが付加または削除されます。数値の小数部においては、必要数の後続ゼロが付加されるか、または末尾の必要数の数字が除去されます。
- **2 進数または浮動小数点数**が 10 進数または数字の列またはホスト変数に割り当てられた場合は、その数値はまず一時的に 10 進数または数字に変換され、その後で必要があれば受け入れ先の精度および位取りに変換されます。
  - 位取りが 0 の **ハーフワード 2 進数** (SMALLINT) が 10 進数または数字に変換される場合には、一時的な結果は精度が 5 で位取りが 0 となります。
  - **フルワード 2 進数** (INTEGER) が 10 進数または数字に変換される場合には、一時的な結果は精度が 11 で位取りが 0 となります。
  - **ダブル・フルワード 2 進数** (BIGINT) が 10 進数または数字に変換される場合には、一時的な結果は精度が 19 で位取りが 0 となります。
  - **浮動小数点数**が 10 進数または数字に変換される場合には、一時的な結果は精度が 31 となり、最大の位取りは、有効数字も正確度も失わずに数値の整数部分全体を表すものになります。

SQL ステートメントでのホスト変数の日付、時刻、およびタイム・スタンプの割り当て規則

日付がホスト変数に割り当てられるときは、その日付は CRTSQLxxx コマンドの DATFMT パラメーターおよび DATSEP パラメーターで指定された文字表現に変換

されます。先行ゼロは文字表現のどの部分からも省かれません。ホスト変数は固定長または可変長の文字ストリング変数であって、その長さは *USA、*EUR、*JIS、または *ISO の日付形式の場合は少なくとも 10 バイト、*MDY、*DMY、または *YMD の日付形式の場合は 8 バイト、*JUL の日付形式の場合は 6 バイトでなければなりません。長さが 10 を超えるときは、文字列の右側にブランクが埋められます。 ILE RPG および ILE COBOL では、ホスト変数は日付変数にも使用できます。

時刻がホスト変数に割り当てられると、その時刻は、CRTSQLxxx コマンドの TIMFMT パラメーターおよび TIMSEP パラメーターで指定された文字表現に変換されます。先行ゼロは省かれません。ホスト変数は固定長または可変長の文字ストリング変数でなければなりません。ホスト変数の長さが時刻の文字表現よりも大きいときは、その文字列は右側にブランクが埋められます。 ILE RPG および ILE COBOL では、ホスト変数は時刻変数にも使用できます。

- *USA 形式を使用するときは、ホスト変数の長さは 8 以上でなければなりません。
- *HMS、*ISO、*EUR, または *JIS 形式を使用する場合は、ホスト変数の長さは、秒を含める場合は少なくとも 8 バイトに、時と分だけが必要な場合は 5 バイトにしなければなりません。この場合は、SQLWARN0 および SQLWARN1 (SQLCA の中の) は W にセットされ、標識変数の指定があれば、切り捨てられた実際の秒数にセットされます。

タイム・スタンプがホスト変数に割り当てられるときは、そのタイム・スタンプはその文字表現に変換されます。先行ゼロはどの部分からも省かれません。ホスト変数は固定長または可変長の文字ストリング変数であって、その長さは少なくとも 19 バイトでなければなりません。長さが 26 未満のときは、ホスト変数にはマイクロ秒の全桁が収まりません。長さが 26 より大きいときは、ホスト変数は右側にブランクが埋められます。 ILE RPG および ILE COBOL では、ホスト変数はタイム・スタンプ変数にも使用できます。

SQL を使用するアプリケーションでの標識変数

標識変数はハーフワードの整数変数であり、関連するホスト変数にヌル値が割り当てられているかどうかを示すために使用されます。

- 結果の列の値がヌルである場合には、SQL は標識変数に -1 を入れます。
- 標識変数を使用していないときに結果列がヌル値の場合には、負の SQLCODE が返されます。
- 結果の列が原因でデータ・マッピング・エラーが起こった場合には、 SQL は標識変数を -2 にセットします。

標識変数を使用すると、検索された文字列が切り捨てられたかどうかを確かめることができます。切り捨てが行われた場合は、標識変数には文字列の元の長さを指定する正の整数値が入っています。文字列がラージ・オブジェクト (LOB) を表し、その文字列の元の長さが 32767 より長い場合は、標識変数に保管されている値は 32767 になります。これは、32767 より長い値をハーフワードの整数に保管できないからです。

結果列の値がデータベース・マネージャーから返されたときは、標識変数を調べることができます。標識変数の値がゼロより小さければ、結果列の値がヌルであるこ

とが分かります。データベース・マネージャーがヌル値を返したときは、ホスト変数は結果列の省略時値にセットされます。

標識変数は、ホスト変数の直後 (前にコロンを付けて) か、キーワード INDICATOR の直後に指定します。以下に、例を示します。

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

次に INDNULL に負の値が入っているかどうかを検証することができます。負の値が入っていれば、SQL からヌルの値が返されたことが分かります。

列が NULL であるかどうかを検査するときは、必ず **IS NULL** 述部を使用してください。以下に、例を示します。

```
WHERE expression IS NULL
```

次の方法で NULL を検査しないでください。

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

EQUAL 述部は、ヌル値を比較するときは必ず偽と評価されます。この例の結果では、行は選択されません。

ホスト構造で使用される標識変数

ホスト構造をサポートするために**標識構造** (ハーフワードの整数変数の配列として定義されているもの) を使用することもできます。ホスト構造に返される結果列の値がヌルになり得る場合には、ホスト構造名に標識構造名を付加することができます。これにより、SQL は、ホスト構造内のホスト変数にヌル値が返されるたびに、それをユーザーのプログラムに知らせることができます。

次は COBOL で書いた例です。

```
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.
```

この例では、SALIND は 3 つの値を含む配列であり、それらの値のおのおのについてそれが負の値であるかどうかをテストすることができます。たとえば、SALIND(1) に負の値が入っていれば、ホスト構造の中の対応するホスト変数 (すなわち MIN-SAL) は、選択された行については変更されません。

上記の例では、SQL は、行の列の値を選択してホスト構造に入れます。したがって、選択されたどの列の値がヌルであるかを判別するためには、ユーザーは対応する構造を標識変数として使用しなければなりません。

ヌル値をセットするために使用する標識変数

標識変数を使用すると、列にヌル値をセットすることができます。 UPDATE ステートメントまたは INSERT ステートメントを処理するとき、SQL は標識変数 (存在する場合) を検査します。標識変数に負の値が入っていれば、列の値はヌルにセットされます。-1 より大きい値が入っていれば、それに対応するホスト変数には列の値が入っています。

たとえば、値を列に入れること (INSERT ステートメントまたは UPDATE ステートメントを使用して) を指定できますが、入力データにその値が指定されていたかどうか確かでないことがあります。列にヌル値をセットできるようにするには、次のようなステートメントを書くことができます。

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

NEWPHONE にヌル値以外の値が入っている場合は、ステートメントの前に次のものを置くことによって PHONEIND をゼロにセットします。

```
MOVE 0 to PHONEIND.
```

そうでない場合は、NEWPHONE にヌル値が入っていることを SQL に伝えるには、次のように PHONEIND を負の値にセットします。

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

SQL のエラー戻りコードの処理

ユーザーのプログラムで SQL ステートメントが処理されると、 SQL は SQLCODE フィールドと SQLSTATE フィールドに戻りコードを入れます。戻りコードは、ステートメントの実行が正常に完了したか、失敗したかを示します。 SQL がステートメントを処理している途中でエラーを見つけると、SQLCODE は負の値となり、SUBSTR(SQLSTATE,1,2) は '00'、 '01'、または '02' のいずれでもなくなります。SQL がステートメントを処理している途中で例外条件を見つけたものの、それが有効な条件であれば、 SQLCODE は正の数となり、SUBSTR(SQLSTATE,1,2) は '01' または '02' となります。SQL ステートメントの処理中にエラー条件も、警告条件も見つからなければ、SQLCODE はゼロになり、SQLSTATE は '00000' になります。

注: ユーザーのプログラムにゼロの SQLCODE が返された場合でも、結果が満足すべきものでないことがあります。たとえば、あるプログラムの実行の結果、ある一部が切り捨てられたとしても、プログラムに返される SQLCODE はゼロです。ただし、SQL 警告標識の 1 つ (SQLWARN1) に、切り捨てが生じたことが示されます。この場合、SQLSTATE は '00000' ではありません。

重要: SQLCODE が負かどうかをテストしない場合や、WHENEVER SQLERROR ステートメントの指定がない場合は、プログラムは次のステートメントに進みます。エラーがあったあと実行を続けると、予期しない結果が生じることがあります。

SQLSTATE の主目的は、異なる IBM リレーショナル・データベース・システム間で共通の戻り条件が起こったとき、共通の戻りコードを出すことにあります。SQLSTATE は、分散データベース操作で問題を処理するときに使用すると、特に便利です。詳細については、「SQL 解説書」を参照してください。

SQLCA は有用な問題診断ツールであるので、SQLCA に入っている情報の一部を表示するために必要な命令をアプリケーション・プログラムに組み込んでおくと便利です。特に重要なのは、次の SQLCA フィールドです。

SQLCODE 戻りコード。

SQLSTATE 戻りコード。

SQLERRD(3) SQL により更新、挿入、または削除された行数。

SQLWARN0 W にセットされた場合、SQL 警告フラグ (SQLWARN1 ～ SQLWARNA) の少なくとも 1 つがセットされます。

SQLCA の詳細については、「SQL 解説書」の『付録 B. SQL 連絡域』を参照してください。特定の SQLCODE、または SQLSTATE を検索するには、SQL メッセージ検索機能を使用してください。DB2 UDB for iSeries の SQLCODE と SQLSTATE の一覧表については、iSeries Information Center の『SQL メッセージおよびコード』を参照してください。

WHENEVER ステートメントによる例外条件の処理

WHENEVER ステートメントが現れると、SQL は SQLSTATE と SQLCODE を検査して、ユーザーのプログラムの処理を続けるか、あるいは SQL ステートメントの実行の結果としてエラー、例外、または警告が生じていれば、プログラム内の別の個所に分岐します。例外条件処理サブルーチン (ユーザーのプログラムの一部) は、 SQLCODE フィールドまたは SQLSTATE フィールドを調べて、発生したエラーまたは例外条件に見合った処置を取ることができます。

注: WHENEVER ステートメントは REXX プロシージャーでは使用できません。REXX での例外条件の処理については、 133 ページの『第 7 章 REXX アプリケーションでの SQL ステートメントのコーディング方法』を参照してください。

WHENEVER ステートメントを使用すると、一般的な条件に該当するときどのような処置をとるべきか指定することができます。同じ条件について複数の WHENEVER ステートメントを指定できます。その場合は、最初の WHENEVER ステートメントは、別の WHENEVER ステートメントが指定されるまでソース・プログラム内の後続ステートメントすべてに適用されます。

WHENEVER ステートメントは次のようになっています。

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

条件として指定できるのは次の 3 つです。

SQLWARNING

SQLWARNING は、SQLWARN0 = W のとき、または SQLCODE が 100 (SUBSTR(SQLSTATE,1,2) ='01') 以外の正の値であるとき、どのような処置をとるかを指示するために指定します。

注: SQLWARN0 がセットされる理由はいくつか考えられます。たとえば、ある列の値がホスト変数に入れられるとき、その一部が切り捨てられたとしても、ユーザーのプログラムはそれをエラーと見なさないことがあります。

SQLERROR

SQLERROR は、 SQL ステートメントの結果としてコードが返されたとき ((SQLCODE < 0) (SUBSTR(SQLSTATE,1,2) > '02'))、どのような処置をとるかを指示するために指定します。

NOT FOUND

NOT FOUND は、次のいずれかの理由で +100 の SQLCODE と '02000' の SQLSTATE が返されたとき、どのような処置をとるかを指定するために使用します。

- 単一行の SELECT が出された後、またはカーソルについて最初の FETCH が出された後で、プログラムで指定したデータが存在しない。
- 2 番目以降の FETCH の後で、カーソル選択ステートメントを満たす検索対象行が残っていない。
- UPDATE、DELETE、または INSERT の後で、探索条件を満たす行がない。

処置として指定できる値は次のいずれかです。

CONTINUE

この値を指定すると、ユーザーのプログラムは次のステートメントに進みます。

GO TO ラベル

これを指定すると、ユーザーのプログラムはプログラム内の別の個所に分岐します。その個所を示すラベルの前にはコロンを付けることができます。WHENEVER ... GO TO ステートメントは次のように指定します。

- COBOL の場合は、セクション名または非修飾段落名でなければなりません。
- PL/I と C の場合は、ラベルです。
- RPG の場合は、TAG のラベルです。

たとえば、カーソルを使用して行を検索しているときに、場合によっては FETCH ステートメントが出されたのに SQL が新たな行を見つけることができないことがあります。このような場合に備えて、WHENEVER NOT FOUND GO TO ... ステー

トメントを指定しておけば、プログラム内の CLOSE ステートメントが出される個所に SQL を分岐させて、カーソルを正しくクローズすることができます。

注: WHENEVER ステートメントは、別の WHENEVER が現れるまでの間の、後続のすべてのソース SQL ステートメントに影響します。

すなわち、2 つの WHENEVER ステートメントの間にコーディングされているすべての SQL ステートメント (WHENEVER が 1 つしかない場合にはその後のすべてのステートメント) は、プログラムが進む経路とは関係なく、最初の WHENEVER ステートメントの影響を受けます。

したがって、WHENEVER ステートメントは、その影響を受ける最初の SQL ステートメントの*前に置かなければなりません。* WHENEVER が SQL ステートメントの*あとに置かれている* と、その SQL ステートメントによってセットされた SQLCODE と SQLSTATE の値に基づく分岐は行われません。ただし、ユーザーのプログラムで SQLCODE または SQLSTATE を直接検査する場合は、その検査は SQL ステートメントの実行後に行う必要があります。

WHENEVER ステートメントには、サブルーチン・オプションに対する CALL を行う働きはありません。そのために、WHENEVER ステートメントを使用するよりも、各 SQL ステートメントの実行後に SQLCODE または SQLSTATE の値を検査してサブルーチンを呼び出すようにした方がよい場合があります。

第 2 章 C および C++ アプリケーションでの SQL ステートメントのコーディング方法

この章では、SQL ステートメントを C または C++ プログラムに組み込む場合に固有のアプリケーションおよびコーディング上の要件について説明します。C プログラムとは、ILE C for iSeries プログラムを意味します。C++ プログラムとは、ILE C++ プログラムを意味します。この章では、ホスト構造およびホスト変数に関する要件についても説明します。詳細については、以下のセクションを参照してください。

SQL ステートメントの使い方を示した詳しいサンプル C プログラムは、『付録 A. DB2 UDB for iSeries ステートメントを使用するサンプル・プログラム』に記載されています。

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

SQL を使用する C および C++ アプリケーションでの SQL 連絡域の定義

SQL ステートメントの入っている C または C++ プログラムは、次のいずれかまたは両方を含んでいなければなりません。

- long SQLCODE として宣言されている SQLCODE 変数
- char SQLSTATE[6] として宣言されている SQLSTATE 変数

または、

- SQLCA (SQLCODE および SQLSTATE 変数が入っている)

SQLCODE 値および SQLSTATE 値は、各 SQL ステートメントが実行された後、データベース・マネージャーによってセットされます。アプリケーションは、SQLCODE 値または SQLSTATE 値を調べて、最後の SQL ステートメントが正しく実行されたかどうかを判定することができます。

SQLCA は、直接または、SQL の INCLUDE ステートメントを使用して、 C または C++ プログラムの中にコーディングすることができます。SQL の INCLUDE ステートメントを使用するときは、次のような標準の宣言を含める必要があります。

```
EXEC SQL INCLUDE SQLCA ;
```

標準の宣言には、構造の定義と 'sqlca' という名前の静的データ域が含まれます。

SQLCODE、SQLSTATE、および SQLCA の各変数は、実行可能ステートメントの前に現れなければなりません。宣言の有効範囲には、プログラムの中のすべての SQL ステートメントの有効範囲が含まれていなければなりません。

SQLCA を組み込んだ C または C++ ソース・ステートメントは次のとおりです。

```
#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;
```

SQLCODE の宣言がプログラムの中にあり、プリコンパイラーが SQLCA を提供している場合、SQLCODE の個所は SQLCADE で置き換えられます。 SQLSTATE の

宣言がプログラムの中にあり、プリコンパイラーが SQLCA を提供している場合、SQLSTATE の個所は SQLSTOTE によって置き換えられます。

注: SQL エラー・メッセージの多くは、可変長のメッセージ・データを含みます。これらのデータ・フィールド長は、SQLCA の sqlerrmc フィールドの値に組み込まれます。この長さが原因で、C または C++ プログラムからの sqlerrmc の値の印刷が予測不可能な結果となる場合があります。

SQLCA の詳細については、「SQL 解説書」の『付録 B. SQL 連絡域』を参照してください。

SQL を使用する C および C++ アプリケーションでの SQL 記述子域の定義

SQLDA を必要とするステートメントには、次のものがあります。

EXECUTE...USING DESCRIPTOR *記述子名*

FETCH...USING DESCRIPTOR *記述子名*

OPEN...USING DESCRIPTOR *記述子名*

DESCRIBE *ステートメント名* INTO *記述子名*

DESCRIBE TABLE *ホスト変数* INTO *記述子名*

PREPARE *ステートメント名* INTO *記述子名*

CALL...USING DESCRIPTOR *記述子名*

SQLCA と異なり、SQLDA を 2 つ以上プログラムの中に置くことができ、また SQLDA の名前は有効であれば、どの名前でも使えます。SQLDA は、直接または、SQL の INCLUDE ステートメントを使用して、 C または C++ プログラムの中にコーディングすることができます。SQL の INCLUDE ステートメントを使用するときは、標準の SQLDA 宣言を組み込む必要があります。

```
EXEC SQL INCLUDE SQLDA;
```

標準の宣言には構造の定義だけが含まれ、その名前は 'sqlda' です。

SQLDA 用として組み込まれる C および C++ 宣言は次のとおりです。

```
#ifndef SQLDASIZE
struct sqlda {
              unsigned char sqldaid[8];
              long sqldabc;
              short sqln;
              short sqld;
              struct sqlvar {
                            short sqltype;
                            short sqllen;
                            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 SQL ステートメントを使用すると、下記のマクロ定義も得られるという利点があります。

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

このマクロを使用すると、SQLVAR 要素の個数を指定して、SQLDA 用の記憶域を割り振ることが簡単になります。次の例では、20 個の SQLVAR 要素を持つ SQLDA 用の記憶域を割り振るために SQLDASIZE マクロが使用されます。

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

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

INCLUDE SQLDA ステートメントに含まれる、その他のマクロ定義を以下に示します。

GETSQLDOUBLED(daptr) daptr がポイントする SQLDA が double である場合に 1、そうでない場合に 0 を戻します。 SQLDA は、SQLDAID フィールドの 7 番目のバイトが '2' に設定されると double となります。

SETSQLDOUBLED(daptr, newvalue)
SQLDAID の 7 番目のバイトを newvalue に設定します。

GETSQLDALONGLEN(daptr,n)
daptr がポイントする SQLDA の n 番目の項目の長さ属性を戻します。これは、SQLDA が double で、n 番目の SQLVAR 項目が LOB データ・タイプを持つ場合のみ使用します。

SETSQLDALONGLEN(daptr,n,len)
daptr がポイントする SQLDA の SQLLONGLEN フィールドの n 番目の項目を len に設定します。これは、SQLDA が double で、n 番目の SQLVAR 項目が LOB データ・タイプを持つ場合のみ使用します。

GETSQLDALENPTR(daptr,n) daptr がポイントする SQLDA の n 番目の項目のデータの実際の長さへのポインターを戻します。SQLDATALEN ポインター・フィールドは、 long (4 バイト) 整数へのポインターを戻します。SQLDATALEN ポインターがゼロの場合、NULL ポインターが戻ります。これは、SQLDA が double の場合のみ使用します。

SETSQLDALENPTR(daptr,n,ptr)
daptr がポイントする SQLDA の n 番目の項目のデータの実際の長さへのポインターを設定します。これは、SQLDA が double の場合のみ使用します。

SQLDA をポインターとして宣言したときは、ポインターとして宣言したホスト変数の場合と全く同じように、それを SQL ステートメントの中で使用するとき宣言したとおりにそれを参照しなければなりません。コンパイラー・エラーを防止するには、SQLDA の sqldata フィールドに割り当てられた値のタイプを、符号なしのポインターにしなければなりません。このように指定すると、コンパイラー・エラーの防止に役立ちます。このようなタイプの指定が必要になるのは、アプリケーション・プログラムがプログラムの中でホスト変数のアドレスを引き渡す時に使用する EXECUTE、OPEN、CALL、および FETCH ステートメントの場合に限られます。たとえば、mydaptr と名付けた SQLDA にポインターを宣言する場合は、PREPARE ステートメントの中で次のように使用します。

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

SQLDA 宣言は、構造の定義が許される場所ならば、どこにでも置くことができます。通常の C 有効範囲規則が適用されます。

動的 SQL は高度なプログラミング技法です。これについては、「*DB2® UDB for iSeries SQL プログラミング 概念*」の『動的 SQL アプリケーション』で説明します。動的 SQL を使用すると、ユーザーのプログラムはその実行と平行して SQL ステートメントを作成し、実行させることができます。変数 SELECT リスト (すなわち、照会の一部として返されるデータのリスト) を指定する SELECT ステートメントは、SQL 記述子域 (SQLDA) を動的に必要とします。これは、SELECT の結果を受け入れるために割り振るべき変数の数とタイプが事前に予測できないからです。

SQLDA の詳細については、「SQL 解説書」のトピック『SQL 記述子域』を参照してください。

SQL を使用する C および C++ アプリケーションへの SQL ステートメントの組み込み

SQL ステートメントは実行可能な C または C++ ステートメントを置くことができる場所ならば、どこにでも置くことができます。

各 SQL ステートメントは EXEC SQL で始まり、セミコロン (;) で終わらなければなりません。 EXEC SQL キーワードは 1 行でなければなりません。 SQL ステートメントの残りの部分は、 2 行以上にまたがっても構いません。

例 : C または C++ プログラムでコーディングされた UPDATE ステートメントは、次のようになります。

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

詳細については、以下のセクションを参照してください。

- 18 ページの『SQL を使用する C および C++ アプリケーションでの注記』
- 18 ページの『SQL を使用する C および C++ アプリケーションでの SQL ステートメントの継続』
- 19 ページの『SQL を使用する C および C++ アプリケーションへのコードの組み込み』

- 19 ページの『SQL を使用する C および C++ アプリケーションでのマージン』
- 19 ページの『SQL を使用する C および C++ アプリケーションでの名前』
- 19 ページの『SQL を使用する C および C++ アプリケーションでの NULL および NUL』
- 20 ページの『SQL を使用する C および C++ アプリケーションでのステートメント・ラベル』
- 20 ページの『SQL を使用する C および C++ アプリケーションでのプリプロセッサー順序』
- 20 ページの『SQL を使用する C および C++ アプリケーションでの 3 文字表記』
- 20 ページの『SQL を使用する C および C++ アプリケーションでの WHENEVER ステートメント』

SQL を使用する C および C++ アプリケーションでの注記

SQL の注記 (--) を使用する他に、ブランクを入れることができる場合はいつでも組み込み SQL ステートメントの中に C の注記 (/* ...*/) を入れることができます。ただし、キーワードの EXEC と SQL の間には入れられません。注記は何行にもまたがることができます。注記をネストすることはできません。C++ では単一行注記 (// で開始する注記) を使用できますが、 C では使用できません。

SQL を使用する C および C++ アプリケーションでの SQL ステートメントの継続

SQL ステートメントは 1 行または 2 行以上に継続することができます。ブランクを置くことができるならば、どこでも SQL ステートメントを分割することができます。円記号 (¥) を使用すると、文字列定数または区切り文字付き識別コードを継続することができます。区切り文字が付いていない識別コードは、継続することができません。

DBCS データを含む定数を複数の行にわたって継続する方法としては、次の 2 とおりが可能です。

- 継続された行の右マージンにある文字がシフトイン文字で、継続行の左マージンにある文字がシフトアウト文字の場合には、左右のマージンにあるシフト文字が除去されます。

  この SQL ステートメントの G'<AABBCCDDEEFFGGHHIIJJKK>' はグラフィック定数として有効です。重複しているシフトは除去されます。

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

- マージンの外側にシフト文字を置くことができます。下記の例では、マージンが 5 と 75 であるとします。この SQL ステートメントの G'<AABBCCDDEEFFGGHHIIJJKK>' はグラフィック定数として有効です。

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

SQL を使用する C および C++ アプリケーションへのコードの組み込み

SQL ステートメント、C ステートメント、または C++ ステートメントは、ソース・コード内に次の SQL ステートメントを組み込むことによって組み入れることができます。

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

C または C++ の #include ステートメントは、SQL ステートメントを組み込んだり、SQL ステートメントの中で参照される C または C++ のホスト変数の宣言を組み込んだりするためには使用できません。

SQL を使用する C および C++ アプリケーションでのマージン

SQL ステートメントは、CRTSQLCI、または CRTSQLCPPI コマンドの MARGINS パラメーターで指定したマージンの範囲内にコーディングしなければなりません。MARGINS パラメーターが *SRCFILE として指定される場合は、ソース・ファイルのレコード長が使用されます。右マージンについて値が指定され、この値がソース・レコード長よりも大きい場合は、レコード全体が読み取られます。この値は任意の組み込みメンバーにも適用されます。たとえば、右マージンが 200 に指定され、ソース・ファイルのレコード長が 80 である場合、データの 80 列までだけがソース・ファイルから読み取られます。同じプリコンパイルに組み込まれるソース・メンバーのレコード長が 200 である場合は、組み込まれたソース・メンバー 200 がすべて読み取られます。

EXEC SQL が指定のマージン内で始まっていないときは、SQL プリコンパイラーはそれを SQL ステートメントと認識しません。CRTSQLCI または CRTSQLCPPI の詳細については、『付録 B. ホスト言語プリコンパイラーの DB2 UDB for iSeries CL コマンド記述』を参照してください。

SQL を使用する C および C++ アプリケーションでの名前

ホスト変数には、任意の有効な C または C++ 変数名を使用することができます。この変数は、次の制限に従っていなければなりません。

'SQL'、'RDI'、または 'DSN' で始まるホスト変数名や外部入り口名は、大文字や小文字をどのように組み合わせても、使用してはなりません。これらの名前はデータベース・マネージャー用に予約されています。ホスト変数名の長さは 128 までです。

SQL を使用する C および C++ アプリケーションでの NULL および NUL

C、C++ および SQL では、ヌル値という語を使用していますが、意味が異なります。C および C++ 言語には、ヌル文字 (NUL)、ヌル・ポインター (NULL)、およびヌル・ステートメント (セミコロンだけ) があります。C の NUL は、0 に相当する単一文字です。 C の NULL は特殊な予備のポインター値であり、有効などのデータ・オブジェクトも指していません。SQL のヌル値は特殊な値であり、これはすべての非ヌル値と区別され、(非ヌル) 値の不在を示します。

SQL を使用する C および C++ アプリケーションでのステートメント・ラベル

実行可能な SQL ステートメントの前には、ラベルを付けることができます。

SQL を使用する C および C++ アプリケーションでのプリプロセッサー順序

C または C++ プリプロセッサーの前に、SQL プリプロセッサーを実行しなければなりません。 SQL ステートメント内で C または C++ プリプロセッサー・ディレクティブを使用することはできません。

SQL を使用する C および C++ アプリケーションでの 3 文字表記

C および C++ 文字セットの文字の中には、一部のキーボードに付いていないものがあります。これらの文字は、*3 文字表記* と呼ばれる 3 文字の列を使用して、C または C++ ソース・プログラムの中に入れることができます。次の 3 文字表記の列は、ホスト変数宣言の中で使用できます。

- ??(左大括弧
- ??) 右大括弧
- ??< 左中括弧
- ??> 右中括弧
- ??= ポンド
- ??/ バックスラッシュ

SQL を使用する C および C++ アプリケーションでの WHENEVER ステートメント

SQL の WHENEVER ステートメントの中の GOTO 文節の対象となるものは、WHENEVER ステートメントの影響が及ぶ SQL ステートメントの有効範囲内になければなりません。

SQL を使用する C および C++ アプリケーションでのホスト変数の使用

SQL ステートメントの中で使用するホスト変数はいずれも明示的に宣言しなければなりません。 SQL ステートメントの中で使用するホスト変数は、SQL ステートメントの中でホスト変数を初めて使用する前に、宣言しておかなければなりません。

C では、ホスト変数を定義するために使用される C ステートメントは、その前に BEGIN DECLARE SECTION ステートメントを置き、その後に END DECLARE SECTION ステートメントを置く必要があります。BEGIN DECLARE SECTION と END DECLARE SECTION を指定した場合、 SQL ステートメントで使用するすべてのホスト変数宣言は、BEGIN DECLARE SECTION ステートメントと END DECLARE SECTION ステートメントとの間になければなりません。typedef 識別コードを使用して宣言されたホスト変数でも BEGIN DECLARE SECTION と END DECLARE SECTION が必要ですが、 typedef 宣言がこの 2 つのセクションの間にある必要はありません。

C++ では、ホスト変数を定義するために使用される C++ ステートメントは、その前に BEGIN DECLARE SECTION ステートメントを置き、その後に END DECLARE SECTION ステートメントを置く必要があります。BEGIN DECLARE SECTION ステートメントと END DECLARE SECTION ステートメントの間にない変数は、ホスト変数として使用することはできません。

SQL ステートメントの中のホスト変数はいずれも、その前にコロン (:) を付けなければなりません。

ホスト変数の名前は、ホスト変数がそれぞれ別のブロックやプロシージャーの中にある場合であっても、1 つのプログラム内では固有になっていなければなりません。

ホスト変数を使用する SQL ステートメントは、その変数が宣言されたステートメントの有効範囲内になければなりません。

ホスト変数を結合要素にすることはできません。

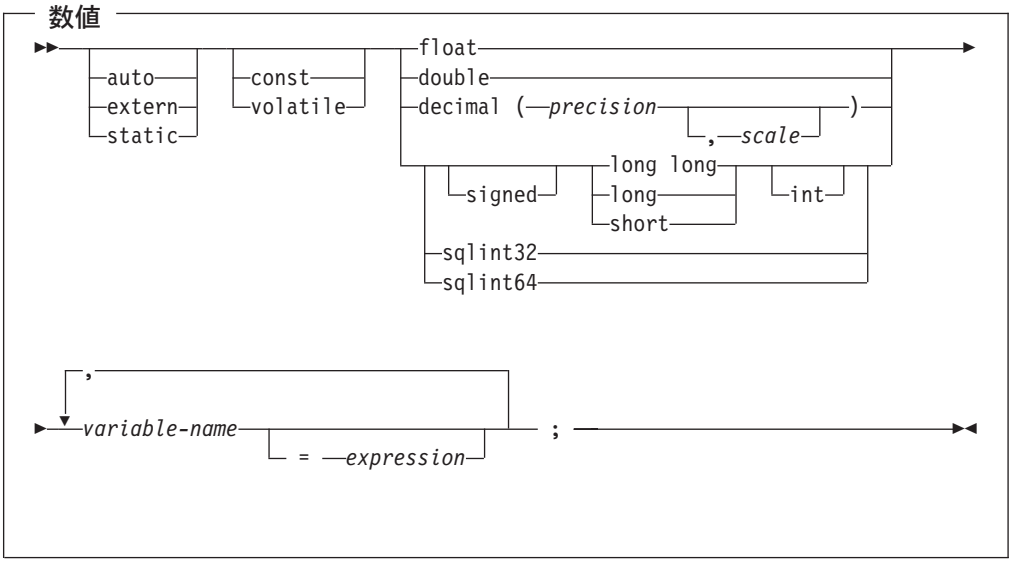
詳細については、『SQL を使用する C および C++ アプリケーションでのホスト変数の宣言』を参照してください。

SQL を使用する C および C++ アプリケーションでのホスト変数の宣言

C および C++ プリコンパイラーは、有効な C および C++ 宣言のサブセットだけを有効なホスト変数宣言として認識します。

SQL を使用する C および C++ アプリケーションでのホスト変数の数値

下図は、有効な数値ホスト変数宣言の構文を示しています。



注:

1. precision (精度) および scale (位取り) は整数定数でなければなりません。precision の範囲は 1 から 31 までです。scale の範囲は 0 から精度の値までです。
2. 10 進データ・タイプを使用している場合は、見出しファイルの decimal.h を取り込む必要があります。
3. sqlint32 または sqlint64 を使用する場合は、ヘッダー・ファイルの sqlsystm.h が組み込まれている必要があります。

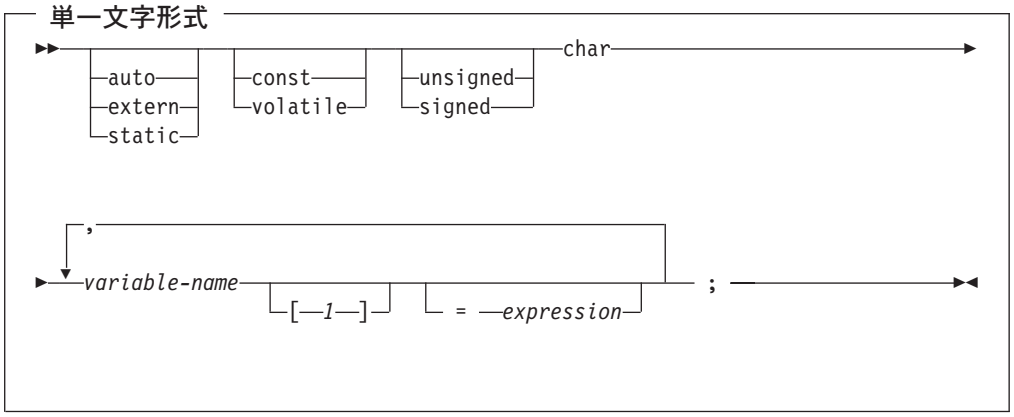
SQL を使用する C および C++ アプリケーションでのホスト変数の文字

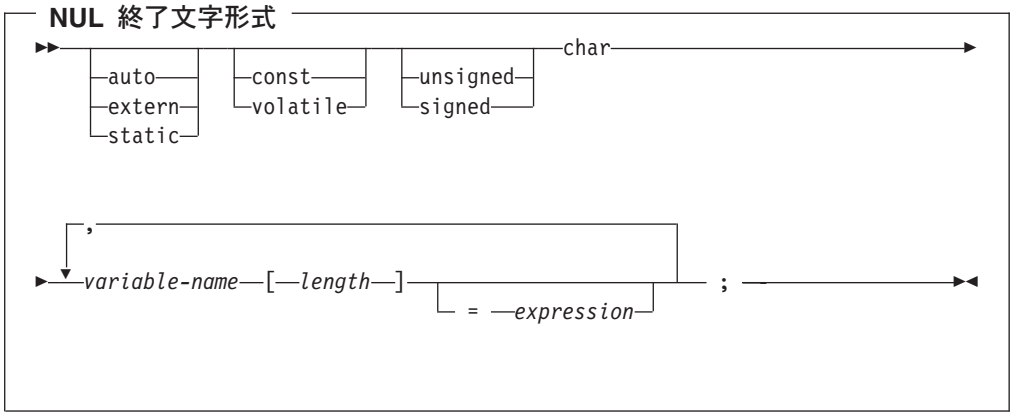
文字ホスト変数には、有効な次の 3 つの形式があります。

- 単一文字形式
- NUL 終了文字形式
- VARCHAR 構造化形式

さらに、SQL VARCHAR 宣言は varchar ホスト変数を定義するために使用できます。

文字タイプはすべて符号なしとして扱われます。



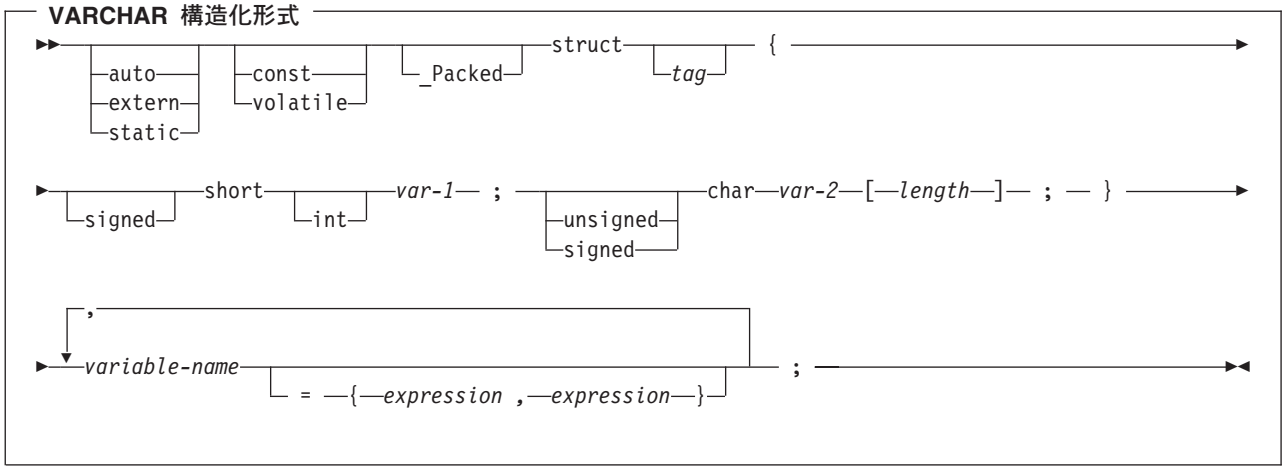


注:

1. length (長さ) は、1 より大きく、32741 以下の整数定数でなければなりません。
2. CRTSQLCI、または CRTSQLCPPI コマンドで *CNULRQD オプションの指定がある場合は、入力ホスト変数に NUL 終了文字を入れる必要があります。出力ホスト変数はブランクで埋め込まれ、最後の文字が NUL 終了文字になります。出力ホスト変数がデータと NUL 終了文字がともに入るだけの大きさになっていないと、次の処置がとられます。
   - データが切り捨てられます。
   - 最後の文字は NUL 終了文字となります。
   - SQLWARN1 は 'W' にセットされます。
3. CRTSQLCI または CRTSQLCPPI コマンドで *NOCNULRQD オプションの指定がある場合は、入力変数に NUL 終了文字を入れる必要はありません。

   出力ホスト変数には以下が適用されます。
   - ホスト変数がデータと NUL 終了文字がともに入る大きさである場合、次の処置がとられます。
     – データは返されますが、ブランクでの埋め込みは行われません。
     – データのすぐ後に NUL 終了文字が入ります。
   - ホスト変数がデータが入る大きさであるが、NUL 終了文字が入る大きさではない場合、次の処置がとられます。
     – データが返されます。
     – NUL 終了文字は返されません。
     – SQLWARN1 は 'N' にセットされます。
   - ホスト変数がデータが入る大きさでない場合、次の処置がとられます。
     – データが切り捨てられます。
     – NUL 終了文字は返されません。
     – SQLWARN1 は 'W' にセットされます。

VARCHAR 構造化形式



注:

1. *length (長さ)* は、0 より大きく、32740 以下の整数定数でなければなりません。
2. *var-1 (変数 1)* と *var-2 (変数 2)* は、単純変数参照にしなければならず、整数ホスト変数および文字ホスト変数として個々に使用することはできません。
3. struct (構造) タグを使用すると、他のデータ域が定義できますが、これらをホスト変数として使用することはできません。
4. VARCHAR 構造化形式は、ヌル値を含む場合のあるビット・データについて使用する必要があります。VARCHAR 構造化形式は、ヌル終了文字を使用して終了されることはありません。
5. _Packed は、C++ では使用してはいけません。代わりに、宣言の前に #pragma pack(1) を、宣言の後に #pragma pack() を指定します。

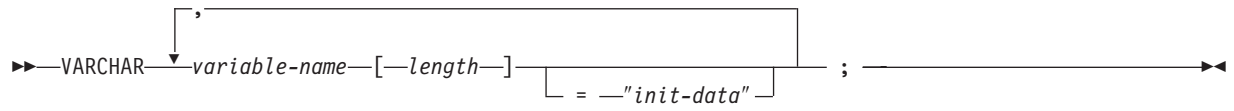
   注: #pragma pack (reset) を #pragma pack() の代わりに使用できます。これらは同じものです。

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

例:

```
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 形式



注:

1. VARCHAR は、大文字小文字混合にすることができます。
2. length (長さ) は、0 より大きく、32740 以下の整数定数でなければなりません。
3. SQL VARCHAR 形式は、NULL 値を含む場合のあるビット・データについて使用する必要があります。SQL VARCHAR 形式は、ヌル終了文字を使用して終了されることはありません。

例:

次のように宣言すると、

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

以下の構造を生成します。

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

次のように宣言すると、

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

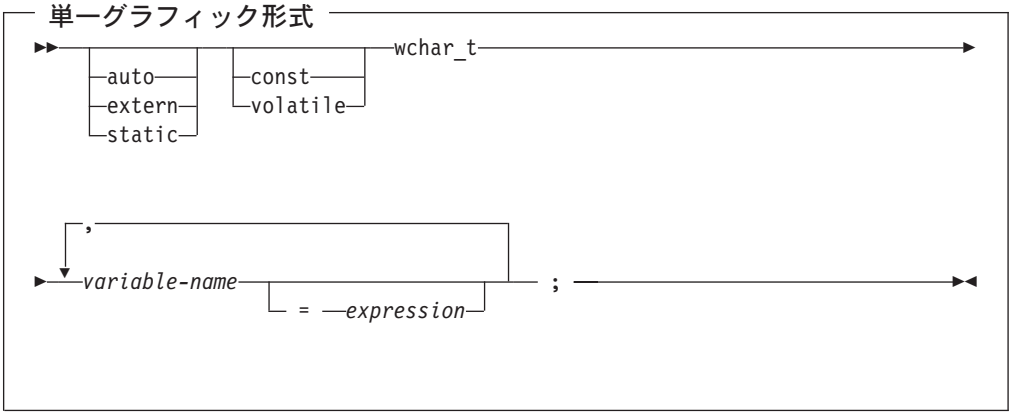
以下の構造を生成します。

```
_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"};
```

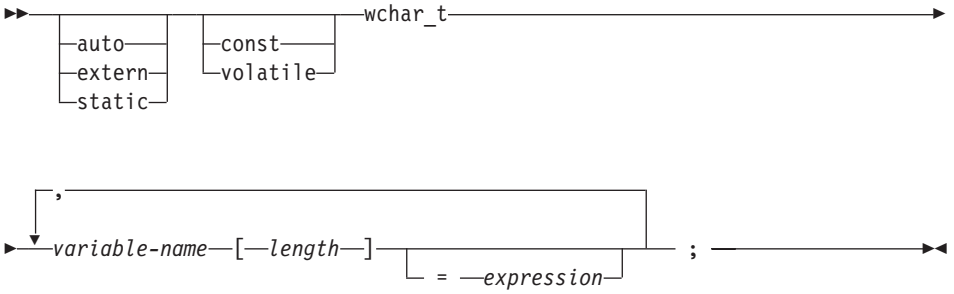
SQL を使用する C および C++ アプリケーションでのグラフィック・ホスト変数

グラフィック・ホスト変数の有効な形式としては、次の 3 つがあります。

- 単一グラフィック形式
- NUL 終了グラフィック形式
- VARGRAPHIC 構造化形式



NUL 終了グラフィック形式

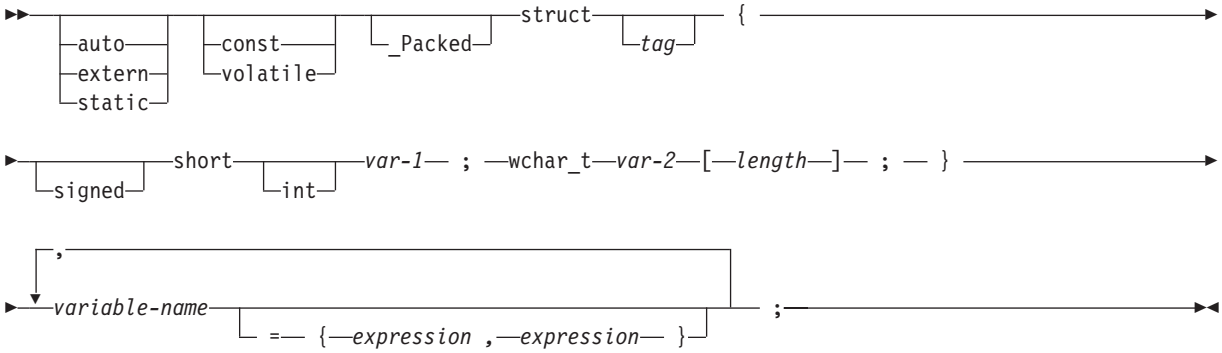


注:

1. *length (長さ)* は、1 より大きく、16371 以下の整数定数でなければなりません。
2. CRTSQLCI または CRTSQLCPPI コマンドで *CNULRQD オプションの指定がある場合は、入力ホスト変数にグラフィック NUL 終了文字 (/0/0) を入れる必要があります。出力ホスト変数は DBCS ブランクで埋め込まれ、最後の文字がグラフィック NUL 終了文字になります。出力ホスト変数がデータと NUL 終了文字がともに入るだけの大きさになっていないと、次の処置がとられます。
   - データが切り捨てられます。
   - 最後の文字はグラフィック NUL 終了文字になります。
   - SQLWARN1 は 'W' にセットされます。

   CRTSQLCI または CRTSQLCPPI コマンドで *NOCNULRQD オプションの指定がある場合は、入力ホスト変数にグラフィック NUL 終了文字を入れる必要はありません。出力ホスト変数の処理は以下のとおりです。
   - ホスト変数がデータとグラフィック NUL 終了文字がともに入る大きさである場合、次の処置がとられます。
     - データは返されますが、DBCS ブランクでの埋め込みは行われません。
     - データのすぐ後に、グラフィック NUL 終了文字が入ります。
   - ホスト変数がデータが入る大きさであるが、グラフィック NUL 終了文字が入る大きさではない場合、次の処置がとられます。
     - データが返されます。
     - グラフィック NUL 終了文字は返されません。
     - SQLWARN1 は 'N' にセットされます。
   - ホスト変数がデータが入る大きさでない場合、次の処置がとられます。
     - データが切り捨てられます。
     - グラフィック NUL 終了文字は返されません。
     - SQLWARN1 は 'W' にセットされます。

VARGRAPHIC 構造化形式



注:

1. *length (長さ)* は、0 より大きく、16370 以下の整数定数でなければなりません。
2. *var-1 (変数 1)* と *var-2 (変数 2)* は、単純変数参照にしなければならず、ホスト変数として使用できません。
3. struct (構造) タグを使用すると、他のデータ域が定義できますが、これらをホスト変数として使用することはできません。
4. _Packed は、C++ では使用してはいけません。代わりに、宣言の前に #pragma pack(1) を、宣言の後に #pragma pack() を指定します。

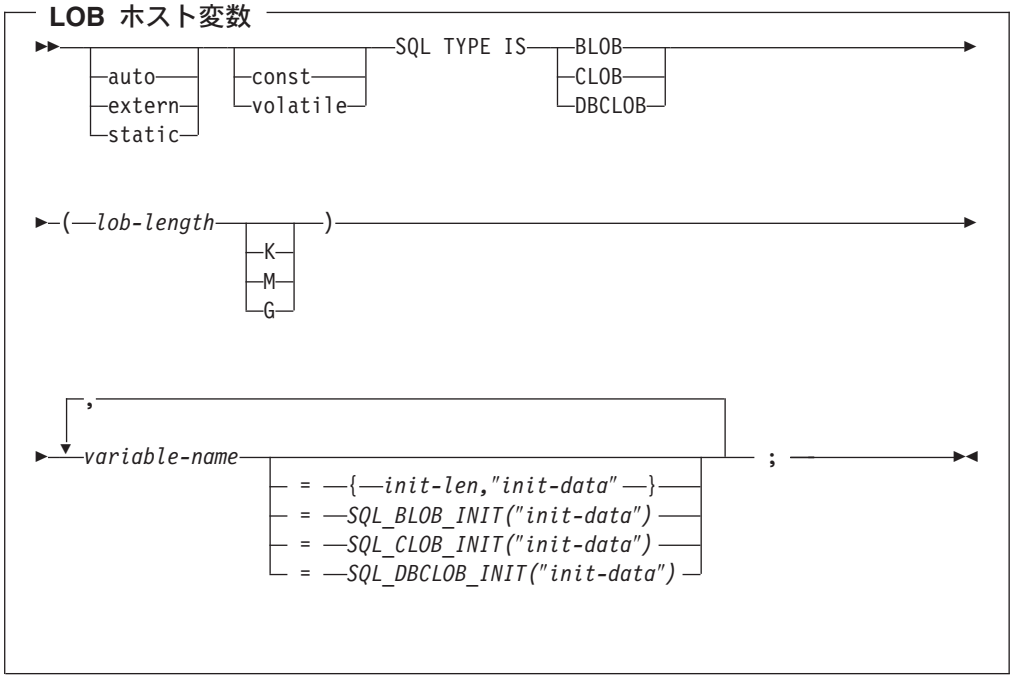
```
#pragma pack(1)
 struct VARGRAPH {
      short len;
      wchar_t s[10];
      } vstring;
#pragma pack()
```

例:

```
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;
```

SQL を使用する C および C++ アプリケーションでの LOB ホスト変数

C および C++ には、LOB (ラージ・オブジェクト) の SQL データ・タイプに対応する変数がありません。これらのデータ・タイプで使用するホスト変数を作成するには、SQL TYPE IS 文節を使用します。SQL プリコンパイラーは、この宣言を出力ソース・メンバー内で、C 言語構造に置き換えます。



注:

1. K は *lob-length* に 1024 を掛けます。 M は *lob-length* に 1,048,576 を掛けます。 G は *lob-length* に 1,073,741,824 を掛けます。
2. BLOB および CLOB の場合、1 <= *lob-length* <= 2,147,483,647 です。
3. DBCLOB の場合、1 <= *lob-length* <= 1,073,741,823 です。
4. SQL TYPE IS、BLOB、CLOB、DBCLOB、K、M、G は大文字小文字混合で構いません。
5. 初期設定文字列に使用できる最大長は 32,766 バイトです。
6. 初期設定の長さ *init-len* は、数値定数でなければなりません (すなわち、K、M、または G を含むことはできません)。
7. LOB の長さは指定しなければなりません。すなわち、以下の宣言は使用できません。

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

8. LOB が宣言内で初期設定されない場合は、初期設定は、プリコンパイラーが生成したコード内で行われません。
9. プリコンパイラーは、ホスト変数のタイプにキャストするために使用される構造体タグを生成します。
10. LOB ホスト変数へのポインターは、他のホスト変数タイプへのポインターの場合と同じ規則と制約事項を使用して宣言することができます。
11. LOB ホスト変数に対する CCSID 処理は、他の文字およびグラフィック・ホスト変数タイプに対する処理と同じです。
12. DBCLOB が初期設定されている場合は、文字ストリングの前に 'L' (ワイド文字ストリングを示す) を付けるのはユーザーの役割です。

BLOB の例

次のように宣言すると、

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

以下の構造を生成します。

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

CLOB の例

次のように宣言すると、

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

プリコンパイラーは C 言語用に以下を生成します。

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

DBCLOB の例

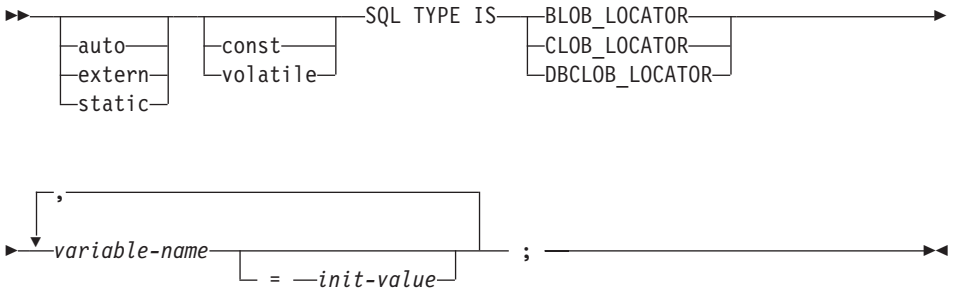
次のように宣言すると、

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

プリコンパイラーは以下を生成します。

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

LOB ロケーター



注:

1. SQL TYPE IS、BLOB_LOCATOR、CLOB_LOCATOR、DBCLOB_LOCATOR は、大文字小文字混合にすることができます。
2. *init-value* により、ポインター・ロケーター変数の初期設定が行えます。他のタイプの初期設定は意味がありません。

3. LOB ロケーターへのポインターは、他のホスト変数タイプへのポインターの場合と同じ規則と制約事項を使用して宣言することができます。

CLOB ロケーターの例

次のように宣言すると、

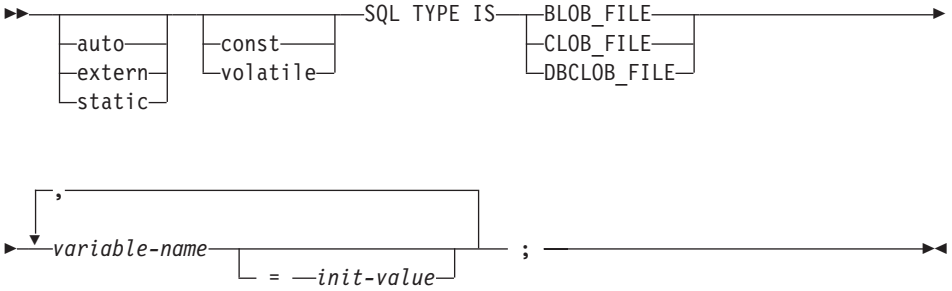
```
static SQL TYPE IS CLOB_LOCATOR my_locator;
```

以下の構造を生成します。

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

BLOB ロケーターおよび DBCLOB ロケーターの構文は、似ています。

LOB ファイル参照変数



注:

1. SQL TYPE IS、BLOB_FILE、CLOB_FILE、DBCLOB_FILE は大文字小文字混合で構いません。
2. LOB ファイル参照変数へのポインターは、他のホスト変数タイプへのポインターの場合と同じ規則と制約事項を使用して宣言することができます。

CLOB ファイル参照の例

次のように宣言すると、

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

以下の構造を生成します。

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

BLOB ファイル参照変数と DBCLOB ファイル参照変数は、類似の構文をもっています。

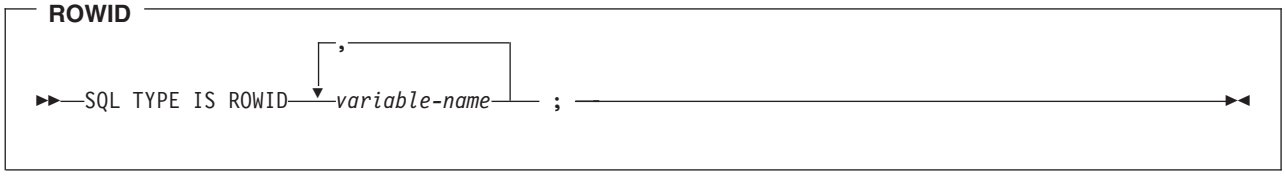
プリコンパイラーは、次のファイル・オプション定数に対する宣言を生成します。ファイル参照ホスト変数を使用する場合、これらの定数を使用して、file_options 変

数を設定できます。これらの値の詳細については、「SQL プログラミング 概念」の『LOB ファイル参照変数』を参照してください。

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

SQL を使用する C および C++ アプリケーションでの ROWID ホスト変数

C および C++ には、SQL データ・タイプ ROWID に対応する変数がありません。このデータ・タイプで使用するホスト変数を作成するには、SQL TYPE IS 文節を使用します。SQL プリコンパイラーは、この宣言を出力ソース・メンバー内で、C 言語構造に置き換えます。



注:

1. SQL TYPE IS ROWID は、大文字小文字混合にすることができます。

ROWID の例

次のように宣言すると、

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

以下の構造を生成します。

プリコンパイラーは C 言語用に以下を生成します。

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

SQL を使用する C および C++ アプリケーションでのホスト構造の使用

C および C++ プログラムの中では、**ホスト構造**が定義できます。これは一連の基本 C または C++ 変数に名前を付けたものです。ホスト構造はそれ自体が複数レベルの構造の中に置かれることがあっても、その最大レベルは 2 レベルまでです。可変長の文字列を宣言するときは、別の構造が必要になるので、この場合だけは例外です。

ホスト構造名は、その従属レベルに基本 C または C++ 変数の名前が指定されているグループ名にすることができます。以下に、例を示します。

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

この例で、b_st は基本項目 c1 と c2 から成るホスト構造の名前です。

構造名は、スカラー・リストを簡略に表記するために使用できますが、これは 2 レベルの構造の場合に限られます。ホスト変数は構造名で修飾することができます (たとえば、structure.field)。ホスト構造は 2 レベルに限定されます。(たとえば、上記のホスト構造の例では、SQL の中で a_st を参照することはできません。) 構造に中間レベルの構造を含めることはできません。上記の例では、a_st はホスト変数として使用することも、SQL ステートメントの中で参照することもできません。SQL データのホスト構造は一連のホスト変数に名前を付けたものと考えられ、レベルは 2 つあります。ホスト構造を定義しておけば、SQL ステートメントの中でいくつかのホスト変数 (ホスト構造を構成するホスト変数の名前) を個別に参照せずに一括して参照することができます。

たとえば、次のようにコーディングすれば、テーブル CORPDATA.EMPLOYEE から選択した行のすべての列の値を検索することができます。

```
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;
```

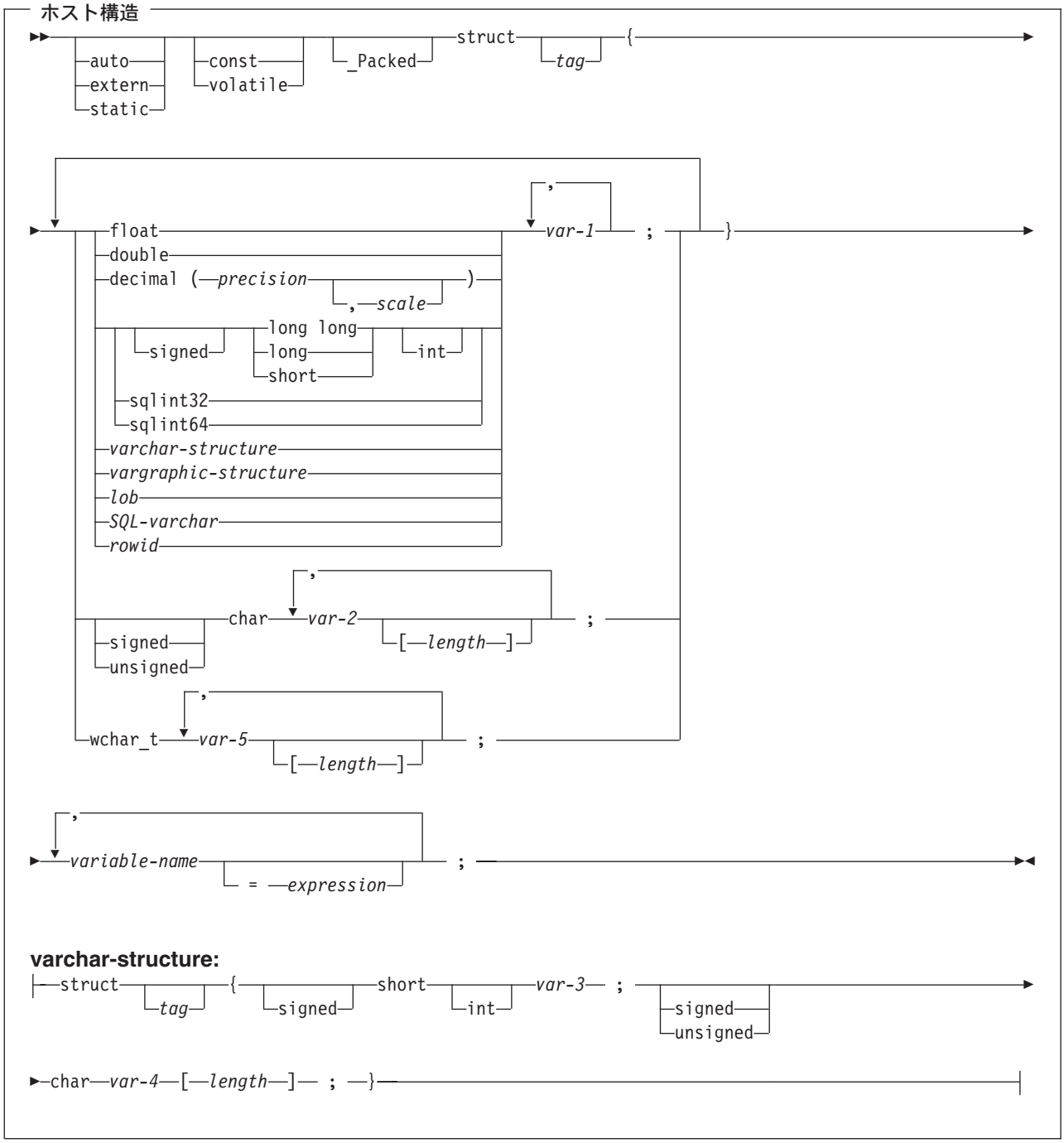
上の例に示すように、pemp1 の宣言には、2 つの可変長文字列要素、firstname と lastname が構造に含まれています。

詳細については、以下のセクションを参照してください。

- 33 ページの『SQL を使用する C および C++ アプリケーションでのホスト構造宣言』
- 35 ページの『SQL を使用する C および C++ アプリケーションでのホスト構造標識配列』

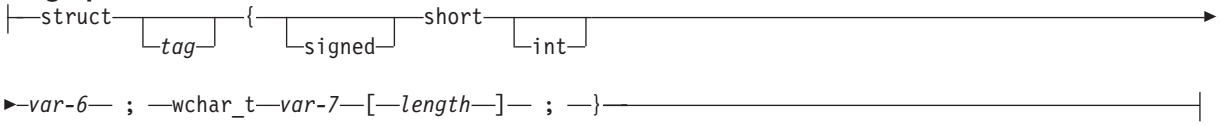
SQL を使用する C および C++ アプリケーションでのホスト構造宣言

次の図は、ホスト構造宣言に関する有効な構文を示しています。

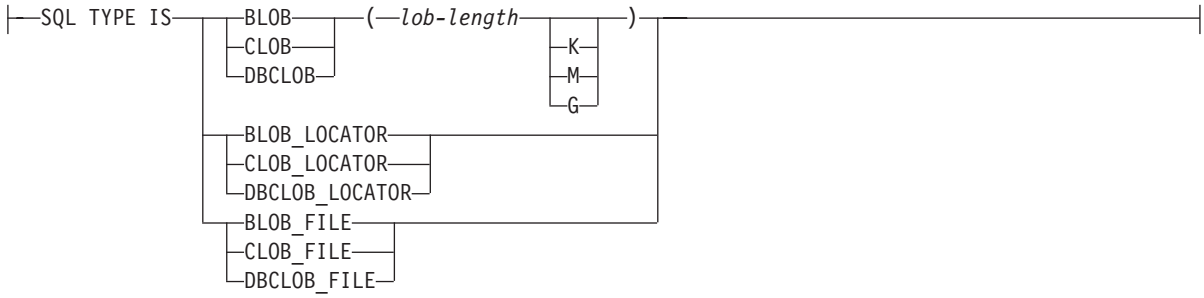


ホスト構造 (続き)

vargraphic-structure:



lob:



SQL-varchar:

rowid:

注:

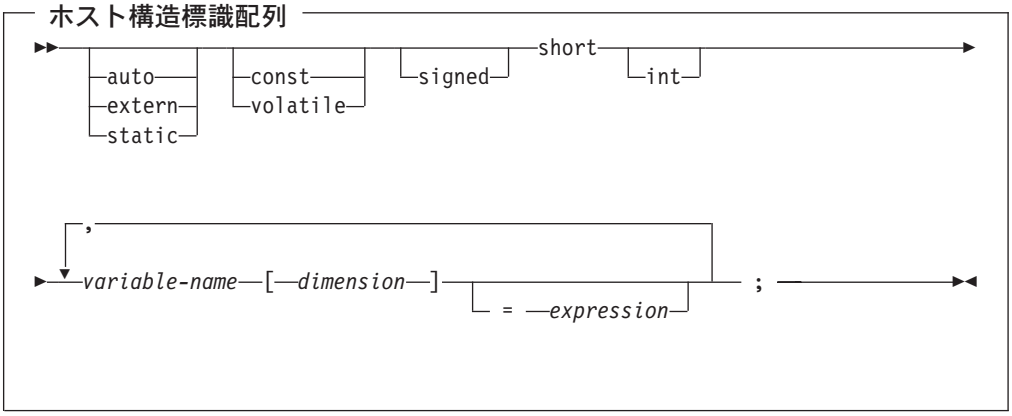
1. 数値ホスト変数、文字ホスト変数、グラフィック・ホスト変数、LOB ホスト変数、および ROWID ホスト変数の宣言の詳細については、数値ホスト変数、文字ホスト変数、グラフィック・ホスト変数、LOB ホスト変数、および ROWID ホスト変数に関する注意事項を参照してください。
2. char 配列または wchar_t 配列が続く short int の構造は、常に SQL C および C++ プリコンパイラーによって VARCHAR 構造または VARGRAPHIC 構造のいずれかとして解釈されます。
3. _Packed は、C++ では使用してはいけません。代わりに、宣言の前に #pragma pack(1) を、宣言の後に #pragma pack() を指定します。

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

4. sqlint32 または sqlint64 を使用する場合は、ヘッダー・ファイルの sqlsystm.h が組み込まれている必要があります。

SQL を使用する C および C++ アプリケーションでのホスト構造標識配列

次の図は、ホスト構造標識配列の有効な構文を示しています。



注: dimension (次元) は 1 から 32767 までの整数定数でなければなりません。

SQL を使用する C および C++ アプリケーションでのホスト構造配列の使用

C および C++ プログラムの中では、ホスト構造配列が定義できます。これには次元属性が付いています。ホスト構造配列はそれ自体が複数レベルの構造の中に置かれることがあっても、その最大レベルは 2 レベルまでです。可変長文字ストリングや可変長漢字ストリングを使用しないのであれば、別の構造は必要ありません。

次の C の例、

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

および次の C++ の例では、

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

以下の条件が該当します。

- b_array 内のすべてのメンバーは、有効な変数宣言でなければなりません。
- _Packed 属性は struct タグに指定しなければなりません。
- b_array は、メンバー c1_var およびメンバー c2_var が入っているホスト構造の配列の名前です。

- b_array はブロック化形式の FETCH ステートメントおよび INSERT ステートメントでのみ使用できます。
- c1_var および c2_var は、SQL ステートメントでは有効なホスト変数ではありません。
- 構造に中間レベルの構造を含めることはできません。

たとえば、C では以下のカーソルから 10 行を取り出すことができます。

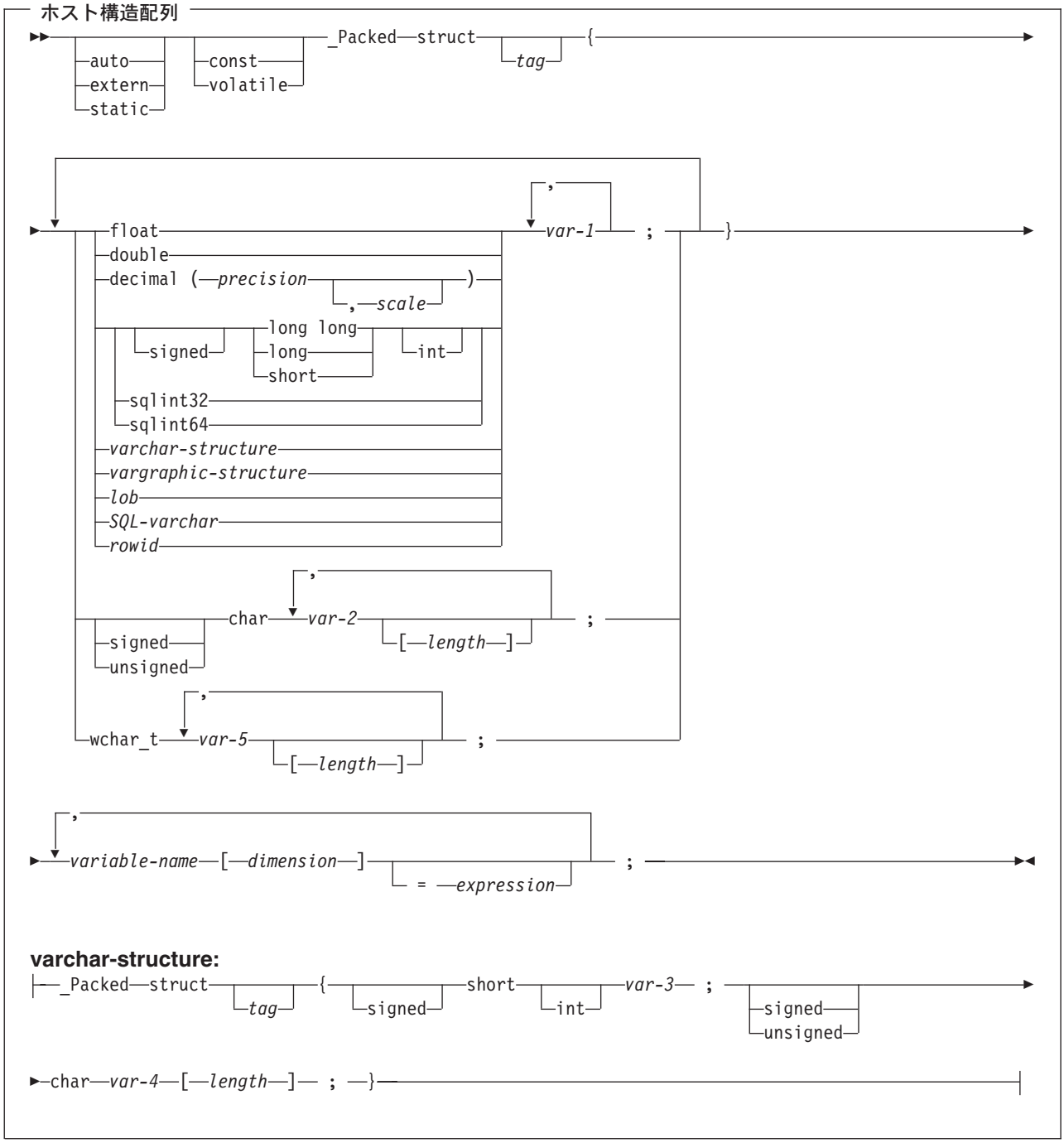
```
_Packed struct {char first_initial;
                char middle_initial;
                _Packed struct {short lastname_len;
                                char lastname_data[15];
                               } lastname;
                double total_salary;
               } employee_rec[10];
struct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

詳細については、以下のセクションを参照してください。

- 37 ページの『SQL を使用する C および C++ アプリケーションでのホスト構造配列』
- 39 ページの『SQL を使用する C および C++ アプリケーションでのホスト構造配列標識構造』

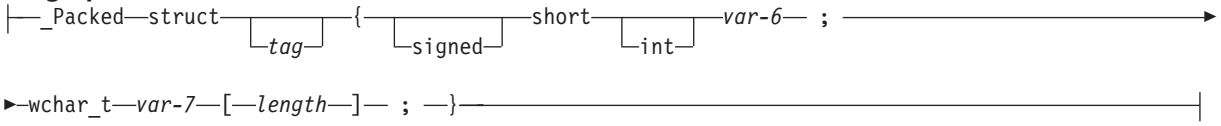
SQL を使用する C および C++ アプリケーションでのホスト構造配列

次の図は、ホスト構造配列宣言の有効な構文を示しています。

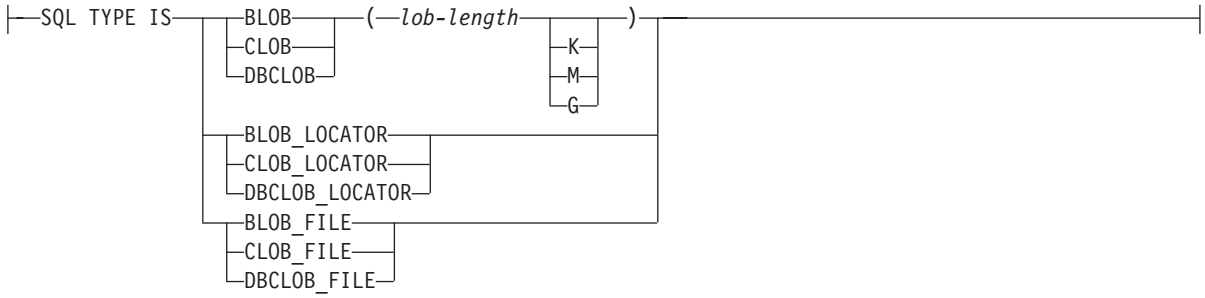


ホスト構造配列 (続き)

vargraphic-structure:



lob:



SQL-varchar:

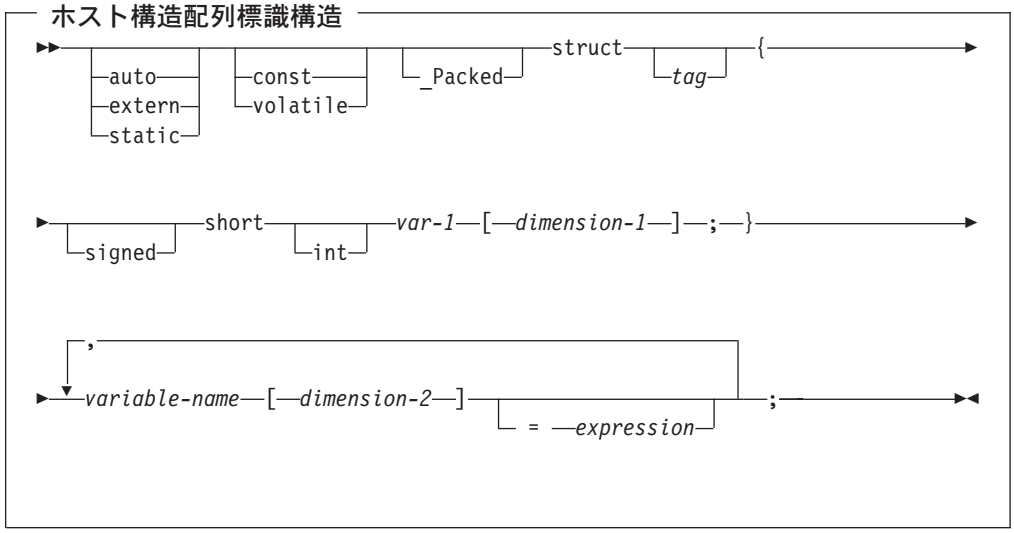
rowid:

注:

1. 数値ホスト変数、文字ホスト変数、グラフィック・ホスト変数、LOB ホスト変数、および ROWID ホスト変数の宣言の詳細については、数値ホスト変数、文字ホスト変数、グラフィック・ホスト変数、LOB ホスト変数、および ROWID ホスト変数に関する注意事項を参照してください。
2. struct (構造) タグを使用すると、他のデータ域が定義できますが、これらをホスト変数として使用することはできません。
3. dimension (次元) は 1 から 32767 までの整数定数でなければなりません。
4. _Packed は、C++ では使用してはいけません。代わりに、宣言の前に #pragma pack(1) を、宣言の後に #pragma pack() を指定します。
5. sqlint32 または sqlint64 を使用する場合は、ヘッダー・ファイルの sqlsystm.h が組み込まれている必要があります。

SQL を使用する C および C++ アプリケーションでのホスト構造配列標識構造

次の図は、ホスト構造配列標識構造宣言の有効な構文を示しています。



注:

1. struct (構造) タグは他のデータ域を定義するために使用できますが、ホスト変数として使用することはできません。
2. dimension-1 (次元 1) および dimension-2 (次元 2) はともに、1 から 32767 までの整数定数でなければなりません。
3. _Packed は、C++ では使用してはいけません。代わりに、宣言の前に #pragma pack(1) を、宣言の後に #pragma pack() を指定します。

SQL を使用する C および C++ アプリケーションでのポインター・データ・タイプの使用

サポートされる C および C++ データ・タイプを指すポインターとなるホスト変数も宣言することができますが、次のような制約があります。

- あるホスト変数をポインターとして宣言するときは、そのホスト変数は、ホスト変数の頭にアスタリスクを付けて宣言しなければなりません。次の例はいずれも有効です。

```
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.                             */
   char mydata[30];
  } *myvarchar;
```

注: 括弧は NUL 終了文字配列を指すポインターを宣言するときだけ許されますが、この場合、括弧は必須です。括弧を使用しないと、配列を指すために必要なポインターではなく、ポインターの配列を宣言することになります。以下に、例を示します。

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

- あるホスト変数をポインターとして宣言するときは、その変数を使って同じソース・ファイル内に他のホスト変数を宣言することはできません。たとえば、下に示す 2 番目の宣言は無効です。

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

- あるホスト変数を SQL ステートメントの中で参照するときは、そのホスト変数は宣言したとおりに参照しなければなりません。ただし、NUL 終了文字配列を指すポインターだけは例外です。たとえば、次の宣言には括弧が必要です。

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

  しかし、ホスト変数が SELECT のような SQL ステートメントの中で参照されるときは、括弧は許されません。

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

- アスタリスクだけがホスト変数名に対する演算子として使用できます。
- ホスト変数名の最大長は、アスタリスクも名前の一部として扱われるので、アスタリスクをいくつ指定したかによって影響されます。
- 構造を指すポインターは、可変長文字構造の場合を除き、ホスト変数として使用できません。また、構造の中のポインター・フィールドもホスト変数として使用できません。
- SQL は基底付きホスト変数のすべての指定記憶域を割り振るよう要求します。記憶域を割り振らないと、予期せぬ結果が生じる場合があります。

SQL を使用する C および C++ アプリケーションでの typedef の使用

typedef 宣言を使用して、short 型、float 型、double 型のような C 型指定子の代わりに使用される独自の識別コードを定義することもできます。ホスト変数の宣言に使用される typedef 識別コードは、typedef 宣言がそれぞれ別のブロックやプロシージャーの中にある場合であっても、1 つのプログラム内では固有になっていなければなりません。プログラムに BEGIN DECLARE SECTION ステートメントと END DECLARE SECTION ステートメントが含まれる場合、typedef 宣言が BEGIN DECLARE SECTION と END DECLARE SECTION と一緒に含まれる必要はありません。typedef 識別コードは BEGIN DECLARE SECTION 内で SQL プリコンパイラーによって認識されます。C および C++ プリコンパイラーは、typedef 宣言のサブセットだけを、ホスト変数宣言の場合と同様に認識します。

有効な typedef ステートメントの例:

- 長い typedef を宣言してから、typedef を参照するホスト変数を宣言します。

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

- 文字配列の長さは、 typedef またはホスト変数宣言のいずれかで指定されますが、両方では指定されません。

```
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 IS ステートメントは typedef で使用されることがあります。

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

- ストレージ・クラス (auto、extern、static)、volatile、または const 修飾子をホスト変数宣言で指定することができます。

```
typdef short INT_T;
typdef short INT2_T;
static INT_T i1;
volatile INT2_T i2;
```

- 構造体の typedef がサポートされます。

```
typedef _Packed struct {char dept[3];
                        char deptname[30];
                        long Num_employees;} DEPT_T;
DEPT_T dept_rec;
DEPT_T dept_array[20]l  /* use for blocked insert or fetch */
```

SQL を使用する C および C++ アプリケーションでの ILE C コンパイラー 外部ファイル記述の使用

C または C++ の #pragma mapinc ディレクティブを #include ディレクティブと一緒に使用すると、プログラムの中に外部ファイル記述を含めることができます。SQL で使用するときは、#pragma mapinc ディレクティブは特定の形式だけが、SQL プリコンパイラーによって受け付けられます。必要とされる要素のすべてが指定されていないと、プリコンパイラーはディレクティブを無視し、ホスト変数構造を生成しません。必要とされる要素とは、次のものです。

- 組み込む名前
- 外部記述ファイル名
- 形式名または形式名のリスト
- オプション
- 変換オプション

ライブラリー名、結合名、変換オプション、および接頭部名は任意選択です。ユーザーがコーディングした typedef ステートメントはプリコンパイラーに認識されませんが、#pragma mapinc ディレクティブと、#include ディレクティブによって生成された typedef ステートメントは認識されます。SQL はオプション・パラメーターの入力値、出力値、およびキー値をサポートします。変換オプションについては、サポートされている値は D、p、z、_P、および 1BYTE_CHAR です。これらのオプションは D と p の両方が指定できない場合を除き、どのような順序でも指定することができます。#pragma mapinc ディレクティブと #include ディレクティブによって生成された typedef 結合を使用して宣言された結合は、SQL ステートメントの中でホスト変数として使用することはできません。しかし、結合のメンバーは使用できます。typedef 構造を含む構造は SQL ステートメントの中では使用できません。しかし、typedef を使用して宣言した構造は使用できます。

「*DB2 UDB for iSeries SQL プログラミング 概念*」の『DB2 UDB for iSeries サンプル・テーブル』に記載されているサンプル・テーブル DEPARTMENT の定義を検索するときは、次のようにコーディングすることができます。

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

Dept_Structure という名前のホスト構造は、次の要素を使用して定義されています。すなわち、DEPTNO、DEPTNAME、MGRNO、および ADMRDEPT です。これらのフィールド名は、SQL ステートメントの中でホスト変数として使用できます。

注: DATE、TIME、および TIMESTAMP の各列からは、文字ホスト変数定義が生成されます。これらは SQL により、DATE、TIME、および TIMESTAMP 列と同じ比較および割り当て規則を使用して扱われます。たとえば、日付ホスト変数は、DATE 列または日付の有効な表現である文字ストリングとだけ突き合わされて比較されます。

GRAPHIC 列または VARGRAPHIC 列が UCS-2 CCSID を持つ場合、生成されるホスト変数には UCS-2 CCSID が割り当てられます。

ゾーン 10 進数、2 進数 (位取りフィールドがゼロでない)、および任意選択で 10 進数は、ILE C for iSeries では文字フィールドにマッピングされますが、SQL はこれらのフィールドを数値として扱います。拡張プログラム・モデル (EPM) ルーチンを使用してこれらのフィールドがゾーンおよびパック 10 進数データを変換するように操作できます。詳細については、「ILE C for iSeries™ Language Reference」  を参照してください。

SQL データ・タイプおよび C または C++ データ・タイプの対応関係の判別

プリコンパイラーは、次の表に基づいて、ホスト変数のベース SQLTYPE とベース SQLLEN を判断します。ホスト変数が標識変数と一緒に記載されているときは、その SQLTYPE はベース SQLTYPE に 1 を加えたものです。

表 1. C または C++ 宣言と代表的 SQL データ・タイプとの対応関係

C または C++ データ・タイプ	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
short int	500	2	SMALLINT
long int	496	4	INTEGER
long long int	492	8	BIGINT
decimal(p,s)	484	バイト 1 には p、バイト 2 には s	DECIMAL (p,s)
float	480	4	FLOAT (単精度)
double	480	8	FLOAT (倍精度)
単一文字形式	452	1	CHAR(1)
NUL 終了文字形式	460	length	VARCHAR (長さ - 1)
VARCHAR 構造化形式 (長さ < 255)	448	length	VARCHAR (長さ)
VARCHAR 構造化形式 (長さ > 254)	456	length	VARCHAR (長さ)
単一グラフィック形式	468	1	GRAPHIC (1)
NUL 終了単一グラフィック形式	400	length	VARGRAPHIC (長さ - 1)

表 1. C または C++ 宣言と代表的 SQL データ・タイプとの対応関係 (続き)

C または C++ データ・タイプ	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
VARGRAPHIC 構造化形式 (長さ < 128)	464	length	VARGRAPHIC (長さ)
VARGRAPHIC 構造化形式 (長さ > 127)	472	length	VARGRAPHIC (長さ)

下表を参照すると、各 SQL データ・タイプに対応する C または C++ データ・タイプを判別することができます。

表 2. SQL データ・タイプと代表的な C または C++ 宣言との対応関係

SQL データ・タイプ	C または C++ データ・タイプ	注
SMALLINT	short int	
INTEGER	long int	
BIGINT	long long int	
DECIMAL(p,s)	decimal(p,s)	p は 1 から 31 までの正の整数。 s は 0 から 31 までの正の整数。
NUMERIC(p,s) または非ゼロ位取り 2 進数	正確に対応するものなし	decimal(p,s) を使用。
FLOAT (単精度)	float	
FLOAT (倍精度)	double	
CHAR(1)	単一文字形式	
CHAR(n)	正確に対応するものなし	*n* > 1 なら、NUL 終了グラフィック形式を使用。
VARCHAR(n)	NUL 終了文字形式	NUL 終了文字を受け入れるには少なくとも *n* +1 が必要。データに NUL (\0) を含めることができる場合は、VARCHAR 構造化形式または SQL VARCHAR を使用。 *n* は正の整数です。 *n* の最大値は 32740。
	VARCHAR 構造化形式	*n* の最大値は 32740。 SQL VARCHAR 形式も使用可能。
BLOB	なし	C または C++ で BLOB を宣言するために SQL TYPE IS を使用。
CLOB	なし	C または C++ で CLOB を宣言するために SQL TYPE IS を使用。
GRAPHIC (1)	単一グラフィック形式	
GRAPHIC (n)	正確に対応するものなし	*n* > 1 なら、NUL 終了グラフィック形式を使用。

表 2. SQL データ・タイプと代表的な C または C++ 宣言との対応関係 (続き)

SQL データ・タイプ	C または C++ データ・タイプ	注
VARGRAPHIC(n)	NUL 終了グラフィック形式	データにグラフィック NUL 値 (/0/0) を含めることができる場合は、VARGRAPHIC 構造化形式を使用。 NUL 終了文字を受け入れるには少なくとも *n* +1 が必要。 *n* は正の整数です。 *n* の最大値は 16370。
	VARGRAPHIC 構造化形式	*n* は正の整数です。 *n* の最大値は 16370。
DBCLOB	なし	C または C++ で DBCLOB を宣言するために SQL TYPE IS を使用。
DATE	NUL 終了文字形式	形式が *USA、*ISO、*JIS、または *EUR のときは、NUL 終了文字を受け入れるには少なくとも 11 文字が必要。形式が *MDY、*YMD、または *DMY のときは、NUL 終了文字を受け入れるには少なくとも 9 文字が必要。形式が *JUL のときは、NUL 終了文字を受け入れるには少なくとも 7 文字が必要。
	VARCHAR 構造化形式	形式が *USA、*ISO、*JIS、または *EUR のときは、少なくとも 10 文字が必要。形式が *MDY、*YMD、または *DMY のときは、少なくとも 8 文字が必要。形式が *JUL のときは、少なくとも 6 文字が必要。
TIME	NUL 終了文字形式	NUL 終了文字を受け入れるには少なくとも 7 文字 (秒を含む場合は、9 文字) が必要。
	VARCHAR 構造化形式	少なくとも 6 文字が必要。秒を含む場合は、8 文字が必要。

表 2. SQL データ・タイプと代表的な C または C++ 宣言との対応関係 (続き)

SQL データ・タイプ	C または C++ データ・タイプ	注
TIMESTAMP	NUL 終了文字形式	NUL ターミネーターを受け入れるには、少なくとも 20 文字 (マイクロ秒を全桁の精度で含める場合 27 文字) は必要。 n が 27 未満のときは、マイクロ秒部分で切り捨てが起こる。
	VARCHAR 構造化形式	少なくとも 19 文字が必要。マイクロ秒を全桁の精度で含める場合は、26 文字必要。数値数が 26 未満の場合は、マイクロ秒部分で切り捨てが起こる。
DATALINK	サポートなし	
ROWID	なし	C または C++ で ROWID を宣言するために SQL TYPE IS を使用。

詳細については、『C および C++ 変数宣言と使用法についての注意』を参照してください。

C および C++ 変数宣言と使用法についての注意

アポストロフィと引用符は、C、C++ および SQL の間で意味が異なります。C および C++ では、引用符は文字列定数を区切るために、アポストロフィは文字定数を区切るために使用されます。SQL では、この区別がなく、引用符は識別名を区切るために、アポストロフィは文字ストリング定数を区切るために使用されます。SQL での文字データは、整数データと区別されています。

SQL を使用する C および C++ アプリケーションでの標識変数の使用

標識変数は 2 バイトの整数 (short int) です。ホスト構造をサポートするために標識構造 (ハーフワードの整数変数の配列として定義されているもの) を指定することもできます。検索されるとき、標識変数はその対応するホスト変数にヌル値が割り当てられているかどうかを示すために使用されます。列に割り当てるときには、ヌル値を割り当てるべきであることを示すために負の標識変数が使用されます。

詳細については、「SQL 解説書」の『標識変数』を参照してください。

標識変数の宣言の仕方は、ホスト変数の場合と同じです。これらの 2 つの変数の宣言は、適切な方法で組み合わせることができます。

例:

次のステートメントがあるとします。

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

変数は次のように宣言することができます。

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

第 3 章 COBOL アプリケーションでの SQL ステートメントのコーディング方法

iSeries では、複数の COBOL コンパイラーがサポートされます。ライセンス・プログラムである DB2 UDB SQL 開発キットがサポートする言語は、 COBOL for iSeries と ILE COBOL for iSeries だけです。この章では、SQL ステートメントを COBOL プログラムに組み込む場合に固有のアプリケーションおよびコーディング上の要件について説明します。ホスト構造およびホスト変数に関する要件についても説明します。

詳細については、以下のセクションを参照してください。

- 『SQL を使用する COBOL アプリケーションでの SQL 連絡域の定義』
- 49 ページの『SQL を使用する COBOL アプリケーションでの SQL 記述子域の定義』
- 50 ページの『SQL を使用する COBOL アプリケーションでの SQL ステートメントの組み込み』
- 53 ページの『SQL を使用する COBOL アプリケーションでのホスト変数の使用』
- 62 ページの『SQL を使用する COBOL アプリケーションでのホスト構造の使用』
- 71 ページの『SQL を使用する COBOL アプリケーションでの外部ファイル記述の使用』
- 72 ページの『SQL データ・タイプと COBOL データ・タイプの対応関係の判別』
- 76 ページの『SQL を使用する COBOL アプリケーションでの標識変数の使用』

SQL ステートメントの使い方を示した詳しいサンプル COBOL プログラムは、161 ページの『付録 A. DB2 UDB for iSeries ステートメントを使用するサンプル・プログラム』に記載されています。

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

SQL を使用する COBOL アプリケーションでの SQL 連絡域の定義

SQL ステートメントが入っている COBOL プログラムは、次のいずれかまたは両方を含んでいなければなりません。

- PICTURE S9(9) BINARY、PICTURE S9(9) COMP-4、または PICTURE S9(9) COMP として宣言されている SQLCODE 変数。
- PICTURE X(5) として宣言されている SQLSTATE 変数

または、

- SQLCA (SQLCODE および SQLSTATE 変数が入っている)

SQLCODE 値および SQLSTATE 値は、各 SQL ステートメントが実行された後、データベース・マネージャーによってセットされます。アプリケーションは、SQLCODE 値または SQLSTATE 値を調べて、最後の SQL ステートメントが正しく実行されたかどうかを判定することができます。

SQLCA は、直後または、SQL の INCLUDE ステートメントの使用によって、COBOL プログラムの中にコーディングすることができます。 SQL の INCLUDE ステートメントを使用するときは、次のような標準の宣言を含める必要があります。

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

SQLCODE、SQLSTATE、および SQLCA の各変数宣言は、ユーザーのプログラムの WORKING-STORAGE SECTION または LINKAGE SECTION に現れる必要があります。これらの変数宣言を置く場所は、これらのセクションにレコード記述項目を指定できる個所ならば、どこでも構いません。

INCLUDE ステートメントを使用すると、SQL の COBOL プリコンパイラーは、SQLCA 用の COBOL ソース・ステートメントを組み込みます。

```
01 SQLCA.
   05 SQLCAID        PIC X(8).
   05 SQLCABC        PIC S9(9) BINARY.
   05 SQLCODE        PIC S9(9) BINARY.
   05 SQLERRM.
      49 SQLERRML    PIC S9(4) BINARY.
      49 SQLERRMC    PIC X(70).
   05 SQLERRP        PIC X(8).
   05 SQLERRD        OCCURS 6 TIMES
                     PIC S9(9) BINARY.
   05 SQLWARN.
      10 SQLWARN0    PIC X.
      10 SQLWARN1    PIC X.
      10 SQLWARN2    PIC X.
      10 SQLWARN3    PIC X.
      10 SQLWARN4    PIC X.
      10 SQLWARN5    PIC X.
      10 SQLWARN6    PIC X.
      10 SQLWARN7    PIC X.
      10 SQLWARN8    PIC X.
      10 SQLWARN9    PIC X.
      10 SQLWARNA    PIC X.
   05 SQLSTATE       PIC X(5).
```

ILE COBOL for iSeries の場合、SQLCA は GLOBAL 文節を使用して宣言されます。SQLCODE の宣言がプログラムの中にあって、SQLCA がプリコンパイラーによって与えられるとき、SQLCODE の個所は SQLCADE で置き換えられます。SQLSTATE の宣言がプログラムの中にあって、SQLCA がプリコンパイラーによって与えられるとき、SQLSTATE の個所は SQLSTOTE で置き換えられます。

SQLCA の詳細については、「SQL 解説書」の『SQL 連絡域』を参照してください。

SQL を使用する COBOL アプリケーションでの SQL 記述子域の定義

SQLDA を必要とするステートメントには、次のものがあります。

EXECUTE...USING DESCRIPTOR *記述子名*

FETCH...USING DESCRIPTOR *記述子名*

OPEN...USING DESCRIPTOR *記述子名*

CALL...USING DESCRIPTOR *記述子名*

DESCRIBE *ステートメント名* INTO *記述子名*

DESCRIBE TABLE *ホスト変数* INTO *記述子名*

PREPARE *ステートメント名* INTO *記述子名*

SQLCA とは違って SQLDA はプログラムの中に 2 つ以上置くことができます。その SQLDA の名前は有効であればどの名前でも使えます。 SQLDA は、COBOL プログラムの中で直接コーディングすることもできますが、INCLUDE ステートメントを使って追加することもできます。 SQL の INCLUDE ステートメントを使用するときは、標準の SQLDA 宣言を組み込む必要があります。

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

SQLDA 用として組み込まれる COBOL 宣言は次のとおりです。

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

図 1. COBOL 用の INCLUDE SQLDA 宣言

SQLDA の宣言は、プログラムの WORKING-STORAGE SECTION か LINKAGE SECTION に置かなければなりません。置く場所は、レコード記述項目をこれらのセクションに指定できるならば、どこでも構いません。 ILE COBOL for iSeries の場合、SQLDA は GLOBAL 文節を使用して宣言されます。

動的 SQL は高度なプログラミング技法です。これについては、「*DB2 UDB for iSeries SQL プログラミング 概念*」の『動的 SQL アプリケーション』で説明します。動的 SQL を使用すると、ユーザーのプログラムはその実行と平行して SQL ステートメントを作成し、実行させることができます。動的に実行される変数 SELECT リスト (すなわち、照会の一部として返されるデータのリスト) を指定す

る SELECT ステートメントには、SQL 記述子域 (SQLDA) が必要です。これは、SELECT の結果を受け入れるために割り振るべき変数の数とタイプが事前に予測できないからです。

SQLDA の詳細については、「SQL 解説書」の『SQL 連絡域』を参照してください。

SQL を使用する COBOL アプリケーションでの SQL ステートメントの組み込み

SQL ステートメントは COBOL プログラム・セクションに次のようにコーディングすることができます。

SQL ステートメント	プログラム・セクション
BEGIN DECLARE SECTION END DECLARE SECTION DECLARE VARIABLE DECLARE STATEMENT	WORKING-STORAGE SECTION または LINKAGE SECTION
INCLUDE SQLCA INCLUDE SQLDA	WORKING-STORAGE SECTION または LINKAGE SECTION
INCLUDE メンバー名	DATA DIVISION または PROCEDURE DIVISION
その他	PROCEDURE DIVISION

COBOL プログラムの中の各 SQL ステートメントは、EXEC SQL で始まり、END-EXEC で終わっていなければなりません。 SQL ステートメントが 2 つの COBOL ステートメントの間に置かれるときは、ピリオドは省略できますが、ピリオドがない方がよい場合があります。EXEC SQL キーワードはいずれも 1 行に置かなければなりませんが、ステートメントの残りの部分は次行とそれ以降の行に継続させることができます。

例:

COBOL プログラムの中にコーディングされる UPDATE ステートメントをコーディングすると、次のようになります。

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

詳細については、以下のセクションを参照してください。

- 51 ページの『SQL を使用する COBOL アプリケーションでの注記』
- 51 ページの『SQL を使用する COBOL アプリケーションでの SQL ステートメントの継続』
- 51 ページの『SQL を使用する COBOL アプリケーションでのコードの組み込み』
- 52 ページの『SQL を使用する COBOL アプリケーションでのマージン』
- 52 ページの『SQL を使用する COBOL アプリケーションでの順序番号』

- 52 ページの『SQL を使用する COBOL アプリケーションでの名前』
- 52 ページの『SQL を使用する COBOL アプリケーションでの COBOL コンパイル時オプション』
- 52 ページの『SQL を使用する COBOL アプリケーションでのステートメント・ラベル』
- 52 ページの『SQL を使用する COBOL アプリケーションでの WHENEVER ステートメント』
- 53 ページの『複数ソース COBOL プログラムおよび SQL COBOL プリコンパイラー』

SQL を使用する COBOL アプリケーションでの注記

SQL の注記 (--) の他に、組み込み SQL ステートメントの中に COBOL の注記行 (7 列目が * または / の行) を組み込むことができます。ただし、キーワードの EXEC と SQL の間には入れられません。COBOL のデバッグ行 (7 行目が D の行) は、プリコンパイラーにより注記行として扱われます。

SQL を使用する COBOL アプリケーションでの SQL ステートメントの継続

SQL ステートメントの場合の行継続の規則は、EXEC SQL を 1 行以内で指定する必要がある点を除けば、他の COBOL ステートメントの場合と同じです。

文字列定数をある行から次の行に継続させる場合は、2 番目の行の最初の非ブランク文字はアポストロフィか引用符でなければなりません。区切り文字付き識別コードをある行から次の行に継続させる場合は、2 番目の行の最初の非ブランク文字はアポストロフィか引用符でなければなりません。

DBCS データを含む定数は、継続される行の 72 桁目にシフトイン文字を入れ、継続行の最初の文字列区切り文字のあとにシフトアウト文字を入れることによって、複数行にわたって継続させることができます。

この SQL ステートメントの G'<AABBCCDDEEFFGGHHIIJJKK>' はグラフィック定数として有効です。重複しているシフトは除去されます。

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

SQL を使用する COBOL アプリケーションでのコードの組み込み

SQL ステートメントまたは COBOL ホスト変数宣言ステートメントは、これらのステートメントが組み込まれるソース・コード内の個所に次の SQL ステートメントを組み込むことによって、挿入することができます。

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

COBOL の COPY ステートメントは、SQL ステートメントまたは SQL ステートメントの中で参照される COBOL ホスト変数の宣言を組み込むためには使用できません。

SQL を使用する COBOL アプリケーションでのマージン

SQL ステートメントは 12 桁目から 72 桁目までにコーディングします。 EXEC SQL が指定したマージンの前 (すなわち、12 桁目の前) から始まっているときは、SQL プリコンパイラーはそのステートメントを認識しません。

SQL を使用する COBOL アプリケーションでの順序番号

SQL プリコンパイラーによって生成されるソース・ステートメントは、 SQL ステートメントと同じ順序番号を使用して生成されます。

SQL を使用する COBOL アプリケーションでの名前

有効な COBOL 変数名ならば、どのような名前でもホスト変数に使用できますが、次のような制約を受けます。

'SQL'、'RDI'、または 'DSN' で始まるホスト変数名や外部入り口名は、使用してはなりません。これらの名前はデータベース・マネージャー用に予約されています。

FILLER が含まれる構造を使用すると、 SQL ステートメントで、期待されるような動作をしないことがあります。 COBOL 構造内のフィールドはすべて、予期しない結果を避けるように指定することをお勧めします。

SQL を使用する COBOL アプリケーションでの COBOL コンパイル時オプション

COBOL PROCESS ステートメントを使用すると、COBOL コンパイラーに対するコンパイル時オプションが指定できます。 PROCESS ステートメントは、プログラムを作成するためにプリコンパイラーによって呼び出されるとき、COBOL コンパイラーによって認識されます。しかし、SQL プリコンパイラー自体は PROCESS ステートメントを認識しません。そのために、APOST や QUOTE のような COBOL ソース・プログラムの構文に影響を与えるオプションは、PROCESS 構文では指定してはなりません。その代わり、*APOST と *QUOTE を CRTSQLCBL と CRTSQLCBLI の各コマンドの OPTION パラメーターで指定する必要があります。

SQL を使用する COBOL アプリケーションでのステートメント・ラベル

PROCEDURE DIVISION の中の実行可能な SQL ステートメントは、その前に段落名を置くことができます。

SQL を使用する COBOL アプリケーションでの WHENEVER ステートメント

SQL WHENEVER ステートメントの中で GOTO 文節の対象となるものは、PROCEDURE DIVISION の中のセクション名または非修飾の段落名でなければなりません。

複数ソース COBOL プログラムおよび SQL COBOL プリコンパイラー

SQL COBOL プリコンパイラーは、PROCESS ステートメントで区切った複数のソース・プログラムのプリコンパイルをサポートしません。

SQL を使用する COBOL アプリケーションでのホスト変数の使用

SQL ステートメントの中で使用するホスト変数はいずれも明示的に宣言しなければなりません。 SQL ステートメントの中で使用するホスト変数は、SQL ステートメントの中でホスト変数を初めて使用する前に、宣言しておかなければなりません。

ホスト変数を定義するために使用される COBOL ステートメントは、その前に BEGIN DECLARE SECTION ステートメントを置き、その後に END DECLARE SECTION ステートメントを置く必要があります。 BEGIN DECLARE SECTION と END DECLARE SECTION を指定した場合、SQL ステートメントで使用するすべてのホスト変数宣言は、BEGIN DECLARE SECTION ステートメントと END DECLARE SECTION ステートメントの間になければなりません。

SQL ステートメントの中のホスト変数はいずれも、その前にコロン (:) を付けなければなりません。

ホスト変数はレコードまたは要素にすることはできません。

COBOL ホスト変数名の中でダッシュを使えるようにするには、負符号の前後にブランクを置かなければなりません。

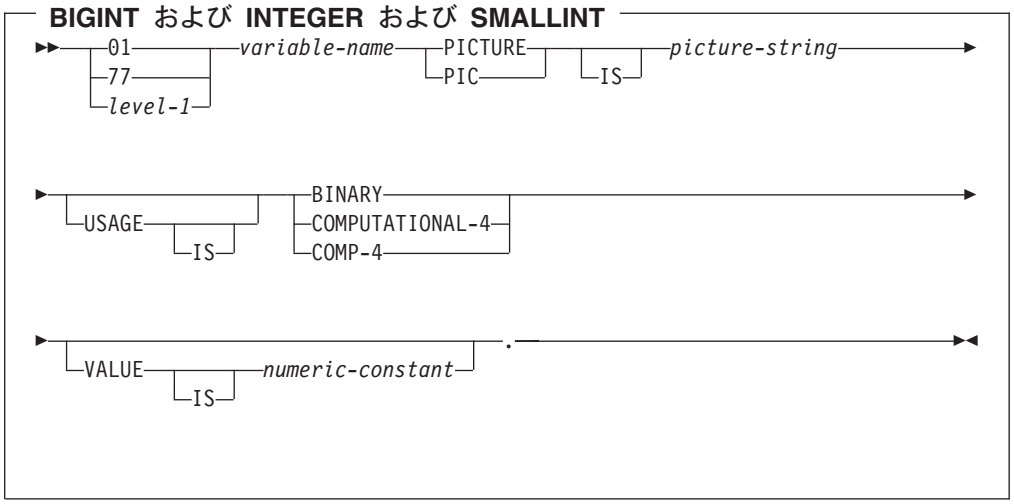
詳細については、『SQL を使用する COBOL アプリケーションでのホスト変数の宣言』を参照してください。

SQL を使用する COBOL アプリケーションでのホスト変数の宣言

COBOL プリコンパイラーは、有効な COBOL 宣言のサブセットだけを有効なホスト変数宣言として認めます。

SQL を使用する COBOL アプリケーションでのホスト変数の数値

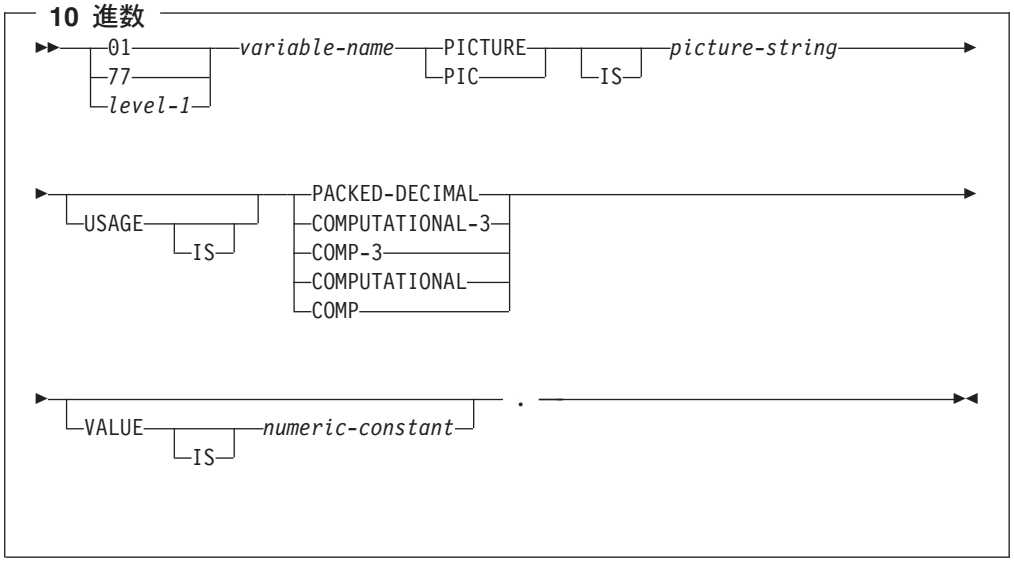
次の図は、有効な整数ホスト変数宣言の構文を示しています。



注:

1. BINARY、COMPUTATIONAL-4、および COMP-4 は同じ働きをします。他のシステムでも実行されるようなアプリケーションの場合には BINARY をコーディングしておくべきです。COMPUTATIONAL-4 と COMP-4 は IBM の拡張機能であり、国際標準化機構 (ISO)/ANSI COBOL ではサポートされないからです。これらのタイプに関連する *picture-string (ピクチャー列)* は S9(i)V9(d) (または 9 のインスタンスが *i* 回および *d* 回現れる S9...9V9...9) の形式になっていなければなりません。 i + d は 18 以下でなければなりません。
2. level-1 (レベル 1) は 2 から 48 までの COBOL レベルを示します。

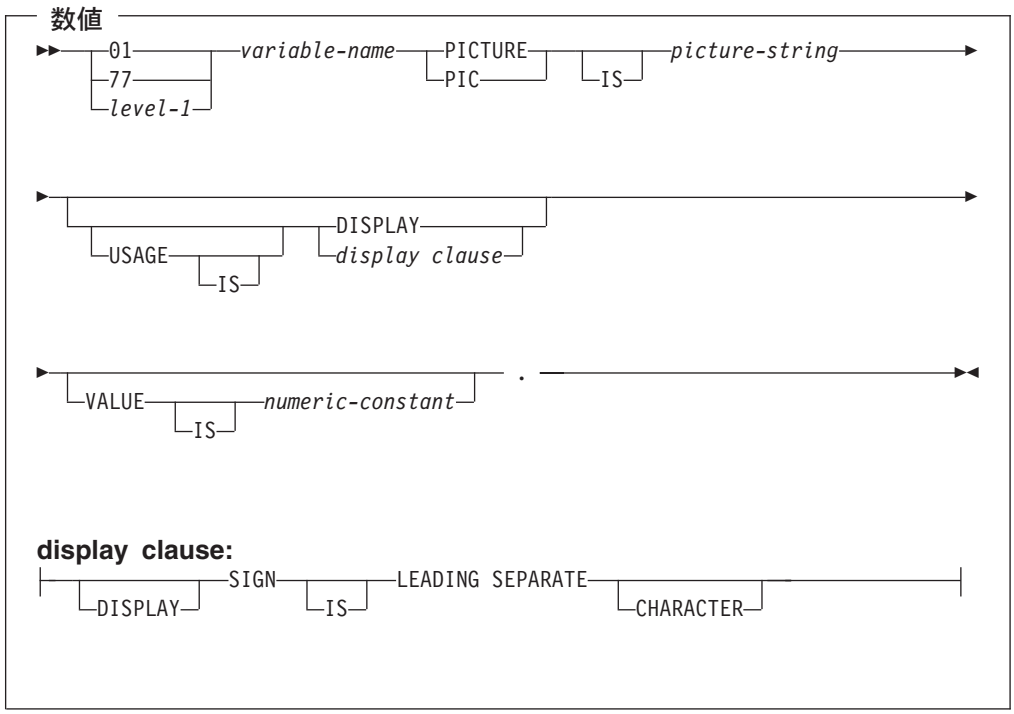
次の図は、有効な 10 進数ホスト変数宣言の構文を示しています。



注:

1. PACKED-DECIMAL、COMPUTATIONAL-3、および COMP-3 は同じ働きをします。他のシステムでも実行されるようなアプリケーションの場合には PACKED-DECIMAL をコーディングしておくべきです。COMPUTATIONAL-3 と COMP-3 は IBM の拡張機能であり、ISO/ANS COBOL ではサポートされないからです。これらのタイプに関連する *picture-string (ピクチャー列)* は S9(i)V9(d) (または 9 のインスタンスが *i* 回および *d* 回現れる S9...9V9...9) の形式になっていなければなりません。 i + d は 18 以下でなければなりません。
2. COMPUTATIONAL と COMP は同じ働きをします。これらのデータ・タイプとそれが表すデータ・タイプに関連するピクチャー列はプロダクト固有になっています。したがって、COMP と COMPUTATIONAL は他のシステムでも実行されるようなアプリケーションの場合には使用してはなりません。 COBOL for iSeries プログラムでは、これらのタイプに関連する *picture-string (ピクチャー列)* は S9(i)V9(d) (または 9 のインスタンスが *i* 回および *d* 回現れる S9...9V9...9) の形式になっていなければなりません。 i + d は 18 以下でなければなりません。
3. level-1 (レベル 1) は 2 から 48 までの COBOL レベルを示します。

下図は、有効な数値ホスト変数宣言の構文を示しています。

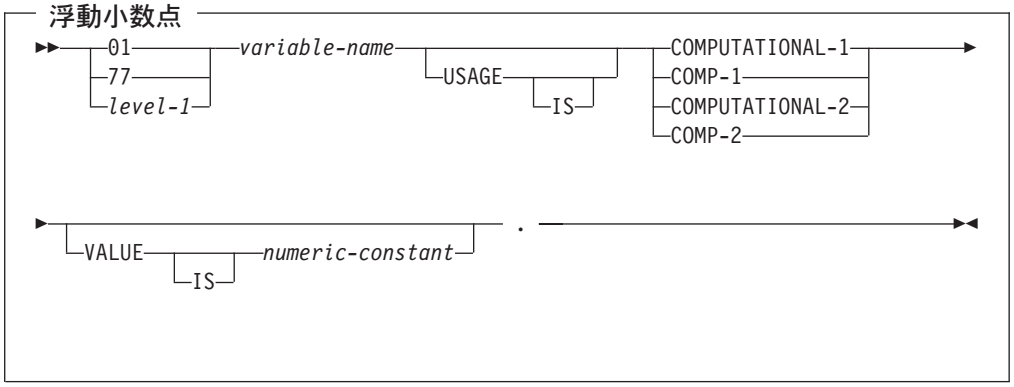


注:

1. SIGN LEADING SEPARATE および DISPLAY に関連する *picture-string (ピクチャー列)* は S9(i)V9(d) (または 9 のインスタンスが *i* 回および *d* 回現れる S9...9V9...9) の形式になっていなければなりません。i + d は 18 以下でなければなりません。
2. level-1 (レベル 1) は 2 から 48 までの COBOL レベルを示します。

SQL を使用する COBOL アプリケーションでの浮動小数点ホスト変数

次の図は、有効浮動小数点ホスト変数の宣言の構文を示しています。浮動小数点ホスト変数は、ILE COBOL for iSeries のみでサポートされます。



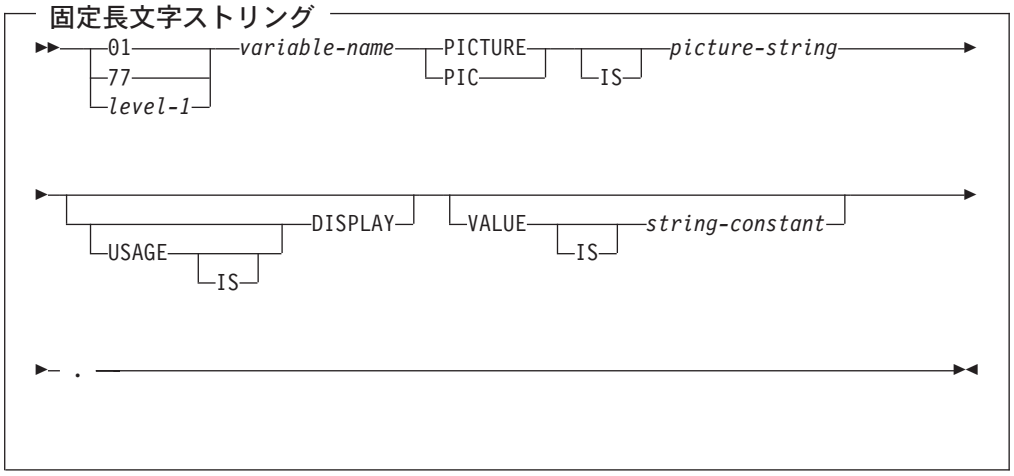
注:

1. COMPUTATIONAL-1 と COMP-1 は同じ働きをします。 COMPUTATIONAL-2 と COMP-2 は同じ働きをします。
2. level-1 (レベル 1) は 2 から 48 までの COBOL レベルを示します。

SQL を使用する COBOL アプリケーションでの文字ホスト変数

文字ホスト変数には、次の 2 つの形式があります。

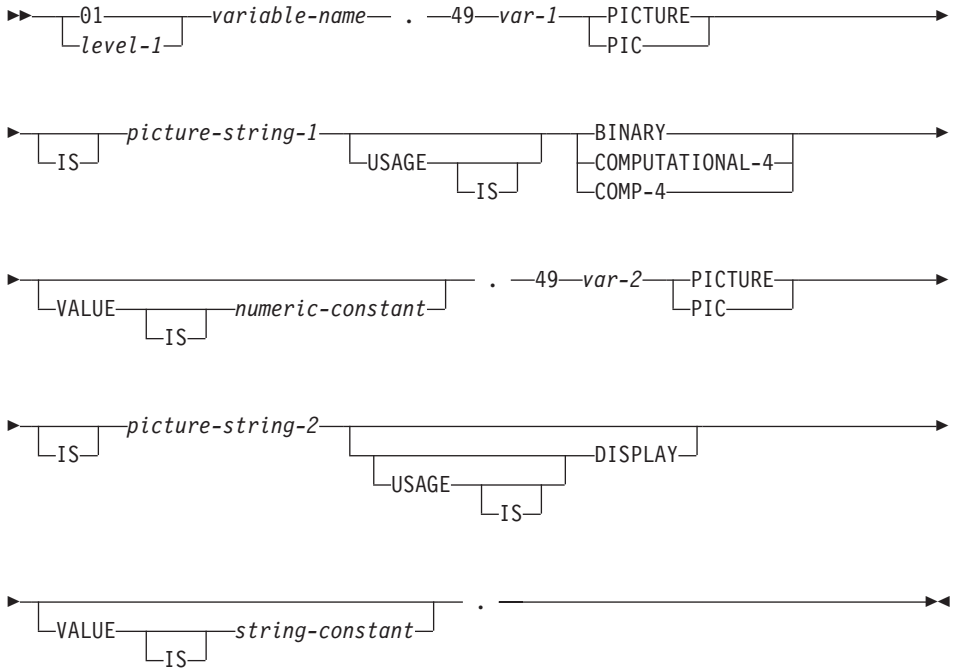
- 固定長文字列
- 可変長文字列



注:

1. これらの形式に関連する *picture-string (ピクチャー列)* は X(m) (または X のインスタンスが m 回現れる XXX...X) で、1 ≤ m ≤ 32 766 になっていなければなりません。
2. level-1 (レベル 1) は 2 から 48 までの COBOL レベルを示します。

可変長文字ストリング



注:

1. これらの形式に関連する *picture-string-1 (ピクチャー列 1)* は、 S9(m) または 9 のインスタンスが m 回現れる S9...9 でなければなりません。m は、1 から 4 までででなければなりません。

   iSeries 上の COBOL で値が指定の精度までしか認識されない場合があっても、データベース・マネージャーは S9(m) 変数の全桁を使用することに注意してください。この結果、COBOL ステートメントの実行中にデータ打ち切りエラーが生じることになるので、可変長文字列の最大長が事実上、指定の精度までに制限される場合があります。
2. これらの形式に関連する *picture-string-2 (ピクチャー列 2)* は X(m) または X のインスタンスが m 回現れる XX...X であって、 1 ≤ m ≤ 32 740 でなければなりません。
3. *var-1 (変数 1)* と *var-2 (変数 2)* はホスト変数として使用できません。
4. level-1 (レベル 1) は 2 から 48 までの COBOL レベルを示します。

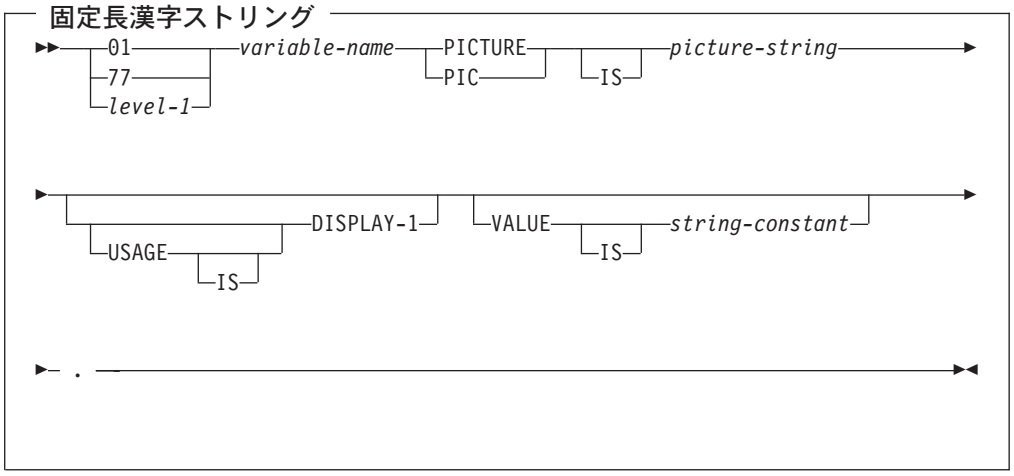
SQL を使用する COBOL アプリケーションでのグラフィック・ホスト変数

グラフィック・ホスト変数は ILE COBOL for iSeries のみでサポートされます。

グラフィック・ホスト変数には、次の 2 つの形式があります。

- 固定長漢字ストリング
- 可変長漢字ストリング

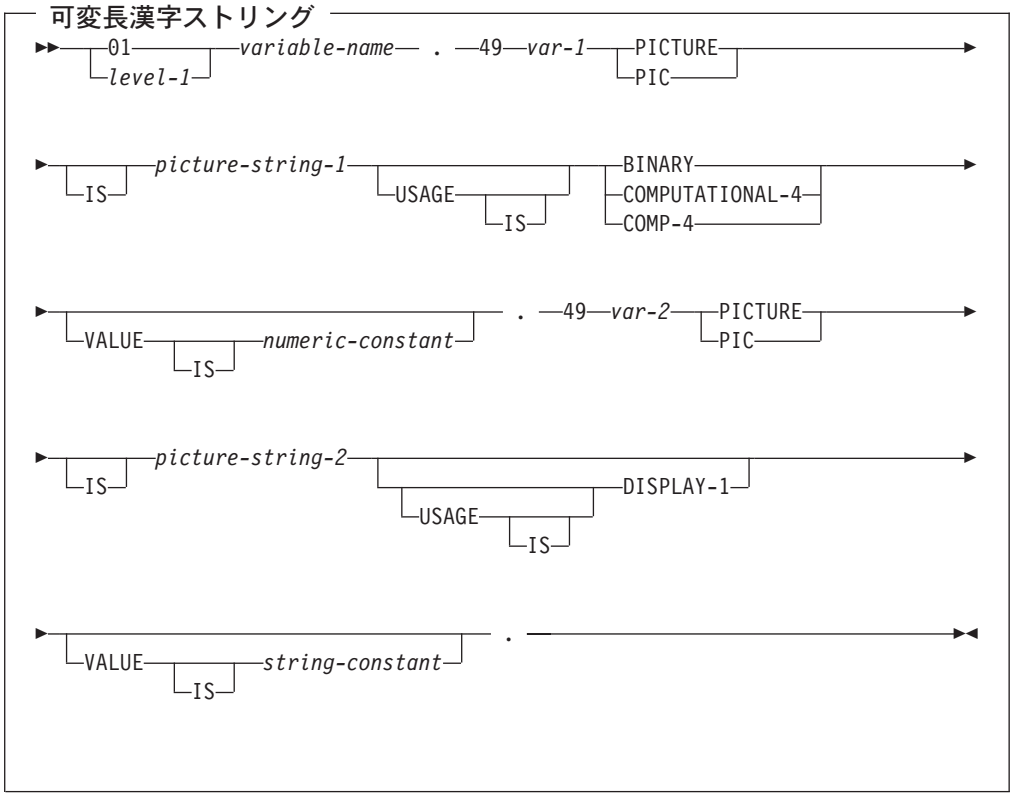
固定長漢字ストリング



注:

1. これらの形式に関連する *picture-string (ピクチャー列)* は G(m) (または G のインスタンスが m 回現れる GGG...G) あるいは、N(m) (または N のインスタンスが m 回現れる NNN...N) で 1 ≤ m ≤ 16 383 になっていなければなりません。
2. level-1 (レベル 1) は 2 から 48 までの COBOL レベルを示します。

可変長漢字ストリング



注:

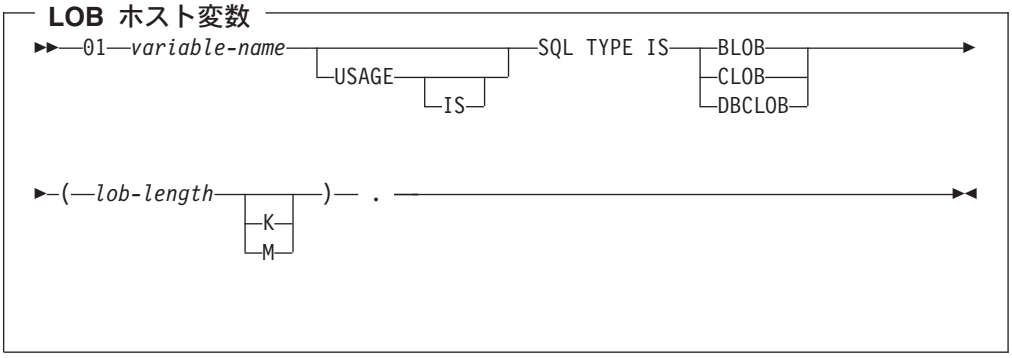
1. これらの形式に関連する *picture-string-1 (ピクチャー列 1)* は、 S9(m) または 9 のインスタンスが m 回現れる S9...9 でなければなりません。m は、1 から 4 までででなければなりません。

   iSeries 上の COBOL で値が指定の精度までしか認識されない場合があっても、データベース・マネージャーは S9(m) 変数の全桁を使用することに注意してください。この結果、COBOL ステートメントの実行中にデータ打ち切りエラーが生じることになるので、可変長漢字ストリングの最大長が事実上、指定の精度までに制限される場合があります。
2. これらの形式に関連する *picture-string-2 (ピクチャー列 2)* は G(m)、 G のインスタンスが m 回現れる GG...G、N(m)、または N のインスタンスが m 回現れる NN...N で、1 ≤ m ≤ 16 370 になっていなければなりません。
3. *var-1 (変数 1)* と *var-2 (変数 2)* はホスト変数として使用できません。
4. level-1 (レベル 1) は 2 から 48 までの COBOL レベルを示します。

SQL を使用する COBOL アプリケーションでの LOB ホスト変数

COBOL には、LOB (ラージ・オブジェクト) の SQL データ・タイプに対応する変数がありません。これらのデータ・タイプで使用するホスト変数を作成するには、SQL TYPE IS 文節を使用します。SQL プリコンパイラーは、この宣言を出力ソース・メンバー内で、COBOL 言語構造に置き換えます。

LOB ホスト変数は、ILE COBOL for iSeries でのみサポートされています。



注:

1. BLOB および CLOB の場合、1 <= lob-length <= 15,728,640 です。
2. DBCLOB の場合、1 <= lob-length <= 7,864,320 です。
3. SQL TYPE IS、BLOB、CLOB、DBCLOB は大文字小文字にすることができます。

BLOB の例

次のように宣言すると、

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

以下の構造を生成します。

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

CLOB の例

次のように宣言すると、

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

以下の構造を生成します。

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

DBCLOB の例

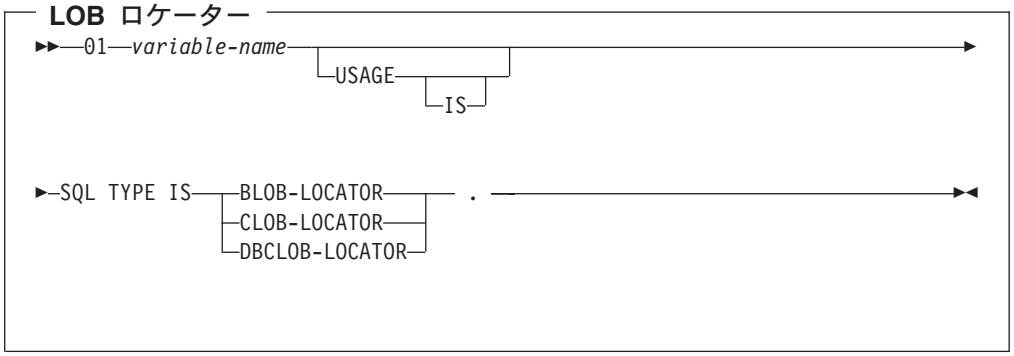
次のように宣言すると、

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

以下の構造を生成します。

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

LOB ロケーター



注:

1. SQL TYPE IS、BLOB-LOCATOR、CLOB-LOCATOR、DBCLOB-LOCATOR は大文字小文字混合にすることができます。
2. LOB ロケーターは、SQL TYPE IS ステートメントの中で初期設定することはできません。

BLOB ロケーターの例

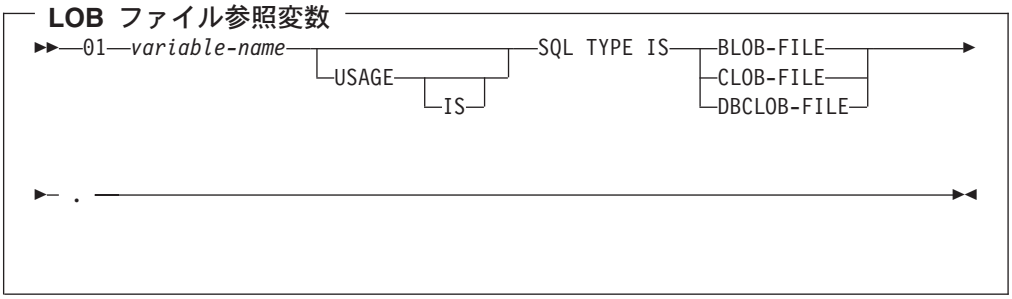
次のように宣言すると、

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

以下の構造を生成します。

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

CLOB ロケーターおよび DBCLOB ロケーターの構文は、似ています。



注: SQL TYPE IS、BLOB-FILE、CLOB-FILE、DBCLOB-FILE は大文字小文字混合にすることができます。

BLOB ファイル参照の例

次のように宣言すると、

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

以下の構造を生成します。

```
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).
```

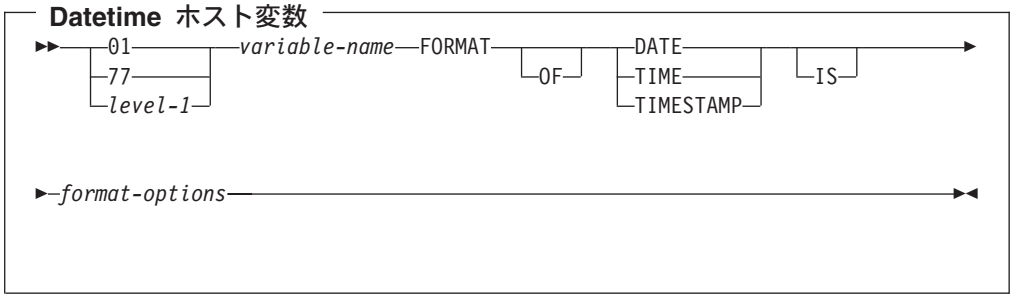
CLOB ファイル参照変数と DBCLOB ファイル参照変数は、類似の構文をもっています。

プリコンパイラーは、次のファイル・オプション定数に対する宣言を生成します。ファイル参照ホスト変数を使用する場合、これらの定数を使用して、xxx-FILE-OPTIONS 変数を設定できます。これらの値の詳細については、「SQL プログラミング 概念」の『LOB ファイル参照変数』を参照してください。

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

SQL を使用する COBOL アプリケーションでの Datetime ホスト変数

次の図は、有効な日付、時刻、およびタイム・スタンプのホスト変数宣言の構文を示しています。 Datetime ホスト変数は、ILE COBOL for iSeries でのみサポートされています。

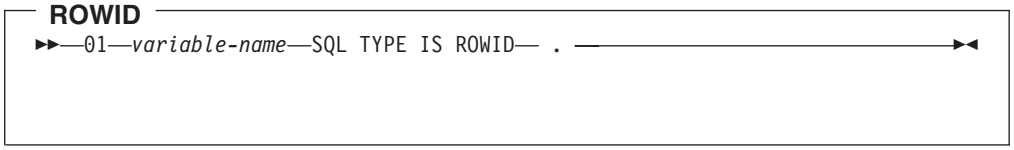


注:

1. *level-1 (レベル 1)* は、2 ～ 48 の COBOL レベルを示します。
2. *format-options* は、 COBOL コンパイラーでサポートされる有効な datetime オプションを示します。詳細は、「ILE COBOL 解説書」 を参照してください。

SQL を使用する COBOL アプリケーションでの ROWID ホスト変数

COBOL には、ROWID の SQL データ・タイプに対応する変数がありません。このデータ・タイプで使用するホスト変数を作成するには、SQL TYPE IS 文節を使用します。SQL プリコンパイラーは、この宣言を出力ソース・メンバー内で、COBOL 言語構造に置き換えます。



注: SQL TYPE IS ROWID は、大文字小文字混合にすることができます。

ROWID の例

次のように宣言すると、

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

以下の構造を生成します。

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

SQL を使用する COBOL アプリケーションでのホスト構造の使用

ホスト構造は、ユーザーのプログラムの DATA DIVISION の中で定義されている一連のホスト変数に名前を付けたものです。ホスト構造はそれ自身が複数レベルの構造の中に置かれることがあっても、その最大レベルは 2 レベルまでです。可変長の文字ストリングの宣言だけは例外で、その場合は別のレベルが必要であり、これはレベル 49 でなければなりません。

ホスト構造名をグループ名とし、その従属レベルで基本データ項目の名前を指定することができます。以下に、例を示します。

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

この例では、B は基本データ項目 C1 と C2 から成るホスト構造の名前です。

修飾ホスト変数名 (たとえば、構造内のフィールドを識別するために) を使用して SQL ステートメントを書くときは、構造の名前の後にピリオドとフィールドの名前を続けます (すなわち、PL/I 形式を使用します)。たとえば、 C1 OF B または C1 IN B ではなく B.C1 を指定してください。ただし、 PL/I 形式は、SQL ステートメントの中の修飾名だけに適用されます。この技法を用いて、COBOL ステートメントに修飾名を書くことはできません。

次のいずれかの項目を検出した場合は、ホスト構造は完全と見なされます。

- 領域 A で始まる必要のある COBOL 項目
- 任意の SQL ステートメント (ただし、SQL INCLUDE を除く)

ホスト構造を定義しておけば、いくつかのホスト変数 (ホスト構造を構成しているデータ項目の名前) を個別に指定しなくても、SQL ステートメントの中でそのホスト構造を参照することができます。

たとえば、次のようにコーディングすれば、テーブル CORPDATA.EMPLOYEE から選択した行のすべての列の値を検索することができます。

```
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.
```

上の例に示すように、PEMPL の宣言の中で、2 つの可変長文字列要素、FIRSTNME と LASTNAME が構造に組み込まれています。

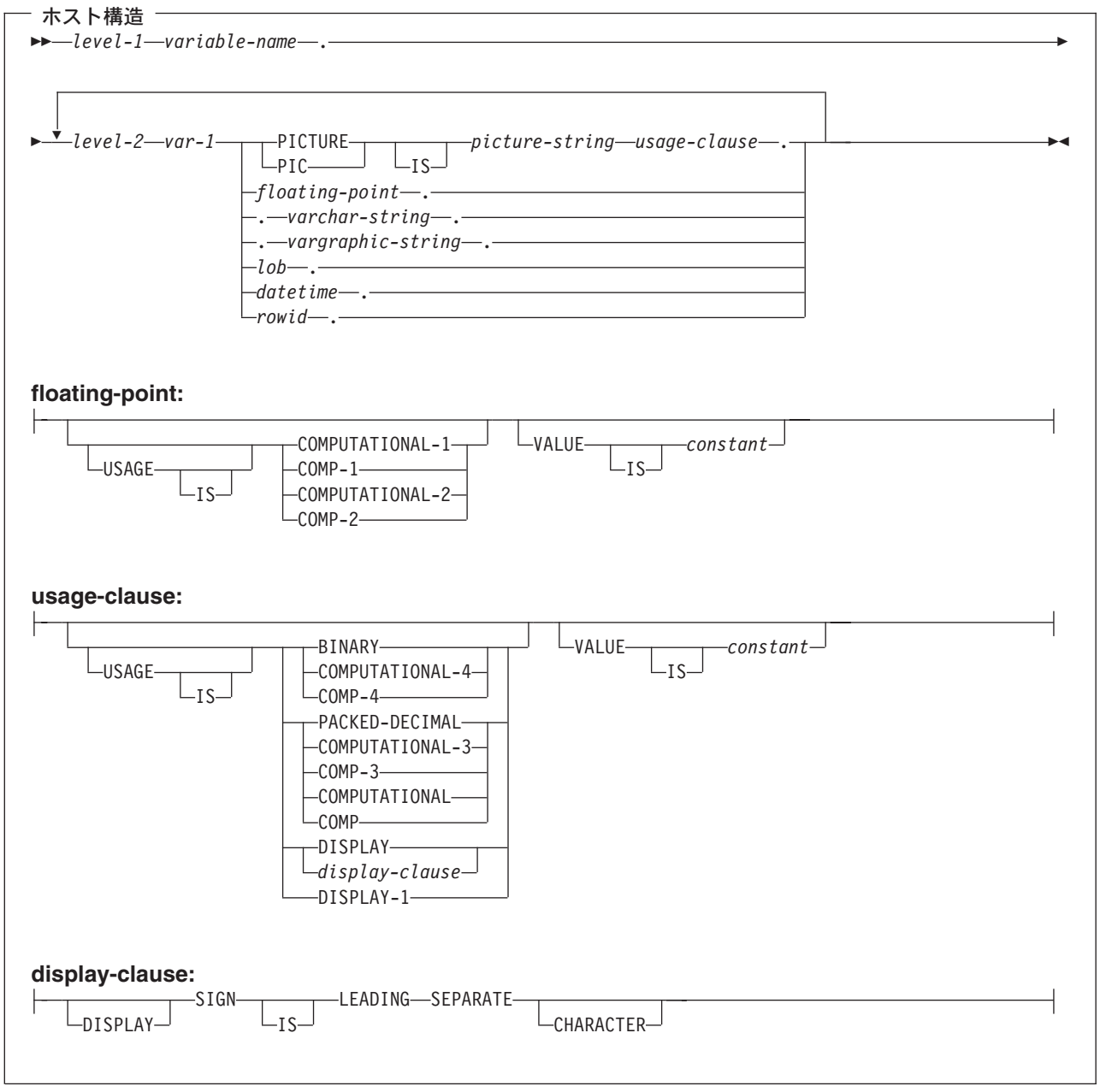
詳細については、以下のセクションを参照してください。

- 64 ページの『SQL を使用する COBOL アプリケーションでのホスト構造』
- 66 ページの『SQL を使用する COBOL アプリケーションでのホスト構造標識配列』
- 67 ページの『SQL を使用する COBOL アプリケーションでのホスト構造配列の使用』

- 68 ページの『SQL を使用する COBOL アプリケーションでのホスト構造配列』
- 70 ページの『SQL を使用する COBOL アプリケーションでのホスト配列標識構造』

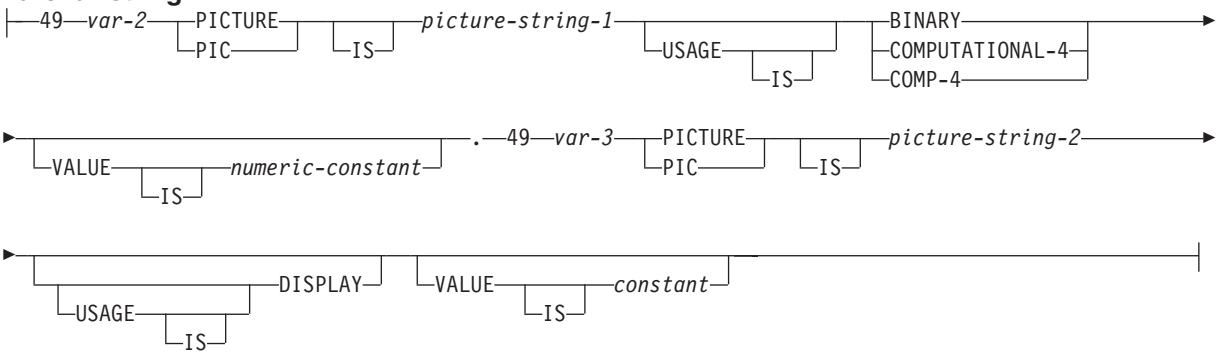
SQL を使用する COBOL アプリケーションでのホスト構造

次の図は、有効なホスト構造の構文を示しています。

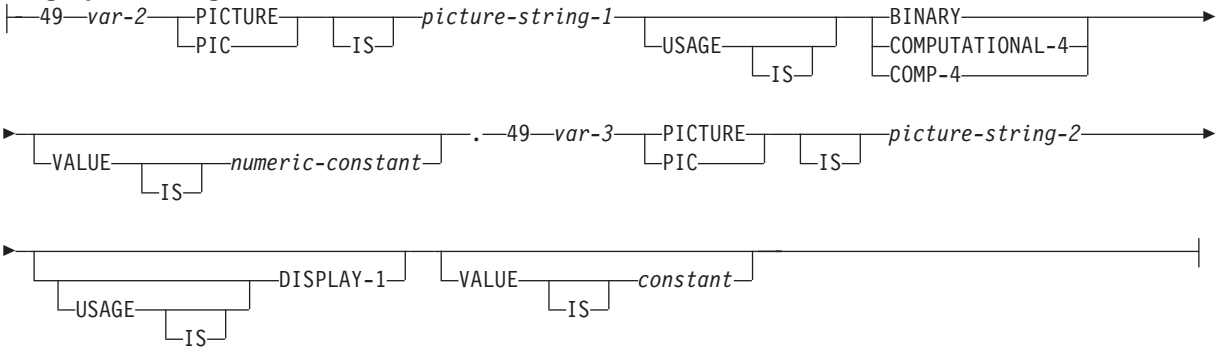


ホスト構造 (続き)

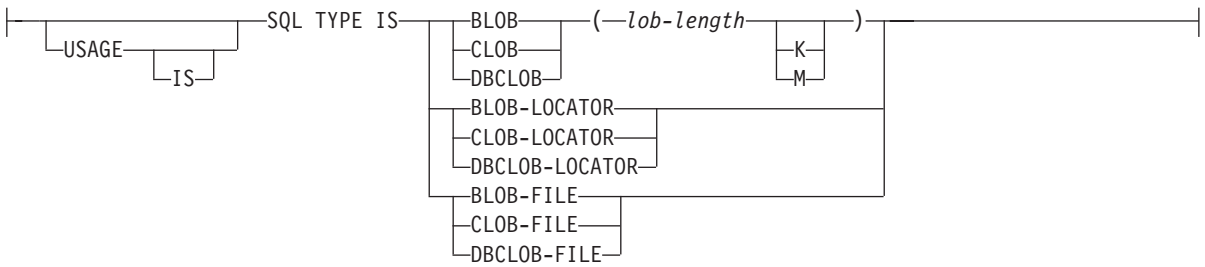
varchar-string:



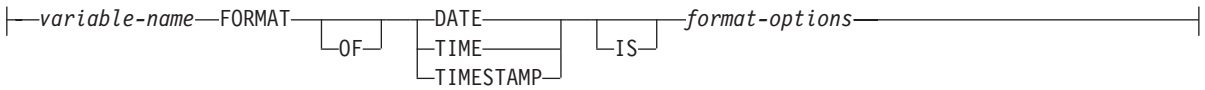
vargraphic-string:



lob:



datetime:



rowid:

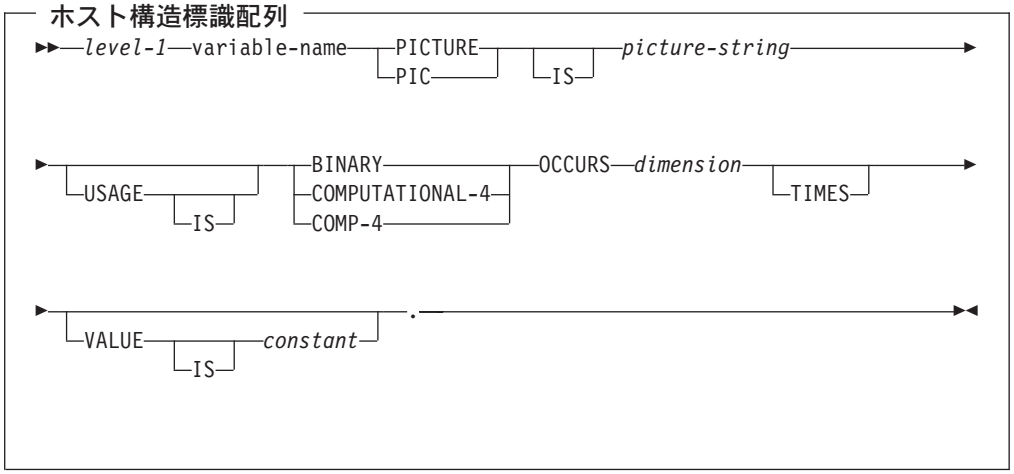
注:

1. level-1 (レベル 1) は 1 から 47 までの COBOL レベルを示します。

2. level-2 (レベル 2) は 2 から 48 までの COBOL レベルを示します。ただし、level-2 > level-1 でなければなりません。
3. グラフィック・ホスト変数、LOB ホスト変数、および浮動小数点ホスト変数は、 ILE COBOL for iSeries のみでサポートされます。
4. 数値ホスト変数、文字ホスト変数、グラフィック・ホスト変数、LOB ホスト変数、および ROWID ホスト変数の宣言の詳細については、数値ホスト変数、文字ホスト変数、グラフィック・ホスト変数、LOB ホスト変数、および ROWID ホスト変数に関する注意事項を参照してください。
5. *format-options* は、 COBOL コンパイラーでサポートされる有効な datetime オプションを示します。詳細は、「ILE COBOL 解説書」 を参照してください。

SQL を使用する COBOL アプリケーションでのホスト構造標識配列

次の図は有効な標識配列宣言の構文を示しています。



注:

1. dimension (次元) は 1 から 32767 までの整数でなければなりません。
2. level-1 (レベル 1) は 2 から 48 までの整数でなければなりません。
3. BINARY、COMPUTATIONAL-4、および COMP-4 は同じ働きをします。他のシステムでも実行されるようなアプリケーションの場合には BINARY をコーディングしておくべきです。COMPUTATIONAL-4 と COMP-4 は IBM の拡張機能であり、ISO/ANSI COBOL ではサポートされないからです。これらのタイプに関連する *picture-string (ピクチャー列)* は S9(i) (または 9 のインスタンスが i 回現れる S9...9) でなければなりません。 i は 4 以下でなければなりません。

SQL を使用する COBOL アプリケーションでのホスト構造配列の使用

ホスト構造配列は、プログラムのデータ部の中で定義され、OCCURS 文節が付いている一連のホスト変数に名前を付けたものです。ホスト構造が複数レベルの構造に現れる場合があっても、ホスト構造配列のレベルは最高 2 レベルまでです。可変長文字列には、別のレベル、すなわちレベル 49 が必要です。ホスト構造配列名をグループ名とし、その従属レベルで基本データ項目の名前を指定することができます。

このような例では、以下の条件が該当します。

- B-ARRAY のすべてのメンバーは有効でなければなりません。
- B-ARRAY は修飾できません。
- B-ARRAY はブロック化形式の FETCH および INSERT ステートメントでのみ使用できます。
- B-ARRAY は、データ項目 C1-VAR および C2-VAR からなるホスト構造の配列名です。
- SYNCHRONIZED 属性は指定してはなりません。
- C1-VAR および C2-VAR は、SQL ステートメントでは有効なホスト変数ではありません。構造に中間レベルの構造を含めることはできません。

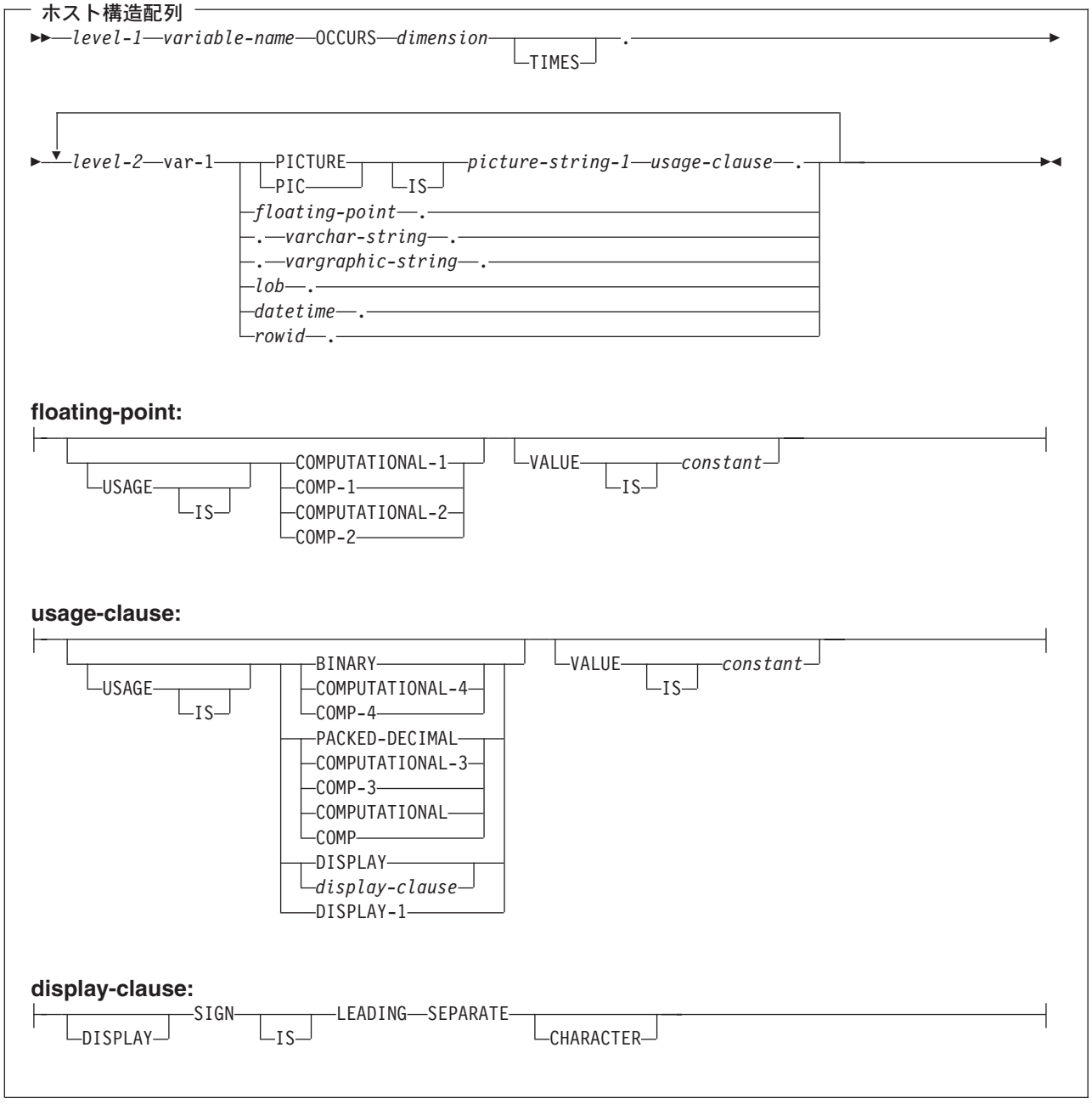
```
01  A-STRUCT.
    02 B-ARRAY OCCURS 10 TIMES.
       03 C1-VAR PIC X(20).
       03 C2-VAR PIC S9(4).
```

CORPDATA.DEPARTMENT テーブルから 10 行分検索する場合は、次の例を使用してください。

```
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.
```

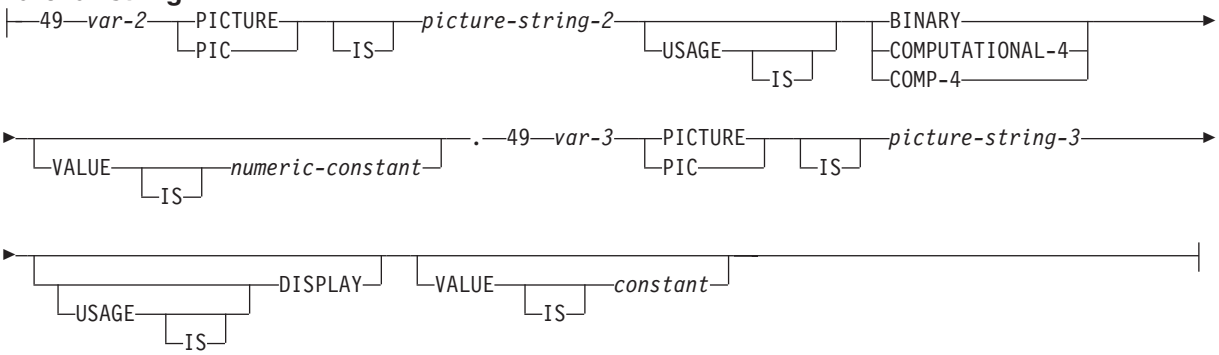
SQL を使用する COBOL アプリケーションでのホスト構造配列

次の図は、有効なホスト構造配列宣言の構文を示しています。

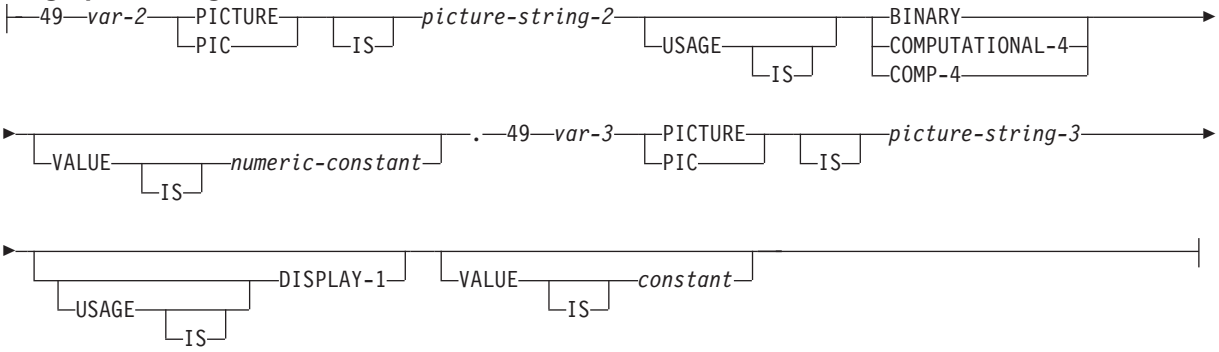


ホスト構造配列 (続き)

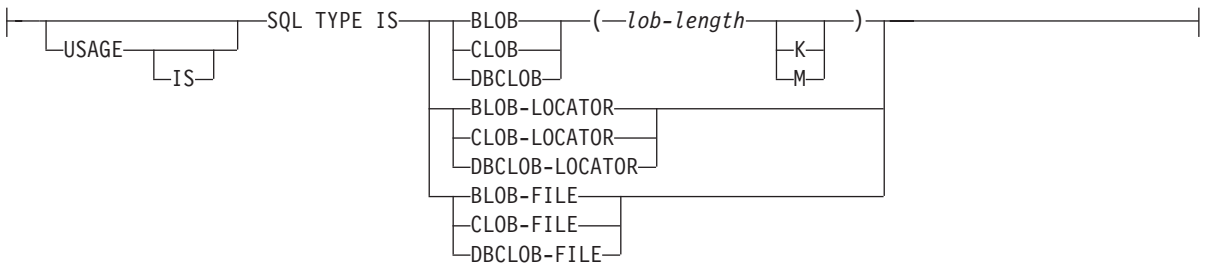
varchar-string:



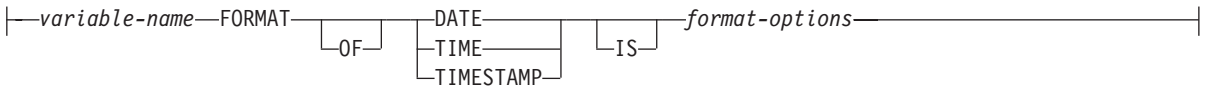
vargraphic-string:



lob:



datetime:



rowid:

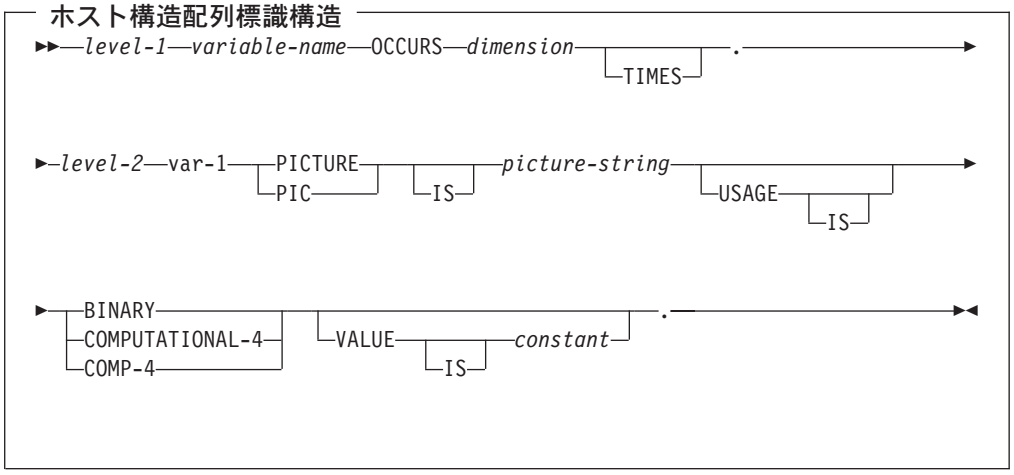
注:

1. level-1 (レベル 1) は 2 から 47 までの COBOL レベルを示します。

2. level-2 (レベル 2) は 3 から 48 までの COBOL レベルを示します。ただし、level-2 > level-1 でなければなりません。
3. グラフィック・ホスト変数、LOB ホスト変数、および浮動小数点ホスト変数は、 ILE COBOL for iSeries のみでサポートされます。
4. 数値ホスト変数、文字ホスト変数、グラフィック・ホスト変数、および LOB ホスト変数の宣言の詳細については、数値ホスト変数、文字ホスト変数、グラフィック・ホスト変数、および LOB ホスト変数に関する注意事項を参照してください。
5. dimension (次元) は 1 から 32767 までの整数定数でなければなりません。
6. *format-options* は、 COBOL コンパイラーでサポートされる有効な datetime オプションを示します。詳細は、「ILE COBOL 解説書」 を参照してください。

SQL を使用する COBOL アプリケーションでのホスト配列標識構造

次の図は、ホスト構造配列標識として有効な構文を示しています。



注:

1. level-1 (レベル 1) は 2 から 48 までの COBOL レベルを示します。
2. level-2 (レベル 2) は 3 から 48 までの COBOL レベルを示します。ただし、level-2 > level-1 でなければなりません。
3. dimension (次元) は 1 から 32767 までの整数定数でなければなりません。
4. BINARY、COMPUTATIONAL-4、および COMP-4 は同じ働きをします。他のシステムでも実行されるようなアプリケーションの場合には BINARY をコーディングしておくべきです。COMPUTATIONAL-4 と COMP-4 は IBM の拡張機能であり、ISO/ANSI COBOL ではサポートされないからです。これらのタイプに関連する *picture-string (ピクチャー列)* は S9(i) (または 9 のインスタンスが i 回現れる S9...9) でなければなりません。 i は 4 以下でなければなりません。

SQL を使用する COBOL アプリケーションでの外部ファイル記述の使用

SQL は、ファイル定義からホスト変数を検索するために、COPY DD 様式名、COPY DD-ALL-FORMATS、COPY DDS 様式名、 COPY DDR 様式名、COPY DDR-ALL-FORMATS、COPY DDSR 様式名、 COPY DDS-ALL-FORMATS、および COPY DDSR-ALL-FORMATS を使用します。 REPLACING オプションの指定があるときは、完全名での書き換えだけが行われます。変数 1 は、様式名およびフィールド名と突き合わされて比較されます。両者が同じであれば、変数 2 が新しい名前として使用されます。

注: COBOL 予約語がフィールド名として使用されているファイル定義からホスト変数を検索することはできません。COBOL ホスト構造の中に、 COPY DDx-format ステートメントを入れなければなりません。

「*DB2 UDB for iSeries SQL プログラミング 概念*」の『DB2 UDB for iSeries サンプル・テーブル』に記載されているサンプル・テーブル DEPARTMENT の定義を検索するときは、次のようにコーディングすることができます。

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

DEPARTMENT-STRUCTURE という名前のホスト構造は、DEPARTMENT-RECORD という名前の 05 レベルのフィールドをもつものとして定義されており、そのフィールドには、さらに DEPTNO、DEPTNAME、MGRNO、および ADMRDEPT という名前の 4 つの 06 レベルのフィールドが含まれています。これらのフィールド名は、SQL ステートメントの中でホスト変数として使用できます。 COBOL COPY verb の詳細については、「COBOL/400® User's Guide (COBOL/400 使用者の手引き、SC88-5201)」 および「ILE COBOL 解説書」 を参照してください。

外部ファイル記述の詳細については、『SQL を使用する COBOL アプリケーションでのホスト構造配列の外部ファイル記述の使用』を参照してください。

SQL を使用する COBOL アプリケーションでのホスト構造配列の外部ファイル記述の使用

COBOL では外部記述データを組み入れるときに追加のレベルを作成するので、その前の 04 レベルに OCCURS 文節を置かなければなりません。 05 レベルで追加の宣言を指定しても、その宣言は構造に入れることはできません。

FILLER として生成されたフィールドがファイルに含まれている場合は、構造をホスト構造配列として使用できません。

装置ファイルの場合は、INDARA の指定がなく、ファイルに標識が入っているときには、その宣言をホスト構造配列として使用できません。生成された構造には標識域が組み入れられ、その標識域があるためにレコード用の記憶域が連続しなくなります。

たとえば、次の例は、COPY-DDS を使用してホスト構造配列を生成し、10 行取り出してそのホスト構造配列に入れる方法を示しています。

```
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.
```

注: DATE、TIME、および TIMESTAMP の各列からは、文字ホスト変数定義が生成されます。SQL により、DATE、TIME、および TIMESTAMP 列と同じ比較規則と割り当て規則を使用して扱われます。たとえば、日付ホスト変数は、DATE 列または日付の有効な表現である文字列とだけ突き合わされて比較されます。

COBOL for iSeries では、GRAPHIC と VARGRAPHIC は文字変数にマップされますが、SQL はこれらを GRAPHIC 変数および VARGRAPHIC 変数と見なします。 GRAPHIC 列または VARGRAPHIC 列が UCS-2 CCSID を持つ場合、生成されるホスト変数には UCS-2 CCSID が割り当てられます。

SQL データ・タイプと COBOL データ・タイプの対応関係の判別

プリコンパイラーは、次の表に基づいて、ホスト変数のベース SQLTYPE とベース SQLLEN を判断します。ホスト変数が標識変数と一緒に記載されているときは、その SQLTYPE はベース SQLTYPE に 1 を加えたものです。

表 3. COBOL 宣言と代表的 SQL データ・タイプとの対応関係

COBOL データ・タイプ	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
S9(i)V9(d) COMP-3 または S9(i)V9(d) COMP または S9(i)V9(d) PACKED-DECIMAL	484	バイト 1 には i+d、バイト 2 には d	DECIMAL(i+d,d)
S9(i)V9(d) DISPLAY SIGN LEADING SEPARATE	504	バイト 1 には i+d、バイト 2 には d	正確に対応するものなし。DECIMAL(i+d,d) または NUMERIC(i+d,d)
S9(i)V9(d)DISPLAY	488	バイト 1 には i+d、バイト 2 には d	NUMERIC(i+d,d)
S9(i) BINARY または S9(i) COMP-4 (i は 1 から 4 まで)	500	2	SMALLINT
S9(i) BINARY または S9(i) COMP-4 (i は 5 から 9 まで)	496	4	INTEGER
S9(i) BINARY または S9(i) COMP-4 (i は 10 から 18 まで) COBOL for iSeries ではサポートなし。	492	8	BIGINT

表 3. COBOL 宣言と代表的 SQL データ・タイプとの対応関係 (続き)

COBOL データ・タイプ	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
S9(i)V9(d) BINARY または S9(i)V9(d) COMP-4 (i+d ≤ 4)	500	バイト 1 には i+d、バイト 2 には d	正確に対応するものなし。DECIMAL(i+d,d) または NUMERIC(i+d,d)
S9(i)V9(d) BINARY または S9(i)V9(d) COMP-4 (4 < i+d ≤ 9)	496	バイト 1 には i+d、バイト 2 には d	正確に対応するものなし。DECIMAL(i+d,d) または NUMERIC(i+d,d)
COMP-1 COBOL for iSeries ではサポートなし。	480	4	FLOAT (単精度)
COMP-2 COBOL for iSeries ではサポートなし。	480	8	FLOAT (倍精度)
固定長文字データ	452	m	CHAR(m)
可変長文字データ m < 255	448	m	VARCHAR(m)
可変長文字データ (m > 254)	456	m	VARCHAR(m)
固定長グラフィック・データ COBOL for iSeries ではサポートなし。	468	m	GRAPHIC(m)
可変長グラフィック・データ (m < 128) COBOL for iSeries ではサポートなし。	464	m	VARGRAPHIC(m)
可変長グラフィック・データ (m > 127) COBOL for iSeries ではサポートなし。	472	m	VARGRAPHIC(m)
DATE COBOL for iSeries ではサポートなし。	384		DATE
TIME COBOL for iSeries ではサポートなし。	388		TIME
TIMESTAMP COBOL for iSeries ではサポートなし。	392	26	TIMESTAMP

下表を参照すると、各 SQL データ・タイプに対応する COBOL データ・タイプを判断することができます。

表 4. SQL データ・タイプと代表的な COBOL 宣言との対応関係

SQL データ・タイプ	COBOL データ・タイプ	注
SMALLINT	S9(m) COMP-4	m は 1 から 4 まで。
INTEGER	S9(m) COMP-4	m は 5 から 9 まで。
BIGINT	ILE COBOL for iSeries の S9(m) COMP-4。 COBOL for iSeries ではサポートなし。	m は 10 から 18 まで。
DECIMAL(p,s)	p<19 の場合: S9(p-s)V9(s) PACKED-DECIMAL または S9(p-s)V9(s) COMP または S9(p-s)V9(s) COMP-3。p>18 の場合: サポートなし。	*p* は精度、*s* は位取りです。0<=s<=p<=18。s=0 の場合は S9(p) または S9(p)V を使用します。 s=p の場合は、SV9(s) を使用します。
NUMERIC(p,s)	p<19 の場合: S9(p-s)V9(s) DISPLAY。p>18 の場合: サポートなし。	*p* は精度、*s* は位取りです。0<=s<=p<=18。s=0 の場合は S9(p) または S9(p)V を使用します。 s=p の場合は、SV9(s) を使用します。
FLOAT (単精度)	ILE COBOL for iSeries の COMP-1。 COBOL for iSeries ではサポートなし。	
FLOAT (倍精度)	ILE COBOL for iSeries の COMP-2。 COBOL for iSeries ではサポートなし。	
CHAR(n)	固定長文字ストリング	32766≥n≥1
VARCHAR(n)	可変長文字ストリング	32740≥n≥1
BLOB	なし	SQL TYPE IS を使用して BLOB を宣言します。 ILE COBOL for iSeries。 COBOL for iSeries ではサポートなし。
CLOB	なし	SQL TYPE IS を使用して CLOB を宣言します。 ILE COBOL for iSeries。 COBOL for iSeries ではサポートなし。
GRAPHIC(n)	ILE COBOL for iSeries の固定長漢字ストリング。 COBOL for iSeries ではサポートなし。	16383≥n≥1

表 4. SQL データ・タイプと代表的な COBOL 宣言との対応関係 (続き)

SQL データ・タイプ	COBOL データ・タイプ	注
VARGRAPHIC(n)	ILE COBOL for iSeries の可変長漢字ストリング。 COBOL for iSeries ではサポートなし。	16370≥n≥1
DBCLOB	なし	SQL TYPE IS を使用して DBCLOB を宣言します。 ILE COBOL for iSeries。 COBOL for iSeries ではサポートなし。
DATE	固定長文字ストリング、または DATE (ILE COBOL for iSeries)	形式が *USA、*JIS、*EUR、または *ISO のときは、少なくとも 10 文字が必要。形式が *YMD、*DMY、または *MDY のときは、少なくとも 8 文字が必要。形式が *JUL のときは、少なくとも 6 文字が必要。
TIME	固定長文字ストリング、または TIME (ILE COBOL for iSeries)	少なくとも 6 文字が必要。秒を含む場合は、8 文字が必要。
TIMESTAMP	固定長文字ストリング、または TIMESTAMP (ILE COBOL for iSeries)	n は少なくとも 19 が必要。マイクロ秒を全桁の精度で含める場合は、n は少なくとも 26 が必要。n が 26 未満のときは、マイクロ秒部分で切り捨てが起こる。
DATALINK	サポートなし	
ROWID	なし	SQL TYPE IS を使用して ROWID を宣言します。

詳細については、『COBOL 変数宣言と使用上の注意事項』を参照してください。

COBOL 変数宣言と使用上の注意事項

レベル 77 のデータ記述項目の後に 1 つ以上の REDEFINES 項目を続けることができます。しかし、これらの項目に入っている名前は、SQL ステートメントの中では使用できません。

構造体が FILLER 項目より下に定義されたレベルを含むとき、異常な結果を生じる場合があります。

SMALLINT、BIGINT、および INTEGER データ・タイプの COBOL 宣言は、数桁の 10 進数で表されます。データベース・マネージャーは整数の全桁を使用するので、COBOL 宣言の中の指定した桁数で許容されるよりも大きな値をホスト変数に入れることができます。しかし、これが行われると、COBOL ステートメントの実行時にデータ切り捨てやサイズのエラーが起こります。アプリケーションの中の数の大きさが宣言した桁数の範囲内にあることを確かめておくべきです。

SQL を使用する COBOL アプリケーションでの標識変数の使用

標識変数は 2 バイトの整数です (PIC S9(m) USAGE BINARY、ただし、m は 1 から 4 までです)。ホスト構造をサポートするために標識構造 (ハーフワードの整数変数の配列として定義されているもの) を指定することもできます。検索されるとき、標識変数はその対応するホスト変数にヌル値が割り当てられているかどうかを示すために使用されます。列に割り当てるときには、ヌル値を割り当てるべきであることを示すために負の標識変数が使用されます。

詳細については、「SQL 解説書」の『標識変数』を参照してください。

標識変数の宣言の仕方はホスト変数の場合と同じであり、これらの 2 つの変数の宣言をプログラマーに適切と思われる方法で組み合わせることができます。

例:

次のステートメントがあるとします。

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

変数は次のように宣言することができます。

```
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.
```

第 4 章 PL/I アプリケーションでの SQL ステートメントのコーディング方法

この章では、SQL ステートメントを iSeries PL/I プログラムに組み込む場合に固有のアプリケーションおよびコーディング上の要件について説明します。ホスト構造およびホスト変数に関する要件についても説明します。

詳細については、以下のセクションを参照してください。

- 『SQL を使用する PL/I アプリケーションでの SQL 連絡域の定義』
- 78 ページの『SQL を使用する PL/I アプリケーションでの SQL 記述子域の使用』
- 79 ページの『SQL を使用する PL/I アプリケーションでの SQL ステートメントの組み込み』
- 81 ページの『SQL を使用する PL/I アプリケーションでのホスト変数の使用』
- 87 ページの『SQL を使用する PL/I アプリケーションでのホスト構造の使用』
- 89 ページの『SQL を使用する PL/I アプリケーションでのホスト構造配列の使用』
- 91 ページの『SQL を使用する PL/I アプリケーションでの外部ファイル記述の使用』
- 92 ページの『SQL データ・タイプと PL/I データ・タイプの対応関係の判別』
- 94 ページの『SQL を使用する PL/I アプリケーションでの標識変数の使用』
- 95 ページの『構造パラメーター受け渡し技法による PL/I での相違』

SQL ステートメントの使い方を示した詳しいサンプル PL/I プログラムは、161 ページの『付録 A. DB2 UDB for iSeries ステートメントを使用するサンプル・プログラム』に記載されています。

注: コード例についての詳細は、viii ページの『コードについての特記事項』を参照してください。

SQL を使用する PL/I アプリケーションでの SQL 連絡域の定義

SQL ステートメントを含んでいる PL/I プログラムには、次のいずれかまたは両方を持っている必要があります。

- FIXED BINARY(31) として宣言している SQLCODE 変数
- CHAR(5) として宣言している SQLSTATE 変数

または、

- SQLCA (SQLCODE および SQLSTATE 変数が入っている)

SQLCODE 値および SQLSTATE 値は、各 SQL ステートメントが実行された後、データベース・マネージャーによってセットされます。アプリケーションは、

SQLCODE 値または SQLSTATE 値を調べて、最後の SQL ステートメントが正しく実行されたかどうかを判定することができます。

SQLCA は、直接または、SQL の INCLUDE ステートメントを使用して、PL/I プログラムの中にコーディングすることができます。SQL の INCLUDE ステートメントを使用するときは、標準の SQLCA 宣言を組み込む必要があります。

```
EXEC SQL INCLUDE SQLCA ;
```

SQLCODE、SQLSTATE、および SQLCA の各変数の有効範囲には、プログラムの中のすべての SQL ステートメントの有効範囲が含まれていなければなりません。

SQLCA を組み込んだ PL/I ソース・ステートメントは次のとおりです。

```
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);
```

SQLCODE の宣言がプログラムの中にあって、SQLCA がプリコンパイラーによって与えられるとき、SQLCODE の個所は SQLCADE で置き換えられます。
SQLSTATE の宣言がプログラムの中にあって、SQLCA がプリコンパイラーによって与えられるとき、SQLSTATE の個所は SQLSTOTE で置き換えられます。

SQLCA の詳細については、「SQL 解説書」の『SQL 連絡域』を参照してください。

SQL を使用する PL/I アプリケーションでの SQL 記述子域の使用

SQLDA を必要とするステートメントには、次のものがあります。

EXECUTE...USING DESCRIPTOR *記述子名*

FETCH...USING DESCRIPTOR *記述子名*

OPEN...USING DESCRIPTOR *記述子名*

CALL...USING DESCRIPTOR *記述子名*

DESCRIBE *ステートメント名* INTO *記述子名*

DESCRIBE TABLE *ホスト変数* INTO *記述子名*

PREPARE *ステートメント名* INTO *記述子名*

SQLCA と異なり、SQLDA を 2 つ以上プログラムの中に置くことができ、また SQLDA の名前は有効であれば、どの名前でも使えます。 SQLDA は、直接プログ

ラムするか、または SQL INCLUDE ステートメントを使用して PL/I プログラムにコーディングすることができます。 SQL の INCLUDE ステートメントを使用するときは、標準の SQLDA 宣言を組み込む必要があります。

```
EXEC SQL INCLUDE SQLDA ;
```

SQLDA 用に組み込まれる PL/I ソース・ステートメントは次のとおりです。

```
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;
```

動的 SQL は高度なプログラミング技法です。これについては、「*DB2 UDB for iSeries SQL プログラミング 概念*」の『動的 SQL アプリケーション』で説明します。動的 SQL を使用すると、ユーザーのプログラムはその実行と平行して SQL ステートメントを作成し、実行させることができます。動的に実行される変数 SELECT リスト (すなわち、照会の一部として返されるデータのリスト) を指定する SELECT ステートメントには、SQL 記述子域 (SQLDA) が必要です。これは、SELECT の結果を受け入れるために割り振るべき変数の数とタイプが事前に予測できないからです。

SQLDA の詳細については、「SQL 解説書」の『SQL 記述子域』を参照してください。

SQL を使用する PL/I アプリケーションでの SQL ステートメントの組み込み

PL/I プログラムの最初のステートメントは PROCEDURE ステートメントでなければなりません。

SQL ステートメントは、実行可能なステートメントを置くことができる個所ならば、PL/I プログラム内のどこにでもコーディングすることができます。

PL/I プログラムの各 SQL ステートメントは EXEC SQL で始まり、セミコロン (;) で終わらなければなりません。キーワード EXEC SQL は 1 行に置かなければなりませんが、ステートメントの残りの部分は次の行とそれ以降の行に置くことができます。

詳細については、以下のセクションを参照してください。

- 80 ページの『例: SQL を使用する PL/I ステートメントでの SQL ステートメントの組み込み』
- 80 ページの『SQL を使用する PL/I アプリケーションでの注記』
- 80 ページの『SQL を使用する PL/I アプリケーションでの SQL ステートメントの継続』

- 『SQL を使用する PL/I アプリケーションでのコードの組み込み』
- 81 ページの『SQL を使用する PL/I アプリケーションでのマージン』
- 81 ページの『SQL を使用する PL/I アプリケーションでの名前』
- 81 ページの『SQL を使用する PL/I アプリケーションでのステートメント・ラベル』
- 81 ページの『SQL を使用する PL/I アプリケーションでの WHENEVER ステートメント』

例: SQL を使用する PL/I ステートメントでの SQL ステートメントの組み込み

PL/I プログラムの中でコーディングする UPDATE ステートメントを書くと、次のようになります。

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

SQL を使用する PL/I アプリケーションでの注記

SQL の注記 (--) の他に、PL/I の注記 (/*...*/) は、組み込み SQL ステートメント内のブランクが許されている場所のどこにでも置くことができます。ただし、キーワードの EXEC と SQL の間には入れられません。

SQL を使用する PL/I アプリケーションでの SQL ステートメントの継続

SQL ステートメントの場合の行継続に関する規則は、EXEC SQL を 1 行に指定する点を除けば、他の PL/I ステートメントの場合と同じです。

DBCS データを含む定数は、シフトイン文字とシフトアウト文字をマージンの外側に置くことによって、複数行にわたって継続させることができます。この例では、マージンは 2 桁目と 72 桁目に限定されているものとします。この SQL ステートメントの G'<AABBCCDDEEFFGGHHIIJJKK>' はグラフィック定数として有効です。

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

SQL を使用する PL/I アプリケーションでのコードの組み込み

SQL ステートメントまたは PL/I ホスト変数宣言ステートメントは、ステートメントを組み込むソース・コード内の個所に次の SQL ステートメントを置くことによって挿入することができます。

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

PL/I プリプロセッサー・ディレクティブは、SQL ステートメントの中では許されません。 PL/I の %INCLUDE ステートメントは、SQL ステートメントまたは SQL ステートメントの中で参照される PL/I ホスト変数の宣言を組み入れるためには使用できません。

SQL を使用する PL/I アプリケーションでのマージン

SQL ステートメントは、CRTSQLPLI コマンドの MARGINS パラメーターで指定したマージンの範囲内でコーディングしてください。 EXEC SQL が指定のマージン内で始まっていないときは、SQL プリコンパイラーはそれを SQL ステートメントと認識しません。 CRTSQLPLI コマンドの詳しい説明については、 199 ページの『付録 B. ホスト言語プリコンパイラーの DB2 UDB for iSeries CL コマンド記述』を参照してください。

SQL を使用する PL/I アプリケーションでの名前

有効な PL/I 変数名であれば、どの変数名でもホスト変数に使用できますが、次のような制約を受けます。

'SQL'、'RDI'、または 'DSN' で始まるホスト変数名や外部入り口名は、使用してはなりません。これらの名前はデータベース・マネージャー用に予約されています。

SQL を使用する PL/I アプリケーションでのステートメント・ラベル

実行可能なすべての SQL ステートメントは、PL/I ステートメントと同様、ラベル接頭部を持つことができます。

SQL を使用する PL/I アプリケーションでの WHENEVER ステートメント

SQL WHENEVER ステートメントの GOTO 文節の対象となるものは、PL/I ソース・コード内のラベルでなければならず、WHENEVER ステートメントの影響が及ぶ SQL ステートメントの有効範囲内になければなりません。

SQL を使用する PL/I アプリケーションでのホスト変数の使用

SQL ステートメントの中で使用するホスト変数はいずれも明示的に宣言しなければなりません。

ホスト変数を定義するために使用される PL/I ステートメントは、その前に BEGIN DECLARE SECTION ステートメントを置き、その後に END DECLARE SECTION ステートメントを置く必要があります。 BEGIN DECLARE SECTION と END DECLARE SECTION を指定した場合、SQL ステートメントで使用するすべてのホスト変数宣言は、BEGIN DECLARE SECTION ステートメントと END DECLARE SECTION ステートメントの間になければなりません。

SQL ステートメントの中のホスト変数はいずれも、その前にコロン (:) を付けなければなりません。

ホスト変数の名前は、ホスト変数がそれぞれ別のブロックやプロシージャーの中にある場合であっても、1 つのプログラム内では固有になっていなければなりません。

ホスト変数を使用する SQL ステートメントは、その変数が宣言されたステートメントの有効範囲内になければなりません。

ホスト変数はスカラー変数でなければなりません。これらは配列の要素にすることはできません。

詳細については、『SQL を使用する PL/I アプリケーションでのホスト変数の宣言』を参照してください。

SQL を使用する PL/I アプリケーションでのホスト変数の宣言

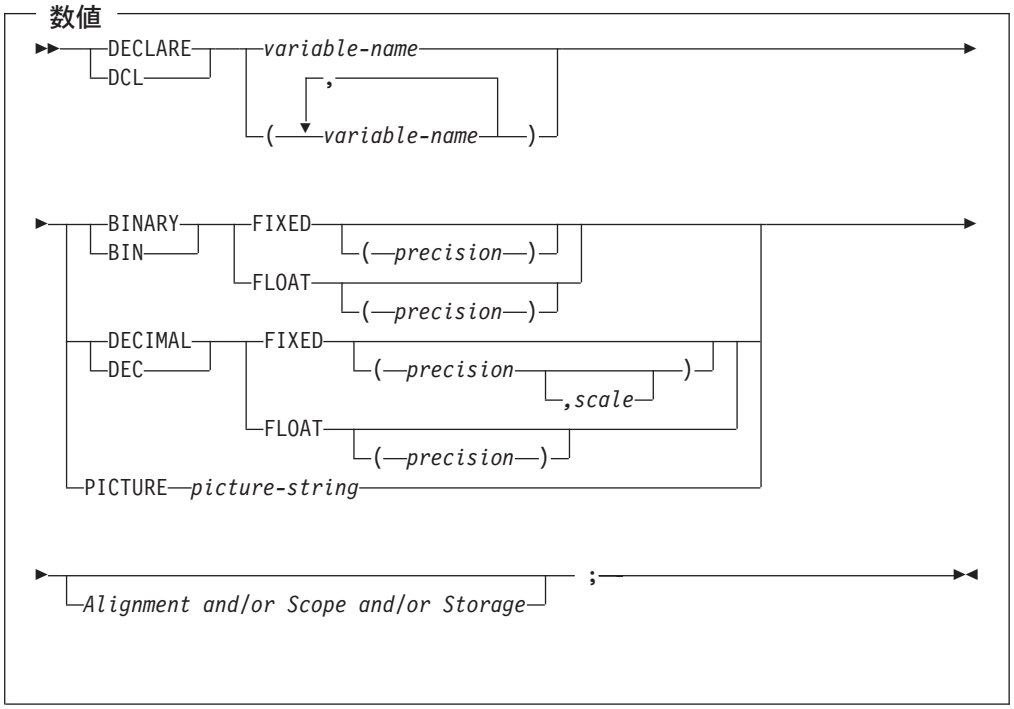
PL/I プリコンパイラーは、有効な PL/I 宣言のサブセットだけを有効なホスト変数宣言として認識します。

変数の名前とデータ属性だけがプリコンパイラーによって使用されます。境界合わせ、有効範囲、および記憶域属性は無視されます。境界合わせ、有効範囲、および記憶域属性が無視される場合であっても、その使い方にいくつかの制約があり、もし無視されると、プリコンパイラーによって生成される PL/I ソース・コードをコンパイルするとき問題が起こることがあります。その制約とは、次のものです。

- EXTERNAL 有効範囲属性と STATIC 記憶属性をもつ宣言は INITIAL 記憶属性ももたなければなりません。
- BASED 記憶属性をコーディングする場合は、そのあとに PL/I 要素ロケーター式を置かなければなりません。

SQL を使用する PL/I アプリケーションでの数値ホスト変数

次の図は、有効なスカラー数値ホスト変数宣言の構文を示しています。



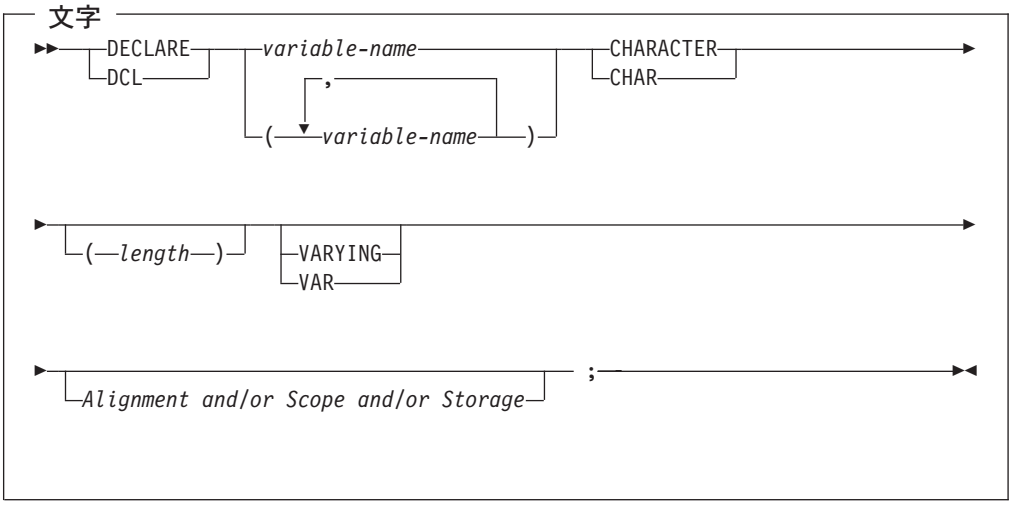
注:

1. (BINARY、BIN、DECIMAL、または DEC) および (FIXED または FLOAT) および (precision, scale) はどの順序でも指定できます。

2. '9...9V9...R' の形式の picture-string (ピクチャー列) は数値のホスト変数を示しています。 R が必要です。任意選択の V は、暗黙の小数点を示しています。
3. 'S9...9V9...9' の形式の picture-string (ピクチャー列) は符号先行の分離ホスト変数を示しています。 S が必要です。任意選択の V は、暗黙の小数点を示しています。

SQL を使用する PL/I アプリケーションでの文字ホスト変数

次の図は、有効なスカラー文字ホスト変数の構文を示しています。



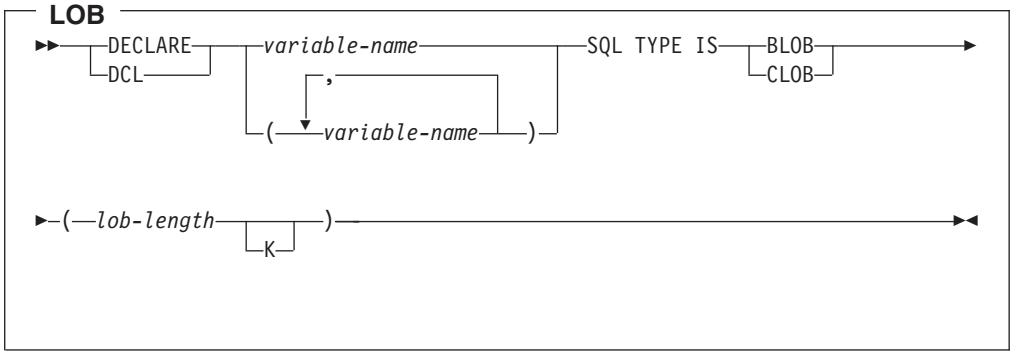
注:

1. VARYING または VAR を指定しない場合、*length (長さ)* は 32766 以下の整数定数でなければなりません。
2. VARYING または VAR を指定する場合、*length (長さ)* は 32740 以下の定数でなければなりません。

SQL を使用する PL/I アプリケーションでの LOB ホスト変数

PL/I には、LOB (ラージ・オブジェクト) の SQL データ・タイプに対応する変数がありません。これらのデータ・タイプで使用するホスト変数を作成するには、SQL TYPE IS 文節を使用します。SQL プリコンパイラーは、この宣言を出力ソース・メンバー中、PL/I 言語構造に置き換えます。

次の図は、有効な LOB ホスト変数の構文を示しています。



注:

1. BLOB および CLOB の場合、1 <= lob-length <= 32,766
2. SQL TYPE IS、BLOB、CLOB は、大文字小文字混合にすることができます。

BLOB 例 :

次のように宣言すると、

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

以下の構造を生成します。

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

CLOB 例 :

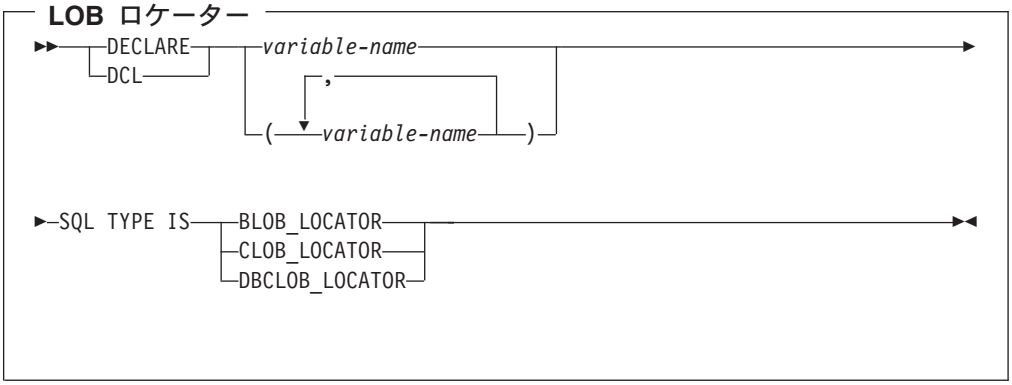
次のように宣言すると、

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

以下の構造を生成します。

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

次の図は、有効な LOB ロケーターの構文を示しています。



注: SQL TYPE IS、BLOB_LOCATOR、CLOB_LOCATOR、DBCLOB_LOCATOR は、大文字小文字混合にすることができます。

CLOB ロケーターの例:

次のように宣言すると、

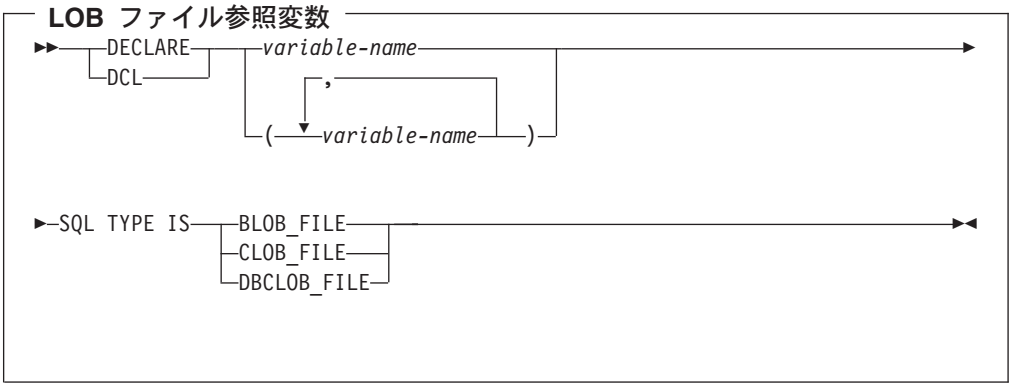
```
DCL MY_LOCATOR SQL TYPE IS CLOB_LOCATOR;
```

以下の構造を生成します。

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

BLOB ロケーターおよび DBCLOB ロケーターの構文は、似ています。

次の図は、有効な LOB ファイル参照変数の構文を示しています。



注: SQL TYPE IS、BLOB_FILE、CLOB_FILE、および DBCLOB_FILE は大文字小文字混合にすることができます。

CLOB ファイル参照の例:

次のように宣言すると、

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

以下の構造を生成します。

```
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 ロケーターおよび DBCLOB ロケーターの構文は、似ています。

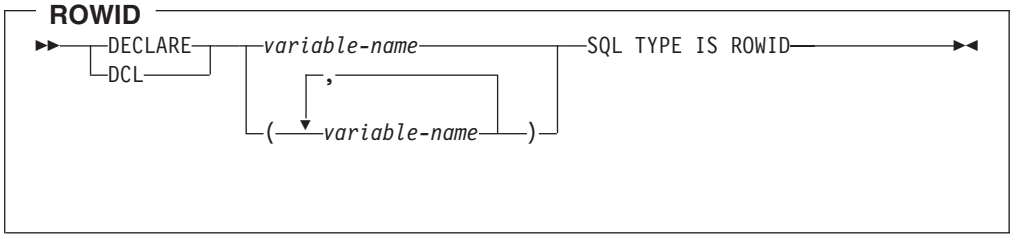
プリコンパイラーは、次のファイル・オプション定数に対する宣言を生成します。

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

これらの値の詳細については、「SQL プログラミング 概念」の『LOB ファイル参照変数』を参照してください。

SQL を使用する PL/I アプリケーションでの ROWID ホスト変数

PL/I には、ROWID の SQL データ・タイプに対応する変数がありません。このデータ・タイプで使用するホスト変数を作成するには、SQL TYPE IS 文節を使用します。SQL プリコンパイラーは、この宣言を出力ソース・メンバー中、PL/I 言語構造に置き換えます。



注: SQL TYPE IS ROWID は、大文字小文字混合にすることができます。

ROWID の例

次のように宣言すると、

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

以下の構造を生成します。

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

SQL を使用する PL/I アプリケーションでのホスト構造の使用

PL/I プログラムの中では、**ホスト構造**が定義できます。これは一組の基本 PL/I 変数に名前を付けたものです。ホスト構造名は、その従属レベルに基本 PL/I 変数の名前が指定されているグループ名にすることができます。以下に、例を示します。

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

この例では、B は基本項目 C1 と C2 から成るホスト構造の名前です。

構造名は、スカラーのリストの省略表現として使用することができます。ホスト変数は構造名で修飾することができます (たとえば、STRUCTURE.FIELD)。ホスト構造は 2 レベルに限定されます。(たとえば、上記のホスト構造の例では、SQL の中で A を参照することはできません。) 構造に中間レベルの構造を含めることはできません。上記の例では、A はホスト変数として使用することも、SQL ステートメントの中で参照することもできません。しかし、B は第 1 レベルの構造です。B は SQL ステートメントの中で参照することができます。SQL データのホスト構造は 2 レベルの深さであり、一連のホスト変数に名前を付けたものと考えることができます。ホスト構造を定義しておけば、SQL ステートメントの中でいくつかのホスト変数 (ホスト構造を構成するホスト変数の名前) を個別に参照せずに一括して参照することができます。

たとえば、次のようにコーディングすれば、テーブル CORPDATA.EMPLOYEE から選択した行のすべての列の値を検索することができます。

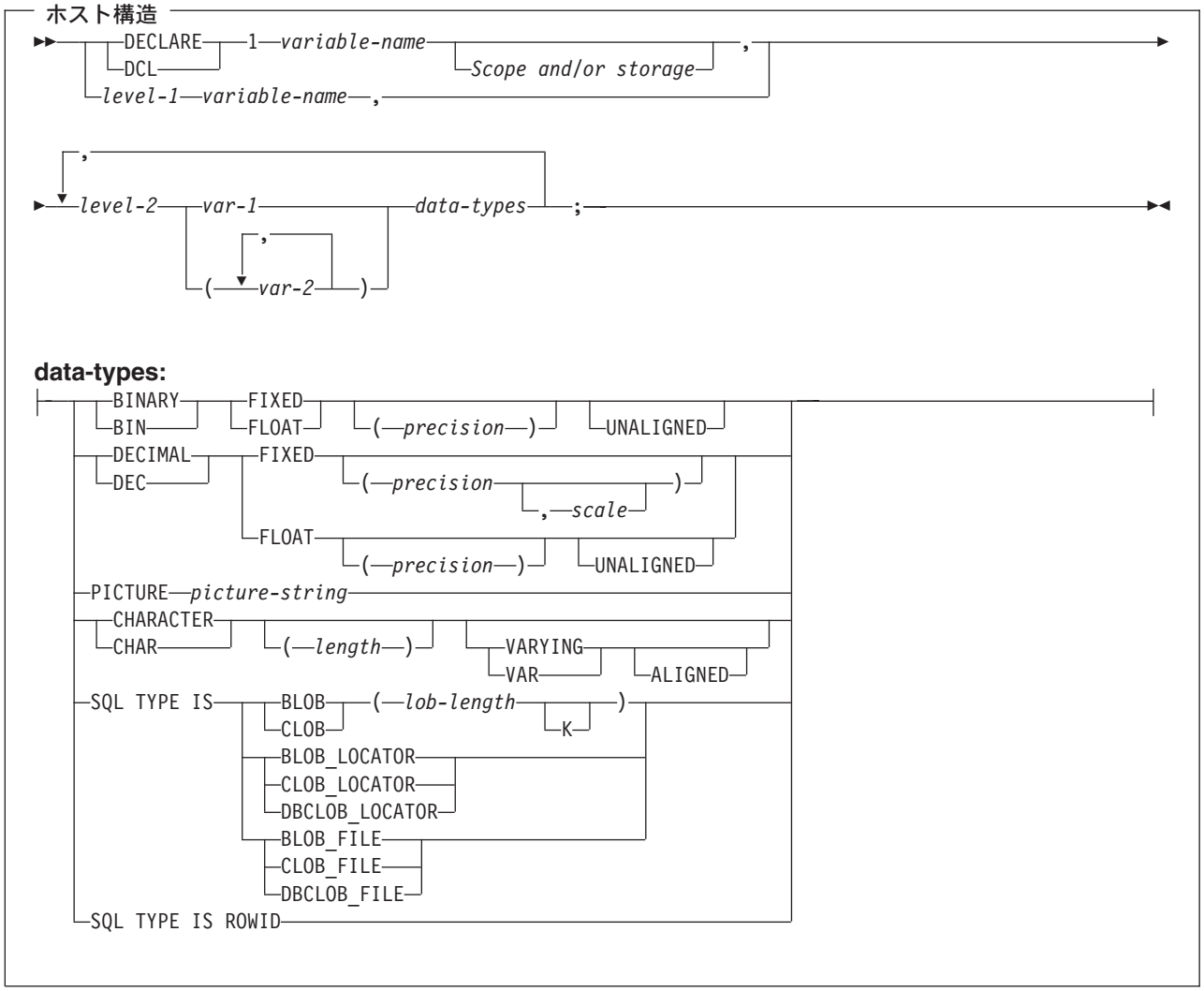
```
DCL 1 PEMPL,
      5 EMPNO    CHAR(6),
      5 FIRSTNME CHAR(12) VAR,
      5 MIDINIT  CHAR(1),
      5 LASTNAME CHAR(15) VAR,
      5 WORKDEPT CHAR(3);
 ...
EMPID = '000220';
 ...
   EXEC SQL
    SELECT *
    INTO :PEMPL
    FROM CORPDATA.EMPLOYEE
    WHERE EMPNO = :EMPID;
```

詳細については、以下のセクションを参照してください。

- 88 ページの『SQL を使用する PL/I アプリケーションでのホスト構造』
- 89 ページの『SQL を使用する PL/I アプリケーションでのホスト構造標識配列』

SQL を使用する PL/I アプリケーションでのホスト構造

次の図は、有効なホスト構造宣言の構文を示しています。

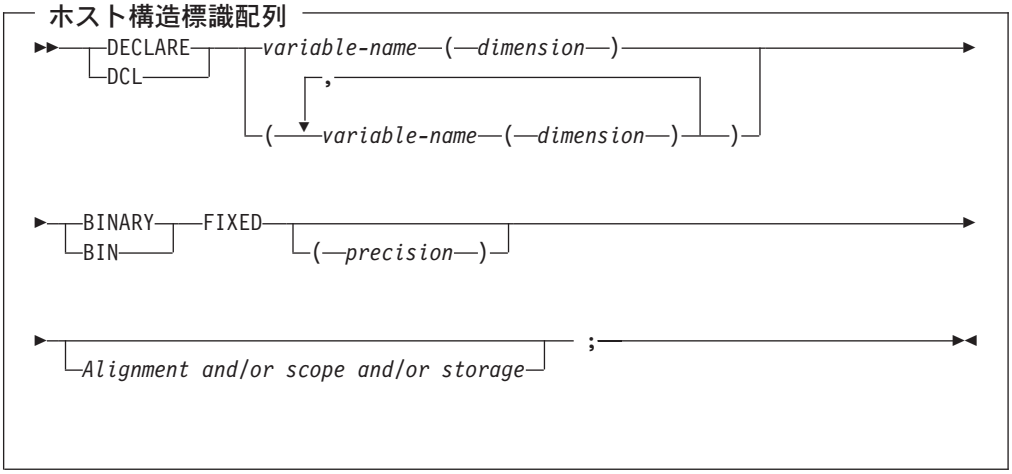


注:

1. level-1 (レベル-1) は、中間レベル構造があることを示しています。
2. level-1 (レベル-1) は、1 から 254 までの整数定数でなければなりません。
3. level-2 (レベル-2) は、2 から 255 までの整数定数でなければなりません。
4. 数値ホスト変数、文字ホスト変数、LOB ホスト変数、および ROWID ホスト変数の宣言の詳細については、数値ホスト変数、文字ホスト変数、LOB ホスト変数、および ROWID ホスト変数に関する注意事項を参照してください。

SQL を使用する PL/I アプリケーションでのホスト構造標識配列

次の図は、有効な標識配列の構文を示しています。



注: 次元は 1 から 32766 までの整数定数でなければなりません。

SQL を使用する PL/I アプリケーションでのホスト構造配列の使用

PL/I プログラムでは、ホスト構造配列を定義することができます。このような例では、以下の条件が該当します。

- B_ARRAY は、項目 C1_VAR と C2_VAR が入っているホスト構造配列の名前です。
- B_ARRAY は修飾できません。
- B_ARRAY はブロック化形式の FETCH および INSERT ステートメントでのみ使用できます。
- B_ARRAY 内のすべての項目は有効なホスト変数でなければなりません。
- C1_VAR および C2_VAR は、SQL ステートメントでは有効なホスト変数ではありません。構造に中間レベルの構造を含めることはできません。A_STRUCT に次元属性を入れることはできません。

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

CORPDATA.DEPARTMENT テーブルから 10 行検索するには、次のようにコーディングします。

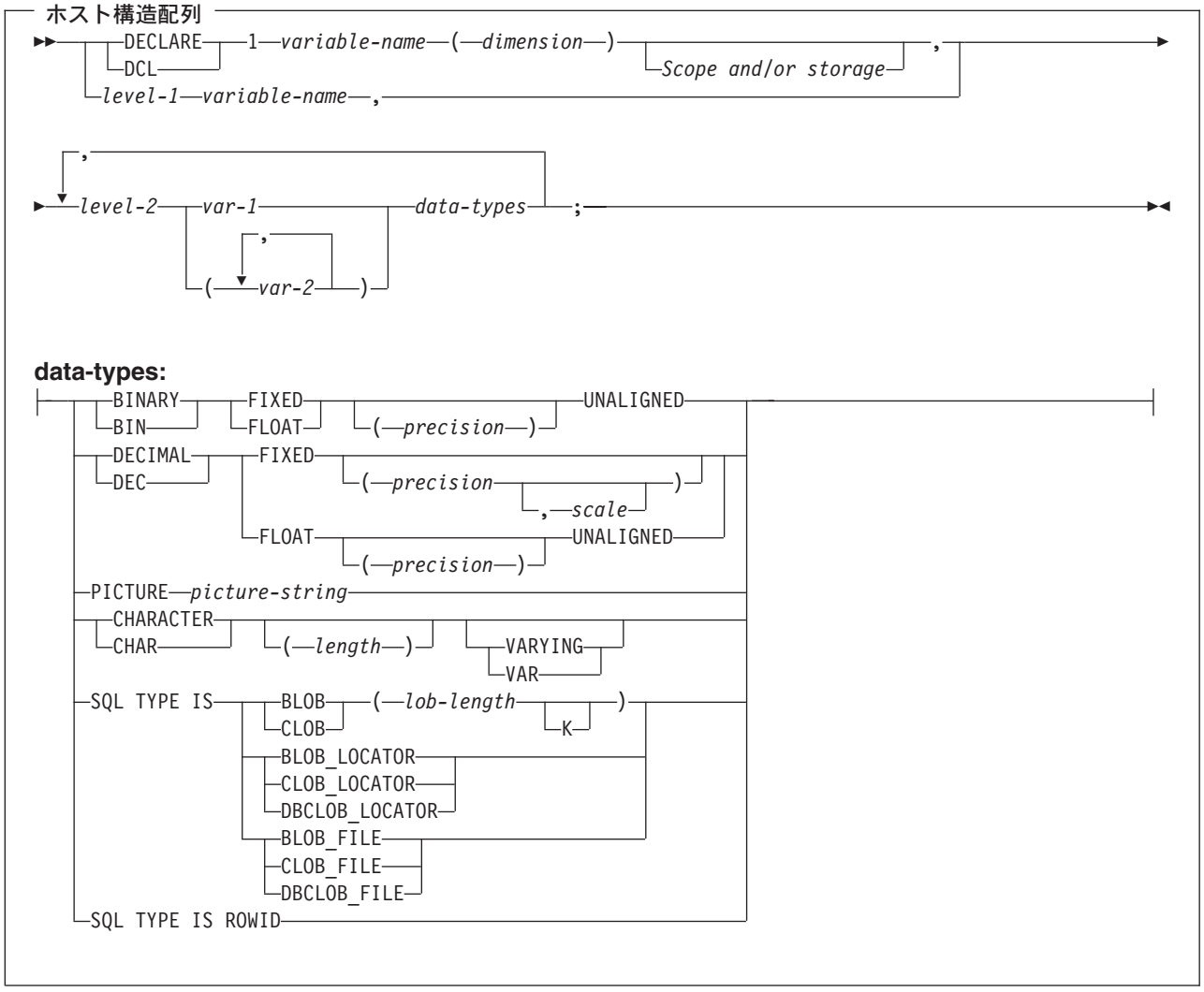
```
DCL 1 DEPT(10),
      5 DEPTPNO CHAR(3),
      5 DEPTNAME CHAR(29) VAR,
      5 MGRNO CHAR(6),
      5 ADMRDEPT CHAR (3);
DCL 1 IND_ARRAY(10),
      5 INDS(4) FIXED BIN(15);
EXEC SQL
  DECLARE C1 CURSOR FOR
     SELECT *
```

```
      FROM CORPDATA.DEPARTMENT;

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

詳細については、『SQL を使用する PL/I アプリケーションでのホスト構造配列』を参照してください。

SQL を使用する PL/I アプリケーションでのホスト構造配列

以下の構文図は、有効な構造配列宣言の構文を示しています。



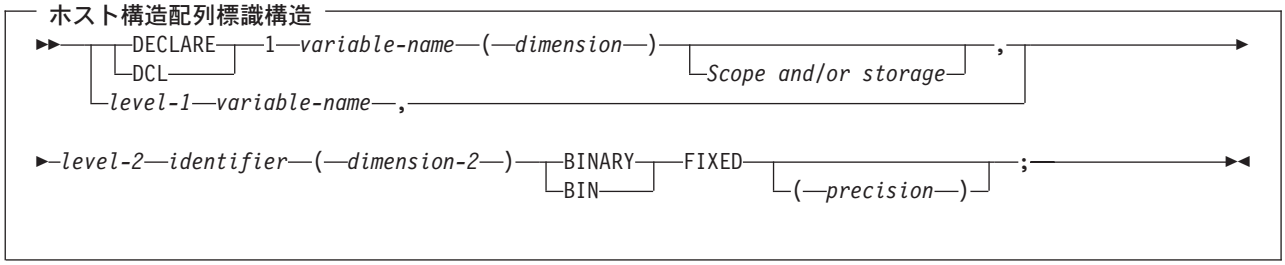
注:

1. level-1 (レベル-1) は、中間レベル構造があることを示しています。
2. level-1 (レベル-1) は、1 から 254 までの整数定数でなければなりません。
3. level-2 (レベル-2) は、2 から 255 までの整数定数でなければなりません。

4. 数値ホスト変数、文字ホスト変数、LOB ホスト変数、および ROWID ホスト変数の宣言の詳細については、数値ホスト変数、文字ホスト変数、LOB ホスト変数、および ROWID ホスト変数に関する注意事項を参照してください。
5. dimension (次元) は 1 から 32767 までの整数定数でなければなりません。

SQL を使用する PL/I アプリケーションでのホスト構造配列標識

次の図は、有効なホスト構造配列標識構造宣言の構文を示しています。



注:

1. level-1 (レベル-1) は、中間レベル構造があることを示しています。
2. level-1 (レベル-1) は、1 から 254 までの整数定数でなければなりません。
3. level-2 (レベル-2) は、2 から 255 までの整数定数でなければなりません。
4. dimension-1 (次元 1) および dimension-2 (次元 2) は、1 から 32767 までの整数定数でなければなりません。

SQL を使用する PL/I アプリケーションでの外部ファイル記述の使用

PL/I %INCLUDE ディレクティブを使用すると、外部記述ファイルの定義をソース・プログラムに組み入れることができます。SQL で使用するときは、特定形式の %INCLUDE ディレクティブだけが SQL プリコンパイラーによって認識されます。そのディレクティブの形式は、次の 3 つの要素またはパラメーター値を持っていなければなりません。そうでないと、プリコンパイラーはそのディレクティブを無視します。必要な 3 つの要素とは、*ファイル名*、*様式名*、および*要素タイプ* です。このほかに、SQL プリコンパイラーによってサポートされる任意選択の要素が 2 つあります。それは接頭部名と COMMA です。

構造は、通常、レコードまたはキー構造の最後のデータ要素で終わります。しかし、%INCLUDE ディレクティブの中で COMMA 要素の指定があるときは、構造はそこで終わりません。

「*DB2 UDB for iSeries SQL プログラミング 概念*」の『DB2 UDB for iSeries サンプル・テーブル』に記載されているサンプル・テーブル DEPARTMENT の定義を組み込むには、次のようにコーディングすることができます。

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

上記の例で、TDEPT_STRUCTURE という名前のホスト構造は、4 つのフィールドをもつことが定義されます。これらのフィールドは DEPTNO、DEPTNAME、MGRNO、および ADMRDEPT となります。

装置ファイルの場合は、INDARA の指定がなく、ファイルに標識が入っているときには、その宣言をホスト構造配列として使用できません。生成された構造には標識域が組み入れられ、その標識域があるために記憶域が連続しなくなります。

```
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;
```

注: DATE、TIME、および TIMESTAMP の各列からはホスト変数定義が生成され、これらは DATE、TIME、および TIMESTAMP 列と同じ比較と割り当ての規則を使用して、SQL によって扱われます。たとえば、日付ホスト変数は、DATE 列または日付の有効な表現である文字ストリングとだけ比較することができます。

精度が 15 を超える 10 進数フィールドとゾーン・フィールド、および位取りフィールドがゼロでない 2 進数は、PL/I では文字フィールド変数に対応していますが、SQL では、これらのフィールドは数値であると見なされます。

GRAPHIC と VARGRAPHIC は PL/I で文字変数にマップされますが、SQL はこれらを GRAPHIC ホスト変数および VARGRAPHIC ホスト変数と見なします。GRAPHIC 列または VARGRAPHIC 列が UCS-2 CCSID を持つ場合、生成されるホスト変数には UCS-2 CCSID が割り当てられます。

SQL データ・タイプと PL/I データ・タイプの対応関係の判別

プリコンパイラーは、次の表に基づいて、ホスト変数のベース SQLTYPE とベース SQLLEN を判断します。ホスト変数が標識変数と一緒に記載されているときは、その SQLTYPE はベース SQLTYPE に 1 を加えたものです。

表 5. PL/I 宣言と代表的 SQL データ・タイプとの対応関係

PL/I データ・タイプ	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
BIN FIXED(p) (p は 1 から 15 までの範囲)	500	2	SMALLINT
BIN FIXED(p) (p は 16 から 31 までの範囲)	496	4	INTEGER
DEC FIXED(p,s)	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s)

表 5. PL/I 宣言と代表的 SQL データ・タイプとの対応関係 (続き)

PL/I データ・タイプ	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
BIN FLOAT(p) (p は 1 から 24 までの範囲)	480	4	FLOAT (単精度)
BIN FLOAT(p) (p は 25 から 53 までの範囲)	480	8	FLOAT (倍精度)
DEC FLOAT(m) (m は 1 から 7 までの範囲)	480	4	FLOAT (単精度)
DEC FLOAT(m) (m は 8 から 16 までの範囲)	480	8	FLOAT (倍精度)
PICTURE ピクチャー列 (数値)	488	バイト 1 には p、バイト 2 には s	NUMERIC(p,s)
PICTURE ピクチャー列 (符号先行の分離)	504	バイト 1 には p、バイト 2 には s	正確に対応するものなし。NUMERIC(p,s) を使用。
CHAR(n)	452	n	CHAR(n)
CHAR(n) VARYING (n < 255)	448	n	VARCHAR(n)
CHAR(n) VARYING (n > 254)	456	n	VARCHAR(n)

下表を参照すると、各 SQL データ・タイプに対応する PL/I データ・タイプを判別することができます。

表 6. SQL データ・タイプと代表的な PL/I 宣言との対応関係

SQL データ・タイプ	対応する PL/I 宣言	説明と注
SMALLINT	BIN FIXED(p)	p は 1 から 15 までの正の整数です。
INTEGER	BIN FIXED(p)	p は 16 から 31 までの正の整数です。
BIGINT	正確に対応するものなし	DEC FIXED(18) を使用。
DECIMAL(p,s) または NUMERIC(p,s)	DEC FIXED(p) または DEC FIXED(p,s) または PICTURE ピクチャー列	*s* (位取り係数) および *p* (精度) は正の整数です。*p* は 1 から 31 までの正の整数です。*s* は 0 から *p* までの正の整数です。
FLOAT (単精度)	BIN FLOAT(p) または DEC FLOAT(m)	*p* は 1 から 24 までの正の整数です。 *m* は 1 から 7 までの正の整数です。
FLOAT (倍精度)	BIN FLOAT(p) または DEC FLOAT(m)	*p* は 25 から 53 までの正の整数です。 *m* は 8 から 16 までの正の整数です。

表 6. SQL データ・タイプと代表的な PL/I 宣言との対応関係 (続き)

SQL データ・タイプ	対応する PL/I 宣言	説明と注
CHAR(n)	CHAR(n)	*n* は 1 から 32766 までの正の整数です。
VARCHAR(n)	CHAR(n) VAR	*n* は 1 から 32740 までの正の整数です。
BLOB	なし	SQL TYPE IS を使用して BLOB を宣言します。
CLOB	なし	SQL TYPE IS を使用して CLOB を宣言します。
GRAPHIC(n)	サポートなし	サポートなし。
VARGRAPHIC(n)	サポートなし	サポートなし。
DBCLOB	なし	SQL TYPE IS を使用して DBCLOB を宣言します。
DATE	CHAR(n)	形式が *USA、*JIS、*EUR、または *ISO のときは、*n* は少なくとも 10 文字にする必要があります。形式が *YMD、*DMY、または *MDY のときは、*n* は少なくとも 8 文字にする必要があります。形式が *JUL のときは、*n* は少なくとも 6 文字にする必要があります。
TIME	CHAR(n)	*n* は少なくとも 6 文字に、秒を含める場合は、 *n* は少なくとも 8 文字にする必要があります。
TIMESTAMP	CHAR(n)	*n* は少なくとも 19 が必要。マイクロ秒を全桁の精度で含める場合には、*n* は少なくとも 26 にする必要があります。 *n* が 26 未満のときは、マイクロ秒部分で切り捨てが起こります。
DATALINK	サポートなし	サポートなし
ROWID	なし	SQL TYPE IS を使用して ROWID を宣言します。

SQL を使用する PL/I アプリケーションでの標識変数の使用

標識変数は 2 バイトの整数です (BIN FIXED(p)、ただし、p は 1 から 15 までです)。ホスト構造をサポートするために標識構造 (ハーフワードの整数変数の配列として定義されているもの) を指定することもできます。検索されるとき、標識変数はその対応するホスト変数にヌル値が割り当てられているかどうかを示すために使用されます。列に割り当てるときには、ヌル値を割り当てるべきであることを示すために負の標識変数が使用されます。

詳細については、「SQL 解説書」の『標識変数』を参照してください。

標識変数の宣言の仕方はホスト変数の場合と同じであり、これらの 2 つの変数の宣言をプログラマーに適切と思われる方法で組み合わせることができます。

例:

次のステートメントがあるとします。

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

変数は次のように宣言することができます。

```
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;
```

構造パラメーター受け渡し技法による PL/I での相違

PL/I プリコンパイラーは、可能ならば、構造パラメーター受け渡し技法の使用を試みます。この構造パラメーター受け渡し技法を使用すると、SQL を使用するほとんどの PL/I プログラムのパフォーマンスが向上します。プリコンパイラーは、以下の条件に該当する場合、各ホスト変数が分離パラメーターになっているコードを生成します。

- 外部テキストをソース・プログラムにコピーする PL/I %INCLUDE コンパイラー指示が見つかった場合。
- ステートメントで参照されているホスト変数のデータ長が 32703 を超えている場合。 SQL は構造のうち 64 バイトを使用するため、データ構造の最大長は 32703 + 64 = 32767 となります。
- PL/I プリコンパイラーがユーザー定義名の PL/I 限度を超えると推定した場合。
- 符号先行分離ホスト変数が SQL ステートメントのホスト変数リストに見つかった場合。

構造パラメーター受け渡し技法の詳細については、「*DB2 UDB for iSeries データベース・パフォーマンスおよび Query 最適化*」の『データベース・アプリケーション設計のヒント: 構造パラメーター受け渡し方法の使用』を参照してください。

第 5 章 RPG for iSeries アプリケーションでの SQL ステートメントのコーディング方法

RPG for iSeries ライセンス・プログラムは RPG II プログラムと RPG III プログラムを共にサポートします。SQL ステートメントは、RPG III プログラムの中でしか使用できません。RPG II と報告書簡易作成機能はサポートされません。本書で RPG という場合は、RPG III または ILE RPG のいずれかを指しています。

この章では、SQL ステートメントを RPG for iSeries プログラムに組み込む場合の固有のアプリケーションおよびコーディング上の要件について説明します。ホスト変数に必要な要件についても説明します。

詳細については、以下のセクションを参照してください。

- 98 ページの『SQL を使用する RPG for iSeries アプリケーションでの SQL 連絡域の定義』
- 99 ページの『SQL を使用する RPG for iSeries アプリケーションでの SQL 記述子域の定義』
- 99 ページの『SQL を使用する RPG for iSeries アプリケーションでの SQL ステートメントの組み込み』
- 102 ページの『SQL を使用する RPG for iSeries アプリケーションでのホスト変数の使用』
- 102 ページの『SQL を使用する RPG for iSeries アプリケーションでのホスト構造の使用』
- 103 ページの『SQL を使用する RPG for iSeries アプリケーションでのホスト構造配列の使用』
- 104 ページの『SQL を使用する RPG for iSeries アプリケーションでの外部ファイル記述の使用』
- 106 ページの『SQL データ・タイプと RPG for iSeries データ・タイプの対応関係の判別』
- 109 ページの『SQL を使用する RPG for iSeries アプリケーションでの標識変数の使用』
- 109 ページの『構造パラメーター受け渡し技法による RPG for iSeries での相違』
- 110 ページの『呼び出された SQL を使用する RPG for iSeries プログラムの正しい終了方法』

SQL ステートメントの使い方を示した詳しいサンプル RPG for iSeries プログラムは、『付録 A. DB2 UDB for iSeries ステートメントを使用するサンプル・プログラム』に記載されています。

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

RPG を使用するプログラミングの詳細については、「RPG/400® 使用者の手引き」 および「RPG/400 解説書」 を参照してください。

SQL を使用する RPG for iSeries アプリケーションでの SQL 連絡域の定義

SQL プリコンパイラーは、RPG iSeries 用プログラムの最初の演算仕様の前の入力仕様に SQLCA を自動的に入れます。 INCLUDE SQLCA をソース・プログラムにコーディングする必要はありません。ソース・プログラムに INCLUDE SQLCA を指定した場合、そのステートメントは受け入れられますが、余分なものとして扱われます。 SQLCA を RPG for iSeries 用に定義すると、次のようになります。

```
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
```

注: RPG for iSeries では変数名は 6 文字までに制限されています。したがって、標準 SQLCA 名は 6 文字の長さに変更されています。RPG for iSeries では、配列を拡張仕様にも定義しておかないと、配列をデータ構造に定義することはできません。SQLERR は、要素の名前に使用される SQLER1 から SQLER6 までの文字として定義されます。

詳細については、「SQL 解説書」の『SQL 連絡域』を参照してください。

SQL を使用する RPG for iSeries アプリケーションでの SQL 記述子域の定義

SQLDA を必要とするステートメントには、次のものがあります。

EXECUTE...USING DESCRIPTOR *記述子名*

FETCH...USING DESCRIPTOR *記述子名*

OPEN...USING DESCRIPTOR *記述子名*

CALL...USING DESCRIPTOR *記述子名*

DESCRIBE *ステートメント名* INTO *記述子名*

DESCRIBE TABLE *ホスト変数* INTO *記述子名*

PREPARE *ステートメント名* INTO *記述子名*

SQLCA と異なり、SQLDA を 2 つ以上プログラムの中に置くことができ、また SQLDA の名前は有効であれば、どの名前でも使えます。

動的 SQL は高度なプログラミング技法です。これについては、「*DB2 UDB for iSeries SQL プログラミング 概念*」の『動的 SQL アプリケーション』で説明します。動的 SQL を使用すると、ユーザーのプログラムはその実行と平行して SQL ステートメントを作成し、実行させることができます。動的に実行される変数 SELECT リスト (すなわち、照会の一部として返されるデータのリスト) を指定する SELECT ステートメントには、SQL 記述子域 (SQLDA) が必要です。これは、SELECT の結果を受け入れるために割り振るべき変数の数とタイプが事前に予測できないからです。

SQLDA ではポインター変数を使用しますが、これは RPG for iSeries ではサポートされないので、INCLUDE SQLDA ステートメントは RPG for iSeries プログラムでは指定できません。 SQLDA を使用するためには、C、COBOL、PL/I、または ILE RPG の各プログラムでセットアップして RPG プログラムに渡さなければなりません。

SQLDA の詳細については、「*SQL 解説書*」の『SQL 記述子域』を参照してください。

SQL を使用する RPG for iSeries アプリケーションでの SQL ステートメントの組み込み

RPG iSeries プログラムの中にコーディングする SQL ステートメントは、演算部分に置かなければなりません。このためには C を 6 桁目に入れる必要があります。SQL ステートメントは、明細演算にも、合計計算にも、RPG for iSeries サブルーチンにも置くことができます。 SQL ステートメントは、RPG for iSeries ステートメントのロジックに基づいて実行されます。

キーワード EXEC は SQL ステートメントの始まりを示します。 EXEC はソース・ステートメントの 8 桁目から 16 桁目までに置き、7 桁目にスラッシュ (/) を置かなければなりません。 SQL ステートメントを 17 桁目から始めて、74 桁目まで続けることができます。

キーワード END-EXEC は SQL ステートメントの終わりを示します。 END-EXEC はソース・ステートメントの 8 桁目から 16 桁目までに置き、7 桁目にスラッシュ (/) を置かなければなりません。 17 桁目から 74 桁目まではブランクにしておきます。

SQL ステートメントでは大文字と小文字の両方が使用できます。

詳細については、以下のセクションを参照してください。

- 『例: SQL を使用する RPG for iSeries アプリケーションでの SQL ステートメントの組み込み』
- 『SQL を使用する RPG for iSeries アプリケーションでの注記』
- 『SQL を使用する RPG for iSeries アプリケーションでの SQL ステートメントの継続』
- 101 ページの『SQL を使用する RPG for iSeries アプリケーションでのコードの組み込み』
- 101 ページの『SQL を使用する RPG for iSeries アプリケーションでの順序番号』
- 101 ページの『SQL を使用する RPG for iSeries アプリケーションでの名前』
- 101 ページの『SQL を使用する RPG for iSeries アプリケーションでのステートメント・ラベル』
- 101 ページの『SQL を使用する RPG for iSeries アプリケーションでの WHENEVER ステートメント』

例: SQL を使用する RPG for iSeries アプリケーションでの SQL ステートメントの組み込み

RPG for iSeries プログラムの中にコーディングされる UPDATE ステートメントをコーディングすると、次のようになります。

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

SQL を使用する RPG for iSeries アプリケーションでの注記

SQL の注記 (--) の他に、RPG for iSeries の注記は、SQL ステートメント内のブランクが許される場所にはどこにでも入れることができます。ただし、キーワードの EXEC と SQL の間には入れられません。 RPG for iSeries の注記を SQL ステートメントに組み込むときは、7 桁目にアスタリスク (*) を置きます。

SQL を使用する RPG for iSeries アプリケーションでの SQL ステートメントの継続

SQL ステートメントを収めるために追加レコードが必要なときは、9 桁目から 74 桁目が使用できます。 7 桁目は + (正符号) にし、8 桁目はブランクにしなければなりません。

DBCS データを含む定数は、継続される行の 75 桁目にシフトイン文字を入れ、継続行の 8 桁目にシフトアウト文字を入れることによって、複数行にわたって継続さ

せることができます。この SQL ステートメントの
G'<AABBCCDDEEFFGGHHIIJJKK>' はグラフィック定数として有効です。

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

SQL を使用する RPG for iSeries アプリケーションでのコードの組み込み

SQL ステートメントと RPG for iSeries 演算仕様は、SQL ステートメントを組み込むことによって取り込むことができます。

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

/COPY ステートメントを使用すると、SQL ステートメントまたは RPG for iSeries 仕様を取り込むことができます。

SQL を使用する RPG for iSeries アプリケーションでの順序番号

SQL プリコンパイラーによって生成されるソース・ステートメントの順序番号は、CRTSQLRPG コマンドの OPTION パラメーターの *NOSEQSRC/*SEQSRC キーワードに基づきます。 *NOSEQSRC が指定されると、入力ソース・メンバーからの順序番号が使用されます。*SEQSRC を指定すると、順序番号は 000001 から始まり、1 ずつ増えます。

SQL を使用する RPG for iSeries アプリケーションでの名前

有効な RPG 変数名であれば、どの名前でもホスト変数に使えますが、次のような制約を受けます。

'SQ'、'SQL'、'RDI'、または 'DSN' で始まるホスト変数名や外部入り口名は、使用してはなりません。これらの名前はデータベース・マネージャー用に予約されています。

SQL を使用する RPG for iSeries アプリケーションでのステートメント・ラベル

どの SQL ステートメントの場合も、その前に TAG ステートメントを置くことができます。 TAG ステートメントは EXEC SQL の前の行にコーディングします。

SQL を使用する RPG for iSeries アプリケーションでの WHENEVER ステートメント

GOTO 文節の対象となるものは、TAG ステートメントのラベルでなければなりません。 GOTO/TAG の有効範囲に従わなければなりません。

SQL を使用する RPG for iSeries アプリケーションでのホスト変数の使用

SQL ステートメントの中で使用するホスト変数はいずれも明示的に宣言しなければなりません。 LOB ホスト変数および ROWID ホスト変数は、RPG for iSeries ではサポートされません。

RPG for iSeries に組み込まれた SQL は、ホスト変数を識別するために SQL の BEGIN DECLARE SECTION および END DECLARE SECTION ステートメントを使用しません。これらのステートメントをソース・プログラムに入れてはなりません。

SQL ステートメントの中のホスト変数はいずれも、その前にコロン (:) を付けなければなりません。

ホスト変数の名前は、プログラムの中で固有でなければなりません。

詳細については、『SQL を使用する RPG for iSeries アプリケーションでのホスト変数の宣言』を参照してください。

SQL を使用する RPG for iSeries アプリケーションでのホスト変数の宣言

SQL RPG for iSeries プリコンパイラーは、有効なホスト変数宣言として RPG for iSeries 宣言のサブセットしか認識しません。

RPG for iSeries で定義した変数はいずれも、次のものを除き SQL ステートメントの中で使用できます。

標識フィールド名 (*INxx)

テーブル

UDATE

UDAY

UMONTH

UYEAR

先読みフィールド

名前付き定数

ホスト変数として使用されるフィールドは、RPG for iSeries の CALL/PARM 機能を使用して SQL に渡されます。PARM の結果フィールドで使用できないフィールドは、ホスト変数として使用できません。

SQL を使用する RPG for iSeries アプリケーションでのホスト構造の使用

RPG for iSeries データ構造名は、データ構造にサブフィールドがあれば、**ホスト構造**名として使用できます。SQL ステートメントの中でデータ構造名を使用したときは、そのデータ構造を構成するサブフィールド名のリストを暗黙に指定したことになります。

データ構造にサブフィールドがないときは、そのデータ構造名は文字タイプのホスト変数です。データ構造は最大 9999 までにできるので、これにより、文字変数を 256 より大きくすることができます。

次の例では、BIGCHR はサブフィールドのない RPG for iSeries データ構造です。BIGCHR が参照された場合、SQL はそれを長さが 642 の文字ストリングとして扱います。

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

次の例では、PEMPL は EMPNO、FIRSTN、MIDINT、LASTNAME、および DEPTNO のサブフィールドから成るホスト構造の名前です。 PEMPL が参照されると、これらのサブフィールドが使用されます。たとえば、EMPLOYEE の最初の列は *EMPNO* に入れられ、2 番目の列は *FIRSTN* に入れられます (以下同様です)。

```
    *...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
```

SQL ステートメントを書くとき、サブフィールドに対する参照を修飾することができます。それには、データ構造の名前の後にピリオドとサブフィールドの名前を付けます。たとえば、PEMPL.MIDINT は MIDINT だけを指定したのと同じです。

SQL を使用する RPG for iSeries アプリケーションでのホスト構造配列の使用

ホスト構造は、オカレンス・データ構造として定義されます。オカレンス・データ構造は、複数行を取り出すときに SQL の FETCH ステートメントで使用することができます。このような例では、以下の条件が該当します。

- BARRAY 内のすべての項目は有効なホスト変数でなければなりません。
- BARRAY 内のすべての項目は連続していなければなりません。最初の FROM の位置は 1 行でなければならず、TO と FROM の位置はオーバーラップできません。
- 複数行用 FETCH およびブロック化 INSERT 以外のすべてのステートメントの場合、オカレンス・データ構造を使用すると、現行オカレンスが使用されます。複数行用 FETCH および INSERT の場合、オカレンスは 1 にセットします。

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

以下の例では、DEPT というホスト構造配列および複数行用 FETCH ステートメントを使用して、DEPARTMENT テーブルから 10 行を取り出しています。

```
    *...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
```

SQL を使用する RPG for iSeries アプリケーションでの外部ファイル記述の使用

SQL プリコンパイラーは、ILE RPG for iSeries コンパイラーと全く同じ方法で RPG for iSeries ソース・コードを処理します。このことは、プリコンパイラーがホスト変数の定義のために /COPY ステートメントを処理することを意味します。異なる名前が定義されているときは、外部記述ファイルのフィールド定義が取り出されて、名前が変更されます。データ構造の外部定義形式を使用すると、列名のコピーをとって、それをホスト変数として使用することができます。

次の例では、サンプル・テーブル DEPARTMENT が ILE RPG for iSeries プログラムの中でファイルとして使用されています。 SQL プリコンパイラーは DEPARTMENT のフィールド (列) 定義を取り出して、ホスト変数として使用します。

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

注: RPG for iSeries のステートメントを使用してファイルに入出力操作を行う場合にだけ、RPG プログラムの中のファイルに F 仕様をコーディングしてください。 SQL ステートメントだけを使用してファイルに入出力操作を行うときは、外部データ構造を使用すると、外部定義を組み込むことができます。

次の例では、サンプル・テーブルは外部データ構造として指定されています。SQL プリコンパイラーはフィールド (列) 定義をデータ構造のサブフィールドとして検索します。サブフィールド名はホスト変数名として、データ構造名 TDEPT はホスト構造名として使用できます。フィールド名は、長さが 6 文字を超えているので変更しなければなりません。

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

注: DATE、TIME、および TIMESTAMP の各列からはホスト変数定義が生成され、その定義は DATE、TIME、および TIMESTAMP 列と同じ比較と割り当ての規則を使用して、SQL によって扱われます。たとえば、日付ホスト変数は、DATE 列または日付の有効な表現である文字ストリングとだけ突き合わされて比較されます。

可変長の列からは固定長の文字ホスト変数定義が生成されますが、SQL はそれを可変長文字変数と見なします。

GRAPHIC 列と VARGRAPHIC 列は RPG for iSeries で文字変数にマップされますが、 SQL はこれらを GRAPHIC 変数および VARGRAPHIC 変数と見なします。GRAPHIC 列または VARGRAPHIC 列が UCS-2 CCSID を持つ場合、生成されるホスト変数には UCS-2 CCSID が割り当てられます。

別の例については、『SQL を使用する RPG for iSeries アプリケーションでのホスト構造配列の外部ファイル記述に関する考慮事項』を参照してください。

SQL を使用する RPG for iSeries アプリケーションでのホスト構造配列の外部ファイル記述に関する考慮事項

ファイルに浮動小数点フィールドがあるときは、そのファイルはホスト構造配列として使用できません。装置ファイルの場合は、INDARA の指定がなく、ファイルに標識が入っているときには、その宣言をホスト構造配列として使用できません。生成された構造には標識域が組み入れられ、その標識域があるために記憶域が連続しなくなります。

以下の例では、DEPARTMENT テーブルは RPG for iSeries プログラムに取り込まれて、ホスト構造配列を宣言するために使用されます。その後、複数行用 FETCH ステートメントが使用されて、10 行をホスト構造配列に取り出します。

```
*...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
```

SQL データ・タイプと RPG for iSeries データ・タイプの対応関係の判別

プリコンパイラーは、次の表に基づいて、ホスト変数のベース SQLTYPE とベース SQLLEN を判断します。ホスト変数が標識変数と一緒に記載されているときは、その SQLTYPE はベース SQLTYPE に 1 を加えたものです。

表 7. RPG for iSeries 宣言と代表的 SQL データ・タイプとの対応関係

RPG for iSeries データ・タイプ	43 桁目	52 桁目	その他の RPG for iSeries コーディング	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
データ構造サブフィールド	ブランク	ブランク	長さ = n (n ≤ 256)	452	n	CHAR(n)
データ構造 (サブフィールドなし)	適用外	適用外	長さ = n (n ≤ 9999)	452	n	CHAR(n)
入力フィールド	ブランク	ブランク	長さ = n (n ≤ 256)	452	n	CHAR(n)
計算結果フィールド	適用外	ブランク	長さ = n (n ≤ 256)	452	n	CHAR(n)
データ構造サブフィールド	B	0	長さ = 2	500	2	SMALLINT
データ構造サブフィールド	B	0	長さ = 4	496	4	INTEGER
データ構造サブフィールド	B	1〜4	長さ = 2	500	2	DECIMAL(4,s) (s = 52 桁目)
データ構造サブフィールド	B	1〜9	長さ = 4	496	4	DECIMAL(9,s) (s = 52 桁目)
データ構造サブフィールド	P	0 から 9 まで	長さ = n (n は 1 から 16 まで)	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (p = n*2-1、s = 52 桁目)
入力フィールド	P	0 から 9 まで	長さ = n (n は 1 から 16 まで)	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (p = n*2-1、s = 52 桁目)
入力フィールド	ブランク	0 から 9 まで	長さ = n (n は 1 から 30 まで)	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (p = n、s = 52 桁目)
入力フィールド	B	n = 2 であれば 0 から 4 まで、n = 4 であれば 0 から 9 まで	長さ = 2 または 4	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (n =2 であれば p =4、n =4 かつ s = 52 桁目であれば 9)
計算結果フィールド	適用外	0 から 9 まで	長さ = n (n は 1 から 30 まで)	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (p = n、s = 52 桁目)

表 7. RPG for iSeries 宣言と代表的 SQL データ・タイプとの対応関係 (続き)

RPG for iSeries データ・タイプ	43 桁目	52 桁目	その他の RPG for iSeries コーディング	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
データ構造サブフィールド	ブランク	0 から 9 まで	長さ = n (n は 1 から 30 まで)	488	バイト 1 には p、バイト 2 には s	NUMERIC(p,s) (p = n、s = 52 桁目)

下表を参照すると、各 SQL データ・タイプに対応する RPG for iSeries データ・タイプを判別することができます。

表 8. SQL データ・タイプと代表的な RPG for iSeries 宣言との対応関係

SQL データ・タイプ	RPG for iSeries データ・タイプ	注
SMALLINT	データ構造のサブフィールド。サブフィールド仕様の 43 桁目が B、長さが 2、52 桁目が 0。	
INTEGER	データ構造のサブフィールド。サブフィールド仕様の 43 桁目が B、長さが 4、52 桁目が 0。	
BIGINT	正確に対応するものなし	サブフィールド仕様の 43 桁目は P、52 桁目は 0 とする。
DECIMAL	データ構造のサブフィールド。サブフィールド仕様の 43 桁目は P、52 桁目は 0 から 9 とする。 または 数値として定義され、データ構造のサブフィールドではないもの。	最大長は 16 (精度 30)、最大位取りは 9。
NUMERIC	データ構造のサブフィールド。サブフィールドの 43 桁目がブランク、52 桁目が 0 から 9 まで。	最大長は 30 (精度 30)、最大位取りは 9。
FLOAT (単精度)	正確に対応するものなし	上述した代替数値データ・タイプの 1 つを使用してください。
FLOAT (倍精度)	正確に対応するものなし	上述した代替数値データ・タイプの 1 つを使用してください。
CHAR(n)	データ構造のサブフィールドまたは入力フィールド。仕様の 43 桁目と 52 桁目がブランク。 または 小数部の桁なしで定義された計算結果フィールド。	n は 1 から 256 まで可能。
CHAR(n)	データ構造にサブフィールドがないデータ構造名。	*n* は 1 から 9999 まで可能。
VARCHAR(n)	正確に対応するものなし	予想される最大の VARCHAR 値が収まるだけの大きさの文字ホスト変数を使用してください。

表 8. SQL データ・タイプと代表的な RPG for iSeries 宣言との対応関係 (続き)

SQL データ・タイプ	RPG for iSeries データ・タイプ	注
BLOB	サポートなし	サポートなし
CLOB	サポートなし	サポートなし
GRAPHIC(n)	サポートなし	サポートなし
VARGRAPHIC(n)	サポートなし	サポートなし
DBCLOB	サポートなし	サポートなし
DATE	データ構造のサブフィールド。サブフィールド仕様の 52 桁目がブランク。 または 小数部の桁なしで定義されたフィールド。	形式が *USA、*JIS、*EUR、または *ISO のときは、長さは少なくとも 10 が必要。形式が *YMD、*DMY、または *MDY のときは、長さは少なくとも 8 が必要。形式が *JUL のときは、長さは少なくとも 6 が必要。
TIME	データ構造のサブフィールド。サブフィールド仕様の 52 桁目がブランク。 または 小数部の桁なしで定義されたフィールド。	長さは少なくとも 6 が必要。秒を含めるときは、長さは少なくとも 8 が必要。
TIMESTAMP	データ構造のサブフィールド。サブフィールド仕様の 52 桁目がブランク。 または 小数部の桁なしで定義されたフィールド。	長さは少なくとも 19 が必要。マイクロ秒を全桁の精度で含める場合は、長さは少なくとも 26 が必要。長さが 26 未満のときは、マイクロ秒部分で切り捨てが起こります。
DATALINK	サポートなし	サポートなし
ROWID	サポートなし	サポートなし

詳細については、『SQL を使用する RPG for iSeries アプリケーションでの RPG for iSeries 変数宣言と使用法に関する注意事項』を参照してください。

SQL を使用する RPG for iSeries アプリケーションでの RPG for iSeries 変数宣言と使用法に関する注意事項

SQL を使用する RPG for iSeries アプリケーションでの割り当て規則

RPG for iSeries は、精度と位取りをすべての数値タイプと関連付けます。 RPG for iSeries は、データがパック形式であるものとして、数値演算を定義します。すなわち、2 進数の変数が関係する演算は暗黙的にパック形式に変換されてから、演算が行われます (必要ならば、2 進数に逆変換されます)。データは暗黙の小数点位置に合わされて、SQL 演算が行われます。

SQL を使用する RPG for iSeries アプリケーションでの標識変数の使用

標識変数は 2 バイトの整数です (106 ページの表 7 の SMALLINT SQL データ・タイプの項目を参照)。

標識構造は、要素の長さが 4,0 の配列として変数を宣言し、43 桁目が B のデータ構造のサブフィールドとして配列名を宣言することによって定義することができます。検索されるとき、標識変数はその対応するホスト変数にヌル値が割り当てられているかどうかを示すために使用されます。列に割り当てるときには、ヌル値を割り当てるべきであることを示すために負の標識変数が使用されます。

詳細については、「SQL 解説書」の『標識変数』を参照してください。

標識変数の宣言の仕方はホスト変数の場合と同じであり、これらの 2 つの変数の宣言をプログラマーに適切と思われる方法で組み合わせることができます。

標識変数の使用例については、『例: SQL を使用する RPG for iSeries アプリケーションでの標識変数の使用』を参照してください。

例: SQL を使用する RPG for iSeries アプリケーションでの標識変数の使用

次のステートメントがあるとします。

```
*...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
```

変数は次のように宣言することができます。

```
*...1....+....2....+....3....+....4....+....5....+....6....+....7...*
I           DS
I                                        1   7 CLSCD
I                                    B   8   9 0DAY
I                                    B  10  11 0DAYIND
I                                       12  19 BGN
I                                    B  20  21 0BGNIND
I                                       22  29 END
I                                    B  30  31 0ENDIND
```

構造パラメーター受け渡し技法による RPG for iSeries での相違

SQL RPG for iSeries プリコンパイラーは、可能ならば、構造パラメーター受け渡し技法の使用を試みます。プリコンパイラーは、以下の条件に該当する場合、各ホスト変数が分離パラメーターになっているコードを生成します。

- ステートメントで参照されているホスト変数のデータ長が 9935 を超えている場合。 SQL は構造のうち 64 バイトを使用するため、データ構造の最大長は 9935 + 64 = 9999 となります。
- ステートメントで標識が指定されており、指標付き標識名の長さに必要な標識値を加えたものが 6 文字を超えている場合。プリコンパイラーは結果フィールドの標識名を用いて標識に関する割り当てステートメントを生成しなければなりませんが、この長さは 6 文字に制限されています (″INDIC,1″ には 7 文字が必要)。

- ホスト変数名の長さが 256 を超える場合。これは、サブフィールドがないデータ構造がホスト変数として使用され、その長さが 256 を超えていると、起こります。サブフィールドは、256 を超える長さで定義することはできません。

構造パラメーター受け渡し技法の詳細については、「*DB2 UDB for iSeries データベース・パフォーマンスおよび Query 最適化*」の『データベース・アプリケーション設計のヒント: 構造パラメーター受け渡し方法の使用』を参照してください。

呼び出された SQL を使用する RPG for iSeries プログラムの正しい終了方法

SQL は実行時に、ホスト変数を含む各 SQL ステートメントについて、データ域 (内部 SQLDA) を作成し、保持します。これらの内部 SQLDA は、ステートメントが最初に実行されるときに作成され、その後のステートメント実行時に再使用され、パフォーマンスの向上が図られます。内部 SQLDA は、少なくとも 1 つの SQL プログラムが活動している限り、再使用されます。SQL プリコンパイラーは、SQL 実行時に使用される静的記憶域を割り振り、内部 SQLDA を正しく管理します。

SQL を含む RPG for iSeries プログラムが、やはり SQL を含む他のプログラムから呼び出された場合、RPG for iSeries プログラムは最終レコード (LR) 標識をオンに設定してはなりません。LR 標識をオンに設定すると、静的記憶域は RPG for iSeries プログラムが次に実行されるときに再初期設定されます。静的記憶域の再初期設定は内部 SQLDA の再作成をもたらし、したがってパフォーマンスが低下します。

SQL ステートメントを含んでいる RPG for iSeries プログラムが、やはり SQL ステートメントを含んでいるプログラムから呼び出されたとき、その RPG for iSeries プログラムは以下の 2 つの方法のいずれかによって終了しなければなりません。

- RETRN ステートメントによって
- RT 標識をオンにセットすることによって

このような方法によれば、内部 SQLDA が再使用可能になり、合計実行時間を減らすことができます。

第 6 章 ILE RPG for iSeries アプリケーションでの SQL ステートメントのコーディング方法

この章は、SQL ステートメントを ILE RPG for iSeries プログラムに組み込む場合の固有のアプリケーションおよびコーディング上の要件について説明します。ホスト変数に関するコーディング要件についても説明します。

詳細については、以下のセクションを参照してください。

- 112 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの SQL 連絡域の定義』
- 113 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの SQL 記述子域の定義』
- 114 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの SQL ステートメントの組み込み』
- 117 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでのホスト変数の使用』
- 118 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでのホスト構造の使用』
- 119 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでのホスト構造配列の使用』
- 120 ページの『SQL を使用する ILE RPG iSeries アプリケーションでの LOB ホスト変数の宣言』
- 122 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの ROWID 変数』
- 123 ページの『SQL を使用する ILE RPG iSeries アプリケーションでの外部ファイル記述の使用』
- 125 ページの『SQL データ・タイプと RPG データ・タイプの対応関係の判別』
- 129 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの標識変数の使用』
- 130 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの複数行領域取り出し用 SQLDA の例』
- 131 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの動的 SQL の例』

SQL ステートメントの使い方を示した詳しい ILE RPG プログラムは、 187 ページの『例: ILE RPG for iSeries プログラム内の SQL ステートメント』に記載されています。

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

ILE RPG を使用するプログラミングの詳細については、「ILE RPG プログラマーの手引き」  および「ILE RPG 解説書」  を参照してください。

SQL を使用する ILE RPG for iSeries アプリケーションでの SQL 連絡域の定義

SQL プリコンパイラーは、ILE RPG for iSeries プログラムの最初の演算仕様の前の定義仕様に SQLCA を自動的に入れます。 INCLUDE SQLCA をソース・プログラムにコーディングする必要はありません。ソース・プログラムに INCLUDE SQLCA を指定した場合、そのステートメントは受け入れられますが、余分なものとして扱われます。 SQLCA を ILE RPG for iSeries 向けに定義すると、次のようになります。

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

注: RPG for iSeries では変数名は 6 文字までに制限されています。標準 SQLCA 名は RPG for iSeries 向けに 6 文字の長さに変更されています。ILE RPG for iSeries に変換される RPG for iSeries プログラムとの互換性を保つために、SQLCA 用の名前をそのまま RPG for iSeries で使用されるようにします。ILE RPG for iSeries に対して定義されている SQLCA は、6 個の整数配列として定義されているフィールド SQLERRD を追加しています。SQLERRD は SQLERR 定義をオーバーレイするよう定義されます。

SQLCA の詳細については、「*SQL 解説書*」の『SQL 連絡域』を参照してください。

SQL を使用する ILE RPG for iSeries アプリケーションでの SQL 記述子域の定義

SQLDA を必要とするステートメントには、次のものがあります。

EXECUTE...USING DESCRIPTOR *記述子名*

FETCH...USING DESCRIPTOR *記述子名*

OPEN...USING DESCRIPTOR *記述子名*

CALL...USING DESCRIPTOR *記述子名*

DESCRIBE *ステートメント名* INTO *記述子名*

DESCRIBE TABLE *ホスト変数* INTO *記述子名*

PREPARE *ステートメント名* INTO *記述子名*

SQLCA と異なり、SQLDA を 2 つ以上プログラムの中に置くことができ、また SQLDA の名前は有効であれば、どの名前でも使えます。

動的 SQL は、SQL プログラマーの手引きで説明している高度なプログラミング技法です。動的 SQL を使用すると、ユーザーのプログラムはその実行と平行して SQL ステートメントを作成し、実行させることができます。動的に実行される変数 SELECT リスト (すなわち、照会の一部として返されるデータのリスト) を指定する SELECT ステートメントには、SQL 記述子域 (SQLDA) が必要です。これは、SELECT の結果を受け入れるために割り振るべき変数の数とタイプが事前に予測できないからです。

INCLUDE SQLDA ステートメントは、ILE RPG for iSeries プログラムで指定することができます。このステートメントの形式は次のとおりです。

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

INCLUDE SQLDA は、次のデータ構造を生成します。

```
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
```

ユーザーは、SQL_NUM の定義を行わなければなりません。 SQL_NUM は、SQL_VAR に必要な次元を持つ数値定数として定義する必要があります。

INCLUDE SQLDA は、2 つのデータ構造を生成します。2 番目のデータ構造を使用すると、フィールド記述の入っている SQLDA の部分をセットアップして参照できます。

SQLDA のフィールド記述をセットするには、プログラムは SQLVAR のサブフィールドにフィールド記述をセットアップしてから、SQLVAR の MOVEA を SQL_VAR,n に行います。ここで、n は SQLDA の中のフィールドの数を表しています。この動作は、すべてのフィールド記述がセットされるまで繰り返されます。

SQLDA フィールド記述を参照するときに、ユーザーは SQL_VAR,n の MOVEA を SQLVAR に行います。ここで、n は処理されるフィールド記述の数を表しています。

SQLDA の詳細については、「*SQL 解説書*」の『SQL 記述子域』を参照してください。

SQL を使用する ILE RPG for iSeries アプリケーションでの SQL ステートメントの組み込み

ILE RPG プログラムの中にコーディングする SQL ステートメントは、演算部分に置かなければなりません。このためには C を 6 桁目に入れる必要があります。SQL ステートメントは、明細演算にも、合計演算にも、RPG サブルーチンにも置くことができます。 SQL ステートメントは、RPG ステートメントのロジックに基づいて実行されます。

キーワード EXEC は SQL ステートメントの始まりを示します。 EXEC SQL はソース・ステートメントの 8 桁目から 16 桁目までに置き、7 桁目にスラッシュ (/) を置かなければなりません。 SQL ステートメントを 17 桁目から始めて、80 桁目まで続けることができます。

キーワード END-EXEC は SQL ステートメントの終わりを示します。 END-EXEC はソース・ステートメントの 8 桁目から 16 桁目までに置き、7 桁目にスラッシュ (/) を置かなければなりません。 17 桁目から 80 桁目まではブランクにしておきます。

SQL ステートメントでは大文字と小文字の両方が使用できます。

ILE RPG for iSeries プログラムの中にコーディングされる UPDATE ステートメントは、次のようにコーディングされます。

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

詳細については、以下のセクションを参照してください。

- 115 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの注記』

- 『SQL を使用する ILE RPG for iSeries アプリケーションでの SQL ステートメントの継続』
- 116 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでのコードの組み込み』
- 116 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでのディレクティブの使用』
- 116 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの順序番号』
- 116 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの名前』
- 116 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでのステートメント・ラベル』
- 117 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの WHENEVER ステートメント』

SELECT ステートメントと UPDATE ステートメントの間の行のロックについての詳細は、「*SQL プログラミング 概念*」の『コミットメント制御』を参照してください。

SQL を使用する ILE RPG for iSeries アプリケーションでの注記

SQL 注記 (--) の他に、ILE RPG for iSeries の注記は、 SQL ステートメント内のブランク文字が許されている場所のどこにでも置くことができます。 ILE RPG for iSeries の注記を SQL ステートメントに組み込むときは、7 桁目にアスタリスク (*) を置きます。

SQL を使用する ILE RPG for iSeries アプリケーションでの SQL ステートメントの継続

SQL ステートメントを収めるために追加レコードが必要なときは、9 桁目から 80 桁目が使用できます。 7 桁目は + (正符号) にし、8 桁目はブランクにしなければなりません。継続される行の 80 桁目は、継続行の 9 桁目と連結します。

DBCS データが入っている定数は、継続される行の 80 桁目にシフトイン文字を入れ、継続行の 8 桁目にシフトアウト文字を入れることによって、複数行にわたって継続させることができます。

この例では、SQL ステートメントの G'<AABBCCDDEEFFGGHHIIJJKK>' はグラフィック定数として有効です。

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

SQL を使用する ILE RPG for iSeries アプリケーションでのコードの組み込み

SQL ステートメントと RPG 演算仕様は、SQL ステートメントを使用して取り込むことができます。

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

RPG /COPY ディレクティブを使用すると、SQL ステートメントまたは RPG 仕様を組み込むことができます。ネストされた /COPY ステートメントは、プリコンパイラーではサポートされません。 RPG /INCLUDE ディレクティブは、プリコンパイラーには認識されません。 RPG /INCLUDE ディレクティブは、SQL によって処理される必要のない RPG コードを組み込むのに使用することができます。これは、条件付きディレクティブを含むコードや、その他の /COPY ブロックでのネストに利用できます。

SQL を使用する ILE RPG for iSeries アプリケーションでのディレクティブの使用

/COPY 以外のディレクティブは SQL プリコンパイラーに無視され、コンパイラーに渡されて、処理されます。これは、条件付き論理ブロック内にある RPG ステートメントおよび SQL ステートメントはすべて、プリコンパイラーによって無条件に処理されるということです。

SQL を使用する ILE RPG for iSeries アプリケーションでの順序番号

SQL プリコンパイラーによって生成されるソース・ステートメントの順序番号は、CRTSQLRPGI コマンドの OPTION パラメーターの有効な *NOSEQSRC/*SEQSRC キーワードに基づきます。 *NOSEQSRC が指定されると、入力ソース・メンバーからの順序番号が使用されます。 *SEQSRC を指定すると、順序番号は 000001 から始まり、1 ずつ増えます。

SQL を使用する ILE RPG for iSeries アプリケーションでの名前

有効な ILE RPG for iSeries 変数名であれば、どの名前でもホスト変数に使用できますが、次のような制約を受けます。

'SQ'、'SQL'、'RDI'、または 'DSN' の文字で始まるホスト変数や外部入り口名は、使用してはなりません。これらの名前はデータベース・マネージャー用に予約されています。ホスト変数名の長さは 64 までです。

SQL を使用する ILE RPG for iSeries アプリケーションでのステートメント・ラベル

どの SQL ステートメントの場合も、その前に TAG ステートメントを置くことができます。 TAG ステートメントは EXEC SQL の前の行にコーディングします。

SQL を使用する ILE RPG for iSeries アプリケーションでの WHENEVER ステートメント

GOTO 文節の対象となるものは、TAG ステートメントのラベルでなければなりません。 GOTO/TAG の有効範囲に従わなければなりません。

SQL を使用する ILE RPG for iSeries アプリケーションでのホスト変数の使用

SQL ステートメントの中で使用するホスト変数はいずれも明示的に宣言しなければなりません。

ILE RPG for iSeries に組み込まれた SQL は、ホスト変数を識別するために SQL の BEGIN DECLARE SECTION および END DECLARE SECTION ステートメントを使用しません。これらのステートメントをソース・プログラムに入れてはなりません。

SQL ステートメントの中のホスト変数はいずれも、その前にコロン (:) を付けなければなりません。

ホスト変数の名前は、ホスト変数がそれぞれ別のプロシージャーの中にある場合であっても、1 つのプログラム内では固有になっていなければなりません。

ホスト変数を使用する SQL ステートメントは、その変数が宣言されたステートメントの有効範囲内になければなりません。

詳細については、『SQL を使用する ILE RPG for iSeries アプリケーションでのホスト変数の宣言』を参照してください。

SQL を使用する ILE RPG for iSeries アプリケーションでのホスト変数の宣言

SQL ILE RPG for iSeries プリコンパイラーは、有効なホスト変数として有効な ILE RPG for iSeries 宣言のサブセットしか認識しません。

ILE RPG for iSeries で定義した変数はほとんど、SQL ステートメントの中で使用できます。サポートされていない変数の一部を以下にリストします。

ポインター
テーブル
UDATE
UDAY
UMONTH
UYEAR
先読みフィールド
名前付き定数
複数次元の配列
*SIZE または *ELEM の解決を必要とする定義
定数を OCCURS または DIM で使用していない限り、定数の解決が必要な定義

ホスト変数として使用されるフィールドは、ILE RPG for iSeries の CALL/PARM 機能を使用して SQL に渡されます。PARM の結果フィールドで使用できないフィールドは、ホスト変数として使用できません。

日付および時刻ホスト変数は、SQL プリコンパイラーが生成する構造の中の対応する日付および時刻サブフィールドに常に割り当てられます。生成された日付およびサブフィールドは、CRTSQLRPGI コマンド上の DATFMT、DATSEP、TIMFMT、および TIMSEP の各パラメーターで指定された形式および分離文字により宣言されます。ユーザーが宣言したホスト変数形式からプリコンパイルを指定した形式への変換は、SQL 生成構造への割り当て時および SQL 生成構造からの割り当て時に起こります。 DATFMT パラメーター値がシステム形式 (*MDY、*YMD、*DMY、あるいは *JUL) である場合は、すべての入出力ホスト変数に 1940 ～ 2039 の範囲内の日付値が入っていなければなりません。日付値がこの範囲外である場合は、プリコンパイル時に DATFMT を *ISO、*USA、*EUR、または *JIS の IBM SQL 形式の 1 つとして指定する必要があります。

SQL を使用する ILE RPG for iSeries アプリケーションでのホスト構造の使用

ILE RPG for iSeries データ構造名は、データ構造にサブフィールドがあれば、**ホスト構造**名として使用できます。SQL ステートメントの中でデータ構造名を使用したときは、そのデータ構造を構成するサブフィールド名のリストを暗黙に指定したことになります。

データ構造にサブフィールドがないときは、そのデータ構造名は文字タイプのホスト変数です。これにより、256 より大きな文字変数を使用できます。このサポートは、フィールドを最大長 32766 で定義することができるので、追加機能を提供することはありませんが、RPG for iSeries プログラムとの互換性を保つ上で必要です。

次の例では、BIGCHR はサブフィールドのない ILE RPG for iSeries データ構造となっています。BIGCHR が参照された場合、SQL はそれを長さが 642 の文字ストリングとして扱います。

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

次の例では、PEMPL は EMPNO、FIRSTN、MIDINT、LASTNAME、および DEPTNO のサブフィールドから成るホスト構造の名前です。 PEMPL が参照されると、これらのサブフィールドが使用されます。たとえば、CORPDATA.EMPLOYEE の最初の例は *EMPNO* に入れられ、2 番目の列は *FIRSTN* に入れられます (以下同様です)。

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

...
C                   MOVE      '000220'      EMPNO

...
C/EXEC SQL
```

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

SQL ステートメントを書くとき、サブフィールドに対する参照を修飾することができます。それには、データ構造の名前の後にピリオドとサブフィールドの名前を付けます。たとえば、PEMPL.MIDINT は MIDINT だけを指定したのと同じです。

SQL を使用する ILE RPG for iSeries アプリケーションでのホスト構造配列の使用

ホスト構造は、オカレンス・データ構造として定義されます。オカレンス・データ構造は、複数行を処理するときに SQL の FETCH または INSERT ステートメントで使用することができます。次の項目リストは、複数の行ブロック化サポートによりデータ構造を使用する場合に考慮する必要があります。

- すべてのサブフィールドは有効なホスト変数でなければなりません。
- すべてのサブフィールドは連続していなければなりません。最初の FROM の位置は 1 行でなければならず、TO と FROM の位置はオーバーラップできません。
- ホスト構造内の日付と時刻形式、および日付と時刻サブフィールドの分離文字が、CRTSQLRPGI コマンド上の DATFMT、DATSEP、TIMFMT、および TIMSEP の各パラメーターと同じでない場合には、ホスト構造配列は使用できません。

ブロック化した FETCH およびブロック化した INSERT 以外のすべてのステートメントについて、オカレンス・データ構造を使用する場合、現行のオカレンスが使用されます。ブロック化した FETCH およびブロック化した INSERT の場合、オカレンスは 1 にセットされます。

以下の例では、DEPT というホスト構造配列およびブロック化した FETCH ステートメントを使用して、DEPARTMENT テーブルから 10 行を取り出しています。

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

DIND_ARRAY        DS                 OCCURS(10)
D INDS                            4B 0 DIM(4)
...
C/EXEC SQL
C+ DECLARE C1 CURSOR FOR
C+    SELECT *
C+       FROM  CORPDATA.DEPARTMENT
C/END-EXEC
...

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

SQL を使用する ILE RPG iSeries アプリケーションでの LOB ホスト変数の宣言

ILE RPG for iSeries には、LOB (ラージ・オブジェクト) の SQL データ・タイプに対応する変数がありません。これらのデータ・タイプで使用するホスト変数を作成するには、SQLTYPE キーワードを使用します。SQL プリコンパイラーは、この宣言を出力ソース・メンバー中、ILE RPG for iSeries 言語構造に置き換えます。LOB 宣言は、独立して、またはデータ構造内に置くことができます。

詳細については、以下のセクションを参照してください。

- 『SQL を使用する ILE RPG for iSeries アプリケーションでの LOB ホスト変数』
- 121 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの LOB ロケーター』
- 122 ページの『SQL を使用する ILE RPG for iSeries アプリケーションでの LOB ファイル参照変数』

SQL を使用する ILE RPG for iSeries アプリケーションでの LOB ホスト変数

BLOB の例

次のように宣言すると、

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

以下の構造を生成します。

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

CLOB の例

次のように宣言すると、

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

以下の構造を生成します。

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

DBCLOB の例

次のように宣言すると、

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

以下の構造を生成します。

```
D MYDBCLOB          DS
D MYDBCLOB_LEN                10U
D MYDBCLOB_DATA               400G
```

注:

1. BLOB、CLOB の場合、1 <= lob-length <= 32,766
2. DBCLOB の場合 1<= lob-length <= 16,383
3. LOB ホスト変数はホスト構造内で宣言することができます。
4. LOB ホスト変数は、ホスト構造配列内では使用できません。代わりに LOB ロケーターを使用します。
5. 構造配列内に宣言された LOB ホスト変数は、独立型ホスト変数として使用することはできません。
6. SQLTYPE、BLOB、CLOB、DBCLOB は大文字小文字混合にすることができます。
7. SQLTYPE は、44 から 80 桁の間にする必要があります。
8. LOB が独立型ホスト変数として宣言された場合、24 桁目は文字 'S' とし、25 桁目はブランクにする必要があります。
9. 24 桁目にある独立型のフィールド標識 'S' は、LOB がホスト構造で宣言されている場合は、省略します。
10. LOB ホスト変数は、初期化できません。

SQL を使用する ILE RPG for iSeries アプリケーションでの LOB ロケーター

BLOB ロケーターの例

次のように宣言すると、

```
D MYBLOB          S          SQLTYPE(BLOB_LOCATOR)
```

以下の構造を生成します。

```
D MYBLOB          S          10U
```

CLOB ロケーターおよび DBCLOB ロケーターの構文は、似ています。

注:

1. LOB ロケーターは、ホスト構造内で宣言することができます。
2. SQLTYPE、BLOB_LOCATOR、CLOB_LOCATOR、DBCLOB_LOCATOR は大文字小文字混合にすることができます。
3. SQLTYPE は、44 から 80 桁の間にする必要があります。
4. LOB ロケーターが独立型ホスト変数として宣言された場合、24 桁目は文字 'S' とし、25 桁目はブランクにする必要があります。
5. 24 桁目にある独立型のフィールド標識 'S' は、LOB ロケーターがホスト構造で宣言されている場合は、省略します。
6. LOB ロケーターは、初期化できません。

SQL を使用する ILE RPG for iSeries アプリケーションでの LOB ファイル参照変数

CLOB ファイル参照の例

次のように宣言すると、

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

以下の構造を生成します。

```
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 ロケーターおよび DBCLOB ロケーターの構文は、似ています。

注:

1. LOB ファイル参照変数は、ホスト構造内で宣言することができます。
2. SQLTYPE、BLOB_FILE、CLOB_FILE、DBCLOB_FILE は大文字小文字混合にすることができます。
3. SQLTYPE は、44 から 80 桁の間にする必要があります。
4. LOB ファイル参照が独立型ホスト変数として宣言された場合、24 桁目は文字 'S' とし、25 桁目はブランクにする必要があります。
5. 24 桁目にある独立型のフィールド標識 'S' は、LOB ファイル参照変数がホスト構造で宣言されている場合は、省略します。
6. LOB ファイル参照変数は、初期化できません。

プリコンパイラーは、次のファイル・オプション定数に対する宣言を生成します。ファイル参照ホスト変数を使用する場合、これらの定数を使用して、xxx_FO 変数を設定できます。これらの値の詳細については、「SQL プログラミング 概念」の『LOB ファイル参照変数』を参照してください。

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

SQL を使用する ILE RPG for iSeries アプリケーションでの ROWID 変数

ILE RPG for iSeries には、ROWID の SQL データ・タイプに対応する変数がありません。このデータ・タイプで使用するホスト変数を作成するには、SQLTYPE キーワードを使用します。SQL プリコンパイラーは、この宣言を出力ソース・メンバー中、ILE RPG for iSeries 言語宣言に置き換えます。ROWID 宣言は、独立して、またはデータ構造内に置くことができます。

ROWID の例

次のように宣言すると、

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

以下の構造を生成します。

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

注:

1. SQLTYPE、ROWID は大文字小文字混合にすることができます。
2. ROWID ホスト変数はホスト構造内で宣言することができます。
3. SQLTYPE は、44 から 80 桁の間にする必要があります。
4. ROWID が独立型ホスト変数として宣言された場合、24 桁目は文字 'S' とし、25 桁目はブランクにする必要があります。
5. 24 桁目にある独立型のフィールド標識 'S' は、ROWID がホスト構造で宣言されている場合は、省略します。
6. ROWID ホスト変数は、初期化できません。

SQL を使用する ILE RPG iSeries アプリケーションでの外部ファイル記述の使用

SQL プリコンパイラーは、ILE RPG for iSeries コンパイラーと全く同じ方法で ILE RPG for iSeries ソース・コードを処理します。このことは、プリコンパイラーがホスト変数の定義のために /COPY ステートメントを処理することを意味します。異なる名前が定義されているときは、外部記述ファイルのフィールド定義が取り出されて、名前が変更されます。データ構造の外部定義形式を使用すると、列名のコピーをとって、それをホスト変数として使用することができます。

SQL プリコンパイラーにより日付および時刻フィールド定義を検索および処理する方法は、*NOCVTDT または *CVTDT のいずれを CRTSQLRPGI コマンドの OPTION パラメーターに指定するかによって異なります。 *NOCVTDT を指定した場合は、日付および時刻フィールド定義は形式および分離文字と共に取り出されます。 *CVTDT を指定した場合は、日付および時刻フィールド定義を取り出したときに形式と分離文字は無視され、プリコンパイラーは、変数宣言が文字形式の日付 / 時刻ホスト変数であると想定します。 *CVTDT は RPG for iSeries プリコンパイラー用の互換性オプションです。

次の例では、サンプル・テーブル DEPARTMENT が ILE RPG for iSeries プログラムの中でファイルとして使用されています。 SQL プリコンパイラーは DEPARTMENT のフィールド (列) 定義を取り出して、ホスト変数として使用します。

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

注: ILE RPG for iSeries ステートメントを使用してファイルに入出力操作を行う場合にだけ、ILE RPG for iSeries プログラムの中のファイルに F 仕様書をコーディングしてください。 SQL ステートメントだけを使用してファイルに入出力操作を行うときは、外部データ構造を使用して、ファイル (テーブル) の外部定義を組み込むことができます。

次の例では、サンプル・テーブルは外部データ構造として指定されています。SQL プリコンパイラーはフィールド (列) 定義をデータ構造のサブフィールドとして検索します。サブフィールド名はホスト変数名として、データ構造名 TDEPT はホスト構造名として使用できます。次の例は、プログラムが要求する場合にはフィールド名を変更できることを示します。

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

GRAPHIC 列または VARGRAPHIC 列が UCS-2 CCSID を持つ場合、生成されるホスト変数には UCS-2 CCSID が割り当てられます。

詳細については、『SQL を使用する ILE RPG for iSeries アプリケーションでのホスト構造配列の外部ファイル記述に関する考慮事項』を参照してください。

SQL を使用する ILE RPG for iSeries アプリケーションでのホスト構造配列の外部ファイル記述に関する考慮事項

装置ファイルの場合は、INDARA の指定がなく、ファイルに標識が入っているときには、その宣言をホスト構造配列として使用できません。生成された構造には標識域が組み入れられ、その標識域があるために記憶域が分離されます。

OPTION(*NOCVTDT) を指定して、ファイル内の日付および時刻形式と日付および時刻フィールド定義の分離文字が CRTSQLRPGI コマンド上の DATFMT、DATSEP、TIMFMT、および TIMSEP の各パラメーターと同じでない場合は、ホスト構造配列は使用できません。

以下の例では、DEPARTMENT テーブルは ILE RPG for iSeries プログラムに取り込まれて、ホスト構造配列を宣言するために使用されます。次にブロック化した FETCH ステートメントを使用して、10 行をホスト構造配列に取り出します。

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

...

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

...

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

SQL データ・タイプと RPG データ・タイプの対応関係の判別

プリコンパイラーは、下表に基づいて、ホスト変数のベース SQLTYPE とベース SQLLEN を判断します。ホスト変数が標識変数と一緒に記載されているときは、その SQLTYPE はベース SQLTYPE に 1 を加えたものです。

表 9. ILE RPG for iSeries 宣言と代表的 SQL データ・タイプとの対応関係

RPG データ・タイプ	D 仕様 40 桁目	D 仕様 41、42 桁目	その他の RPG コーディング	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
データ構造 (サブフィールドなし)	ブランク	ブランク	長さ = n (n ≤ 32766)	452	n	CHAR(n)
計算結果フィールド (69、70 桁目 = ブランク)	適用外	適用外	長さ = n (n ≤ 32766 (59 ～ 63 桁目))	452	n	CHAR(n)
定義仕様	A	ブランク	長さ = n (n は 1 から 254 まで) VARYING は 44 ～ 80 桁目	448	n	VARCHAR(n)
定義仕様	A	ブランク	長さ =n (n > 254) VARYING は 44 ～ 80 桁目	456	n	VARCHAR(n)
定義仕様	B	0	長さ ≤ 4	500	2	SMALLINT
定義仕様	I	0	長さ = 5	500	2	SMALLINT
定義仕様	B	0	長さ ≤ 9 かつ ≥ 5	496	4	INTEGER
定義仕様	I	0	長さ = 10	496	4	INTEGER
定義仕様	I	0	長さ = 20	492	8	BIGINT
定義仕様	B	1～4	長さ = 2	500	2	DECIMAL(4,s) s = 41、42 桁目
定義仕様	B	1～9	長さ = 4	496	4	DECIMAL(9,s) s = 41、42 桁目
定義仕様	P	0 から 30 まで	長さ = n (n は 1 から 16 まで)	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (p = n*2-1 かつ s = 41、42 桁目)
定義仕様	F	ブランク	長さ = 4	480	4	FLOAT (単精度)
定義仕様	F	ブランク	長さ = 8	480	8	FLOAT (倍精度)
定義仕様はサブフィールドでない	ブランク	0 から 30 まで	長さ = n (n は 1 から 16 まで)	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (p = n*2-1 かつ s = 41、42 桁目)
入力フィールド (36 = P 桁目)	適用外	適用外	長さ = n (n は 1 から 16 (37 ～ 46 桁目))	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (p = n*2-1 かつ s = 47、48 桁目)
入力フィールド (36 桁目 = ブランクまたは S)	適用外	適用外	長さ = n (n は 1 から 30 (37 ～ 46 桁目))	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (p = n かつ s = 47、48 桁目)

表 9. ILE RPG for iSeries 宣言と代表的 SQL データ・タイプとの対応関係 (続き)

RPG データ・タイプ	D 仕様 40 桁目	D 仕様 41、42 桁目	その他の RPG コーディング	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
入力フィールド (36 桁目 = B)	適用外	適用外	長さ = n (n は 2 から 4 (37 ～ 46 桁目))	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (n = 2 の場合 p = 4、または n = 4 かつ s = 47、48 桁目であれば 9)
計算結果フィールド (69、70 桁目 ≠ ブランク)	適用外	適用外	長さ = n (n は 1 から 30 (59 ～ 63 桁目))	484	バイト 1 には p、バイト 2 には s	DECIMAL(p,s) (p = n かつ s = 64、65 桁目)
データ構造サブフィールド	ブランク	0 から 30 まで	長さ = n (n は 1 から 30 まで)	488	バイト 1 には p、バイト 2 には s	NUMERIC(p,s) (p = n かつ s = 41、42 桁目)
定義仕様	S	0 から 30 まで	長さ = n (n は 1 から 30 まで)	488	バイト 1 には p、バイト 2 には s	NUMERIC(p,s) (p = n かつ s = 41、42 桁目)
入力フィールド (36 桁目 = G)	適用外	適用外	長さ = n (n は 1 から 32766 (37 ～ 46 桁目))	468	m	GRAPHIC(m) (m = n/2、m = (TO-FROM-1)/2)
定義仕様	G	ブランク	長さ = n (n は 1 から 127 まで) VARYING は 44 ～ 80 桁目	464	n	VARGRAPHIC (n)
定義仕様	C	ブランク	長さ = n (n<16383)	468	n	CCSID 13488 を持つ GRAPHIC(n)
定義仕様	G	ブランク	長さ =n (n > 127) VARYING は 44 ～ 80 桁目	472	n	VARGRAPHIC (n)
定義仕様	D	ブランク	長さ = n (n は 6、8、または 10)	384	n	DATE (44 ～ 80 桁目に指定した DATFMT、DATSEP)
入力フィールド (36 桁目 = D)	適用外	適用外	長さ = n (n は 6、8、または 10 (37 ～ 46 桁目))	384	n	DATE (31 ～ 34 桁目で指定した形式)
定義仕様	T	ブランク	長さ = n (n は 8)	388	n	TIME (44 ～ 80 桁目に指定した TIMFMT、TIMSEP)
入力フィールド (36 桁目 = T)	適用外	適用外	長さ = n (n は 8 (37 ～ 46 桁目))	388	n	TIME (31 ～ 34 桁目で指定した形式)
定義仕様	Z	ブランク	長さ = n (n は 26)	392	n	TIMESTAMP
入力フィールド (36 桁目 = Z)	適用外	適用外	長さ = n (n は 26 (37 ～ 46 桁目))	392	n	TIMESTAMP

注:

1. 最初の列では、用語「定義仕様」には、特に他に記載がない限り、データ構造サブフィールドが含まれます。
2. 定義仕様では、2 進数フィールドの長さ (40 桁目に B) は次のように決定されます。
   - FROM (26 ～ 32 桁目) はブランクではない。この場合、長さ = TO-FROM+1。
   - FROM (26 ～ 32 桁目) はブランク。この場合、33 ～ 39 桁目 < 5 であれば長さ = 2、または 33 ～ 39 桁目 > 4 であれば長さ = 4。
3. SQL は CRTSQLRPGI コマンドで指定した DATE/TIME 形式を使用して日付 / 時刻サブフィールドを作成します。ホスト変数 DATE/TIME 形式への変換は、ホスト変数と SQL が生成したサブフィールド間でマッピングを行う際に行われます。

下表を参照すると、各 SQL データ・タイプに対応する RPG データ・タイプを判別することができます。

表 10. SQL データ・タイプと代表的な RPG 宣言との対応関係

SQL データ・タイプ	RPG データ・タイプ	注
SMALLINT	定義仕様。40 桁目が I、長さ 5、42 桁目が 0。 または 定義仕様。40 桁目が B、長さ ≤ 4、42 桁目が 0。	
INTEGER	定義仕様。40 桁目が I、長さ 10、42 桁目が 0。 または 定義仕様。40 桁目が B、長さ ≤ 9 かつ ≥ 5、42 桁目が 0。	
BIGINT	定義仕様。40 桁目が I、長さ 20、42 桁目が 0。	
DECIMAL	定義仕様。サブフィールドでない場合、40 桁目が P またはブランク、41、42 桁目が 0 なら 30 まで。 または 非定義仕様で数値として定義される。	最大長は 16 (精度 30)、最大位取りは 30。
NUMERIC	定義仕様。サブフィールドの 40 桁目が S またはブランク、41、42 桁目が 0 なら 30 まで。	最大長は 30 (精度 30)、最大位取りは 30。
FLOAT (単精度)	定義仕様。40 桁目が F、長さ 4。	
FLOAT (倍精度)	定義仕様。40 桁目が F、長さ 8。	

表 10. SQL データ・タイプと代表的な RPG 宣言との対応関係 (続き)

SQL データ・タイプ	RPG データ・タイプ	注
CHAR(n)	定義仕様。40 桁目が A またはブランク、41、42 桁目がブランク。 または 小数部の桁なしで定義してある入力フィールド。 または 小数部の桁なしで定義された計算結果フィールド。	n は 1 から 32766 まで可能。
CHAR(n)	データ構造にサブフィールドがないデータ構造名。	n は 1 から 32766 まで可能。
VARCHAR(n)	定義仕様。40 桁目が A またはブランク、 44 ～ 80 桁目が VARYING。	n は 1 から 32740 まで可能。
BLOB	サポートなし	SQLTYPE キーワードを使用して BLOB を宣言。
CLOB	サポートなし	SQLTYPE キーワードを使用して CLOB を宣言。
GRAPHIC(n)	定義仕様。40 桁目が G。 または 36 桁目に G を定義してある入力フィールド。	n は 1 から 16383 まで可能。
VARGRAPHIC(n)	定義仕様。40 桁目が G、 44 ～ 80 桁目が VARYING。	n は 1 から 16370 まで可能。
DBCLOB	サポートなし	SQLTYPE キーワードを使用して DBCLOB を宣言。
DATE	文字フィールド または 40 桁目に D を指定した定義仕様。 または 36 桁目に D を定義してある入力フィールド。	形式が *USA、*JIS、*EUR、または *ISO のときは、長さは少なくとも 10 が必要。形式が *YMD、*DMY、または *MDY のときは、長さは少なくとも 8 が必要。形式が *JUL のときは、長さは少なくとも 6 が必要。
TIME	文字フィールド または 40 桁目に T を指定した定義仕様。 または 36 桁目に T を定義してある入力フィールド。	長さは少なくとも 6 が必要。秒を含めるときは、長さは少なくとも 8 が必要。

表 10. SQL データ・タイプと代表的な RPG 宣言との対応関係 (続き)

SQL データ・タイプ	RPG データ・タイプ	注
TIMESTAMP	文字フィールド または 40 桁目に Z を指定した定義仕様。 または 36 桁目に Z を定義してある入力フィールド。	長さは少なくとも 19 が必要。マイクロ秒を含めるときは、長さは少なくとも 26 が必要。長さが 26 未満のときは、マイクロ秒部分で切り捨てが起こります。
DATALINK	サポートなし	
ROWID	サポートなし	SQLTYPE キーワードを使用して ROWID を宣言。

詳細については、『ILE RPG for iSeries 変数宣言と使用法に関する注意事項』を参照してください。

ILE RPG for iSeries 変数宣言と使用法に関する注意事項

SQL を使用する ILE RPG for iSeries アプリケーションでの割り当て規則

ILE RPG for iSeries は、精度と位取りをすべての数値タイプと関連付けます。 ILE RPG for iSeries は、データがパック形式であるものとして、数値演算を定義します。すなわち、2 進数の変数が関係する演算は暗黙的にパック形式に変換されてから、演算が行われます (必要ならば、2 進数に逆変換されます)。データは暗黙の小数点位置に合わされて、SQL 演算が行われます。

SQL を使用する ILE RPG for iSeries アプリケーションでの標識変数の使用

標識変数は長さ 5 未満の (2 バイト) の 2 進数フィールドです。

標識配列は、要素の長さが 4,0 の変数として宣言し、定義仕様に DIM を指定することによって定義できます。

検索されるとき、標識変数はその対応するホスト変数にヌル値が割り当てられているかどうかを示すために使用されます。列に割り当てるときには、ヌル値を割り当てるべきであることを示すために負の標識変数が使用されます。

詳細については、「SQL 解説書」の『標識変数』を参照してください。

標識変数の宣言の仕方はホスト変数の場合と同じであり、これらの 2 つの変数の宣言をプログラマーに適切と思われる方法で組み合わせることができます。

ILE RPG での標識変数の使用例については、130 ページの『例: SQL を使用する ILE RPG for iSeries アプリケーションでの標識変数の使用』を参照してください。

例: SQL を使用する ILE RPG for iSeries アプリケーションでの標識変数の使用

次のステートメントがあるとします。

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

変数は次のように宣言することができます。

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

SQL を使用する ILE RPG for iSeries アプリケーションでの複数行領域取り出し用 SQLDA の例

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

DIND_ARRAY        DS                  OCCURS(10)
D INDS                         4B 0 DIM(4)
...
C* setup number of sqlda entries and length of the sqlda
C                   eval      sqld = 4
C                   eval      sqln = 4
C                   eval      sqldabc = 336
C*
C* setup the first entry in the sqlda
C*
C                   eval      sqltype = 453
C                   eval      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 :SQLDA
C+     INTO :DEPARTMENT:IND_ARRAY
C/END-EXEC
```

SQL を使用する ILE RPG for iSeries アプリケーションでの動的 SQL の例

```
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
```

第 7 章 REXX アプリケーションでの SQL ステートメントのコーディング方法

REXX プロシージャーは、プリプロセスを行う必要はありません。実行時に REXX インタープリターは、理解できないステートメントを現行活動コマンド環境の処理のために渡します。このコマンド環境を *EXECSQL に変更し、全未知ステートメントをデータベース・マネージャーに渡すことができますが、その方法は 2 つあります。

1. STRREXPRC CL コマンドの CMDENV パラメーター
2. ADDRESS REXX コマンドのアドレス定位置パラメーター

詳細については、以下のセクションを参照してください。

- 『REXX アプリケーションでの SQL 連絡域の定義』
- 134 ページの『REXX アプリケーションでの SQL 記述子域の使用』
- 136 ページの『REXX アプリケーションでの SQL ステートメントの組み込み』
- 139 ページの『SQL を使用する REXX アプリケーションでのホスト変数の使用』
- 142 ページの『SQL を使用する REXX アプリケーションでの標識変数の使用』

STRREXPRC CL コマンドまたは ADDRESS REXX コマンドの詳細については、「REXX/400 Programmer's Guide」  および「REXX/400 Reference」  を参照してください。

SQL ステートメントの使い方を示した詳しいサンプル REXX プログラムは、193 ページの『例: REXX プログラム内の SQL ステートメント』に記載されています。

注: コード例についての詳細は、viii ページの『コードについての特記事項』を参照してください。

REXX アプリケーションでの SQL 連絡域の定義

SQL 連絡域 (SQLCA) を構成するフィールドは、SQL/REXX インターフェースにより自動的に組み込まれます。INCLUDE SQLCA ステートメントは不要でしかも使用できません。SQLCA の SQLCODE フィールドおよび SQLSTATE フィールドには、SQL 戻りコードが入っています。これらの値は、各 SQL ステートメントが実行された後、データベース・マネージャーによってセットされます。アプリケーションは、SQLCODE 値または SQLSTATE 値を調べて、最後の SQL ステートメントが正しく実行されたかどうかを判定することができます。

SQL/REXX インターフェースは、SQLCA を典型的な SQL の使用法に従って使用します。ただし、SQL/REXX インターフェースは SQLCA のフィールドを連続したデータ域としてではなく個別の変数として維持します。 SQL/REXX インターフェースが SQLCA 用に維持する変数の定義は次のとおりです。

SQLCODE 1 次 SQL 戻りコード。

SQLERRMC エラーおよび警告メッセージ・トークン。

SQLERRP 製品コード。エラーがあった場合は、エラーを返したモジュールの名前。

SQLERRD.*n* 診断情報が入っている 6 個の変数 (*n* は、1 から 6 までの数字)。

SQLWARN.*n* 警告フラグが入っている 11 個の変数 (*n* は、0 から 10 までの数字)。

SQLSTATE 代替 SQL 戻りコード。

SQLCA の詳細については、「*SQL 解説書*」の『SQL 連絡域』を参照してください。

REXX アプリケーションでの SQL 記述子域の使用

SQLDA を必要とするステートメントには、次のものがあります。

EXECUTE...USING DESCRIPTOR *記述子名*

FETCH...USING DESCRIPTOR *記述子名*

OPEN...USING DESCRIPTOR *記述子名*

CALL...USING DESCRIPTOR *記述子名*

DESCRIBE *ステートメント名* INTO *記述子名*

DESCRIBE TABLE *ホスト変数* INTO *記述子名*

SQLCA とは違い、プロシージャーに複数の SQLDA を入れることができ、しかも有効な名前であれば SQLDA にどのような名前を付けても構いません。各 SQLDA は、共通ステムを持つ 1 組の REXX 変数からなります。ステムの名前は、該当する SQL ステートメントからの*記述子名* です。これは単純なステムでなければなりません。すなわち、ステム自体にピリオドが含まれていてはなりません。
SQL/REXX インターフェースは、各固有の記述子名について SQLDA のフィールドを自動的に用意します。 INCLUDE SQLDA ステートメントは不要でしかも使用できません。

SQL/REXX インターフェースは、SQLDA を典型的な SQL の使用法に従って使用します。ただし、SQL/REXX インターフェースは SQLDA のフィールドを連続したデータ域としてではなく個別の変数として維持します。

SQLDA の詳細については、「SQL 解説書」の『SQL 記述子域』を参照してください。

次の変数は、DESCRIBE、DESCRIBE TABLE、または PREPARE INTO の各ステートメントの後にアプリケーションに返されます。

stem.n.SQLNAME
結果テーブルの中の n 番目の列の名前。

次の変数は、EXECUTE...USING DESCRIPTOR、OPEN...USING DESCRIPTOR、CALL...USING DESCRIPTOR、または FETCH...USING DESCRIPTOR の各ステート

メントの前にアプリケーションにより指定される必要があります。これらの変数は、DESCRIBE、DESCRIBE TABLE、または PREPARE INTO ステートメントの後にアプリケーションに返されます。

stem.SQLD
SQLDA に実際に入っている変数要素の数。

stem.n.SQLTYPE
n 番目の要素のデータ・タイプを表す整数 (たとえば、最初の要素は stem.1.SQLTYPE にあります)。

次のデータ・タイプは使用できません。

400/401 NUL 終了漢字ストリング

404/405 BLOB ホスト変数

408/409 CLOB ホスト変数

412/413 DBCLOB ホスト変数

460/461 NUL 終了文字ストリング

476/477 PASCAL L 文字列

496/497 長精度整数 (位取りが 0 より大きい)

500/501 短精度整数 (位取りが 0 より大きい)

504/505 DISPLAY SIGN LEADING SEPARATE

904/905 ROWID

916/917 BLOB ファイル参照変数

920/921 CLOB ファイル参照変数

924/925 DBCLOB ファイル参照変数

960/961 BLOB ロケーター

964/965 CLOB ロケーター

968/969 DBCLOB ロケーター

stem.n.SQLLEN
SQLTYPE が DECIMAL または NUMERIC データ・タイプを指示していない場合、データの最大長は stem.n.SQLDATA に入っています。

stem.n.SQLLEN.SQLPRECISION
データ・タイプが DECIMAL または NUMERIC の場合、これには数の精度が入ります。

stem.n.SQLLEN.SQLSCALE
データ・タイプが DECIMAL または NUMERIC の場合、これには数の位取りが入ります。

stem.n.SQLCCSID
データの n 列目の CCSID。

次の変数は、EXECUTE...USING DESCRIPTOR または OPEN...USING DESCRIPTOR の各ステートメントの前にアプリケーションによって必ず提供される必要があります。これらの FETCH...USING DESCRIPTOR ステートメントの後でア

プリケーションに返されます。これらの変数は、DESCRIBE、DESCRIBE TABLE、または PREPARE INTO の各ステートメントの後では使用されません。

stem.n.SQLDATA
これには、アプリケーションから提供された入力値、または SQL が取り出した出力値が含まれます。

この値は、SQLTYPE、SQLLEN、SQLPRECISION、および SQLSCALE で指定した属性に変換されます。

stem.n.SQLIND
入力値または出力値が NULL の場合、この値は負の数になります。

REXX アプリケーションでの SQL ステートメントの組み込み

SQL ステートメントは REXX コマンドを入れられるところならどこにでも入れることができます。

REXX プロシージャー内の各 SQL ステートメントは、必ず EXECSQL で始まり (大文字と小文字はどのように組み合わせても構いません)、次に以下のいずれかが続かなければなりません

- 一重引用符または二重引用符で囲まれている SQL ステートメント。または
- ステートメントが入っている REXX 変数。 REXX 変数に SQL ステートメントが入っている場合は、REXX 変数の前にコロンを入れてはなりません。

以下に、例を示します。

```
EXECSQL "COMMIT"
```

上記の例は次に相当します。

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

このコマンドは通常の REXX 規則に従います。たとえば、任意選択でコマンドの後にセミコロン (;) を入れて、1 行に複数の REXX ステートメントを入れることができます。また、REXX を使用すると、一重引用符の中にコマンド名を入れることができます。たとえば、次のとおりです。

```
'EXECSQL COMMIT'
```

SQL/REXX インターフェースは、次の SQL ステートメントをサポートします。

ALTER TABLE
CALL [3]
CLOSE
COMMENT ON
COMMIT
CREATE ALIAS
CREATE DISTINCT TYPE
CREATE FUNCTION
CREATE INDEX
CREATE PROCEDURE
CREATE SCHEMA
CREATE TABLE
CREATE TRIGGER
CREATE VIEW
DECLARE CURSOR [3]
DECLARE GLOBAL TEMPORARY TABLE
DELETE [3]
DESCRIBE
DESCRIBE TABLE
DROP
EXECUTE
EXECUTE IMMEDIATE
FETCH [2]
GRANT
INSERT [2], [3]
LABEL ON
LOCK TABLE
OPEN
PREPARE
RELEASE SAVEPOINT
RENAME
REVOKE
ROLLBACK
SAVEPOINT
SET OPTION [4]
SET PATH
SET SCHEMA
SET TRANSACTION
SET variable (変数) [3]
UPDATE [3]
VALUES INTO [3]

次の SQL ステートメントは、SQL/REXX インターフェースではサポートされません。

BEGIN DECLARE SECTION
CONNECT
DECLARE PROCEDURE
DECLARE STATEMENT
DECLARE VARIABLE
DISCONNECT
END DECLARE SECTION
FREE LOCATOR
HOLD LOCATOR
INCLUDE
RELEASE
SELECT INTO
SET CONNECTION
SET RESULT SETS
WHENEVER[5]

詳細については、以下のセクションを参照してください。

- 138 ページの『SQL を使用する REXX アプリケーションでの注記』
- 138 ページの『SQL を使用する REXX アプリケーションでの SQL ステートメントの継続』
- 138 ページの『SQL を使用する REXX アプリケーションでのコードの組み込み』

---

2. このステートメントのブロック化形式はサポートされていません。

3. これらのステートメントがホスト変数を含む場合には、直接実行することはできません。これらのステートメントは、PREPARE のオブジェクト、次に EXECUTE のオブジェクトでなければなりません。

4. SET OPTION ステートメントを REXX プロシージャーで使用し、SQL ステートメントを実行するために使用される処理オプションのいくつかを変更することができます。これらのオプションにはコミットメント制御レベルや日付形式が含まれます。 SET OPTION ステートメントの詳細については、「SQL 解説書」を参照してください。

5. 詳細は、 139 ページの『SQL を使用する REXX アプリケーションでのエラーおよび警告の処理』を参照してください。

- 『SQL を使用する REXX アプリケーションでのマージン』
- 『SQL を使用する REXX アプリケーションの名前』
- 『SQL を使用する REXX アプリケーションでのヌル』
- 『SQL を使用する REXX アプリケーションでのステートメント・ラベル』
- 139 ページの『SQL を使用する REXX アプリケーションでのエラーおよび警告の処理』

SQL を使用する REXX アプリケーションでの注記

SQL の注記 (--) も REXX の注記も SQL ステートメントを表している文字列に入れることはできません。

SQL を使用する REXX アプリケーションでの SQL ステートメントの継続

SQL ステートメントの入っている文字列は、標準 REXX の使用法に従ってコンマや連結記号で分離することによって、複数行の複数の文字列に分割することができます。

SQL を使用する REXX アプリケーションでのコードの組み込み

他のホスト言語と違い、外部で定義されたステートメントの組み込みについてのサポートはありません。

SQL を使用する REXX アプリケーションでのマージン

SQL/REXX インターフェースの場合、特殊なマージンの規則はありません。

SQL を使用する REXX アプリケーションの名前

ホスト変数には、ピリオド (.) で終わらない任意の有効な REXX 名を使用できます。名前は必ず 64 文字以下でなければなりません。

変数名は、'SQL'、'RDI'、'DSN'、'RXSQL'、または 'QRW' の文字で始まってはなりません。

SQL を使用する REXX アプリケーションでのヌル

ヌル という用語は、REXX でも SQL でも使用していますが、この用語はこれら 2 つの言語では異なる意味を持っています。 REXX には、ヌル・ストリング (ゼロ長の文字列) とヌル文節 (ブランクと注記のみから成る文節) があります。 SQL のヌル値は、すべての非ヌル値とは異なる特殊な値で、 (ヌルでない) 値がないことを表しています。

SQL を使用する REXX アプリケーションでのステートメント・ラベル

REXX コマンド・ステートメントは、通常どおりにラベル付けすることができます。

SQL を使用する REXX アプリケーションでのエラーおよび警告の処理

WHENEVER ステートメントは、SQL/REXX インターフェースではサポートしていません。その代わり、次のどれを使用しても構いません。

- データベース・マネージャーが出したエラーおよび警告条件 (ただし、SQL/REXX インターフェースが出したものでない) を検出するための各 SQL ステートメントの後の REXX SQLCODE 変数または SQLSTATE 変数のテスト。
- エラーおよび警告条件を検出するための各 SQL ステートメントの後の REXX RC 変数のテスト。 EXECSQL コマンドを使用するたびに RC 変数は次のようにセットされます。

  0 ステートメントが正常に完了。

  +10 SQL 警告が発生。

  -10 SQL エラーが発生。

  -100 SQL/REXX インターフェース・エラーが発生。

  これを使用すると、データベース・マネージャーまたは SQL/REXX インターフェースのいずれかによって出されたエラーおよび警告を検出することができます。
- SIGNAL ON ERROR 機能および SIGNAL ON FAILURE 機能を使用すると、エラー (負の RC 値) を検出することができますが、警告を検出することはできません。

SQL を使用する REXX アプリケーションでのホスト変数の使用

REXX には変数宣言がありません。 LOB ホスト変数および ROWID ホスト変数は、REXX ではサポートされません。新規の変数は、それが割り当てステートメントに現れることによって認識されます。したがって、宣言部分はなく、BEGIN DECLARE SECTION ステートメントおよび END DECLARE SECTION ステートメントをサポートしていません。

SQL ステートメントの中のホスト変数はいずれも、その前にコロン (:) を付けなければなりません。

SQL/REXX インターフェースは、ステートメントをデータベース・マネージャーに渡す前に複合変数で置換を行います。以下に、例を示します。

```
a = 1
b = 2
EXECSQL 'OPEN c1 USING :x.a.b'
```

上記により、x.1.2 の内容が SQL に渡されます。

詳細については、以下のセクションを参照してください。

- 140 ページの『SQL を使用する REXX アプリケーションでの入力ホスト変数のデータ・タイプの判別』
- 141 ページの『SQL を使用する REXX アプリケーションでの出力ホスト変数のフォーマット』

- 141 ページの『SQL を使用する REXX アプリケーションでの REXX 変換の回避』

SQL を使用する REXX アプリケーションでの入力ホスト変数のデータ・タイプの判別

REXX におけるすべてのデータは、文字列形式になっています。入力ホスト変数のデータ・タイプ (すなわち、EXECUTE ステートメントまたは OPEN ステートメント内の 'USING ホスト変数' 文節で使用しているホスト変数) は、表 11 に従って、実行時にデータ・タイプ管理プログラムによって変数の内容から推論されます。

これらの規則は、数値、文字値、またはグラフィック値のいずれかを定義します。数値は、任意のタイプの数値列への入力として使用することができます。文字値は、任意のタイプの文字列への入力、または日付、時刻、またはタイム・スタンプの各列への入力として使用することができます。グラフィック値は、任意のタイプのグラフィック列への入力として使用することができます。

表 11. REXX でのホスト変数のデータ・タイプの判別

ホスト変数の内容	想定データ・タイプ	SQL タイプ・コード	SQL タイプ記述
未定義の変数	値を割り当てていない変数	なし	無効なデータを検出した。
先行と後書きアポストロフィ (') または引用符 (") をもつ文字列で、2 つの区切り文字を除く長さが n である。 または、先行 X または先行 x の後にアポストロフィ (') または引用符 (")、および後書きアポストロフィ (') または引用符 (") をもつ文字列。この文字列の長さは、X または x と 2 つの区切り文字を取り除くと 2n になる。残りの文字の各対は、単一文字の 16 進表示。 または、このテーブルの他の規則により文字、数値またはグラフィックとして認識できない長さ n の文字列。	可変長文字ストリング	448/449	VARCHAR(n)
前にコロンの付いている先行および後書きアポストロフィ (') または引用符 (") をもつ文字列。[6] • G、g、N または n で始まる文字列。その後にはアポストロフィまたは引用符とシフトアウト (x'0E') が続く。その後には、n 個のグラフィック文字が続く。それぞれの長さは 2 文字。この文字列は、必ずシフト・イン (X'0F') とアポストロフィまたは引用符 (文字列の最初にあるいずれか一方) で終わること。 • 文字列の先行は GX、Gx、gX、または gx とし、次にアポストロフィまたは引用符とシフトアウト (x'0E') が続く。その後には、n 個のグラフィック文字が続く。それぞれの長さは 2 文字。この文字列は、必ずシフト・イン (X'0F') とアポストロフィまたは引用符 (文字列の最初にあるいずれか一方) で終わること。GX または 区切り文字を取り除くと、この文字列の長さは 4n になる。残りの 4 文字の各グループは、単一グラフィック文字の 16 進表示となる。	可変長漢字ストリング	464/465	VARGRAPHIC(n)

表 11. REXX でのホスト変数のデータ・タイプの判別 (続き)

ホスト変数の内容	想定データ・タイプ	SQL タイプ・コード	SQL タイプ記述
科学または技術的表記での数字 (すなわち、直後に 'E' または 'e'、任意選択で正符号または負符号、および一連の数字が続く)。先頭に正符号または負符号が付いても構わない。	浮動小数点	480/481	FLOAT
小数点を含む数字。ただし、指数は含まれない。 または、小数点または指数を含まない数で、2147483647 より大きいか、-2147483647 より小さい数。 先頭に正符号または負符号が付いても構わない。 *m* は、数の総桁数。*n* は、小数点の左にある桁数 (存在する場合)。	パック 10 進数	484/485	DECIMAL(m,n)
小数点も指数も入っていない数字。先頭に正符号または負符号が付いても構わない。	符号付き整数	496/497	INTEGER

SQL を使用する REXX アプリケーションでの出力ホスト変数のフォーマット

出力ホスト変数 (すなわち、FETCH ステートメントの 'INTO ホスト変数' 文節で使用されるホスト変数) のデータ・タイプを判別する必要はありません。出力値は、次のようにホスト変数に割り当てられます。

- 文字値は、前後のアポストロフィなしで割り当てられます。
- グラフィック値は、先行 G またはアポストロフィ、後書きアポストロフィ、シフトアウトとシフトインの文字のいずれも伴うことなく割り当てられます。
- 数値は、文字列に変換されます。
- 整数値は先行ゼロを保持しません。負の値には、先頭に負符号が付いています。
- 10 進数値は、それらの精度と位取りに従って先行ゼロと後続ゼロを持っています。負の値には、先頭に負符号が付いています。正の値には、先頭に正符号は付いていません。
- 浮動小数点値は浮動小数になっていて、小数点の左に 1 桁入っています。 'E' は、大文字になっています。

SQL を使用する REXX アプリケーションでの REXX 変換の回避

文字列を確実に数に変換しないようにするか、またはグラフィック・タイプと見なす場合は、文字列を "'" の引用符で囲む必要があります。ただアポストロフィで囲んでも認識されません。以下に、例を示します。

```
stringvar = '100'
```

上記のように指定すると、REXX は、変数 *stringvar* を文字の 100 (アポストロフィなし) の文字列にセットします。これは、SQL/REXX インターフェースで数字 100 として評価され、そのまま SQL に渡されます。

---

6. 先行アポストロフィの直後のバイトは X'0E' シフトアウトで、後書きアポストロフィの直前のバイトは X'0F' シフトインです。

一方、次の例を見てください。

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

上記のように指定すると、REXX は、変数 *stringvar* を文字 '100' (アポストロフィを含む) の文字列にセットします。これは、SQL/REXX インターフェースにより文字列 100 として評価され、そのまま SQL に渡されます。

SQL を使用する REXX アプリケーションでの標識変数の使用

標識変数は整数です。検索時に、標識変数が使用され、その関連ホスト変数がヌル値に割り当てられたかどうかを示します。列に割り当てるときには、ヌル値を割り当てるべきであることを示すために負の標識変数が使用されます。

その他の言語と違い、たとえその関連標識変数に負の値が入っていても、必ずホスト変数に有効な値を指定する必要があります。

詳細については、「SQL 解説書」の『標識変数』を参照してください。

第 8 章 SQL ステートメントを含むプログラムの準備と実行

この章では、アプリケーション・プログラムの準備と実行に必要な作業のいくつかについて説明します。詳細については、以下のセクションを参照してください。

- 『SQL プリコンパイラーの基本処理』
- 152 ページの『非 ILE SQL プリコンパイラー・コマンド』
- 153 ページの『ILE SQL プリコンパイラー・コマンド』
- 155 ページの『SQL を使用するアプリケーションでのコンパイル・エラーの解釈』
- 156 ページの『SQL を使用するアプリケーションのバインド』
- 158 ページの『SQL プリコンパイラー・オプションの表示』
- 158 ページの『組み込み SQL を使用したプログラムの実行』

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

SQL プリコンパイラーの基本処理

SQL ステートメントが組み込まれているアプリケーション・プログラムを実行させるには、その前にプログラムをプリコンパイルし、コンパイルしなければなりません。

注: REXX プロシージャー内の SQL ステートメントは、プリコンパイルもコンパイルも行われません。

このようなプログラムのプリコンパイルは SQL プリコンパイラーによって行われます。SQL プリコンパイラーは、アプリケーション・プログラムのソース仕様の各ステートメントを走査して、次のことを行います。

- **SQL ステートメントおよびホスト変数名の定義を見つける。**変数名とその定義は、SQL ステートメントの妥当性を検査するために使用されます。ユーザーは、SQL プリコンパイラーが処理を完了した後、リストを調べて、エラーの有無を確かめることができます。
- **各ステートメントが有効で構文エラーがないことを確かめる。**この妥当性検査のプロシージャーは、出力リストにエラー・メッセージを出すので、エラーがあればそれを訂正するのに役立ちます。
- **データベースの記述に基づいて SQL ステートメントの妥当性を検査する。**プリコンパイルの過程で、テーブル、ビュー、および列名が有効であることを確かめるために SQL ステートメントが検査されます。指定したテーブルまたはビューが存在しない場合や、プリコンパイルまたはコンパイル時にテーブルまたはビューを使用する権限がない場合は、妥当性検査は実行時に行われます。実行時にテーブルやビューが存在しない時は、エラーが起こります。

  注:

  1. 一時変更の処理は外部定義の検査時に行われます。詳細については、「データベース・プログラミング」および「ファイル管理」を参照してください。

2. SQL ステートメントが有効であることを確かめるためには、SQL ステートメントで参照されているテーブルやビューに対する何らかの権限 (少なくとも *OBJOPR) が必要です。 SQL ステートメントの処理に必要な実際の権限は、実行時に検査されます。SQL ステートメントの詳細については、「SQL 解説書」を参照してください。
3. CRTSQLxxx コマンドで RDB パラメーターを指定すると、プリコンパイラーは、指定したリレーショナル・データベースにアクセスして、テーブル記述とビュー記述を入手します。

- **ホスト言語でのコンパイルに備えて各 SQL ステートメントの準備を行う。**ほとんどの SQL ステートメントの場合、SQL プリコンパイラーは、注記と CALL ステートメントを次の SQL インターフェース・モジュールの 1 つに挿入します。
  - QSQROUTE
  - QSQLOPEN
  - QSQLCLSE
  - QSQLCMIT

  一部の SQL ステートメント (たとえば、DECLARE ステートメント) では、SQL プリコンパイラーは注記だけを生成し、ホスト言語ステートメントを生成しません。
- **プリコンパイルされた SQL ステートメントに関する情報を生成する。**この情報は一時ソース・ファイル・メンバーに内部的に保管され、バインド処理の際に使用されます。

プリコンパイルするときに完全な診断情報を入手するには、次のいずれかを指定してください。

- CRTSQLxxx の場合 (ただし、xxx=CBL、PLI、または RPG)、OPTION(*SOURCE *XREF)
- CRTSQLxxx の場合 (ただし、xxx=CI、CPPI、CBLI、または RPGI) OPTION(*XREF) OUTPUT(*PRINT)

詳細については、以下のセクションを参照してください。

- 『SQL プリコンパイラーへの入力』
- 145 ページの『SQL プリコンパイラーのソース・ファイル CCSID』
- 146 ページの『SQL プリコンパイラーの出力』

SQL プリコンパイラーへの入力

SQL プリコンパイラーへの主要な入力は、アプリケーション・プログラミング・ステートメントと組み込み SQL ステートメントです。PL/I、C、および C++ プログラムでは、SQL ステートメントは、 CRTSQLPLI、CRTSQLCI、および CRTSQLCPPI コマンドの MARGINS パラメーターに指定されたマージンを使用しなければなりません。

SQL プリコンパイラーは、ホスト言語ステートメントが構文的に正しいと想定します。ホスト言語ステートメントが構文的に正しくないと、プリコンパイラーは SQL ステートメントとホスト変数宣言を正しく識別できないことがあります。プリコンパイラーを経由して渡すことができるソース・ステートメントの形式には制限があ

ります。アプリケーション言語コンパイラーに受け付けられないリテラルと注記は、プリコンパイラーによるソース・ステートメントの走査処理を阻害し、エラーの原因となるおそれがあります。

SQL INCLUDE ステートメントを使用すると、CRTSQLxxx[7] の INCFILE パラメーターで指定されたファイルから 2 次入力を得ることができます。SQL INCLUDE ステートメントは、指定したメンバーの終わりに達するまでそのメンバーから入力の読み取りを行います。このようにして組み込まれるメンバーには、プリコンパイラーの他の INCLUDE ステートメントが含まれていてはなりませんが、アプリケーション・プログラム・ステートメントと SQL ステートメントの両方を含むことができます。

SQL プリコンパイラーの前に、別の前処理プログラムでソース・ステートメントを処理しておくこともできます。ただし、SQL プリコンパイルの前に実行される前処理プログラムは、SQL ステートメントの受け渡しが可能なものでなければなりません。

アプリケーション・プログラムのソース・ステートメントで混合 DBCS 定数が指定されている場合には、ソース・ファイルは混合 CCSID でなければなりません。

SQL SET OPTION ステートメントを使用することにより、入力ソース・メンバーに多くのプリコンパイラー・オプションを指定することができます。 SET OPTION 構文については、「SQL 解説書」を参照してください。

SQL プリコンパイラーのソース・ファイル CCSID

SQL プリコンパイラーは、ソース・ファイルの CCSID を使用してソース・レコードを読み取ります。 SQL INCLUDE ステートメントを処理するとき、組み込まれるソース・ファイルは、必要ならば、元のソース・ファイルの CCSID に変換されます。組み込まれるソース・ファイルが元のソース・ファイルの CCSID に変換できないときは、エラーが起こります。

SQL プリコンパイラーは、ソース・ファイルの CCSID を使用して SQL ステートメントを処理します。この影響を最も受けるのは、変換文字です。たとえば、 NOT 記号 (¬) は CCSID 500 では 'BA'X に置かれています。これは、ソース・ファイルの CCSID が 500 である場合に、SQL が NOT 記号 (¬) が 'BA'X に置かれるよう要求したことを意味します。

ソース・ファイルの CCSID が 65535 の場合、SQL は、CCSID が 37 であるものとして可変文字を処理します。すなわち、SQL は NOT 記号 (¬) が '5F'X にあるものとして探します。

7. このコマンドの xxx はホスト言語標識を表しています。 COBOL for iSeries 言語の場合は CBL、ILE COBOL for iSeries 言語の場合は CBLI、iSeries PL/I 言語の場合は PLI、ILE C for iSeries 言語の場合は CI、 RPG for iSeries 言語の場合は RPG、ILE RPG for iSeries 言語の場合は RPGI、 ILE C++/400 言語の場合は CPPI です。

SQL プリコンパイラーの出力

以下のセクションでは、プリコンパイラーから得られる種々の出力について説明します。

リスト

出力リストは、CRTSQLxxx コマンドの PRTFILE パラメーターで指定した印刷ファイルに送られます。印刷ファイルに書き込まれる項目には、次のものがあります。

- プリコンパイラー・オプション

  CRTSQLxxx コマンドで指定されているオプション。
- プリコンパイラー・ソース・ステートメント

  リスト・オプションが有効な場合、この出力は、プリコンパイラー・ソース・ステートメントと、プリコンパイラーによって割り当てられたレコード番号を提供します。
- プリコンパイラー相互参照

  OPTION パラメーターに *XREF を指定すると、この出力は相互参照リストを提供します。このリストは、参照されるホスト名と列名が入っている SQL ステートメントのプリコンパイラー・レコード番号を示します。
- プリコンパイラー診断情報

  この出力からは、エラーのあるステートメントのプリコンパイラー・レコード番号を示す診断メッセージが得られます。

  印刷ファイルに送られる出力には、CCSID 値の 65535 が使用されます。印刷ファイルに書き込まれるとき、データは変換されません。

SQL プリコンパイラーにより作成される一時ソース・ファイル・メンバー

プリコンパイラーによって処理されたソース・ステートメントは、出力ソース・ファイルに書き出されます。プリコンパイラーによって変更されたソース・コードでは、SQL ステートメントは、注記と SQL 実行時への呼び出しに変換されています。 SQL によって処理される組み込みは拡張されます。

出力ソース・ファイルは、CRTSQLxxx コマンドの TOSRCFILE パラメーターで指定されます。C および C++ 以外の言語について、省略時ファイルは、QTEMP ライブラリーの QSQLTEMP (ILE RPG for iSeries の場合は QSQLTEMP1) です。*CALC が出力ソース・ファイルとして指定されるときの C および C++ については、ソース・ファイルのレコード長が 92 以下である場合に QSQLTEMP が使用されます。レコード長が 92 より大きい C または C++ ソース・ファイルについては、出力ソース・ファイル名が QSQLTxxxxx として生成されます。ここで xxxxx はレコード長です。出力ソース・ファイル・メンバーの名前は、CRTSQLxxx コマンドの PGM パラメーターまたは OBJ パラメーターに指定した名前と同じになります。このメンバーは、コンパイラーへの入力として使用する前に変更することはできません。 SQL が出力ソース・ファイルを作成するとき、ソース・ファイルの CCSID 値を新しいファイルの CCSID 値として使用します。

プリコンパイルで QTEMP のソース・ファイルに出力を生成する場合で、あとでコンパイルを行いたい場合は、プリコンパイル後にそのファイルを永続ライブラリー

に移しておくことができます。ソース・メンバーのレコードを変更することはできません。変更すると、コンパイルは失敗します。

プリコンパイルの結果として SQL によって生成されるソース・メンバーは、別のプリコンパイル・ステップへの入力メンバーとして編集および再利用してはなりません。最初のプリコンパイル中にソース・メンバーとともに保管される追加の SQL 情報によって、2 番目のプリコンパイルが正しく行われなくなります。この情報は一度ソース・メンバーに付加されると、そのメンバーが削除されるまでこのメンバーとともに存在し続けます。

SQL プリコンパイラーは、CRTSRCPF コマンドを使用して、出力ソース・ファイルを作成します。このコマンドの省略時値が変更されていると、結果が予測できないものになる可能性があります。SQL プリコンパイラーではなくユーザーがソース・ファイルを作成した場合、ファイル属性も異なる場合があります。ユーザーが SQL に出力ソース・ファイルを作成させるようお勧めします。このファイルは、いったん SQL によって作成されたら、後のプリコンパイルで再利用することができます。

SQL プリコンパイラー出力のサンプル

プリコンパイラーの出力は、プログラムのソース仕様に関する情報を提供することができます。リストを作成するには、次のように行ってください。

- 非 ILE プリコンパイラーの場合は、*SOURCE(*SRC) および *XREF の各オプションを CRTSQLxxx コマンドの OPTION パラメーターに指定してください。
- ILE プリコンパイラーの場合は、OPTION(*XREF) および OUTPUT(*PRINT) を CRTSQLxxx コマンドに指定してください。

プリコンパイラーの出力の形式は次のとおりです。

```
5722ST1 V5R2M0 020719         Create SQL COBOL Program          CBLTEST1                          08/06/02 11:14:21   Page   1
Source type...............COBOL
Program name..............CORPDATA/CBLTEST1
Source file...............CORPDATA/SRC
Member....................CBLTEST1
To source file............QTEMP/QSQLTEMP
 1 Options..................*SRC      *XREF     *SQL
Target release............V5R2M0
INCLUDE file..............*LIBL/*SRCFILE
Commit....................*CHG
Allow copy of data........*YES
Close SQL cursor..........*ENDPGM
Allow blocking............*READ
Delay PREPARE.............*NO
Generation level..........10
Printer file..............*LIBL/QSYSPRT
Date format...............*JOB
Date separator............*JOB
Time format...............*HMS
Time separator ...........*JOB
Replace...................*YES
Relational database.......*LOCAL
User .....................*CURRENT
RDB connect method........*DUW
Default Collection........*NONE
Package name..............*PGMLIB/*PGM
Path......................*NAMING
Created object type.......*PGM
User profile..............*NAMING
Dynamic User Profile......*USER
Sort Sequence.............*JOB
Language ID...............*JOB
IBM SQL flagging..........*NOFLAG
ANS flagging..............*NONE
Text......................*SRCMBRTXT
Source file CCSID.........65535
Job CCSID.................65535
 2  Source member changed on 06/06/00  10:16:44
```

1 SQL プリコンパイラーの呼び出し時に指定したオプションのリスト。

2 ソース・メンバーが最後に変更された日付。

図 2. サンプル COBOL プリコンパイラー出力の形式 (1/4)

```
5722ST1 V5R2M0 020719           Create SQL COBOL Program          CBLTEST1                      08/06/02 11:14:21   Page   2
1 Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   2 SEQNBR 3 Last Change

  1          IDENTIFICATION DIVISION.                                                          100
  2          PROGRAM-ID.  CBLTEST1.                                                            200
  3          ENVIRONMENT DIVISION.                                                             300
  4          CONFIGURATION SECTION.                                                            400
  5          SOURCE-COMPUTER. IBM-AS400.                                                       500
  6          OBJECT-COMPUTER. IBM-AS400.                                                       600
  7          INPUT-OUTPUT SECTION.                                                             700
  8          FILE-CONTROL.                                                                     800
  9              SELECT OUTFILE, ASSIGN TO PRINTER-QPRINT,                                     900
 10                 FILE STATUS IS FSTAT.                                                     1000
 11          DATA DIVISION.                                                                   1100
 12          FILE SECTION.                                                                    1200
 13          FD  OUTFILE                                                                      1300
 14              DATA RECORD IS REC-1,                                                        1400
 15              LABEL RECORDS ARE OMITTED.                                                   1500
 16          01  REC-1.                                                                       1600
 17              05  CC                     PIC X.                                         1700
 18              05  DEPT-NO                PIC X(3).                                      1800
 19              05  FILLER                 PIC X(5).                                      1900
 20              05  AVERAGE-EDUCATION-LEVEL PIC ZZZ.                                      2000
 21              05  FILLER                  PIC X(5).                                     2100
 22              05  AVERAGE-SALARY          PIC ZZZZ9.99.                                 2200
 23          01  ERROR-RECORD.                                                             2300
 24              05  CC                      PIC X.                                        2400
 25              05  ERROR-CODE              PIC S9(5).                                    2500
 26              05  ERROR-MESSAGE           PIC X(70).                                    2600
 27          WORKING-STORAGE SECTION.                                                       2700
 28              EXEC SQL                                                                   2800
 29                INCLUDE SQLCA                                                            2900
 30              END-EXEC.                                                                  3000
 31          77  FSTAT                       PIC XX.                                       3100
 32          01  AVG-RECORD.                                                               3200
 33              05  WORKDEPT                PIC X(3).                                     3300
 34              05  AVG-EDUC                PIC S9(4) USAGE COMP-4.                       3400
 35              05  AVG-SALARY              PIC S9(6)V99 COMP-3.                          3500
 36          PROCEDURE DIVISION.                                                            3600
 37         ****************************************************************              3700
 38         *  This program will get the average education level and the  *               3800
 39         *  average salary by department.                              *               3900
 40         ****************************************************************              4000
 41          A000-MAIN-PROCEDURE.                                                          4100
 42              OPEN OUTPUT OUTFILE.                                                      4200
 43         ****************************************************************              4300
 44         * Set-up WHENEVER statement to handle SQL errors.             *               4400
 45         ****************************************************************              4500
 46              EXEC SQL                                                                  4600
 47                WHENEVER SQLERROR GO TO B000-SQL-ERROR                                  4700
 48              END-EXEC.                                                                 4800
 49         ****************************************************************              4900
 50         * Declare cursor                                              *               5000
 51         ****************************************************************              5100
 52              EXEC SQL                                                                  5200
 53                DECLARE CURS CURSOR FOR                                                 5300
 54                  SELECT WORKDEPT, AVG(EDLEVEL), AVG(SALARY)                            5400
 55                    FROM CORPDATA.EMPLOYEE                                              5500
 56                    GROUP BY WORKDEPT                                                   5600
 57              END-EXEC.                                                                 5700
 58         ****************************************************************              5800
 59         *  Open cursor                                                *               5900
 60         ****************************************************************              6000
 61              EXEC SQL                                                                  6100
 62                OPEN CURS                                                               6200
 63              END-EXEC.                                                                 6300
```

1 プリコンパイラーがソース・レコードを読み取った時に割り当てるレコード番号。レコード番号は、エラー・メッセージや SQL 実行時処理でソース・レコードを識別するのに使用されます。

2 ソース・レコードから取られた順序番号。この順序番号は、原始ステートメント入力ユーティリティー (SEU) を使ってソース・メンバーを編集するときに表示される番号です。

3 ソース・レコードが最後に変更された日付。 Last Change がブランクの場合は、そのレコードが作成されてから変更されていないことを示します。

図 2. サンプル COBOL プリコンパイラー出力の形式 (2/4)

```
5722ST1 V5R2M0 020719           Create SQL COBOL Program          CBLTEST1                        08/06/02 11:14:21   Page     3
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
   64         ************************************************************                6400
   65         *  Fetch all result rows                                   *                6500
   66         ************************************************************                6600
   67              PERFORM A010-FETCH-PROCEDURE THROUGH A010-FETCH-EXIT                    6700
   68                UNTIL SQLCODE IS = 100.                                               6800
   69         ************************************************************                6900
   70         *  Close cursor                                            *                7000
   71         ************************************************************                7100
   72              EXEC SQL                                                               7200
   73                CLOSE CURS                                                           7300
   74              END-EXEC.                                                              7400
   75              CLOSE OUTFILE.                                                         7500
   76              STOP RUN.                                                              7600
   77         ************************************************************                7700
   78         *  Fetch a row and move the information to the output record. *             7800
   79         ************************************************************                7900
   80          A010-FETCH-PROCEDURE.                                                      8000
   81              MOVE SPACES TO REC-1.                                                  8100
   82              EXEC SQL                                                               8200
   83                FETCH CURS INTO :AVG-RECORD                                          8300
   84              END-EXEC.                                                              8400
   85              IF SQLCODE IS = 0                                                      8500
   86                MOVE WORKDEPT TO DEPT-NO                                             8600
   87                MOVE AVG-SALARY TO AVERAGE-SALARY                                    8700
   88                MOVE AVG-EDUC TO AVERAGE-EDUCATION-LEVEL                             8800
   89                WRITE REC-1 AFTER ADVANCING 1 LINE.                                  8900
   90          A010-FETCH-EXIT.                                                           9000
   91              EXIT.                                                                  9100
   92         ************************************************************                9200
   93         *  An SQL error occurred.  Move the error number to the error *             9300
   94         *  record and stop running.                                *                9400
   95         ************************************************************                9500
   96          B000-SQL-ERROR.                                                            9600
   97              MOVE SPACES TO ERROR-RECORD.                                           9700
   98              MOVE SQLCODE TO ERROR-CODE.                                            9800
   99              MOVE "AN SQL ERROR HAS OCCURRED" TO ERROR-MESSAGE.                     9900
  100              WRITE ERROR-RECORD AFTER ADVANCING 1 LINE.                            10000
  101              CLOSE OUTFILE.                                                        10100
  102              STOP RUN.                                                             10200
* * * * *  E N D  O F  S O U R C E  * * * * *
```

図 2. サンプル COBOL プリコンパイラー出力の形式 (3/4)

```
5722ST1 V5R2M0 0020719           Create SQL COBOL Program         CBLTEST1                    08/06/02 11:14:21   Page     4
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 データ名 (Data Names) は、ソース・ステートメントで使用されている記号名です。

2 定義 (Define) の列には、その名前が定義されている行番号が指定されます。行番号は、SQL プリコンパイラーにより生成されます。 **** は、そのオブジェクトが定義されていないか、あるいはプリコンパイラーが宣言を認識することができなかったことを意味します。

3 参照 (Reference) の列には次の 2 種類の情報が示されます。

- 記号名がどのように定義されているか **4**
- 記号名が現れる行の番号 **5**

記号名が有効なホスト変数を参照している場合には、データ・タイプ **6** またはデータ構造 **7** も注記されます。

図 2. サンプル COBOL プリコンパイラー出力の形式 (4/4)

非 ILE SQL プリコンパイラー・コマンド

DB2 UDB SQL 開発キットには、以下のホスト言語のための非 ILE プリコンパイラー・コマンドがあります。 CRTSQLCBL (COBOL for iSeries)、CRTSQLPLI (iSeries PL/I)、および CRTSQLRPG (RPG III。これは RPG for iSeries の一部) です。オプションによっては特定の言語だけに適用されます。たとえば、*APOST と *QUOTE は COBOL に固有のオプションであり、他の言語のコマンドには用意されていません。詳しい説明については、199 ページの『付録 B. ホスト言語プリコンパイラーの DB2 UDB for iSeries CL コマンド記述』を参照してください。

詳細については、『SQL を使用する非 ILE アプリケーション・プログラムのコンパイル』を参照してください。

SQL を使用する非 ILE アプリケーション・プログラムのコンパイル

プリコンパイルが正常に完了すると、*NOGEN が指定されている場合を除き、SQL プリコンパイラーは自動的にホスト言語コンパイラーを呼び出します。そして、プログラム名、ソース・ファイル名、プリコンパイラーにより作成されたソース・メンバー名、テキスト、および USRPRF を指定して CRTxxxPGM コマンドが実行されます。

これらの言語では、次のパラメーターが渡されます。

- COBOL では、*QUOTE または *APOST は CRTCBLPGM コマンドに渡されます。
- RPG と COBOL では、SAAFLAG (*FLAG) は CRTxxxPGM コマンドに渡されます。
- RPG および COBOL では、CRTSQLxxx コマンドからの SRTSEQ および LANGID パラメーターは CRTxxxPGM コマンドで指定されます。
- RPG と COBOL では、CVTOPT (*DATETIME *VARCHAR) は常に CRTxxxPGM コマンドで指定されます。
- COBOL および RPG では、CRTSQLxxx コマンドからの TGTRLS パラメーターは、CRTxxxPGM コマンドで指定されます。 TGTRLS は CRTPLIPGM コマンドでは指定されません。プログラムは、CRTSQLPLI コマンドの TGTRLS パラメーターで指定されたレベルで保管したり、または復元することができます。
- PL/I では、MARGINS は一時ソース・ファイルの中にセットされます。
- どの言語でも、CRTSQLxxx コマンドからの REPLACE パラメーターは CRTxxxPGM コマンドで指定されます。

  パッケージをプリコンパイル処理の一部として作成した場合、CRTSQLxxx コマンドからの REPLACE パラメーター値は CRTSQLPKG コマンドで指定されます。
- どの言語の場合も、USRPRF(*USER) または USRPRF(*NAMING) によるシステム命名が指定される場合、USRPRF(*USER) は CRTxxxPGM コマンドで指定されます。 USRPRF(*OWNER) または USRPRF(*NAMING) によって SQL 命名 (*SQL) が指定される場合、USRPRF (*OWNER) が CRTxxxPGM コマンドで指定されます。

CRTxxxPGM コマンドの他のすべてのパラメーターには、省略時値が使用されます。

プリコンパイラー・コマンドの OPTION パラメーターに *NOGEN を指定することにより、ホスト言語コンパイラーの呼び出しを中断することができます。 *NOGEN では、ホスト言語コンパイラーを呼び出さないことを指定します。CRTSQLxxx コマンドでメンバー名としてオブジェクト名を使用すると、プリコンパイラーはソース・メンバーを出力ソース・ファイル (CRTSQLxxx コマンドの TOSRCFILE パラメーターとして指定) の中に作成します。この後、ホスト言語コンパイラーを明示的に呼び出し、出力ソース・ファイルの中のソース・メンバーを指定し、省略時値を変更することができます。プリコンパイルとコンパイルを別々のステップとして行ったときは、CRTSQLPKG コマンドを使用して分散プログラム用の SQL パッケージを作成することができます。

注: CRTxxxPGM コマンドを出す前に QTEMP/QSQLTEMP の中のソース・メンバーを変更してはなりません。変更すると、コンパイルは正常に実行されません。

ILE SQL プリコンパイラー・コマンド

DB2 UDB SQL 開発キット には、次の ILE プリコンパイラー・コマンドがあります。CRTSQLCI、CRTSQLCBLI、CRTSQLRPGI、および CRTSQLCPPI です。また、次の各ホスト言語ごとにプリコンパイラー・コマンドがあります。 ILE C for iSeries、ILE COBOL for iSeries、および ILE RPG for iSeriesです。言語ごとの個別コマンドにより、必須パラメーターを指定して、残りのパラメーターについては省略時値をとるようにすることができます。省略時値は、現在使用中の言語だけに適用できます。たとえば、*APOST と *QUOTE は COBOL に固有のオプションであり、他の言語のコマンドには用意されていません。詳しい説明については、199 ページの 『付録 B. ホスト言語プリコンパイラーの DB2 UDB for iSeries CL コマンド記述』を参照してください。

詳細については、以下のセクションを参照してください。

- 『SQL を使用する ILE アプリケーション・プログラムのコンパイル』

SQL を使用する ILE アプリケーション・プログラムのコンパイル

CRTSQLxxx コマンドのプリコンパイルが正常に完了すると、*NOGEN が指定されている場合を除き、SQL プリコンパイラーは自動的にホスト言語コンパイラーを呼び出します。*MODULE オプションを指定すると、SQL プリコンパイラーが CRTxxxMOD コマンドを出してモジュールを作成します。*PGM オプションを指定すると、SQL プリコンパイラーは CRTBNDxxx コマンドを出してプログラムを作成します。*SRVPGM オプションを指定すると、SQL プリコンパイラーは CRTxxxMOD コマンドを出してモジュールを作成してから、サービス・プログラム作成 (CRTSRVPGM) コマンドを出してサービス・プログラムを作成します。CRTSQLCPPI コマンドは *MODULE オブジェクトだけを作成します。

これらの言語では、次のパラメーターが渡されます。

- CRTSQLxxx コマンドで DBGVIEW(*SOURCE) を指定すると、DBGVIEW(*ALL) が CRTxxxMOD コマンドおよび CRTBNDxxx コマンドの両方で指定されます。

- CRTSQLxxx コマンドで OUTPUT(*PRINT) を指定すると、これは CRTxxxMOD コマンドおよび CRTBNDxxx コマンドの両方に渡されます。

  CRTSQLxxx コマンドで OUTPUT(*NONE) を指定すると、これは CRTxxxMOD コマンドでも CRTBNDxxx コマンドでも指定されません。
- CRTSQLxxx コマンドからの TGTRLS パラメーター値は、CRTxxxMOD、CRTBNDxxx、およびサービス・プログラム作成 (CRTSRVPGM) コマンドで指定されます。
- CRTSQLxxx コマンドからの REPLACE パラメーター値は、CRTxxxMOD、CRTBNDxxx、および CRTSRVPGM の各コマンドで指定されます。

  パッケージをプリコンパイル処理の一部として作成した場合、CRTSQLxxx コマンドからの REPLACE パラメーター値は CRTSQLPKG コマンドで指定されます。
- OBJTYPE が *PGM または *SRVPGM のいずれかで、USRPRF(*USER) または USRPRF(NAMING) を伴うシステム (*SYS) 命名を使用している場合、USRPRF(*USER) が CRTBNDxxx コマンドまたは CRTSRVPGM コマンドで指定されます。

  OBJTYPE が *PGM または *SRVPGM のいずれかで、USRPRF(*OWNER) または USRPRF(*NAMING) を伴う SQL(*SQL) 命名を使用している場合、USRPRF(*OWNER) が CRTBNDxxx コマンドまたは CRTSRVPGM コマンドで指定されます。
- C および C++ では、MARGINS は一時ソース・ファイルの中にセットされます。

  LOB ホスト変数の全長が 15M に近いとプリコンパイラーが計算する場合は、TERASPACE(*YES *TSIFC) オプションが CRTCMOD、CRTBNDC、または CRTCPPMOD コマンドで指定されます。
- COBOL では、*QUOTE または *APOST が CRTBNDCBL コマンドまたは CRTCBLMOD コマンドに渡されます。
- RPG および COBOL では、CRTSQLxxx コマンドからの SRTSEQ および LANGID パラメーターを CRTxxxMOD コマンドおよび CRTBNDxxx コマンドで指定します。
- COBOL では、CVTOPT(*VARCHAR *DATETIME *PICGGRAPHIC *FLOAT) は、常に CRTCBLMOD コマンドおよび CRTBNDCBL コマンドで指定されます。OPTION(*NOCVTDT) を指定すると (出荷時のコマンドの省略時値)、CVTOPT について追加のオプション *DATE *TIME *TIMESTAMP も指定されます。
- RPG では、OPTION(*CVTDT) を指定すると、CVTOPT(*DATETIME) は CRTRPGMOD コマンドおよび CRTBNDRPG コマンドで指定されます。

プリコンパイラー・コマンドの OPTION パラメーターに *NOGEN を指定することにより、ホスト言語コンパイラーの呼び出しを中断することができます。 *NOGEN では、ホスト言語コンパイラーを呼び出さないことを指定します。 CRTSQLxxx コマンドでメンバー名としてオブジェクト名を使用すると、プリコンパイラーはソース・メンバーを出力ソース・ファイル (TOSRCFILE パラメーター) の中に作成します。この後、ホスト言語コンパイラーを明示的に呼び出し、出力ソース・ファイルの中のソース・メンバーを指定し、省略時値を変更することができます。プリコン

パイルとコンパイルを別々のステップとして行ったときは、CRTSQLPKG コマンドを使用して分散プログラム用の SQL パッケージを作成することができます。

プログラムまたはサービス・プログラムを後から作成する場合、 USRPRF パラメーターが、CRTBNDxxx コマンド、プログラム作成 (CRTPGM) コマンド、またはサービス・プログラム作成 (CRTSRVPGM) コマンドで正しくセットされていない可能性があります。 SQL プログラムは、USRPRF パラメーターを訂正してからでないと、予測どおりに実行されません。システム命名規則を使用している場合、USRPRF パラメーターを *USER にセットしなければなりません。 SQL 命名規則を使用している場合、USRPRF パラメーターは *OWNER にセットしなければなりません。

SQL を使用するアプリケーションでのコンパイル・エラーの解釈

重要: プリコンパイルとコンパイルのステップを分けていて、ソース・プログラムが外部記述ファイルを参照している場合、参照されるファイルをプリコンパイルとコンパイルの間で変更してはなりません。そうしないと、フィールド定義への変更が一時ソース・メンバー内で変更されないため、予期しない結果が生じることがあります。

外部記述ファイルの例を以下に示します。

- COBOL の COPY DDS
- PL/I の %INCLUDE
- C または C++ の #pragma mapinc および #include
- RPG のデータ構造

SQL プリコンパイラーがホスト変数を認識しない時は、ソース・プログラムのコンパイルを行ってください。コンパイラーは EXEC SQL ステートメントを認識せず、これらのエラーを無視します。コンパイラーがその言語の SQL プリコンパイラーで定義したとおりにホスト変数宣言を解釈していることを確かめてください。

詳細については、『SQL を使用するアプリケーション・プログラムのコンパイル時のエラーおよび警告メッセージ』を参照してください。

SQL を使用するアプリケーション・プログラムのコンパイル時のエラーおよび警告メッセージ

以下に説明する条件が起こると、コンパイル処理時にエラー・メッセージまたは警告メッセージが出されることがあります。

PL/I、C、または C++ コンパイル時のエラーおよび警告メッセージ

左マージン (MARGINS パラメーターの指定値、または省略時値) より前で EXEC SQL が始まっている場合には、SQL プリコンパイラーはそのステートメントを SQL ステートメントとして認識しません。その結果、そのステートメントはそのままの形でコンパイラーに渡されます。

COBOL コンパイル時のエラーおよび警告メッセージ

EXEC SQL が 12 桁目より前で始まっている場合には、SQL プリコンパイラーはそのステートメントを SQL ステートメントとして認識しません。その結果、そのステートメントはそのままの形でコンパイラーに渡されます。

RPG コンパイル時のエラーおよび警告メッセージ

EXEC SQL が 8 桁目から 16 桁目までにコーディングされていないで、7 桁目に '/' 文字が置かれている場合には、SQL プリコンパイラーはそのステートメントを SQL ステートメントとして認識しません。その結果、そのステートメントはそのままの形でコンパイラーに渡されます。

詳しい説明については、 13 ページの『第 2 章 C および C++ アプリケーションでの SQL ステートメントのコーディング方法』から 133 ページの『第 7 章 REXX アプリケーションでの SQL ステートメントのコーディング方法』までのそれぞれのプログラミング例を参照してください。

SQL を使用するアプリケーションのバインド

アプリケーション・プログラムを実行するためには、その前に、プログラムと、プログラムで指定したテーブルおよびビューとを関係づけておかなければなりません。このプロセスを**バインド処理**と呼びます。そしてバインド処理の結果を**アクセス・プラン**と呼びます。

アクセス・プランは、各 SQL 要求を満たすのに必要な処置を記述した制御構造です。アクセス・プランには、プログラムに関する情報とそのプログラムが使用しようとするデータに関する情報が収められています。

非分散 SQL プログラムの場合、アクセス・プランはそのプログラム内に保管されます。分散 SQL プログラム (RDB パラメーターが CRTSQLxxx コマンドで指定されている) の場合は、アクセス・プランは、指定したリレーショナル・データベースに置かれている SQL パッケージに保管されます。

SQL は、プログラム・オブジェクトが作成されると、自動的にアクセス・プランをバインドして作成しようとします。非 ILE コンパイルの場合、これは CRTxxxPGM が正常に完了した結果として起こります。 ILE コンパイルの場合、これは CRTBNDxxx、CRTPGM、または CRTSRVPGM の各コマンドが正常に完了した結果として起こります。実行時に、アクセス・プランが無効であること (たとえば、参照テーブルが異なるライブラリーにある)、あるいはパフォーマンスを向上させるような変更 (たとえば、索引の追加) などがデータベースに行われたことを、 DB2 UDB for iSeries が見つけると、新しいアクセス・プランが自動的に作成されます。バインド処理では、次の 3 つのことが行われます。

1. **データベースの記述を用いて SQL ステートメントの妥当性検査を再度行う。**バインド処理の時点で、テーブル、ビュー、および列の名前が有効かどうかについて、SQL ステートメントが検査されます。指定したテーブルまたはビューがプリコンパイルまたはコンパイル時に存在していない時は、妥当性検査は実行時に行われます。実行時にもテーブルまたはビューが存在しない場合には、負の SQLCODE が返されます。
2. **プログラムで処理したいデータにアクセスするのに必要な索引を選択する。**索引を選択する際には、テーブル・サイズ、その他の要因が考慮されて、アクセス・プランが作成されます。データをアクセスするのに使用できるすべての索引が考慮され、データに至るパスを選択するとき、(もしあれば) どの索引を使用するかが決定されます。

3. **アクセス・プランの作成を試みる。**すべての SQL ステートメントが有効であれば、バインド処理はアクセス・プランを作成して、それをプログラムの中に保管します。

プログラムがアクセスするテーブルまたはビューの特性が変わると、それまでのアクセス・プランは有効でなくなる場合があります。有効でないアクセス・プランを含むプログラムを実行しようとすると、システムはアクセス・プランを再作成することを自動的に試みます。アクセス・プランが再作成できない時は、負の SQLCODE が返されます。この場合は、ユーザーはプログラムの SQL ステートメントを変更して、CRTSQLxxx コマンドを再度出して状況を訂正しなければならないことがあります。

たとえば、あるプログラムに、TABLEA の COLUMNA を参照する SQL ステートメントが含まれている場合に、ユーザーが TABLEA を削除し再作成した結果、COLUMNA が存在しなくなったとすると、そのプログラムを呼び出したとき、COLUMNA が存在していないので、自動再バインドは正しく行われません。この場合は、ユーザーはプログラムのソース仕様を変更して、CRTSQLxxx コマンドを出し直さなければなりません。

詳細については、『SQL を使用するアプリケーションでのプログラム参照』を参照してください。

SQL を使用するアプリケーションでのプログラム参照

SQL プログラムの中の SQL ステートメントで参照されるスキーマ、テーブル、ビュー、SQL パッケージ、および索引は、いずれも、プログラムの作成時にライブラリーのオブジェクト情報リポジトリー (OIR) に保管されます。

CL コマンドのプログラム参照表示 (DSPPGMREF) を使用すると、プログラムの中のすべてのオブジェクト参照を表示することができます。 SQL の命名規則が使用されている場合は、ライブラリー名は次のいずれかの形で OIR に保管されます。

1. SQL 名が完全修飾されている場合には、コレクション名が名前の修飾子として保管されます。
2. SQL 名が完全修飾されていないで、DFTRDBCOL パラメーターの指定がない場合は、ステートメントの権限 ID が名前の修飾子として保管されます。
3. SQL 名が完全修飾されていないで、DFTRDBCOL パラメーターの指定がある場合は、DFTRDBCOL パラメーターで指定されたスキーマ名が名前の修飾子として保管されます。

システムの命名規則が使用されている場合には、ライブラリー名は次のいずれかの形で OIR に保管されます。

1. オブジェクト名が完全修飾されている場合には、ライブラリー名が名前の修飾子として保管されます。
2. オブジェクトが完全修飾されていないで、DFTRDBCOL パラメーターの指定がない場合は、*LIBL が保管されます。
3. SQL 名が完全修飾されていないで、DFTRDBCOL パラメーターの指定がある場合は、DFTRDBCOL パラメーターで指定されたスキーマ名が名前の修飾子として保管されます。

SQL プリコンパイラー・オプションの表示

SQL アプリケーション・プログラムのコンパイルが正常に完了したら、モジュール表示 (DSPMOD)、プログラム表示 (DSPPGM)、またはサービス・プログラム表示 (DSPSRVPGM) の各コマンドを使用して、SQL プリコンパイルで指定済みのいくつかのオプションを判別することができます。この情報が必要になるのは、プログラムのソース仕様を変更する必要がある時です。プログラムを再度コンパイルするときに、これらと同じ SQL プリコンパイラー・オプションを CRTSQLxxx コマンドで指定することができます。

SQL 情報印刷 (PRTSQLINF) コマンドを使用しても、SQL プリコンパイルで指定したいくつかのオプションを判別することができます。

組み込み SQL を使用したプログラムの実行

プリコンパイルとコンパイルが正常に完了した後で、SQL ステートメントが組み込まれているホスト言語プログラムを実行する方法は、他のホスト・プログラムを実行する場合と同じです。次のように、

```
CALL pgm-name
```

システム・コマンド行から入力してください。プログラムの実行方法の詳細については、「CL プログラミング」  を参照してください。

注: 新しいリリースを導入した後、ユーザーがプログラムに対する *CHANGE 権限を持っていない場合、構造化照会言語 (SQL) プログラムを使用する QHST で、メッセージ CPF2218 を受け取ることがあります。*CHANGE 権限を持つユーザーがいったんプログラムを呼び出した後は、アクセス・プランが更新され、その旨を知らせるメッセージが出されます。

詳細については、以下のセクションを参照してください。

- 『組み込み SQL を使用したプログラムの実行: OS/400 DDM の考慮事項』
- 『組み込み SQL を使用したプログラム実行: 一時変更に関する考慮事項』
- 159 ページの『組み込み SQL を使用したプログラムの実行: SQL 戻りコード』

組み込み SQL を使用したプログラムの実行: OS/400 DDM の考慮事項

SQL は、DDM (分散データ管理) ファイルを経由するリモート・ファイルのアクセスをサポートしていません。SQL は、DRDA® (分散リレーショナル・データベース体系™) を経由するリモート・アクセスをサポートしています。

組み込み SQL を使用したプログラム実行: 一時変更に関する考慮事項

一時変更 (OVRDBF コマンドによって指定します) を使用すれば、別のテーブルまたはビューを参照したり、あるいは、プログラムまたは SQL パッケージの操作特

性の一部を変更することができます。一時変更を指定した場合には、下記のパラメーターが処理されます。一時変更を指定した場合には、下記のパラメーターが処理されます。

TOFILE

MBR

SEQONLY

INHWRT

WAITRCD

上記以外の一時変更パラメーターはすべて無視されます。 SQL パッケージ内のステートメントの一時変更は、以下の両方の操作によって行われます。

1. OVRDBF コマンドでの OVRSCOPE(*JOB) パラメーターの指定
2. リモート・コマンド投入 (SBMRMTCMD) コマンドの使用による、アプリケーション・サーバーへのコマンドの送信

長名で作成されたテーブルおよびビューを一時変更するには、そのテーブルまたはビューと関連付けられたシステム名を使用して一時変更を作成することができます。SQL ステートメントで長名が指定されている場合、一時変更は対応するシステム名を用いて検出されます。

別名は、実際には DDM ファイルとして作成されます。別名 (DDM ファイル) を参照する一時変更を作成することもできます。この場合、その一時変更を含むファイルを参照する SQL ステートメントは、実際には別名が参照するファイルを使用していることになります。

一時変更の詳細については、「データベース・プログラミング」および「ファイル管理」を参照してください。

組み込み SQL を使用したプログラムの実行: SQL 戻りコード

SQL 戻りコードのリストは、iSeries Information Center の『SQL メッセージおよびコード 』を参照してください。

付録 A. DB2 UDB for iSeries ステートメントを使用するサンプル・プログラム

この付録には、DB2 UDB for iSeries システムがサポートする各言語で SQL ステートメントをコーディングする方法を示したサンプル・アプリケーションが記載されています。

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

SQL ステートメントを使用するプログラム例

以下のプログラム言語について、ホスト言語による SQL ステートメントのコーディング方法の例を示したプログラムが提供されています。

- ILE C および C++
- COBOL および ILE COBOL
- PL/I
- RPG for iSeries
- ILE RPG for iSeries
- REXX

サンプル・アプリケーションは、年俸に基づく昇給を行うものです。

どのサンプル・プログラムでも同じ報告書が作成されますが、その報告書はこの付録の最後に示されています。報告書の最初の部分には、プロジェクトに参加し、昇給を受けたすべての社員がプロジェクト別に示されます。報告書の第 2 の部分には、新しい給与支出額がプロジェクト別にされます。

サンプル・プログラムについての注:

次の注記は、すべてのサンプル・プログラムに適用されます。

SQL ステートメントは大文字でも小文字でも入力できます。

1 このホスト言語ステートメントは、SQL テーブル PROJECT に関する外部定義を検索するためのものです。これらの外部定義は、ホスト変数としてもホスト構造としても使用できます。

注:

1. RPG for iSeries では、外部で記述された構造の中のフィールド名のうち、6 文字より長い名前は変更しなければなりません。
2. REXX では、外部定義の検索はサポートされません。

2 SQL INCLUDE SQLCA ステートメントは、PL/I、C、および COBOL プログラム用の SQLCA を挿入するために使用されます。 RPG プログラムの場合は、SQL プリコンパイラーが入力仕様セクションの最後にあるソース・プログラムに SQLCA データ構造を自動的に置きます。 REXX の場合

は、SQLCA によってマッピングされる連続データ域ではなく、個別の変数の形で SQLCA フィールドが保管されます。

3 この SQL WHENEVER ステートメントは、SQL ステートメントに SQLERROR (SQLCODE < 0) が現れたとき、制御権が渡されるホスト言語ラベルを定義します。この WHENEVER SQLERROR ステートメントは、次の WHENEVER SQLERROR ステートメントが現れるまですべての SQL ステートメントに適用されます。 REXX では、WHENEVER ステートメントはサポートされません。その代わりに、REXX では SIGNAL ON ERROR 機能を使用します。

4 この SQL UPDATE ステートメントは、*SALARY* 列を更新するためのものです。この列には、ホスト変数 PERCENTAGE (RPG では PERCNT) に入っている比率として社員の給与が入ります。更新される行は、社員の年俸が 2000 を超えている行です。 REXX の場合は、ホスト変数があると UPDATE を直接実行できないため、このステートメントは PREPARE と EXECUTE になります。

5 この SQL COMMIT ステートメントは、SQL UPDATE ステートメントによって行われた変更をコミットするためのものです。変更されたすべての行のレコード・ロックが解除されます。

注: プログラムは、COMMIT(*CHG) を使用してプリコンパイルされています。 (REXX の場合は、*CHG が省略時値になります。)

6 この SQL DECLARE CURSOR ステートメントはカーソル C1 を定義しています。カーソル C1 は、2 つのテーブル EMPLOYEE と EMPPROJACT を結合し、昇給を受けた社員 (commission > 2000) の行を返します。行は、プロジェクト番号と社員番号 (PROJNO 列と EMPNO 列) の昇順に返されます。 REXX の場合は、ホスト変数を含む場合にステートメント文字列を指定して DECLARE CURSOR ステートメントを直接指定することができないため、このステートメントは PREPARE と DECLARE CURSOR になります。

7 この SQL OPEN ステートメントはカーソル C1 をオープンして、行の取り出し (FETCH) ができるようにします。

8 この SQL WHENEVER ステートメントは、すべての行が取り出されたとき (SQLCODE = 100) 制御権が渡されるホスト言語ラベルを定義します。 REXX の場合は、SQLCODE を明示的に検査しなければなりません。

9 この SQL FETCH ステートメントはカーソル C1 に関するすべての列を返し、返した値をホスト構造の対応する要素に入れます。

10 すべての行が取り出されると、制御権がこのラベルに返されます。 SQL CLOSE ステートメントはカーソル C1 をクローズします。

11 この SQL DECLARE CURSOR ステートメントはカーソル C2 を定義しています。カーソル C2 は 3 つのテーブル EMPPROJACT、PROJECT、および EMPLOYEE を結合します。その結果は、PROJNO 列と PROJNAME 列によってグループ化されます。 COUNT 関数は各グループの行数を返します。 SUM 関数は新しい給与支給額を各プロジェクト別に計算します。 ORDER BY 1 文節は、最終結果の列 (EMPPROJACT.PROJNO) の内容に基づいて行を検索することを指定しています。REXX の場合は、ホスト変数

を含む場合にステートメント文字列を指定して DECLARE CURSOR ステートメントを直接指定することができないため、このステートメントは PREPARE と DECLARE CURSOR になります。

12 この SQL FETCH ステートメントはカーソル C2 に関する結果の列を返し、返した値をプログラムにより記述されているホスト構造の対応する要素に入れます。

13 この SQL WHENEVER ステートメントには CONTINUE オプションが指定されているので、SQL ROLLBACK ステートメントでエラーが起こったかどうかに関係なく、次のステートメントに移って処理が続行されます。SQL ROLLBACK ステートメントにはエラーがないことが前提とされます。仮にエラーが生じても、このオプションによりプログラムがループに入ることが防止されます。 SQLERROR ステートメントは、次の WHENEVER SQLERROR ステートメントが現れるまですべての SQL ステートメントに適用されます。 REXX では、WHENEVER ステートメントはサポートされません。その代わりに、REXX では SIGNAL OFF ERROR 機能を使用します。

14 この SQL ROLLBACK ステートメントは、更新中にエラーが起こった場合にテーブルを元の状態に復元します。

例: ILE C および C++ プログラム内の SQL ステートメント

このサンプル・プログラムは、C プログラミング言語で作成されています。次の条件が満たされれば、同じプログラムは C++ でも動作します。

- 18 行目の前に SQL BEGIN DECLARE SECTION ステートメントを追加する
- 42 行目の後に SQL END DECLARE SECTION ステートメントを追加する

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

```
5722ST1 V5R2M0 020719            Create SQL ILE C Object         CEX                          08/06/02 15:52:26   Page   1
Source type..............C
Object name..............CORPDATA/CEX
Source file..............CORPDATA/SRC
Member...................CEX
To source file...........QTEMP/QSQLTEMP
Options..................*XREF
Listing option...........*PRINT
Target release...........V5R2M0
INCLUDE file.............*LIBL/*SRCFILE
Commit...................*CHG
Allow copy of data.......*YES
Close SQL cursor.........*ENDACTGRP
Allow blocking...........*READ
Delay PREPARE............*NO
Generation level.........10
Margins..................*SRCFILE
Printer file.............*LIBL/QSYSPRT
Date format..............*JOB
Date separator...........*JOB
Time format..............*HMS
Time separator ..........*JOB
Replace..................*YES
Relational database.....*LOCAL
User ....................*CURRENT
RDB connect method.......*DUW
Default collection.......*NONE
Dynamic default
  collection.............*NO
Package name.............*OBJLIB/*OBJ
Path.....................*NAMING
Created object type.....*PGM
Debugging view...........*NONE
User profile.............*NAMING
Dynamic user profile.....*USER
Sort Sequence............*JOB
Language ID..............*JOB
IBM SQL flagging.........*NOFLAG
ANS flagging.............*NONE
Text.....................*SRCMBRTXT
Source file CCSID........65535
Job CCSID................65535
Source member changed on 06/06/00  17:15:17

Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
    1   #include "string.h"                                                                     100
    2   #include "stdlib.h"                                                                     200
    3   #include "stdio.h"                                                                      300
    4                                                                                           400
    5   main()                                                                                  500
    6   {                                                                                       600
    7   /* A sample program which updates the salaries for those employees   */               700
    8   /* whose current commission total is greater than or equal to the    */               800
    9   /* value of 'commission'. The salaries of those who qualify are      */               900
   10   /* increased by the value of 'percentage' retroactive to 'raise_date'*/              1000
   11   /* A report is generated showing the projects which these employees  */              1100
   12   /* have contributed to ordered by project number and employee ID.    */              1200
   13   /* A second report shows each project having an end date occurring   */              1300
   14   /* after 'raise_date' (is potentially affected by the retroactive    */              1400
   15   /* raises) with its total salary expenses and a count of employees   */              1500
   16   /* who contributed to the project.                                   */              1600
   17                                                                                          1700
```

図 3. SQL ステートメントを使用したサンプル C プログラム (1/5)

```
5722ST1 V5R2M0 020719             Create SQL ILE C Object          CEX                              08/06/02 15:52:26   Page     2
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
   18      short work_days  = 253;          /* work days during in one year */              1800
   19      float commission = 2000.00;      /* cutoff to qualify for raise  */              1900
   20      float percentage = 1.04;         /* raised salary as percentage  */              2000
   21      char raise_date??(12??) = "1982-06-01"; /*  effective raise date */              2100
   22                                                                                       2200
   23      /* File declaration for qprint */                                                2300
   24      FILE *qprint;                                                                    2400
   25                                                                                       2500
   26      /* Structure for report 1 */                                                     2600
   27      1  #pragma mapinc ("project","CORPDATA/PROJECT(PROJECT)","both","p z")           2700
   28      #include "project"                                                               2800
   29      struct {                                                                         2900
   30              CORPDATA_PROJECT_PROJECT_both_t Proj_struct;                             3000
   31              char  empno??(7??);                                                      3100
   32              char  name??(30??);                                                      3200
   33              float salary;                                                            3300
   34              } rpt1;                                                                  3400
   35                                                                                       3500
   36      /* Structure for report 2 */                                                     3600
   37      struct {                                                                         3700
   38              char projno??(7??);                                                      3800
   39              char project_name??(37??);                                               3900
   40              short employee_count;                                                    4000
   41              double total_proj_cost;                                                  4100
   42             } rpt2;                                                                   4200
   43                                                                                       4300
   44      2  exec sql include SQLCA;                                                       4400
   45                                                                                       4500
   46      qprint=fopen("QPRINT","w");                                                      4600
   47                                                                                       4700
   48      /* Update the selected projects by the new percentage. If an error */           4800
   49      /* occurs during the update, ROLLBACK the changes.                 */           4900
   50      3  EXEC SQL WHENEVER SQLERROR GO TO update_error;                               5000
   51      4  EXEC SQL                                                                     5100
   52           UPDATE CORPDATA/EMPLOYEE                                                    5200
   53              SET SALARY = SALARY * :percentage                                        5300
   54              WHERE COMM >= :commission ;                                              5400
   55                                                                                       5500
   56      /* Commit changes */                                                             5600
   57      5  EXEC SQL                                                                     5700
   58           COMMIT;                                                                     5800
   59      EXEC SQL WHENEVER SQLERROR GO TO report_error;                                   5900
   60                                                                                       6000
   61      /* Report the updated statistics for each employee assigned to the */           6100
   62      /* selected projects.                                              */           6200
   63                                                                                       6300
   64      /* Write out the header for Report 1 */                                          6400
   65      fprintf(qprint,"                     REPORT OF PROJECTS AFFECTED \               6500
   66    BY RAISES");                                                                       6600
   67      fprintf(qprint,"¥n¥nPROJECT  EMPID     EMPLOYEE NAME    ");                      6700
   68      fprintf(qprint,  "                  SALARY¥n");                                  6800
   69                                                                                       6900
   70      6  exec sql                                                                     7000
   71           declare c1 cursor for                                                       7100
   72             select distinct projno, empprojact.empno,                                 7200
   73                    lastname||', '||firstnme, salary                                   7300
   74             from corpdata/empprojact, corpdata/employee                               7400
   75             where empprojact.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
```

図 3. SQL ステートメントを使用したサンプル C プログラム (2/5)

```
5722ST1 V5R2M0 020719            Create SQL ILE C Object          CEX                              08/06/02 15:52:26   Page     3
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
   96                                                                                          9600
   97     /* For all projects ending at a date later than the 'raise_date'   * /             9700
   98     /* (i.e. those projects potentially affected by the salary raises) */             9800
   99     /* generate a report containing the project number, project name   */             9900
  100     /* the count of employees participating in the project and the     */            10000
  101     /* total salary cost of the project.                               */            10100
  102                                                                                        10200
  103     /* Write out the header for Report 2 */                                            10300
  104     fprintf(qprint,"¥n¥n¥n                    ACCUMULATED STATISTICS\                 10400
  105    BY PROJECT");                                                                       10500
  106     fprintf(qprint,  "¥n¥nPROJECT                                      \              10600
  107      NUMBER OF       TOTAL");                                                          10700
  108     fprintf(qprint,    "¥nNUMBER   PROJECT NAME                       \              10800
  109      EMPLOYEES       COST¥n");                                                         10900
  110                                                                                        11000
  111   11 EXEC SQL                                                                          11100
  112          DECLARE C2 CURSOR FOR                                                         11200
  113            SELECT EMPPROJACT.PROJNO, PROJNAME, COUNT(*),                               11300
  114               SUM ( ( DAYS(EMENDATE) - DAYS(EMSTDATE) ) * EMPTIME *                    11400
  115                     (DECIMAL( SALARY / :work_days ,8,2)))                              11500
  116            FROM CORPDATA/EMPPROJACT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE               11600
  117            WHERE EMPPROJACT.PROJNO=PROJECT.PROJNO  AND                                 11700
  118                  EMPPROJACT.EMPNO =EMPLOYEE.EMPNO  AND                                 11800
  119                  PRENDATE > :raise_date                                                11900
  120            GROUP BY EMPPROJACT.PROJNO, PROJNAME                                        12000
  121            ORDER BY 1;                                                                 12100
  122     EXEC SQL                                                                           12200
  123          OPEN C2;                                                                      12300
  124                                                                                        12400
  125     /* Fetch and write the rows to QPRINT */                                           12500
  126     EXEC SQL WHENEVER NOT FOUND GO TO done2;                                           12600
  127                                                                                        12700
  128     do {                                                                               12800
  129    12 EXEC SQL                                                                         12900
  130           FETCH C2 INTO :rpt2;                                                         13000
  131       fprintf(qprint,"¥n%6s   %-36s  %6d       %9.2f",                               13100
  132               rpt2.projno,rpt2.project_name,rpt2.employee_count,                       13200
  133               rpt2.total_proj_cost);                                                   13300
  134     }                                                                                  13400
  135     while (SQLCODE==0);                                                                13500
  136                                                                                        13600
  137    done2:                                                                              13700
  138     EXEC SQL                                                                           13800
  139          CLOSE C2;                                                                     13900
  140     goto finished;                                                                     14000
  141                                                                                        14100
  142     /* Error occured while updating table.  Inform user and rollback   */             14200
  143     /* changes.                                                        */             14300
  144    update_error:                                                                       14400
  145    13 EXEC SQL WHENEVER SQLERROR CONTINUE;                                             14500
  146     fprintf(qprint,"*** ERROR Occurred while updating table.  SQLCODE="                14600
  147             "%5d¥n",SQLCODE);                                                          14700
  148    14 EXEC SQL                                                                         14800
  149          ROLLBACK;                                                                     14900
  150     goto finished;                                                                     15000
  151                                                                                        15100
  152     /* Error occured while generating reports.  Inform user and exit.  */             15200
  153    report_error:                                                                       15300
  154     fprintf(qprint,"*** ERROR Occurred while generating reports.  "                    15400
  155             "SQLCODE=%5d¥n",SQLCODE);                                                  15500
  156     goto finished;                                                                     15600
  157                                                                                        15700
  158     /* All done  */                                                                    15800
  159    finished:                                                                           15900
  160     fclose(qprint);                                                                    16000
  161     exit(0);                                                                           16100
  162                                                                                        16200
  163   }                                                                                    16300
* * * * *  E N D  O F  S O U R C E  * * * * *
```

図 3. SQL ステートメントを使用したサンプル C プログラム (3/5)

```
CROSS REFERENCE
Data Names                      Define    Reference
commission                         19       FLOAT(24)
                                            54 75
done1                              ****     LABEL
                                            81
done2                              ****     LABEL
                                            126
employee_count                     40       SMALL INTEGER PRECISION(4,0) IN rpt2
empno                              31       VARCHAR(7) IN rpt1
                                            85
name                               32       VARCHAR(30) IN rpt1
                                            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.EMPPROJACT
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.EMPPROJACT
EMENDATE                           ****     COLUMN
                                            114
EMPLOYEE                           ****     TABLE IN CORPDATA
                                            52 74 116
EMPLOYEE                           ****     TABLE
                                            75 118
EMPNO                              ****     COLUMN IN EMPPROJACT
                                            72 75 76 118
EMPNO                              ****     COLUMN IN EMPLOYEE
                                            75 118
EMPNO                              74       CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
EMPNO                              74       CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMPPROJACT                         ****     TABLE
                                            72 75 113 117 118 120
EMPPROJACT                         ****     TABLE IN CORPDATA
                                            74 116
EMPTIME                            74       DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJACT
EMPTIME                            ****     COLUMN
                                            114
EMSTDATE                           74       DATE(10) COLUMN IN CORPDATA.EMPPROJACT
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
```

図 3. SQL ステートメントを使用したサンプル C プログラム (4/5)

```
5722ST1 V5R2M0 020719            Create SQL ILE C Object          CEX                              08/06/02 15:52:26   Page     5
CROSS REFERENCE

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.EMPPROJACT
PROJNO                          ****     COLUMN IN EMPPROJACT
                                         113 117 120
PROJNO                          ****     COLUMN IN PROJECT
                                         117
PROJNO                          116      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PRSTAFF                         27       DECIMAL(5,2) IN Proj_struct
PRSTAFF                         116      DECIMAL(5,2) COLUMN IN CORPDATA.PROJECT
PRSTDATE                        27       DATE(10) IN Proj_struct
PRSTDATE                        116      DATE(10) COLUMN IN CORPDATA.PROJECT
RESPEMP                         27       VARCHAR(6) IN Proj_struct
RESPEMP                         116      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
SALARY                          ****     COLUMN
                                         53 53 73 115
SALARY                          74       DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
SEX                             74       CHARACTER(1) COLUMN IN CORPDATA.EMPLOYEE
WORKDEPT                        74       CHARACTER(3) COLUMN IN CORPDATA.EMPLOYEE
No errors found in source
163 Source records processed
* * * * *  E N D  O F  L I S T I N G  * * * * *
```

図 3. SQL ステートメントを使用したサンプル C プログラム (5/5)

例: COBOL および ILE COBOL プログラム内の SQL ステートメント

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

```
5722ST1 V5R2M0 020719           Create SQL COBOL Program         CBLEX                        08/06/02 11:09:13   Page   1
Source type..............COBOL
Program name.............CORPDATA/CBLEX
Source file..............CORPDATA/SRC
Member...................CBLEX
To source file...........QTEMP/QSQLTEMP
Options..................*SRC      *XREF
Target release...........V5R2M0
INCLUDE file.............*LIBL/*SRCFILE
Commit...................*CHG
Allow copy of data.......*YES
Close SQL cursor.........*ENDPGM
Allow blocking...........*READ
Delay PREPARE............*NO
Generation level.........10
Printer file.............*LIBL/QSYSPRT
Date format..............*JOB
Date separator...........*JOB
Time format..............*HMS
Time separator ..........*JOB
Replace..................*YES
Relational database.....*LOCAL
User ....................*CURRENT
RDB connect method.......*DUW
Default collection.......*NONE
Dynamic default
  collection.............*NO
Package name.............*PGMLIB/*PGM
Path.....................*NAMING
Created object type.....*PGM
User profile.............*NAMING
Dynamic user profile.....*USER
Sort Sequence............*JOB
Language ID..............*JOB
IBM SQL flagging.........*NOFLAG
ANS flagging.............*NONE
Text.....................*SRCMBRTXT
Source file CCSID........65535
Job CCSID................65535
Source member changed on 07/01/96  09:44:58
    1
    2         ****************************************************************
    3         * A sample program which updates the salaries for those        *
    4         * employees whose current commission total is greater than or   *
    5         * equal to the value of COMMISSION. The salaries of those who   *
    6         * qualify are increased by the value of PERCENTAGE retroactive *
    7         * to RAISE-DATE. A report is generated showing the projects     *
    8         * which these employees have contributed to ordered by the      *
    9         * project number and employee ID. A second report shows each    *
   10         * project having an end date occurring  after RAISE-DATE        *
   11         * (i.e. potentially affected by the retroactive raises ) with   *
   12         * its total salary expenses and a count of employees who        *
   13         * contributed to the project.                                   *
   14         ****************************************************************
   15
   16
   17          IDENTIFICATION DIVISION.
   18
   19          PROGRAM-ID.  CBLEX.
   20          ENVIRONMENT DIVISION.
   21          CONFIGURATION SECTION.
   22          SOURCE-COMPUTER. IBM-AS400.
   23          OBJECT-COMPUTER. IBM-AS400.
   24          INPUT-OUTPUT SECTION.
   25
   26          FILE-CONTROL.
   27              SELECT PRINTFILE ASSIGN TO PRINTER-QPRINT
   28                 ORGANIZATION IS SEQUENTIAL.
   29
   30          DATA DIVISION.
   31
```

図4. SQL ステートメントを使用したサンプル COBOL プログラム (1/7)

```
5722ST1 V5R2M0 020719           Create SQL COBOL Program         CBLEX                      08/06/02 11:09:13        Page   2
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
   32          FILE SECTION.
   33
   34          FD  PRINTFILE
   35              BLOCK CONTAINS 1 RECORDS
   36              LABEL RECORDS ARE OMITTED.
   37          01  PRINT-RECORD PIC X(132).
   38
   39          WORKING-STORAGE SECTION.
   40          77  WORK-DAYS PIC S9(4) BINARY VALUE 253.
   41          77  RAISE-DATE PIC X(11) VALUE "1982-06-01".
   42          77  PERCENTAGE PIC S999V99 PACKED-DECIMAL.
   43          77  COMMISSION PIC S99999V99 PACKED-DECIMAL VALUE 2000.00.
   44
   45         ****************************************************************
   46         *  Structure for report 1.                                     *
   47         ****************************************************************
   48
   49        1  01  RPT1.
   50              COPY DDS-PROJECT OF CORPDATA-PROJECT.
   51              05  EMPNO     PIC X(6).
   52              05  NAME      PIC X(30).
   53              05  SALARY    PIC S9(6)V99 PACKED-DECIMAL.
   54
   55
   56         ****************************************************************
   57         *  Structure for report 2.                                     *
   58         ****************************************************************
   59
   60          01  RPT2.
   61              15  PROJNO PIC X(6).
   62              15  PROJECT-NAME PIC X(36).
   63              15  EMPLOYEE-COUNT PIC S9(4) BINARY.
   64              15  TOTAL-PROJ-COST PIC S9(10)V99 PACKED-DECIMAL.
   65
   66            2  EXEC SQL
   67                   INCLUDE SQLCA
   68              END-EXEC.
   69          77  CODE-EDIT PIC ---99.
   70
   71         ****************************************************************
   72         *  Headers for reports.                                        *
   73         ****************************************************************
   74
   75          01  RPT1-HEADERS.
   76              05  RPT1-HEADER1.
   77                  10  FILLER PIC X(21) VALUE SPACES.
   78                  10  FILLER PIC X(111)
   79                        VALUE "REPORT OF PROJECTS AFFECTED BY RAISES".
   80              05  RPT1-HEADER2.
   81                  10  FILLER PIC X(9) VALUE "PROJECT".
   82                  10  FILLER PIC X(10) VALUE "EMPID".
   83                  10  FILLER PIC X(35) VALUE "EMPLOYEE NAME".
   84                  10  FILLER PIC X(40) VALUE "SALARY".
   85          01  RPT2-HEADERS.
   86              05  RPT2-HEADER1.
   87                  10  FILLER PIC X(21) VALUE SPACES.
   88                  10  FILLER PIC X(111)
   89                          VALUE "ACCUMULATED STATISTICS BY PROJECT".
   90              05  RPT2-HEADER2.
   91                  10  FILLER PIC X(9) VALUE "PROJECT".
   92                  10  FILLER PIC X(38) VALUE SPACES.
   93                  10  FILLER PIC X(16) VALUE "NUMBER OF".
   94                  10  FILLER PIC X(10) VALUE "TOTAL".
   95              05  RPT2-HEADER3.
   96                  10  FILLER PIC X(9) VALUE "NUMBER".
   97                  10  FILLER PIC X(38) VALUE "PROJECT NAME".
   98                  10  FILLER PIC X(16) VALUE "EMPLOYEES".
   99                  10  FILLER PIC X(65) VALUE "COST".
  100          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 ZZZZZ9.99.
  108              05  FILLER    PIC X(96) VALUE SPACES.
```

図 4. SQL ステートメントを使用したサンプル COBOL プログラム (2/7)

```
5722ST1 V5R2M0 020719           Create SQL COBOL Program          CBLEX                            08/06/02 11:09:13   Page     3
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
  109          01  RPT2-DATA.
  110              05  PROJNO PIC X(6).
  111              05  FILLER PIC XXX VALUE SPACES.
  112              05  PROJECT-NAME PIC X(36).
  113              05  FILLER PIC X(4) VALUE SPACES.
  114              05  EMPLOYEE-COUNT PIC ZZZ9.
  115              05  FILLER PIC X(5) VALUE SPACES.
  116              05  TOTAL-PROJ-COST PIC ZZZZZZZZ9.99.
  117              05  FILLER PIC X(56) VALUE SPACES.
  118
  119          PROCEDURE DIVISION.
  120
  121          A000-MAIN.
  122              MOVE 1.04 TO PERCENTAGE.
  123              OPEN OUTPUT PRINTFILE.
  124
  125         ****************************************************************
  126         * Update the selected employees by the new percentage. If an  *
  127         * error occurs during the update, ROLLBACK the changes,       *
  128         ****************************************************************
  129
  130           3  EXEC SQL
  131                   WHENEVER SQLERROR GO TO E010-UPDATE-ERROR
  132              END-EXEC.
  133           4  EXEC SQL
  134                   UPDATE CORPDATA/EMPLOYEE
  135                     SET SALARY = SALARY * :PERCENTAGE
  136                     WHERE COMM >= :COMMISSION
  137              END-EXEC.
  138
  139         ****************************************************************
  140         *  Commit changes.                                             *
  141         ****************************************************************
  142
  143           5  EXEC SQL
  144                   COMMIT
  145              END-EXEC.
  146
  147              EXEC SQL
  148                   WHENEVER SQLERROR GO TO E020-REPORT-ERROR
  149              END-EXEC.
  150
  151         ****************************************************************
  152         *  Report the updated statistics for each employee receiving  *
  153         *  a raise and the projects that s/he participates in         *
  154         ****************************************************************
  155
  156         ****************************************************************
  157         *  Write out the header for Report 1.                          *
  158         ****************************************************************
  159
  160              write print-record from rpt1-header1
  161                    before advancing 2 lines.
  162              write print-record from rpt1-header2
  163                    before advancing 1 line.
  164           6  exec sql
  165                  declare c1 cursor for
  166                    SELECT DISTINCT projno, empprojact.empno,
  167                            lastname||", "||firstnme ,salary
  168                    from corpdata/empprojact, corpdata/employee
  169                    where empprojact.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
```

注: 8 および 9 はこの図の第 5 部にあります。

図 4. SQL ステートメントを使用したサンプル COBOL プログラム (3/7)

```
5722ST1 V5R2M0 020719           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
  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 EMPPROJACT.PROJNO, PROJNAME, COUNT(*),
  211                            SUM ( (DAYS(EMENDATE)-DAYS(EMSTDATE)) *
  212                            EMPTIME * DECIMAL((SALARY / :WORK-DAYS),8,2))
  213                     FROM CORPDATA/EMPPROJACT, CORPDATA/PROJECT,
  214                          CORPDATA/EMPLOYEE
  215                     WHERE EMPPROJACT.PROJNO=PROJECT.PROJNO AND
  216                           EMPPROJACT.EMPNO =EMPLOYEE.EMPNO AND
  217                           PRENDATE > :RAISE-DATE
  218                     GROUP BY EMPPROJACT.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
```

図 4. SQL ステートメントを使用したサンプル COBOL プログラム (4/7)

```
5722ST1 V5R2M0 020719            Create SQL COBOL Program          CBLEX                          08/06/02 11:09:13   Page     5
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
  241         ***************************************************************
  242         *  Fetch and write the rows to PRINTFILE.                     *
  243         ***************************************************************
  244
  245          B000-GENERATE-REPORT1.
  246           8  EXEC SQL
  247                   WHENEVER NOT FOUND GO TO A100-DONE1
  248              END-EXEC.
  249           9  EXEC SQL
  250                   FETCH C1 INTO :PROJECT.PROJNO, :RPT1.EMPNO,
  251                                 :RPT1.NAME, :RPT1.SALARY
  252              END-EXEC.
  253              MOVE CORRESPONDING RPT1 TO RPT1-DATA.
  254              MOVE PROJNO OF RPT1 TO PROJNO OF RPT1-DATA.
  255              WRITE PRINT-RECORD FROM RPT1-DATA
  256                    BEFORE ADVANCING 1 LINE.
  257
  258          B010-GENERATE-REPORT1-EXIT.
  259              EXIT.
  260
  261         ***************************************************************
  262         *  Fetch and write the rows to PRINTFILE.                     *
  263         ***************************************************************
  264
  265          C000-GENERATE-REPORT2.
  266              EXEC SQL
  267                   WHENEVER NOT FOUND GO TO A200-DONE2
  268              END-EXEC.
  269           12  EXEC SQL
  270                   FETCH C2 INTO :RPT2
  271              END-EXEC.
  272              MOVE CORRESPONDING RPT2 TO RPT2-DATA.
  273              WRITE PRINT-RECORD FROM RPT2-DATA
  274                    BEFORE ADVANCING 1 LINE.
  275
  276          C010-GENERATE-REPORT2-EXIT.
  277              EXIT.
  278
  279         ***************************************************************
  280         *  Error occured while updating table.  Inform user and       *
  281         *  rollback changes.                                          *
  282         ***************************************************************
  283
  284          E010-UPDATE-ERROR.
  285           13  EXEC SQL
  286                   WHENEVER SQLERROR CONTINUE
  287              END-EXEC.
  288              MOVE SQLCODE TO CODE-EDIT.
  289              STRING "*** ERROR Occurred while updating table.  SQLCODE="
  290                     CODE-EDIT DELIMITED BY SIZE INTO PRINT-RECORD.
  291              WRITE PRINT-RECORD.
  292           14  EXEC SQL
  293                   ROLLBACK
  294              END-EXEC.
  295              STOP RUN.
  296
  297         ***************************************************************
  298         *  Error occured while generating reports.  Inform user and   *
  299         *  exit.                                                      *
  300         ***************************************************************
  301
  302          E020-REPORT-ERROR.
  303              MOVE SQLCODE TO CODE-EDIT.
  304              STRING "*** ERROR Occurred while generating reports.  SQLCODE
  305         -           "=" CODE-EDIT DELIMITED BY SIZE INTO PRINT-RECORD.
  306              WRITE PRINT-RECORD.
  307              STOP RUN.
                           * * * * *  E N D  O F  S O U R C E  * * * * *
```

図 4. SQL ステートメントを使用したサンプル COBOL プログラム (5/7)

```
5722ST1 V5R2M0 020719           Create SQL COBOL Program         CBLEX                          08/06/02 11:09:13   Page     6
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
```

図 4. SQL ステートメントを使用したサンプル COBOL プログラム (6/7)

```
5722ST1 V5R2M0 020719           Create SQL COBOL Program         CBLEX                          08/06/02 11:09:13   Page     7
CROSS REFERENCE
PERCENTAGE                      42       DECIMAL(5,2)
                                         135
PHONENO                         134      CHARACTER(4) COLUMN IN CORPDATA.EMPLOYEE
PRENDATE                        50       DATE(10) IN PROJECT
PRENDATE                        ****     COLUMN
                                         217
PRENDATE                        213      DATE(10) COLUMN IN CORPDATA.PROJECT
PRINT-RECORD                    37       CHARACTER(132)
PROJECT                         50       STRUCTURE IN RPT1
PROJECT                         ****     TABLE IN CORPDATA
                                         213
PROJECT                         ****     TABLE
                                         215
PROJECT-NAME                    62       CHARACTER(36) IN RPT2
PROJECT-NAME                    112      CHARACTER(36) IN RPT2-DATA
PROJNAME                        50       VARCHAR(24) IN PROJECT
PROJNAME                        ****     COLUMN
                                         210 218
PROJNAME                        213      VARCHAR(24) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PROJNO                          50       CHARACTER(6) IN PROJECT
                                         250
PROJNO                          61       CHARACTER(6) IN RPT2
PROJNO                          101      CHARACTER(6) IN RPT1-DATA
PROJNO                          110      CHARACTER(6) IN RPT2-DATA
PROJNO                          ****     COLUMN
                                         166 171
PROJNO                          168      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
PROJNO                          ****     COLUMN IN EMPPROJACT
                                         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  * * * * *
```

図 4. SQL ステートメントを使用したサンプル COBOL プログラム (7/7)

例: PL/I 内の SQL ステートメント

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

```
5722ST1 V5R2M0 020719            Create SQL PL/I Program          PLIEX                          08/06/02 12:53:36   Page   1
Source type..............PLI
Program name.............CORPDATA/PLIEX
Source file..............CORPDATA/SRC
Member...................PLIEX
To source file...........QTEMP/QSQLTEMP
Options..................*SRC      *XREF
Target release...........V5R2M0
INCLUDE file.............*LIBL/*SRCFILE
Commit...................*CHG
Allow copy of data.......*YES
Close SQL cursor.........*ENDPGM
Allow blocking...........*READ
Delay PREPARE............*NO
Generation level.........10
Margins..................*SRCFILE
Printer file.............*LIBL/QSYSPRT
Date format..............*JOB
Date separator...........*JOB
Time format..............*HMS
Time separator ..........*JOB
Replace..................*YES
Relational database.....*LOCAL
User ....................*CURRENT
RDB connect method.......*DUW
Default collection.......*NONE
Dynamic default
  collection.............*NO
Package name.............*PGMLIB/*PGM
Path.....................*NAMING
User profile.............*NAMING
Dynamic user profile.....*USER
Sort sequence............*JOB
Language ID..............*JOB
IBM SQL flagging.........*NOFLAG
ANS flagging.............*NONE
Text.....................*SRCMBRTXT
Source file CCSID........65535
Job CCSID................65535
Source member changed on 07/01/96  12:53:08


    1    /* A sample program which updates the salaries for those employees   */             100
    2    /* whose current commission total is greater than or equal to the    */             200
    3    /* value of COMMISSION. The salaries of those who qualify are        */             300
    4    /* increased by the value of PERCENTAGE, retroactive to RAISE_DATE.  */             400
    5    /* A report is generated showing the projects which these employees  */             500
    6    /* have contributed to, ordered by project number and employee ID.   */             600
    7    /* A second report shows each project having an end date occurring   */             700
    8    /* after RAISE_DATE (i.e. is potentially affected by the retroactive */             800
    9    /* raises) with its total salary expenses and a count of employees   */             900
   10    /* who contributed to the project.                                   */            1000
   11    /*********************************************************************/            1100
   12                                                                                       1200
```

図 5. SQL ステートメントを使用したサンプル PL/I プログラム (1/6)

```
5722ST1 V5R2M0 020719            Create SQL PL/I Program         PLIEX                          08/06/02 12:53:36   Page   2
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
   13                                                                                          1300
   14    PLIEX: PROC;                                                                          1400
   15                                                                                          1500
   16      DCL RAISE_DATE CHAR(10);                                                            1600
   17      DCL WORK_DAYS  FIXED BIN(15);                                                       1700
   18      DCL COMMISSION FIXED DECIMAL(8,2);                                                  1800
   19      DCL PERCENTAGE FIXED DECIMAL(5,2);                                                  1900
   20                                                                                          2000
   21      /* File declaration for sysprint */                                                 2100
   22      DCL SYSPRINT FILE EXTERNAL OUTPUT STREAM PRINT;                                     2200
   23                                                                                          2300
   24      /* Structure for report 1 */                                                        2400
   25      DCL 1 RPT1,                                                                         2500
   26    1 %INCLUDE PROJECT (PROJECT, RECORD,,COMMA);                                          2600
   27             15 EMPNO     CHAR(6),                                                       2700
   28             15 NAME      CHAR(30),                                                      2800
   29             15 SALARY    FIXED DECIMAL(8,2);                                            2900
   30                                                                                          3000
   31      /* Structure for report 2 */                                                        3100
   32      DCL 1 RPT2,                                                                         3200
   33            15 PROJNO          CHAR(6),                                                   3300
   34            15 PROJECT_NAME    CHAR(36),                                                  3400
   35            15 EMPLOYEE_COUNT  FIXED BIN(15),                                           3500
   36            15 TOTL_PROJ_COST  FIXED DECIMAL(10,2);                                     3600
   37                                                                                          3700
   38     2  EXEC SQL INCLUDE SQLCA;                                                          3800
   39                                                                                          3900
   40      COMMISSION = 2000.00;                                                              4000
   41      PERCENTAGE = 1.04;                                                                 4100
   42      RAISE_DATE = '1982-06-01';                                                         4200
   43      WORK_DAYS  = 253;                                                                  4300
   44      OPEN FILE(SYSPRINT);                                                               4400
   45                                                                                         4500
   46      /* Update the selected employee's salaries by the new percentage. */               4600
   47      /* If an error occurs during the update, ROLLBACK the changes.    */               4700
   48    3  EXEC SQL WHENEVER SQLERROR GO TO UPDATE_ERROR;                                    4800
   49    4  EXEC SQL                                                                          4900
   50           UPDATE CORPDATA/EMPLOYEE                                                     5000
   51              SET SALARY = SALARY * :PERCENTAGE                                         5100
   52              WHERE COMM >= :COMMISSION ;                                               5200
   53                                                                                        5300
   54      /* Commit changes */                                                              5400
   55    5  EXEC SQL                                                                        5500
   56           COMMIT;                                                                     5600
   57      EXEC SQL WHENEVER SQLERROR GO TO REPORT_ERROR;                                   5700
   58                                                                                       5800
   59      /* Report the updated statistics for each project supported by one */            5900
   60      /* of the selected employees.                                       */           6000
   61                                                                                       6100
   62      /* Write out the header for Report 1 */                                          6200
   63      put file(sysprint)                                                               6300
   64          edit('REPORT OF PROJECTS AFFECTED BY EMPLOYEE RAISES')                       6400
   65              (col(22),a);                                                             6500
   66      put file(sysprint)                                                               6600
   67          edit('PROJECT','EMPID','EMPLOYEE NAME','SALARY')                             6700
   68              (skip(2),col(1),a,col(10),a,col(20),a,col(55),a);                        6800
   69                                                                                       6900
   70    6  exec sql                                                                        7000
   71           declare c1 cursor for                                                       7100
   72             select DISTINCT projno, EMPPROJACT.empno,                                 7200
   73                             lastname||', '||firstnme, salary                          7300
   74             from CORPDATA/EMPPROJACT, CORPDATA/EMPLOYEE                               7400
   75             where EMPPROJACT.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
```

図 5. SQL ステートメントを使用したサンプル PL/I プログラム (2/6)

```
5722ST1 V5R2M0 020719            Create SQL PL/I Program          PLIEX                          08/06/02 12:53:36   Page     3
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
   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
  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 occured 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
```

図 5. SQL ステートメントを使用したサンプル PL/I プログラム (3/6)

```
5722ST1 V5R2M0 020719            Create SQL PL/I Program          PLIEX                          08/06/02 12:53:36   Page     4
  150   14  EXEC SQL                                                                              15000
  151           ROLLBACK;                                                                       15100
  152       GO TO FINISHED;                                                                     15200
  153                                                                                           15300
  154       /* Error occured while generating reports.  Inform user and exit.  */              15400
  155     REPORT_ERROR:                                                                         15500
  156       PUT FILE(SYSPRINT) EDIT('*** ERROR Occurred while generating '||                    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
                             * * * * *  E N D  O F  S O U R C E  * * * * *
```

図 5. SQL ステートメントを使用したサンプル PL/I プログラム (4/6)

```
5722ST1 V5R2M0 020719            Create SQL PL/I Program          PLIEX                          08/06/02 12:53:36   Page     5
CROSS REFERENCE
Data Names                     Define    Reference
ACTNO                             74       SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
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.EMPPROJACT
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 EMPPROJACT
                                           72 75 77 120
EMPNO                             ****     COLUMN IN EMPLOYEE
                                           75 120
EMPNO                             74       CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
EMPNO                             74       CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMPPROJACT                        ****     TABLE
                                           72 75 115 119 120 122
EMPPROJACT                        ****     TABLE IN CORPDATA
                                           74 118
EMPTIME                           74       DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJACT
EMPTIME                           ****     COLUMN
                                           116
EMSTDATE                          74       DATE(10) COLUMN IN CORPDATA.EMPPROJACT
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
```

図 5. SQL ステートメントを使用したサンプル PL/I プログラム (5/6)

```
5722ST1 V5R2M0 020719            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.EMPPROJACT
PROJNO                          ****     COLUMN IN EMPPROJACT
                                         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
                              * * * * *  E N D  O F  L I S T I N G  * * * * *
```

図 5. SQL ステートメントを使用したサンプル PL/I プログラム (6/6)

例: RPG for iSeries プログラム内の SQL ステートメント

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

```
5722ST1 V5R2M0 020719            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............V5R2M0
INCLUDE file..............*LIBL/*SRCFILE
Commit....................*CHG
Allow copy of data........*YES
Close SQL cursor..........*ENDPGM
Allow blocking............*READ
Delay PREPARE.............*NO
Generation level..........10
Printer file..............*LIBL/QSYSPRT
Date format...............*JOB
Date separator............*JOB
Time format...............*HMS
Time separator ...........*JOB
Replace...................*YES
Relational database.......*LOCAL
User .....................*CURRENT
RDB connect method........*DUW
Default collection........*NONE
Dynamic default
  collection..............*NO
Package name..............*PGMLIB/*PGM
Path......................*NAMING
User profile..............*NAMING
Dynamic user profile......*USER
Sort sequence.............*JOB
Language ID...............*JOB
IBM SQL flagging..........*NOFLAG
ANS flagging..............*NONE
Text......................*SRCMBRTXT
Source file CCSID.........65535
Job CCSID.................65535
Source member changed on 07/01/96  17:06:17

    1       H                                                                       100
    2       F*  File declaration for QPRINT                                         200
    3       F*                                                                      300
    4       FQPRINT  O   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
```

図 6. SQL ステートメントを使用したサンプル RPG for iSeries プログラム (1/6)

```
5722ST1 V5R2M0 020719            Create SQL RPG Program          RPGEX                          08/06/02 12:55:22   Page   2
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
   34       2  C*                                                                              3400
   35        C                     Z-ADD253       WRKDAY                                      3500
   36        C                     Z-ADD2000.00   COMMI                                       3600
   37        C                     Z-ADD1.04      PERCNT                                      3700
   38        C                     MOVEL'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
   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, EMPPROJACT.EMPNO,                                6800
   70        C+           LASTNAME||', '||FIRSTNME, SALARY                                  6900
   71        C+        FROM CORPDATA/EMPPROJACT, CORPDATA/EMPLOYEE                          7000
   72        C+        WHERE EMPPROJACT.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
```

図 6. SQL ステートメントを使用したサンプル RPG for iSeries プログラム (2/6)

```
5722ST1 V5R2M0 020719            Create SQL RPG Program          RPGEX                          08/06/02 12:55:22   Page     3
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change
  105      11  C/EXEC SQL                                                                      10400
  106        C+  DECLARE C2 CURSOR FOR                                                       10500
  107        C+    SELECT EMPPROJACT.PROJNO, PROJNAME, COUNT(*),                             10600
  108        C+       SUM((DAYS(EMENDATE) - DAYS(EMSTDATE)) * EMPTIME *                      10700
  109        C+           DECIMAL((SALARY/:WRKDAY),8,2))                                     10800
  110        C+    FROM CORPDATA/EMPPROJACT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE             10900
  111        C+    WHERE EMPPROJACT.PROJNO = PROJECT.PROJNO AND                              11000
  112        C+          EMPPROJACT.EMPNO = EMPLOYEE.EMPNO AND                               11100
  113        C+          PRENDATE > :RDATE                                                   11200
  114        C+    GROUP BY EMPPROJACT.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
  125        C           SQLCOD    DOUNE0                                                    12400
  126        C/EXEC SQL                                                                      12500
  127      12  C+   FETCH C2 INTO :RPT2                                                     12600
  128        C/END-EXEC                                                                      12700
  129        C                     EXCPTRECD                                                 12800
  130        C                     END                                                       12900
  131        C           DONE2     TAG                                                       13000
  132        C/EXEC SQL CLOSE C2                                                             13100
  133        C/END-EXEC                                                                      13200
  134        C                     RETRN                                                     13300
  135        C*                                                                              13400
  136        C* Error occured while updating table.  Inform user and rollback               13500
  137        C* changes.                                                                     13600
  138        C*                                                                              13700
  139        C           UPDERR    TAG                                                       13800
  140        C                     EXCPTRECE                                                 13900
  141      13  C/EXEC SQL WHENEVER SQLERROR CONTINUE                                         14000
  142        C/END-EXEC                                                                      14100
  143        C*                                                                              14200
  144      14  C/EXEC SQL                                                                    14300
  145        C+   ROLLBACK                                                                   14400
  146        C/END-EXEC                                                                      14500
  147        C                     RETRN                                                     14600
  148        C*                                                                              14700
  149        C* Error occured while generating reports.  Inform user and exit.              14800
  150        C*                                                                              14900
  151        C           RPTERR    TAG                                                       15000
  152        C                     EXCPTRECF                                                 15100
  153        C*                                                                              15200
  154        C* All done.                                                                    15300
  155        C*                                                                              15400
  156        C           FINISH    TAG                                                       15500
  157        OQPRINT  E 0201           RECA                                                15700
  158        O                                      45 'REPORT OF PROJECTS AFFEC'            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
```

図6. SQL ステートメントを使用したサンプル RPG for iSeries プログラム (3/6)

```
5722ST1 V5R2M0 020719            Create SQL RPG Program          RPGEX                          08/06/02 12:55:22   Page     4
  182        O        E 01            RECD                                                    18200
  195        O                                    57 'CODE='                                  19500
  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
  196        O                        SQLCODL     67                                         19600
                              * * * * *  E N D  O F  S O U R C E  * * * * *
```

図 6. SQL ステートメントを使用したサンプル RPG for iSeries プログラム (4/6)

```
5722ST1 V5R2M0 020719            Create SQL RPG Program          RPGEX                          08/06/02 12:55:22   Page     5
CROSS REFERENCE
Data Names                  Define    Reference
ACTNO                          68       SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
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.EMPPROJACT
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 EMPPROJACT
                                        68 68 68 105
EMPNO                          ****     COLUMN IN EMPLOYEE
                                        68 105
EMPNO                          68       CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
EMPPROJACT                     ****     TABLE
                                        68 68 105 105 105 105
EMPPROJACT                     ****     TABLE IN CORPDATA
                                        68 105
EMPTIME                        68       DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJACT
EMPTIME                        ****     COLUMN
                                        105
EMSTDATE                       68       DATE(10) COLUMN IN CORPDATA.EMPPROJACT
EMSTDATE                       ****     COLUMN
                                        105
FINISH                         156      LABEL
FIRSTNME                       48       VARCHAR(12) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
FIRSTNME                       ****     COLUMN
                                        68
HIREDATE                       48       DATE(10) COLUMN IN CORPDATA.EMPLOYEE
JOB                            48       CHARACTER(8) COLUMN IN CORPDATA.EMPLOYEE
LASTNAME                       48       VARCHAR(15) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
LASTNAME                       ****     COLUMN
                                        68
MAJPRJ                         8        CHARACTER(6) IN RPT1
MAJPROJ                        105      CHARACTER(6) COLUMN IN CORPDATA.PROJECT
MIDINIT                        48       CHARACTER(1) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
NAME                           18       CHARACTER(30)
                                        86
PERCNT                         33       DECIMAL(7,2)
                                        48
PHONENO                        48       CHARACTER(4) COLUMN IN CORPDATA.EMPLOYEE
PNAME                          25       CHARACTER(36) IN RPT2
PRCOST                         27       DECIMAL(9,2) IN RPT2
PREND                          8        DATE(10) IN RPT1
PRENDATE                       ****     COLUMN
                                        105
PRENDATE                       105      DATE(10) COLUMN IN CORPDATA.PROJECT
PRJNUM                         24       CHARACTER(6) IN RPT2
```

図 6. SQL ステートメントを使用したサンプル RPG for iSeries プログラム (5/6)

```
5722ST1 V5R2M0 020719            Create SQL RPG Program          RPGEX                          08/06/02 12:55:22   Page     6
CROSS REFERENCE
PROJECT                          ****     TABLE IN CORPDATA
                                          105
PROJECT                          ****     TABLE
                                          105
PROJNAME                         ****     COLUMN
                                          105 105
PROJNAME                         105      VARCHAR(24) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PROJNM                           8        VARCHAR(24) IN RPT1
PROJNO                           8        CHARACTER(6) IN RPT1
                                          86
PROJNO                           ****     COLUMN
                                          68 68
PROJNO                           68       CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
PROJNO                           ****     COLUMN IN EMPPROJACT
                                          105 105 105
PROJNO                           ****     COLUMN IN PROJECT
                                          105
PROJNO                           105      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PRSTAFF                          105      DECIMAL(5,2) COLUMN IN CORPDATA.PROJECT
PRSTD                            8        DATE(10) IN RPT1
PRSTDATE                         105      DATE(10) COLUMN IN CORPDATA.PROJECT
RDATE                            32       CHARACTER(10)
                                          105
RESEM                            8        CHARACTER(6) IN RPT1
RESPEMP                          105      CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.PROJECT
RPTERR                           151      LABEL
                                          59
RPT1                             8        STRUCTURE
RPT2                             23       STRUCTURE
                                          126
SALARY                           19       DECIMAL(9,2)
                                          86
SALARY                           ****     COLUMN
                                          48 48 68 105
SALARY                           48       DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
SEX                              48       CHARACTER(1) COLUMN IN CORPDATA.EMPLOYEE
STAFF                            8        DECIMAL(5,2) IN RPT1
UPDERR                           139      LABEL
                                          45
WORKDEPT                         48       CHARACTER(3) COLUMN IN CORPDATA.EMPLOYEE
WRKDAY                           30       SMALL INTEGER PRECISION(4,0)
                                          105

No errors found in source
  196 Source records processed
                              * * * * *  E N D  O F  L I S T I N G  * * * * *
```

図 6. SQL ステートメントを使用したサンプル RPG for iSeries プログラム (6/6)

例: ILE RPG for iSeries プログラム内の SQL ステートメント

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

```
5722ST1 V5R2M0 020719           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
Listing option............*PRINT
Target release............V5R2M0
INCLUDE file..............*LIBL/*SRCFILE
Commit....................*CHG
Allow copy of data........*YES
Close SQL cursor..........*ENDMOD
Allow blocking............*READ
Delay PREPARE.............*NO
Generation level..........10
Printer file..............*LIBL/QSYSPRT
Date format...............*JOB
Date separator............*JOB
Time format...............*HMS
Time separator ...........*JOB
Replace...................*YES
Relational database.......*LOCAL
User .....................*CURRENT
RDB connect method........*DUW
Default collection........*NONE
Dynamic default
  collection..............*NO
Package name..............*OBJLIB/*OBJ
Path......................*NAMING
Created object type.......*PGM
Debugging view............*NONE
User profile..............*NAMING
Dynamic user profile......*USER
Sort sequence.............*JOB
Language ID...............*JOB
IBM SQL flagging..........*NOFLAG
ANS flagging..............*NONE
Text......................*SRCMBRTXT
Source file CCSID.........65535
Job CCSID.................65535
Source member changed on 07/01/96  15:55:32

    1       H                                                                       100
    2       F*  File declaration for QPRINT                                         200
    3       F*                                                                      300
    4       FQPRINT    O    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
```

図 7. SQL ステートメントを使用したサンプル ILE RPG for iSeries プログラム (1/6)

```
5722ST1 V5R2M0 020719           Create SQL ILE RPG Object        RPGLEEX                        08/06/02 16:03:02   Page   2
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change       Comments
   29       2  C                   Z-ADD     253           WRKDAY                                2900
   30        C                   Z-ADD     2000.00       COMMI                                 3000
   31        C                   Z-ADD     1.04          PERCNT                                3100
   32        C                   MOVEL     '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
   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
                                                      12000
```

図 7. SQL ステートメントを使用したサンプル ILE RPG for iSeries プログラム (2/6)

```
5722ST1 V5R2M0 020719           Create SQL ILE RPG Object         RPGLEEX                        08/06/02 16:03:02   Page     3
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8   SEQNBR  Last change       Comments
  100     11  C+  DECLARE C2 CURSOR FOR                                                        10000
  101       C+    SELECT 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
  121     12  C+   FETCH C2 INTO :RPT2                                                       12100
  122       C/END-EXEC                                                                        12200
  123       C                   EXCEPT    RECD                                              12300
  124       C                   END                                                         12400
  125       C     DONE2         TAG                                                         12500
  126       C/EXEC SQL CLOSE C2                                                             12600
  127       C/END-EXEC                                                                      12700
  128       C                   RETURN                                                      12800
  129       C*                                                                              12900
  130       C* Error occured 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 occured 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       OQPRINT    E            RECA        0  2 01                                    15100
  152       O                                             42 'REPORT OF PROJECTS AFFEC'     15200
  153       O                                             64 'TED BY EMPLOYEE RAISES'       15300
  154       O          E            RECA        0  1                                      15400
  155       O                                              7 'PROJECT'                      15500
  156       O                                             17 'EMPLOYEE'                     15600
  157       O                                             32 'EMPLOYEE NAME'                15700
  158       O                                             60 'SALARY'                       15800
  159       O          E            RECB        0  1                                      15900
  160       O                       PROJNO                6                               16000
  161       O                       EMPNO                15                               16100
  162       O                       NAME                 50                               16200
  163       O                       SALARY         L     61                               16300
  164       O          E            RECC        2  2                                      16400
  165       O                                             42 'ACCUMULATED STATISTIC'        16500
  166       O                                             54 'S BY PROJECT'                 16600
```

図 7. SQL ステートメントを使用したサンプル ILE RPG for iSeries プログラム (3/6)

```
5722ST1 V5R2M0 020719           Create SQL ILE RPG Object         RPGLEEX                        08/06/02 16:03:02   Page     4
  167        O          E            RECC        0  1                                          16700
  168        O                                                    7 'PROJECT'                    16800
  169        O                                                   56 'NUMBER OF'                  16900
  170        O                                                   67 'TOTAL'                      17000
  171        O          E            RECC        0  2                                          17100
  172        O                                                    6 'NUMBER'                     17200
  173        O                                                   21 'PROJECT NAME'               17300
  174        O                                                   56 'EMPLOYEES'                  17400
  175        O                                                   66 'COST'                       17500
  176        O          E            RECD        0  1                                          17600
  177        O                       PRJNUM               6                                     17700
  178        O                       PNAME               45                                     17800
  179        O                       EMPCNT        L     54                                     17900
  180        O                       PRCOST        L     70                                     18000
  181        O          E            RECE        0  1                                          18100
  182        O                                                   28 '*** ERROR Occurred while'   18200
  183        O                                                   52 ' updating table. SQLCODE'   18300
  184        O                                                   53 '='                          18400
  185        O                       SQLCOD        L     62                                     18500
  186        O          E            RECF        0  1                                          18600
  187        O                                                   28 '*** ERROR Occurred while'   18700
  188        O                                                   52 ' generating reports. SQL'   18800
  189        O                                                   57 'CODE='                      18900
  190        O                       SQLCOD        L     67                                     19000
                               * * * * *  E N D  O F  S O U R C E  * * * * *
```

図 7. SQL ステートメントを使用したサンプル ILE RPG for iSeries プログラム (4/6)

```
5722ST1 V5R2M0 020719           Create SQL ILE RPG Object         RPGLEEX                          08/06/02 16:03:02   Page     5
CROSS REFERENCE
Data Names                    Define    Reference
ACTNO                            62       SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
BIRTHDATE                        42       DATE(10) COLUMN IN CORPDATA.EMPLOYEE
BONUS                            42       DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMM                             ****     COLUMN
                                          42 62
COMM                             42       DECIMAL(9,2) COLUMN IN CORPDATA.EMPLOYEE
COMMI                            25       DECIMAL(7,2)
                                          42 62
CORPDATA                         ****     COLLECTION
                                          42 62 62 99 99 99
C1                               62       CURSOR
                                          71 80 86
C2                               99       CURSOR
                                          112 120 126
DEPTNO                           8        CHARACTER(3) IN RPT1
DEPTNO                           99       CHARACTER(3) COLUMN (NOT NULL) IN CORPDATA.PROJECT
DONE1                            85
DONE1                            ****     LABEL
                                          77
DONE2                            125
DONE2                            ****     LABEL
                                          117
EDLEVEL                          42       SMALL INTEGER PRECISION(4,0) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMENDATE                         62       DATE(10) COLUMN IN CORPDATA.EMPPROJACT
EMENDATE                         ****     COLUMN
                                          99
EMPCNT                           20       SMALL INTEGER PRECISION(4,0) IN RPT2
EMPLOYEE                         ****     TABLE IN CORPDATA
                                          42 62 99
EMPLOYEE                         ****     TABLE
                                          62 99
EMPNO                            11       CHARACTER(6) DBCS-open
                                          80
EMPNO                            42       CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
EMPNO                            ****     COLUMN IN EMPPROJACT
                                          62 62 62 99
EMPNO                            ****     COLUMN IN EMPLOYEE
                                          62 99
EMPNO                            62       CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
EMPPROJACT                       ****     TABLE
                                          62 62 99 99 99 99
EMPPROJACT                       ****     TABLE IN CORPDATA
                                          62 99
EMPTIME                          62       DECIMAL(5,2) COLUMN IN CORPDATA.EMPPROJACT
EMPTIME                          ****     COLUMN
                                          99
EMSTDATE                         62       DATE(10) COLUMN IN CORPDATA.EMPPROJACT
EMSTDATE                         ****     COLUMN
                                          99
FINISH                           150
FIRSTNME                         42       VARCHAR(12) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
FIRSTNME                         ****     COLUMN
                                          62
HIREDATE                         42       DATE(10) COLUMN IN CORPDATA.EMPLOYEE
JOB                              42       CHARACTER(8) COLUMN IN CORPDATA.EMPLOYEE
LASTNAME                         42       VARCHAR(15) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
LASTNAME                         ****     COLUMN
                                          62
MAJPROJ                          8        CHARACTER(6) IN RPT1
MAJPROJ                          99       CHARACTER(6) COLUMN IN CORPDATA.PROJECT
MIDINIT                          42       CHARACTER(1) COLUMN (NOT NULL) IN CORPDATA.EMPLOYEE
NAME                             12       CHARACTER(30) DBCS-open
                                          80
PERCNT                           27       DECIMAL(7,2)
                                          42
PHONENO                          42       CHARACTER(4) COLUMN IN CORPDATA.EMPLOYEE
PNAME                            19       CHARACTER(36) DBCS-open IN RPT2
PRCOST                           21       DECIMAL(9,2) IN RPT2
PRENDATE                         8        DATE(8) IN RPT1
PRENDATE                         ****     COLUMN
                                          99
PRENDATE                         99       DATE(10) COLUMN IN CORPDATA.PROJECT
PRJNUM                           18       CHARACTER(6) DBCS-open IN RPT2
```

図 7. SQL ステートメントを使用したサンプル ILE RPG for iSeries プログラム (5/6)

```
5229ST1 V5R2M0 020719           Create SQL ILE RPG Object         RPGLEEX                        08/06/02 16:03:02   Page     6
CROSS REFERENCE
PROJECT                         ****     TABLE IN CORPDATA
                                         99
PROJECT                         ****     TABLE
                                         99
PROJNAME                        8        VARCHAR(24) IN RPT1
PROJNAME                        ****     COLUMN
                                         99 99
PROJNAME                        99       VARCHAR(24) COLUMN (NOT NULL) IN CORPDATA.PROJECT
PROJNO                          8        CHARACTER(6) IN RPT1
                                         80
PROJNO                          ****     COLUMN
                                         62 62
PROJNO                          62       CHARACTER(6) COLUMN (NOT NULL) IN CORPDATA.EMPPROJACT
PROJNO                          ****     COLUMN IN EMPPROJACT
                                         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
                              * * * * *  E N D  O F  L I S T I N G  * * * * *
```

図 7. SQL ステートメントを使用したサンプル ILE RPG for iSeries プログラム (6/6)

例: REXX プログラム内の SQL ステートメント

注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

```
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
    1   /************************************************************************/
    2   /* A sample program which updates the salaries for those employees   */
    3   /* whose current commission total is greater than or equal to the    */
    4   /* value of COMMISSION. The salaries of those who qualify are        */
    5   /* increased by the value of PERCENTAGE, retroactive to RAISE_DATE.  */
    6   /* A report is generated and dumped to the display which shows the   */
    7   /* projects which these employees have contributed to, ordered by    */
    8   /* project number and employee ID. A second report shows each        */
    9   /* project having an end date occurring after RAISE DATE (i.e. is    */
   10   /* potentially affected by the retroactive raises) with its total    */
   11   /* salary expenses and a count of employees who contributed to the   */
   12   /* project.                                                          */
   13   /************************************************************************/
   14
   15
   16   /* Initialize RC variable */
   17   RC = 0
   18
   19   /* Initialize HV for program usage */
   20   COMMISSION = 2000.00;
   21   PERCENTAGE = 1.04;
   22   RAISE_DATE = '1982-06-01';
   23   WORK_DAYS  = 253;
   24
   25   /* Create the output file to dump the 2 reports. Perform an OVRDBF   */
   26   /* to allow us to use the SAY REXX command to write to the output    */
   27   /* file.                                                             */
   28   ADDRESS '*COMMAND',
   29          'DLTF FILE(CORPDATA/REPORTFILE)'
   30   ADDRESS '*COMMAND',
   31          'CRTPF FILE(CORPDATA/REPORTFILE) RCDLEN(80)'
   32   ADDRESS '*COMMAND',
   33          'OVRDBF FILE(STDOUT) TOFILE(CORPDATA/REPORTFILE) MBR(REPORTFILE)'
   34
   35   /* Update the selected employee's salaries by the new percentage. */
   36   /* If an error occurs during the update, ROLLBACK the changes.    */
   37    3 SIGNAL ON ERROR
   38   ERRLOC = 'UPDATE_ERROR'
   39   UPDATE_STMT = 'UPDATE CORPDATA/EMPLOYEE ',
   40                 'SET SALARY = SALARY * ?  ',
   41                 'WHERE COMM >= ?          '
   42   EXECSQL,
   43          'PREPARE S1 FROM :UPDATE_STMT'
   44    4 EXECSQL,
   45          'EXECUTE S1 USING :PERCENTAGE,',
   46          '                 :COMMISSION '
   47   /* Commit changes */
   48    5 EXECSQL,
   49          'COMMIT'
   50   ERRLOC = 'REPORT_ERROR'
   51
   52   /* Report the updated statistics for each project supported by one */
   53   /* of the selected employees.                                      */
   54
   55   /* Write out the header for Report 1 */
   56   SAY '   '
   57   SAY '   '
   58   SAY '   '
   59   SAY '         REPORT OF PROJECTS AFFECTED BY EMPLOYEE RAISES'
   60   SAY '   '
   61   SAY 'PROJECT  EMPID     EMPLOYEE NAME                     SALARY'
   62   SAY '-------  -----     -------------                     ------'
   63   SAY '   '
   64
   65   SELECT_STMT =  'SELECT DISTINCT PROJNO, EMPPROJACT.EMPNO, ',
   66                  '       LASTNAME||'', ''||FIRSTNME, SALARY ',
   67                  'FROM CORPDATA/EMPPROJACT, CORPDATA/EMPLOYEE  ',
   68                  'WHERE EMPPROJACT.EMPNO = EMPLOYEE.EMPNO AND  ',
   69                  '      COMM >= ?                              ',
   70                  'ORDER BY PROJNO, EMPNO                       '
   71   EXECSQL,
   72          'PREPARE S2 FROM :SELECT_STMT'
   73    6 EXECSQL,
   74          'DECLARE C1 CURSOR FOR S2'
```

図 8. SQL ステートメントを使用したサンプル REXX プロシージャー (1/3)

```
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
   75    7 EXECSQL,
   76           'OPEN C1 USING :COMMISSION'
   77
   78    /* Handle the FETCH errors and warnings inline */
   79    SIGNAL OFF ERROR
   80
   81    /* Fetch all of the rows */
   82    DO UNTIL (SQLCODE <> 0)
   83      9 EXECSQL,
   84             'FETCH C1 INTO :RPT1.PROJNO, :RPT1.EMPNO,',
   85             '              :RPT1.NAME, :RPT1.SALARY  '
   86
   87      /* Process any errors that may have occurred. Continue so that   */
   88      /* we close the cursor for any warnings.                         */
   89      IF SQLCODE < 0 THEN
   90        SIGNAL ERROR
   91
   92      /* Stop the loop when we hit the EOF. Don't try to print out the */
   93      /* fetched values.                                               */
   94     8 IF SQLCODE = 100 THEN
   95        LEAVE
   96
   97      /* Print out the fetched row */
   98      SAY RPT1.PROJNO '   ' RPT1.EMPNO '   ' RPT1.NAME '     ' RPT1.SALARY
   99    END;
  100
  101    10 EXECSQL,
  102           'CLOSE C1'
  103
..+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
  104    /* For all projects ending at a date later than 'raise_date'       */
  105    /* (i.e. those projects potentially affected by the salary raises) */
  106    /* generate a report containing the project number, project name   */
  107    /* the count of employees participating in the project and the     */
  108    /* total salary cost of the project.                               */
  109
  110      /* Write out the header for Report 2 */
  111    SAY '   '
  112    SAY '   '
  113    SAY '   '
  114    SAY '         ACCUMULATED STATISTICS BY PROJECT'
  115    SAY '   '
  116    SAY 'PROJECT  PROJECT NAME                          NUMBER OF      TOTAL'
  117    SAY 'NUMBER                                         EMPLOYEES      COST'
  118    SAY '-------  ------------                          ---------      -----'
  119    SAY '   '
  120
  121
  122    /* Go to the common error handler */
  123    SIGNAL ON ERROR
  124
  125    SELECT_STMT = 'SELECT EMPPROJACT.PROJNO, PROJNAME, COUNT(*),                ',
  126                  '   SUM( (DAYS(EMENDATE) - DAYS(EMSTDATE)) * EMPTIME *     ',
  127                  '         DECIMAL(( SALARY / ? ),8,2) )                    ',
  128                  'FROM CORPDATA/EMPPROJACT, CORPDATA/PROJECT, CORPDATA/EMPLOYEE',
  129                  'WHERE EMPPROJACT.PROJNO = PROJECT.PROJNO  AND               ',
  130                  '      EMPPROJACT.EMPNO = EMPLOYEE.EMPNO  AND                ',
  131                  '      PRENDATE > ?                                  ',
  132                  'GROUP BY EMPPROJACT.PROJNO, PROJNAME                       ',
  133                  'ORDER BY 1                                          '
  134    EXECSQL,
  135           'PREPARE S3 FROM :SELECT_STMT'
  136    11 EXECSQL,
  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)
```

図 8. SQL ステートメントを使用したサンプル REXX プロシージャー (2/3)

```
Record  *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
  146       12 EXECSQL,
  147            'FETCH C2 INTO :RPT2.PROJNO, :RPT2.PROJNAME,    ',
  148            '              :RPT2.EMPCOUNT, :RPT2.TOTAL_COST '
  149
  150      /* Process any errors that may have occurred. Continue so that   */
  151      /* we close the cursor for any warnings.                         */
  152      IF SQLCODE < 0 THEN
  153        SIGNAL ERROR
  154
  155      /* Stop the loop when we hit the EOF. Don't try to print out the */
  156      /* fetched values.                                               */
  157      IF SQLCODE = 100 THEN
  158        LEAVE
  159
  160      /* Print out the fetched row */
  161      SAY RPT2.PROJNO '   ' RPT2.PROJNAME '  ' ,
  162          RPT2.EMPCOUNT '        ' RPT2.TOTAL_COST
  163    END;
  164
  165    EXECSQL,
  166           'CLOSE C2'
  167
  168    /* Delete the OVRDBF so that we will continue writing to the output  */
  169    /* display.                                                          */
  170    ADDRESS '*COMMAND',
  171           'DLTOVR FILE(STDOUT)'
  172
  173    /* Leave procedure with a successful or warning RC */
  174    EXIT RC
  175
  176
  177    /* Error occurred while updating the table or generating the         */
  178    /* reports. If the error occurred on the UPDATE, rollback all of     */
  179    /* the changes. If it occurred on the report generation, display the */
  180    /* REXX RC variable and the SQLCODE and exit the procedure.          */
  181    ERROR:
  182
  183       13 SIGNAL OFF ERROR
  184
  185      /* Determine the error location */
  186      SELECT
  187        /* When the error occurred on the UPDATE statement */
  188        WHEN ERRLOC = 'UPDATE_ERROR' THEN
  190          DO
  191            SAY '*** ERROR Occurred while updating table.',
  192                'SQLCODE = ' SQLCODE
  193             14 EXECSQL,
  194                   'ROLLBACK'
  195          END
  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                          * * * * *  E N D  O F  S O U R C E  * * * * *
```

SQL を使用したサンプル・プログラムにより作成される報告書

前記の各サンプル・プログラムにより作成される報告書は次のとおりです。

```
                    REPORT OF PROJECTS AFFECTED BY RAISES

PROJECT  EMPID     EMPLOYEE NAME                    SALARY

AD3100   000010    HAAS, CHRISTINE                54860.00
AD3110   000070    PULASKI, EVA                   37616.80
AD3111   000240    MARINO, SALVATORE              29910.40
AD3113   000270    PEREZ, MARIA                   28475.20
IF1000   000030    KWAN, SALLY                    39780.00
IF1000   000140    NICHOLLS, HEATHER              29556.80
IF2000   000030    KWAN, SALLY                    39780.00
IF2000   000140    NICHOLLS, HEATHER              29556.80
MA2100   000010    HAAS, CHRISTINE                54860.00
MA2100   000110    LUCCHESSI, VICENZO             48360.00
MA2110   000010    HAAS, CHRISTINE                54860.00
MA2111   000200    BROWN, DAVID                   28849.60
MA2111   000220    LUTZ, JENNIFER                 31033.60
MA2112   000150    ADAMSON, BRUCE                 26291.20
OP1000   000050    GEYER, JOHN                    41782.00
OP1010   000090    HENDERSON, EILEEN              30940.00
OP1010   000280    SCHNEIDER, ETHEL               27300.00
OP2010   000050    GEYER, JOHN                    41782.00
OP2010   000100    SPENSER, THEODORE              27196.00
OP2012   000330    LEE, WING                      26384.80
PL2100   000020    THOMPSON, MICHAEL              42900.00


                    ACCUMULATED STATISTICS BY PROJECT

PROJECT                                          NUMBER OF       TOTAL
NUMBER   PROJECT NAME                            EMPLOYEES       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
```

付録 B. ホスト言語プリコンパイラーの DB2 UDB for iSeries CL コマンド記述

この付録には、本書および「SQL 解説書」で参照または使用されている構文図が記載されています。

詳細については、『SQL プリコンパイラー・コマンド』を参照してください。

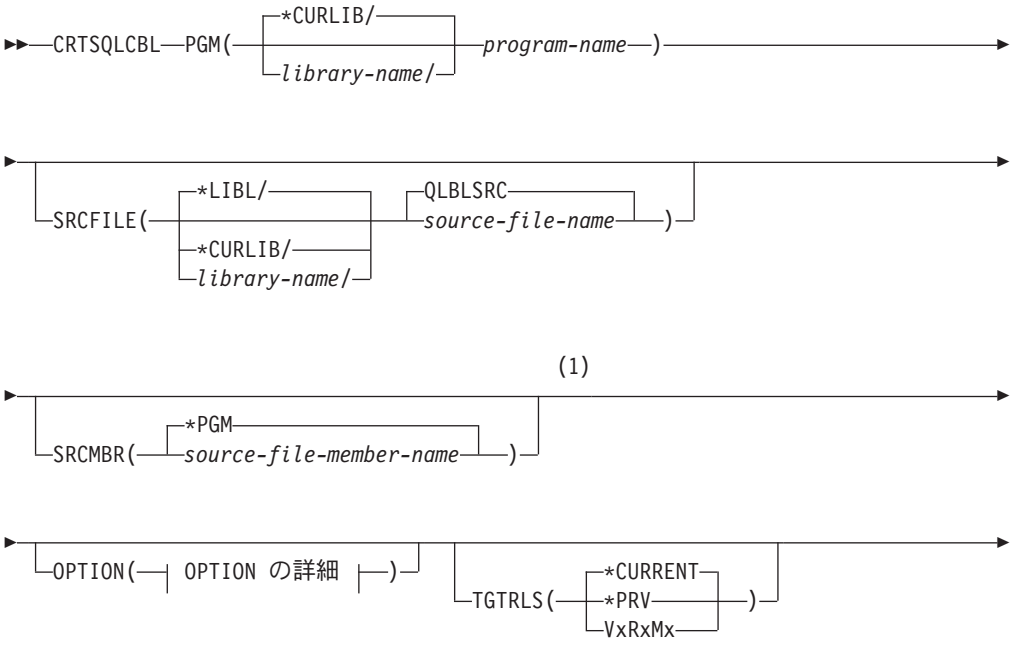
SQL プリコンパイラー・コマンド

DB2 UDB for iSeries は、以下のプログラム言語でコード化されたプログラムのプリコンパイル・コマンドを提供します。

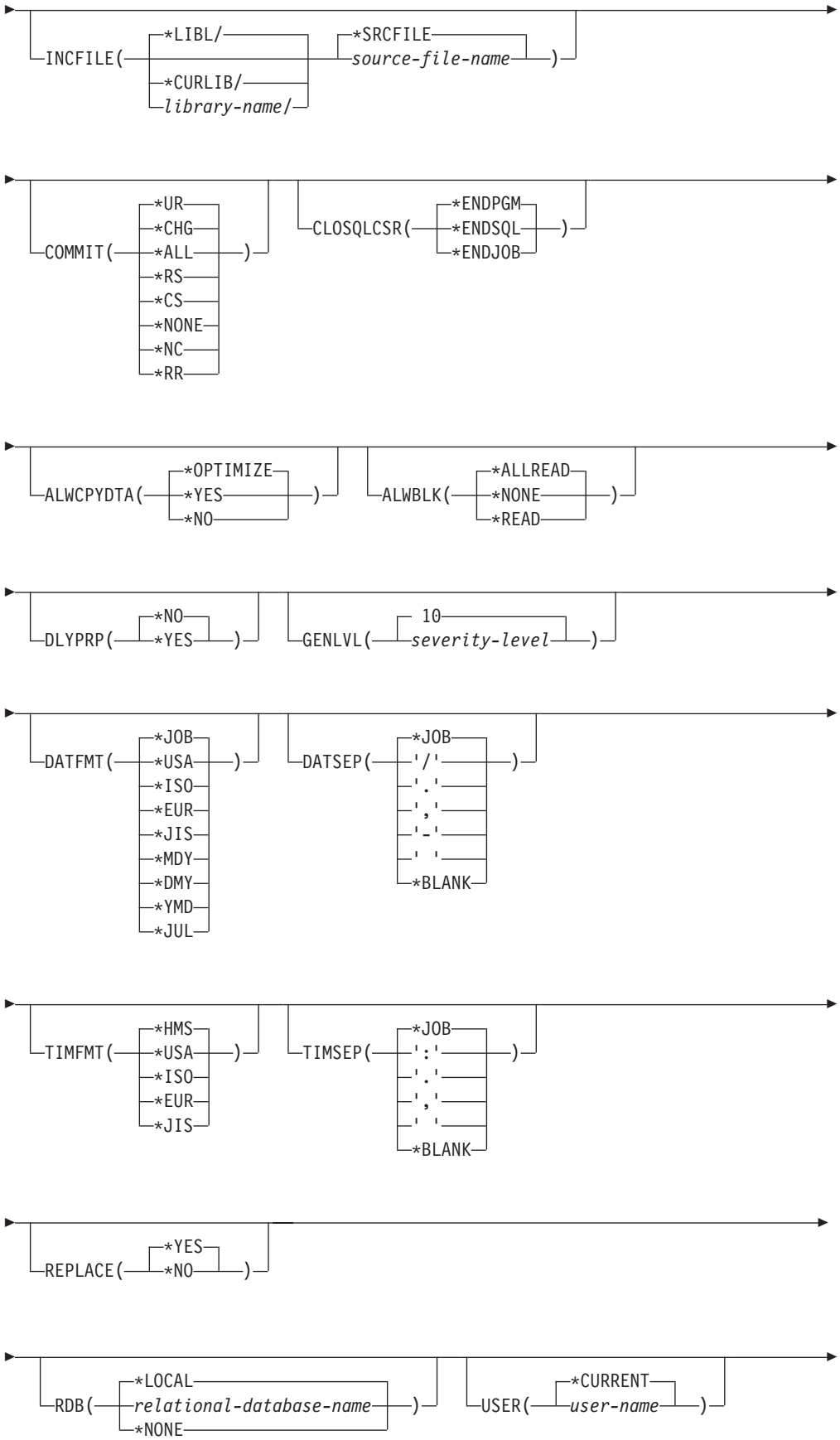
- COBOL
- ILE COBOL
- ILE C
- C++
- PL/I
- RPG
- ILE RPG

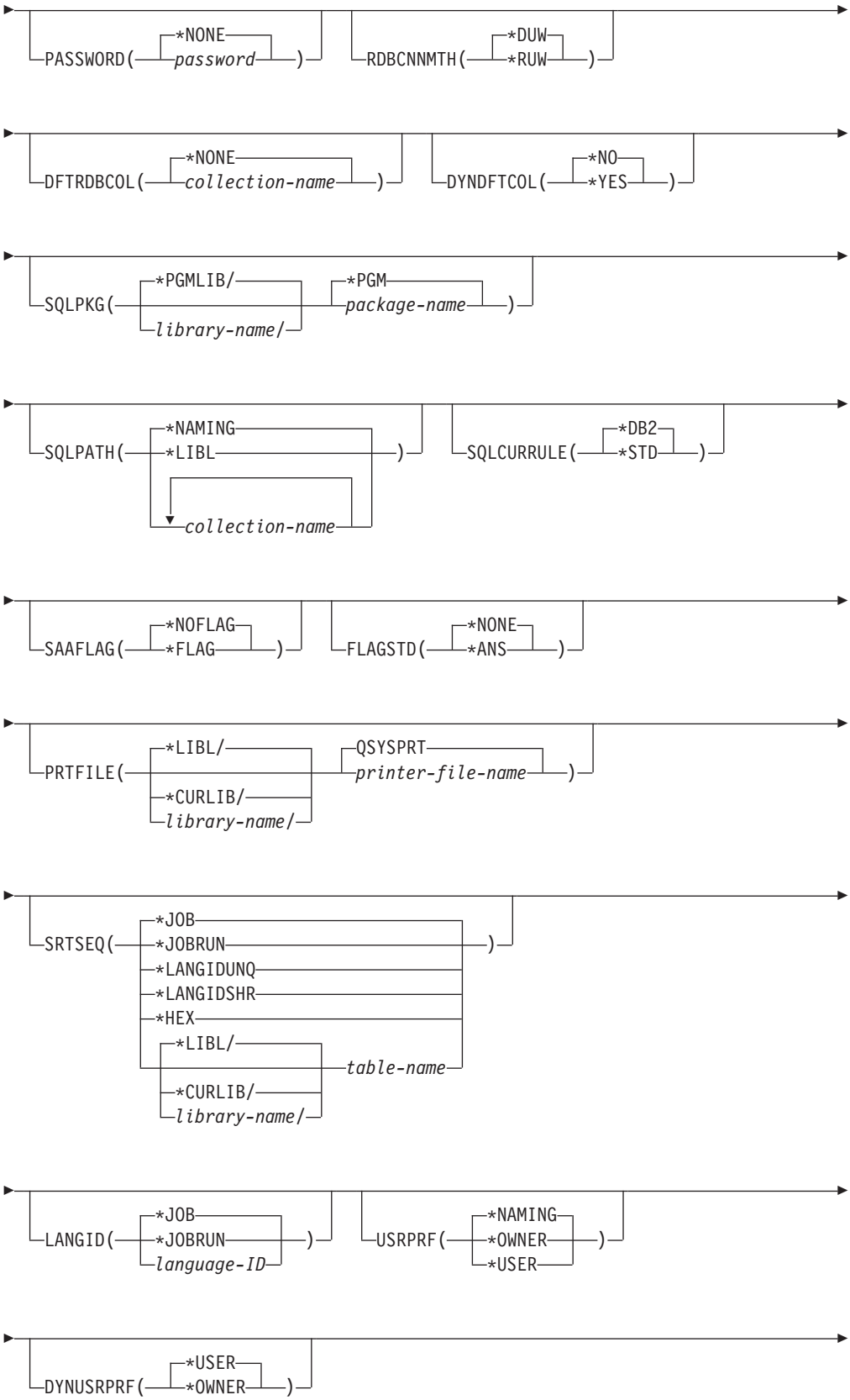
CRTSQLCBL (SQL COBOL プログラム作成) コマンド

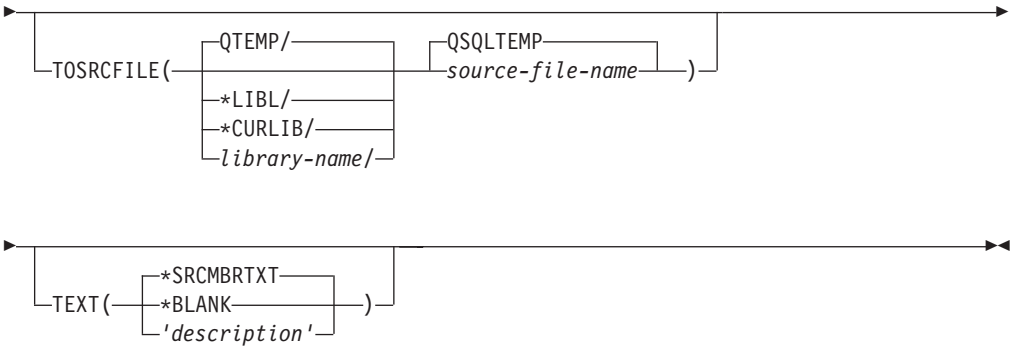
Job: B,I Pgm: B,I REXX: B,I Exec



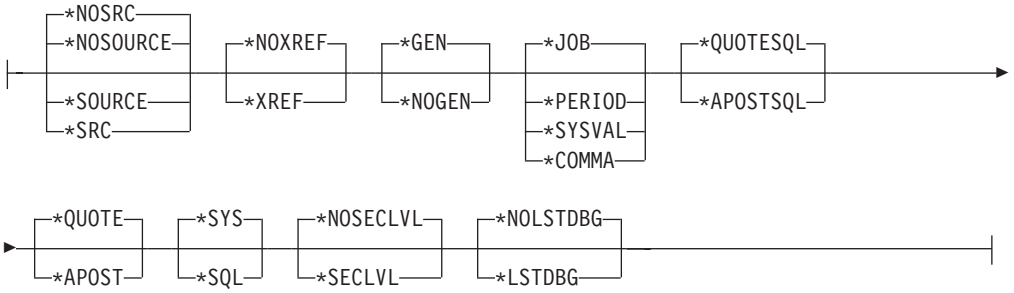
CRTSQLCBL


INCFILE(
*LIBL/
*CURLIB/
library-name/
*SRCFILE
source-file-name
)
COMMIT(
*UR
*CHG
*ALL
*RS
*CS
*NONE
*NC
*RR
)
CLOSQLCSR(
*ENDPGM
*ENDSQL
*ENDJOB
)
ALWCPYDTA(
*OPTIMIZE
*YES
*NO
)
ALWBLK(
*ALLREAD
*NONE
*READ
)
DLYPRP(
*NO
*YES
)
GENLVL(
10
severity-level
)
DATFMT(
*JOB
*USA
*ISO
*EUR
*JIS
*MDY
*DMY
*YMD
*JUL
)
DATSEP(
*JOB
'/'
'.'
','
'-'
' '
*BLANK
)
TIMFMT(
*HMS
*USA
*ISO
*EUR
*JIS
)
TIMSEP(
*JOB
':'
'.'
','
' '
*BLANK
)
REPLACE(
*YES
*NO
)
RDB(
*LOCAL
relational-database-name
*NONE
)
USER(
*CURRENT
user-name
)


PASSWORD(*NONE password)
RDBCNNMTH(*DUW *RUW)
DFTRDBCOL(*NONE collection-name)
DYNDFTCOL(*NO *YES)
SQLPKG(*PGMLIB/ library-name/ *PGM package-name)
SQLPATH(*NAMING *LIBL collection-name)
SQLCURRULE(*DB2 *STD)
SAAFLAG(*NOFLAG *FLAG)
FLAGSTD(*NONE *ANS)
PRTFILE(*LIBL/ *CURLIB/ library-name/ QSYSPRT printer-file-name)
SRTSEQ(*JOB *JOBRUN *LANGIDUNQ *LANGIDSHR *HEX *LIBL/ *CURLIB/ library-name/ table-name)
LANGID(*JOB *JOBRUN language-ID)
USRPRF(*NAMING *OWNER *USER)
DYNUSRPRF(*USER *OWNER)



OPTION の詳細:



注:

1 これより前にあるパラメーターはすべて、定位置形式で指定されます。

目的:

SQL COBOL プログラム作成 (CRTSQLCBL) コマンドは構造化照会言語 (SQL) プリコンパイラーを呼び出すもので、このプリコンパイラーが SQL ステートメントを含む COBOL ソース・プログラムをプリコンパイルし、一時ソース・メンバーを作成してから、任意で COBOL コンパイラーを呼び出してプログラムのコンパイルを行います。

パラメーター:

PGM

コンパイルしたプログラムの修飾名を指定します。

コンパイルした COBOL プログラムの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***CURLIB** ジョブ用の現行ライブラリー内でコンパイルした COBOL プログラムが作成されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: コンパイルした COBOL プログラムが作成されるライブラリーの名前を指定します。

program-name: コンパイルした COBOL プログラムの名前を指定します。

SRCFILE

SQL ステートメントとともに COBOL ソース・プログラムが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QLBLSRC: COBOL ソース・ファイル名の指定がない場合は、 IBM 提供のソース・ファイル QLBLSRC に COBOL ソース・プログラムが入ります。

source-file-name: COBOL ソース・プログラムが入っているソース・ファイルの名前を指定します。このソース・ファイルのレコード長は 92 バイトでなければなりません。このソース・ファイルはデータベース・ファイル、装置ファイル、インライン・データ・ファイルのいずれでも構いません。

SRCMBR

COBOL ソース・プログラムが入っているソース・ファイル・メンバーの名前を指定します。このパラメーターは、SRCFILE パラメーターのソース・ファイル名がデータベース・ファイルである場合にのみ指定します。このパラメーターが指定されない場合は、PGM パラメーターに指定された PGM 名が使用されます。

***PGM:** PGM パラメーターに指定した名前と同じ名前をもつソース・ファイルのメンバーに COBOL ソースがあることを指定します。

source-file-member-name: COBOL ソースが入っているメンバーの名前を指定します。

OPTION

COBOL ソース・プログラムをプリコンパイルするときに次のオプションのうち 1 つ以上が使用されるかどうかを指定します。オプションが 2 度以上使用される場合、または 2 つのオプションが対立する場合は、最後に指定されたオプションが使用されます。

要素 1: ソース・リスト・オプション

***NOSOURCE** または ***NOSRC:** プリコンパイルまたはパッケージ作成時にエラーが検出されなければ、プリコンパイラーによるソース印刷出力は行われません。

***SOURCE または *SRC:** プリコンパイラーは、COBOL ソース入力から成るソース印刷出力を作成します。

要素 2: 相互参照オプション

***NOXREF:** プリコンパイラーは名前の相互参照を行いません。

***XREF:** プリコンパイラーは、プログラムの中の項目と、プログラムの中でそれらの項目を参照しているステートメントの番号との間の相互参照を行います。

要素 3: プログラム作成オプション

***GEN:** コンパイラーはコンパイル後に実行できるプログラムを作成します。リレーショナル・データベース名が RDB パラメーターで指定されている場合、SQL パッケージ・オブジェクトが作成されます。

***NOGEN:** プリコンパイラーは COBOL コンパイラーを呼び出さないので、プログラムと SQL パッケージは作成されません。

要素 4: 小数点オプション

***JOB:** SQL で数値定数の小数点として使用される値は、プリコンパイル時にジョブ用に指定されている小数点を表しています。

***SYSVAL:** SQL ステートメントの中の数値定数に小数点として使用する値は QDECFMT システム値になります。

注: QDECFMT が小数点として使用する値をコンマにすることを指定している場合は、リストの中の (SELECT 文節、VALUES 文節などのように) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

***PERIOD:** SQL ステートメントの中の数値定数に小数点として使用する値はピリオドになります。

***COMMA:** SQL ステートメントの中の数値定数に小数点として使用する値はコンマになります。

注: リストの中の (SELECT 文節、VALUES 文節のような) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

要素 5: 文字列区切り文字オプション

***QUOTESQL:** 2 重引用符 (") が SQL ステートメント内の文字列区切り文字になります。

***APOSTSQL:** アポストロフィ (') が SQL ステートメント内の文字列区切り文字になります。

要素 6: リテラル・オプション

***QUOTE:** COBOL ステートメントの中の非数字リテラルおよびブール・リテラルに 2 重引用符 (") を使用します。

***APOST:** COBOL ステートメントの中の非数字リテラルおよびブール・リテラルにアポストロフィ (') を使用します。

要素 7: 命名規則オプション

***SYS:** システム命名規則 (library-name/file-name) を使用します。

***SQL:** SQL の命名規則 (schema-name.table-name) を使用します。iSeries 以外のリモート・データベース上でプログラムを作成するときは、命名規則として *SQL を指定しなければなりません。

要素 8: 2 次レベル・メッセージ・テキスト・オプション

***NOSECLVL:** 2 次レベルのテキスト記述はリストに追加されません。

***SECLVL:** リストのいずれかのメッセージについても、置換データを含む 2 次レベルのテキストが追加されます。

要素 9: デバッグ・リスト・ビュー

***NOLSTDBG:** エラーとデバッグの情報が生成されません。

***LSTDBG:** SQL プリコンパイラーはリスト・ビュー、およびこのビューに必要なエラーとデバッグの情報を生成します。 *LSTDBG を使用できるのは、CODE/400 プロダクトを使用してユーザーのプログラムをコンパイルしている場合に限られます。

TGTRLS

作成中のオブジェクトを使用するためのオペレーティング・システムのリリース・レベルを指定します。

*CURRENT 値および *PRV 値の例では、VxRxMx 形式でリリースを指定する release-level 値が示されています。ここで Vx はバージョン、Rx はリリース、Mx はモディフィケーション・レベルです。たとえば、V2R3M0 はバージョン 2、リリース 3、モディフィケーション・レベル 0 です。

***CURRENT:** オブジェクトは、ユーザーのシステムで現在稼働しているオペレーティング・システムのリリースで使用されます。たとえば、V2R3M5 がシステムで稼働している場合、*CURRENT はユーザーが V2R3M5 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

注: V2R3M5 がシステム上で稼働しており、V2R3M0 が導入されたシステムでオブジェクトが使用される場合、TGTRLS(*CURRENT) ではなく TGTRLS(V2R3M0) を指定してください。

***PRV:** オブジェクトはオペレーティング・システムのモディフィケーション・レベル 0 の前のリリースで使用されます。たとえば、V2R3M5 がユーザーのシステムで稼働している場合、*PRV はユーザーが V2R2M0 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

release-level: VxRxMx 形式でリリースを指定してください。指定されたリリースのシステム上、あるいは以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することができます。

有効な値は、現行バージョン、リリース、モディフィケーション・レベルによります。また、有効な値は各新規リリースで変わります。コマンドがサポートする

初期リリース・レベルよりもさらに前のリリース・レベルを指定した場合には、エラー・メッセージはこれがサポートする最も初期のリリース・レベルを示して送信されます。

INCFILE

SQL の INCLUDE ステートメントを用いてプログラムに組み込むメンバーが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

***SRCFILE:** SRCFILE パラメーターに修飾名を指定したソース・ファイルには、SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入ります。

source-file-name: SQL INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入っているソース・ファイルの名前を指定します。ここに指定するソース・ファイルのレコード長は、SRCFILE パラメーターに指定したソース・ファイルのレコード長より短くてはなりません。

COMMIT

コンパイル済みプログラム内の SQL ステートメントがコミットメント制御のもとで実行されるかどうかを指定します。ホスト言語ソースの中で参照されているファイルは、このオプションによって影響を受けません。SQL ステートメントの中で参照される SQL テーブル、SQL ビュー、および SQL パッケージだけが影響されます。

注: COBOL ソース・プログラムで参照されているファイルは、このオプションの影響を受けません。

***CHG または *UR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができます。

***ALL または *RS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。

***CS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクショ

ン) の終わりまでロックされることを指定します。選択されたが、更新されていない行は、次の行が選択されるまでロックされます。他のジョブにおけるコミットされていない変更は見ることができません。

***NONE または *NC:** コミットメント制御が使用されないことを指定します。他のジョブにおけるコミットされていない変更は見ることができます。プログラムに SQL の DROP SCHEMA ステートメントが組み込まれている場合は、*NONE または *NC を使用しなければなりません。RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries 以外のシステム上にある場合は、*NONE または *NC を指定することはできません。

***RR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。
SELECT、UPDATE、DELETE、および INSERT ステートメントで参照されているすべてのテーブルは、作業単位 (トランザクション) の終わりまで排他的にロックされます。

CLOSQLCSR

SQL カーソルが暗黙にクローズされるとき、SQL 準備済みステートメントが暗黙に破棄され、LOCK TABLE のロックが解除されることを指定します。
CLOSE、COMMIT、または ROLLBACK (HOLD を指定しない) SQL ステートメントを発行すると、SQL カーソルが明示的にクローズされます。

***ENDPGM:** プログラムが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが破棄されます。呼び出しスタック上の最初の SQL プログラムが終了すると、LOCK TABLE のロックが解除されます。

***ENDSQL:** SQL カーソルは呼び出しと次の呼び出しの間はオープンしたままで、別の SQL OPEN を実行せずに取り出すことができます。呼び出しスタック上でより高い位置にあるプログラムの 1 つは、少なくとも 1 つの SQL ステートメントを実行しているはずです。呼び出しスタック上の最初の SQL プログラムが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが廃棄され、LOCK TABLE のロックが解除されます。呼び出された最初の SQL プログラム (呼び出しスタック上の最初の SQL プログラム) であるプログラムに *ENDSQL が指定される場合、プログラムは *ENDPGM が指定されたかのように扱われます。

***ENDJOB:** SQL カーソルは呼び出しと次の呼び出しの間はオープンしたままで、別の SQL OPEN を実行せずに取り出すことができます。呼び出しスタック上でより高い位置にあるプログラムは、SQL ステートメントを実行させる必要がありません。呼び出しスタック上の最初の SQL プログラムが終了するときに、SQL カーソルはオープンのままになり、SQL 準備済みステートメントが保持され、LOCK TABLE のロックが保留されます。ジョブが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが廃棄され、LOCK TABLE のロックが解除されます。

ALWCPYDTA

SELECT ステートメントでデータのコピーが使用できるかどうかを指定します。

***OPTIMIZE:** データベースから直接検索されたデータを使用するか、データのコピーを使用するかどうかは、システムが決定します。決定は、どの方式が最良のパフォーマンスを提供するかに基づいて行われます。 COMMIT が *CHG または *CS で ALWBLK が *ALLREAD でない場合、あるいは COMMIT が *ALL または *RR の場合、データのコピーが使用されるのは、照会を実行する必要があるときだけです。

***YES:** 必要な場合にデータのコピーが使用されます。

***NO:** データのコピーを使用することはできません。照会を実行するのにデータの一時コピーが必要な場合には、エラー・メッセージが返されます。

ALWBLK

データベース・マネージャーがレコードのブロック化を使用できるかどうか、および読み取り専用カーソルについてどの程度までブロック化が使用できるかを指定します。

***ALLREAD:** COMMIT パラメーターで *NONE または *CHG が指定されている場合は、行は読み取り専用カーソルに対してブロックされます。プログラム内で、明示的に更新することができないすべてのカーソルは、プログラム内に EXECUTE または EXECUTE IMMEDIATE ステートメントがある場合でも、読み取り専用にオープンされます。

*ALLREAD を指定すると、

- *READ の場合に許されるブロック化に加え、コミットメント制御レベル *CHG のもとでレコードのブロック化を行うことができます。
- プログラムの中のほとんどすべての読み取り専用カーソルのパフォーマンスを向上できますが、照会が次のように制限されます。
  - ロールバック (ROLLBACK) コマンド、ホスト言語で書いた ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、*ALLREAD の指定があるとき、読み取り専用カーソルの再位置決めはしません。
  - カーソル内の行を更新するために、位置付けされた UPDATE または DELETE 使用ステートメントの動的実行 (たとえば、EXECUTE IMMEDIATE による) をすることはできません。ただし、カーソルに関する DECLARE ステートメントに FOR UPDATE 文節が含まれている場合を除きます。

***NONE:** カーソルの対象となるデータを検索するとき、行はブロック化されません。

*NONE を指定すると、

- 検索されるデータが最新であることが保証されます。
- 照会の対象となるデータの最初の行を検索するときの所要時間が短縮します。

- 照会の最初の数行だけが検索されてから照会がクローズされるときは、プログラムによって使用されないデータ行のブロックの検索をデータベース・マネージャーに中止させます。
- 大量の行を検索する照会全体のパフォーマンスが低下する可能性があります。

***READ:** 次のとき、カーソルの対象となるデータを読み取り専用で検索するときレコードがブロック化されます。

- COMMIT パラメーターに *NONE の指定があるとき。これは、コミットメント制御が使用されないことを示します。
- カーソルが FOR READ ONLY 文節を使用して宣言されているか、あるいは位置指定の UPDATE または DELETE ステートメントをカーソルに対して実行できる動的ステートメントがないとき。

*READ を指定すると、上記条件を満足する照会の全体のパフォーマンスが向上し、大量のレコードを検索することができます。

DLYPRP

PREPARE ステートメントについての動的ステートメント妥当性検査を、OPEN、EXECUTE、または DESCRIBE ステートメントが実行されるまで遅延させるかどうかを指定します。妥当性検査を遅延させると、余分な妥当性検査が除かれるため、パフォーマンスが向上します。

***NO:** 動的ステートメント妥当性検査を遅延させません。動的ステートメントが準備されるとき、アクセス・プランが妥当性検査されます。動的ステートメントが OPEN または EXECUTE ステートメントで使用される場合、アクセス・プランが再度妥当性検査されます。動的ステートメントによって参照されるオブジェクトの権限または存在は変化する場合があるので、OPEN または EXECUTE ステートメントを出した後、SQLCODE または SQLSTATE を検査して、動的ステートメントがまだ有効であるか確かめる必要があります。

***YES:** 動的ステートメント妥当性検査を、動的ステートメントが OPEN、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅延させます。動的ステートメントが使用されたときは、その妥当性検査が行われて、アクセス・プランが作られます。このパラメーターで *YES を指定する場合は、OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後 SQLCODE と SQLSTATE を調べて、動的ステートメントが有効であるかどうかを確かめておく必要があります。

注: *YES を指定したときは、PREPARE ステートメントで INTO 文節が使用されている場合や、OPEN が動的ステートメントに対して出される前に DESCRIBE ステートメントがその動的ステートメントを使用した場合は、パフォーマンスは向上しません。

GENLVL

作成操作が失敗する時の重大度レベルを指定します。この値と等しいかまたはより大きい重大度レベルのエラーが発生すると、操作が終了します。

10: 省略時の重大度レベルは 10 です。

severity-level: 0 から 40 までの範囲で重大度レベルの値を指定します。

DATFMT

日付結果列にアクセスするときに使用される形式を指定します。すべての出力日付フィールドは、指定された形式で返されます。入力日付文字列については、日付が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、*USA、*ISO、*EUR、または *JIS を指定しなければなりません。

***JOB:** ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***USA:** 米国の日付形式 (mm/dd/yyyy) が使用されます。

***ISO:** 国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR:** ヨーロッパの日付形式 (dd.mm.yyyy) が使用されます。

***JIS:** 日本工業規格の日付形式 (yyyy-mm-dd) が使用されます。

***MDY:** 日付形式 (mm/dd/yy) が使用されます。

***DMY:** 日付形式 (dd/mm/yy) が使用されます。

***YMD:** 日付形式 (yy/mm/dd) が使用されます。

***JUL:** 年間通算日の日付形式 (yy/ddd) が使用されます。

DATSEP

日付結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブに指定された日付区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

'/': スラッシュ (/) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

'-': ダッシュ (-) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

TIMFMT

時刻結果列にアクセスするときに使用される形式を指定します。入力時刻文字列については、時刻が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、時刻形式は、時刻区切り記号がコロンまたはピリオドである *USA、*ISO、*EUR、*JIS、または *HMS でなければなりません。

***HMS:** (hh:mm:ss) 形式が使用されます。

***USA:** 米国の時刻形式 (hh:mm *xx*) が使用されます。ただし、*xx* は AM か PM です。

***ISO:** 国際標準化機構 (ISO) の時刻形式 (hh.mm.ss) が使用されます。

***EUR:** ヨーロッパの時刻形式 (hh.mm.ss) が使用されます。

***JIS:** 日本工業規格の時刻形式 (hh:mm:ss) が使用されます。

TIMSEP

時刻結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブのために指定された時刻区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

':': コロン (:) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

REPLACE

同じライブラリーに同じ名前のプログラムまたは SQL パッケージが存在するときに新しいプログラムまたは SQL パッケージが作成されるかどうかを指定します。このパラメーターの値は、CRTCBLPGM コマンドに渡されます。このパラメーターの詳細については、Information Center の「CL 解説書」の『REPLACE パラメーター』を参照してください。

***YES:** 新しいプログラムまたは SQL パッケージが作成され、指定したライブラリーの中の、同じ名前とタイプを持つ既存のプログラムまたは SQL パッケージは QRPLOBJ に移されます。

***NO:** 指定されたライブラリーに同じ名前およびタイプのオブジェクトがすでに存在している場合は、新規のプログラムまたは SQL パッケージは作成されません。

RDB

SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。

***LOCAL:** プログラムは分散 SQL プログラムとして作成されます。 SQL ステートメントはローカル・データベースにアクセスします。プリコンパイル・プロセスの一部として SQL パッケージ・オブジェクトは作成されません。 SQL パッケージの作成 (CRTSQLPKG) コマンドを使用することができます。

relational-database-name: 新しい SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。ローカル・リレーショナル・データベースの名前を指定すると、作成されるプログラムは分散 SQL プログラムになります。 SQL ステートメントはローカル・データベースにアクセスします。

***NONE:** SQL パッケージ・オブジェクトは作成されません。プログラム・オブジェクトは分散プログラムではなく、 SQL パッケージの作成 (CRTSQLPKG) コマンドは使用できません。

USER

会話の開始時にリモート・システムに送られるユーザー名を指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***CURRENT:** 現在のジョブが実行されているユーザー・プロファイルが使用されます。

user-name: アプリケーション・サーバー・ジョブのユーザー名を指定します。

PASSWORD

リモート・システムで使用されるパスワードを指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***NONE:** パスワードは送られません。この値を指定する場合は、USER(*CURRENT) も指定しなければなりません。

password: USER パラメーターに指定したユーザー名のパスワードを指定します。

RDBCNNMTH

CONNECT ステートメントに使用する意味を指定します。詳細については、「*SQL 解説書*」の CONNECT (タイプ 1) および CONNECT (タイプ 2) を参照してください。

***DUW:** 分散作業単位をサポートするために CONNECT (タイプ 2) の意味が使用されます。追加のリレーショナル・データベースへの連続する CONNECT ステートメントによって、直前の接続が切断されることはありません。

***RUW:** リモート作業単位をサポートするのに CONNECT (タイプ 1) の意味が使用されます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

DFTRDBCOL

テーブル、ビュー、索引および SQL パッケージの非修飾名に使用されるスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用されます。

***NONE:** OPTION パラメーターで定義した命名規則が使用されます。

schema-name: スキーマ識別名を指定します。この値は、OPTION パラメーターで指定した命名規則の代わりに使用されます。

DYNDFTCOL

DFTRDBCOL パラメーター用に指定した省略時スキーマ名が動的ステートメントでも使用されるかどうかを指定します。

***NO:** 動的 SQL ステートメント用のテーブル、ビュー、索引、および SQL パッケージの非修飾名用の DFTRDBCOL パラメーターに指定した値は使用されません。OPTION パラメーターで指定した命名規則が使用されます。

***YES:** DFTRDBCOL パラメーターに指定したスキーマ名は、動的 SQL ステートメント内のテーブル、ビュー、索引、および SQL パッケージの非修飾名用に使用されます。

SQLPKG

このコマンドの RDB パラメーターで指定されたリレーショナル・データベースで作成される SQL パッケージの修飾名を指定します。

ライブラリー値は次のとおりです。

***PGMLIB:** パッケージは、プログラムが置かれているライブラリーと同じ名前をもつライブラリーの中に作成されます。

library-name: パッケージが作成されるライブラリーの名前を指定します。

***PGM:** パッケージ名はプログラム名と同じ名前になります。

package-name: RDB パラメーターで指定したリモート・データベース上で作成されるパッケージの名前を指定します。

SQLPATH

静的 SQL ステートメント内のプロシージャー、関数、およびユーザー定義タイプを検出するために使用するパスを指定します。

***NAMING:** 使用するパスは、OPTION パラメーターで指定した命名規則に従います。

*SYS 命名の場合、使用するパスは、*LIBL です (実行時の現行ライブラリー・リスト)。

*SQL 命名の場合、使用するパスは、″QSYS″、″QSYS2″、″userid″ です。ただし、″userid″ は、USER 特殊レジスターの値です。schema-name が DFTRDBCOL パラメーターに指定された場合、その schema-name がユーザー ID に置き替わります。

***LIBL:** 使用するパスは、実行時のライブラリー・リストです。

schema-name: 1 つ以上のスキーマ名のリストを指定します。最大 268 の個別スキーマを指定できます。

SQLCURRULE

SQL ステートメントに使用する意味を指定します。

***DB2:** SQL ステートメントすべての意味が、DB2 について設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数が文字データとして扱われます。

***STD:** SQL ステートメントすべての意味が、ISO および ANSI の SQL 規格によって設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数がバイナリー・データとして扱われます。

SAAFLAG

IBM SQL フラグ付け機能を指定します。このパラメーターは、SQL ステートメントにフラグ付けし、SQL ステートメントが IBM SQL 構文に準拠しているかを検査します。IBM データベース・プロダクトの IBM SQL 構文に関する詳細は、「*DRDA IBM SQL Reference*」にあります。

***NOFLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行いません。

***FLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行います。

FLAGSTD

米国規格協会 (ANSI) のフラグ機能を指定します。このパラメーターは SQL ステートメントにフラグ付けし、ステートメントが次の標準に準拠するかどうかを検査します。

```
ANSI X3.135-1992 entry
ISO 9075-1992 entry
FIPS 127.2 entry
```

***NONE:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行いません。

***ANS:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行います。

PRTFILE

リストが送られる先の印刷装置ファイルの修飾名を指定します。ファイルは 132 バイト以上のレコード長をもっている必要があります。そうでない場合は情報が失われます。

印刷装置ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

QSYSPRT: ファイル名の指定がない場合、プリコンパイラーの印刷出力は IBM 提供の印刷装置ファイル QSYSPRT に送られます。

printer-file-name: プリコンパイラー印刷出力が送られる印刷装置ファイルの名前を指定します。

SRTSEQ

SQL ステートメントの中の文字列比較に使用される分類順序テーブルを指定します。

注: アプリケーション・サーバーが iSeries システム上にない分散アプリケーション、またはリリース・レベルが V2R3M0 より前の分散アプリケーションでは、このパラメーターに *HEX を指定する必要があります。

***JOB:** ジョブの SRTSEQ 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの SRTSEQ 値は、プログラム実行時に検索されます。分散アプリケーションの場合、SRTSEQ(*JOBRUN) が有効なのは、LANGID(*JOBRUN) も指定されている場合だけです。

***LANGIDUNQ:** LANGID パラメーターで指定した言語用の固有の分類テーブルが使用されます。

***LANGIDSHR:** LANGID パラメーターで指定した言語用の共用分類テーブルが使用されます。

***HEX:** 分類順序テーブルは使用しません。分類順序を決定するには、文字の 16 進値を使用します。

分類順序テーブルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

table-name: 使用する分類順序テーブルの名前を指定します。

LANGID

SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されたときに使用される言語識別コードを指定します。

***JOB:** ジョブの LANGID 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの LANGID 値は、プログラム実行時に検索されます。分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

language-id: プログラムが使用する言語識別コードを指定します。

USRPRF

コンパイル済みプログラム・オブジェクトが実行されるときに使用されるユーザー・プロファイル (プログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトに対して所有する権限を含む) を指定します。プログラム・オブ

ジェクトが使用できるオブジェクトの制御には、プログラム所有者またはプログラム・ユーザーのプロファイルが使用されます。

***NAMING:** ユーザー・プロファイルが命名規則によって判別されます。命名規則が *SQL である場合は、USRPRF(*OWNER) が使用されます。命名規則が *SYS である場合は、USRPRF(*USER) が使用されます。

***USER:** プログラム・オブジェクトを実行するユーザーのプロファイルが使用されます。

***OWNER:** プログラム所有者とプログラム・ユーザーの両方のユーザー・プロファイルがプログラムの実行時に使用されます。

DYNUSRPRF
動的 SQL ステートメントに使用されるユーザー・プロファイルを指定します。

***USER:** ローカルな動的 SQL ステートメントは、ジョブのユーザー・プロファイルに基づいて実行されます。分散動的 SQL ステートメントはアプリケーション・サーバー・ジョブのユーザー・プロファイルに基づいて実行されます。

***OWNER:** ローカルな動的 SQL ステートメントは、プログラムの所有者のユーザー・プロファイルに基づいて実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のユーザー・プロファイルに基づいて実行されます。

TOSRCFILE
SQL プリコンパイラーによって処理された出力ソース・メンバーを含む、ソース・ファイルの修飾名を指定します。指定したソース・ファイルが見つからない場合、作成されます。出力メンバーの名前は、SRCMBR パラメーターに指定した名前と同じになります。

指定できるライブラリー値は次のとおりです。

QTEMP: ライブラリー QTEMP が使用されます。

***LIBL:** 指定したファイルがジョブのライブラリー・リストで検索されます。ライブラリー・リストに載せられているどのライブラリーにもファイルが見つからなければ、現行ライブラリー内にファイルが作成されます。

***CURLIB:** ジョブ用の現行ライブラリーが使用されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 出力ソース・ファイルを含むライブラリーの名前を指定します。

QSQLTEMP: ソース・ファイル QSQLTEMP が使用されます。

source-file-name: 出力ソース・メンバーを含むソース・ファイルの名前を指定します。

TEXT
プログラムおよびその機能を簡単に記述するテキストを指定します。このパラメーターの詳細については、Information Center の「CL 解説書」の『TEXT パラメーター』を参照してください。

***SRCMBRTXT:** テキストは COBOL プログラムを作成するために使用されるソース・ファイル・メンバーから取られます。データベース・ソース・メンバーの

テキストを追加または変更するには、原始ステートメント入力ユーティリティー開始 (STRSEU) コマンドを使用するか、物理ファイル・メンバー追加 (ADDPFM) または物理ファイル・メンバー変更 (CHGPFM) コマンドを使用します。ソース・ファイルがインライン・ファイルまたは装置ファイルの場合は、テキストはブランクになります。

***BLANK:** テキストは指定されません。

'description': 50 文字以下のテキストをアポストロフィで囲んで指定します。

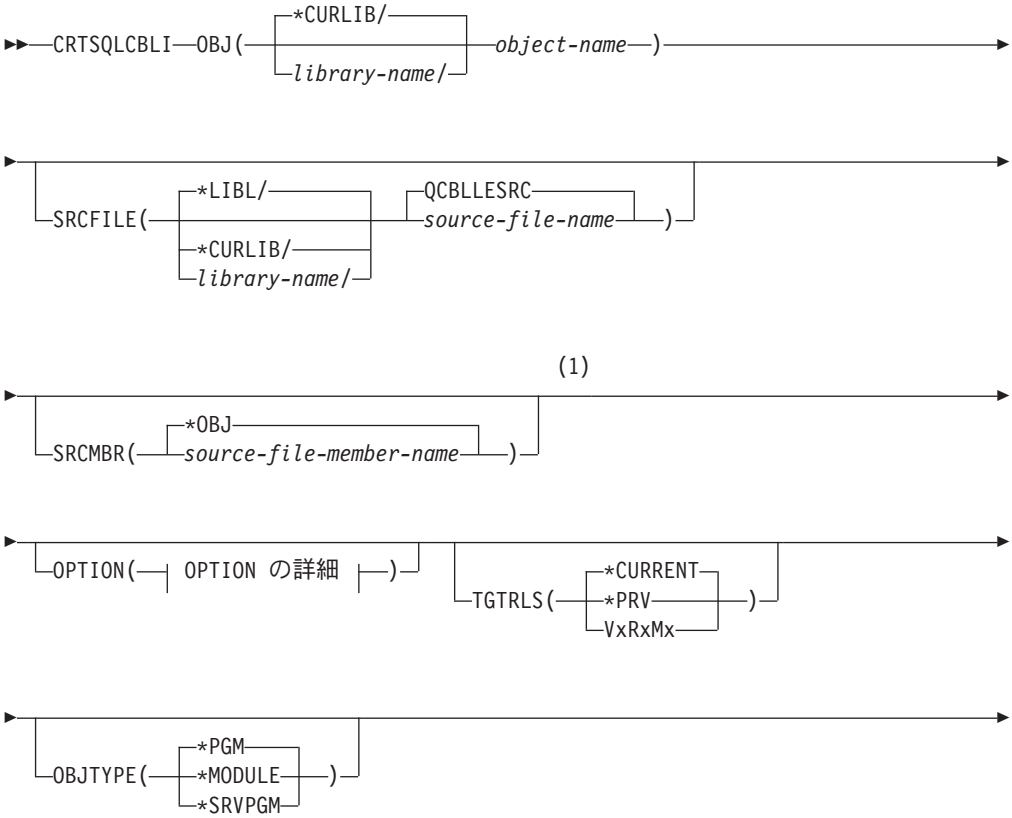
例:

```
CRTSQLCBL   PGM(ACCTS/STATS)  SRCFILE(ACCTS/ACTIVE)
  TEXT('Statistical analysis program for
  active accounts')
```

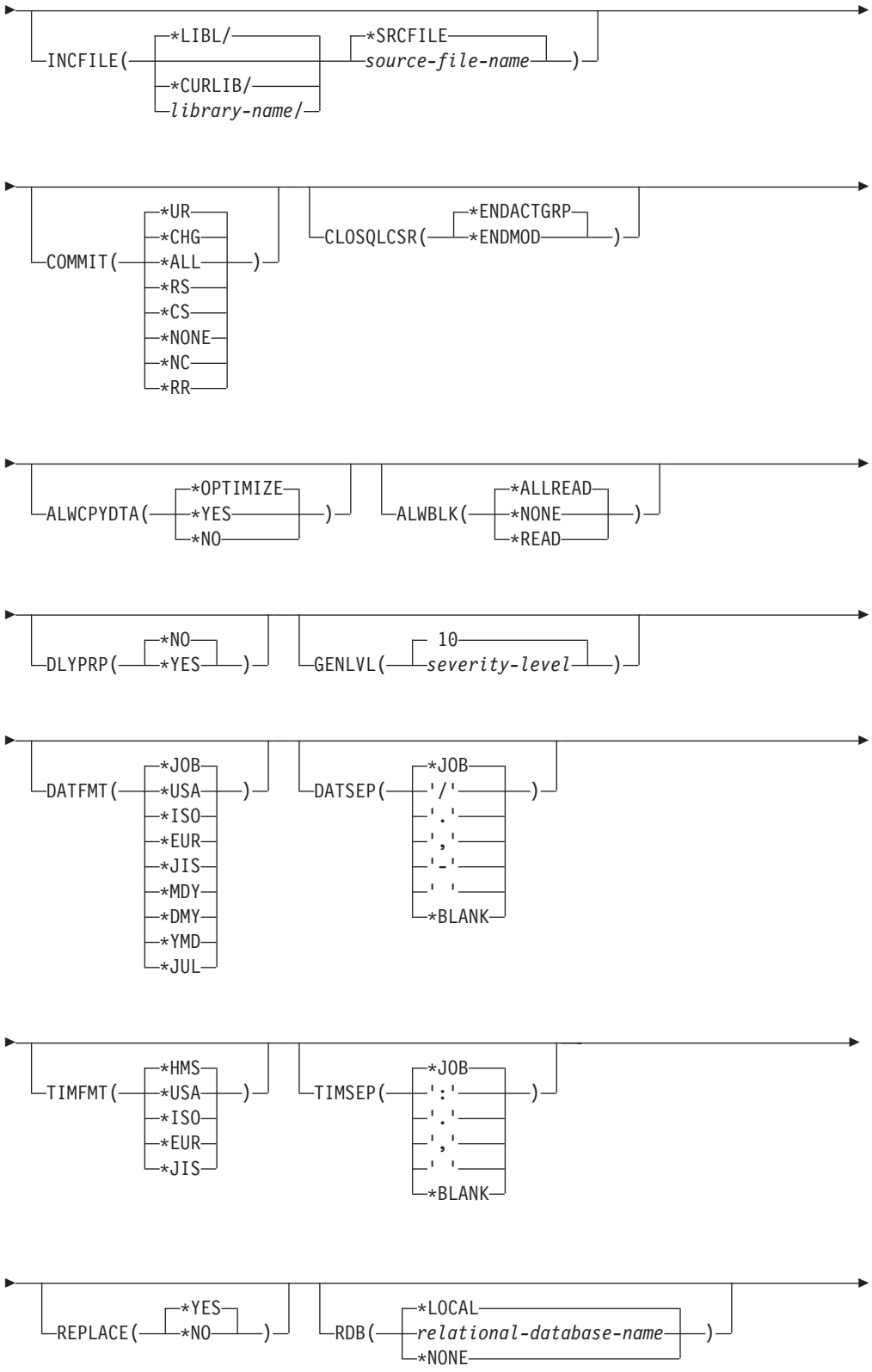
このコマンドは SQL プリコンパイラーを実行させるもので、そのプリコンパイラーはソース・プログラムをプリコンパイルし、変更したソース・プログラムをライブラリー QTEMP のファイル QSQLTEMP 内のメンバー STATS の中に保管します。 SQL プリコンパイラーが作成したソース・メンバーを使用してライブラリー ACCTS の中でプログラム STATS を作成するために COBOL コンパイラーが呼び出されます。

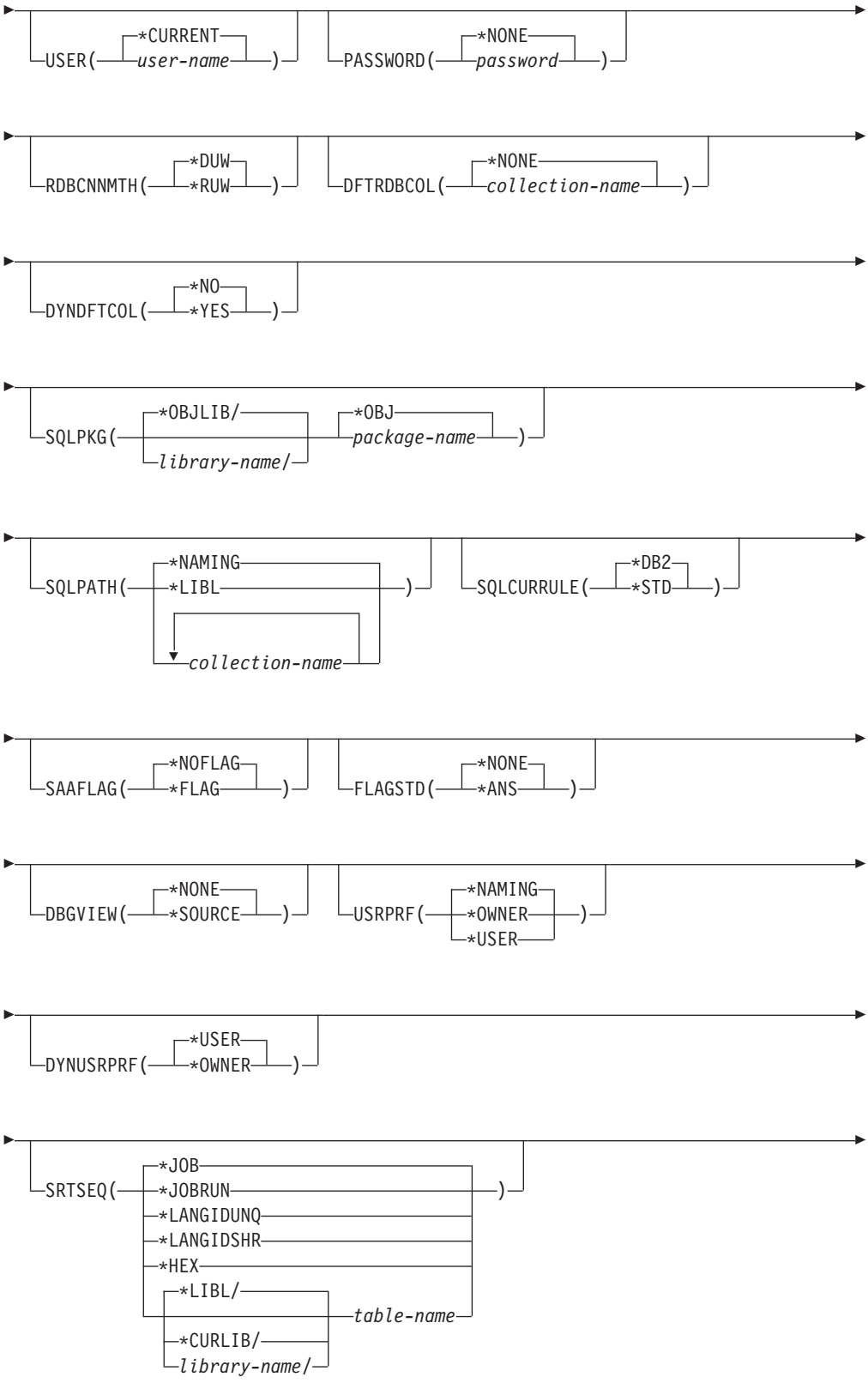
CRTSQLCBLI (SQL ILE COBOL オブジェクト作成) コマンド

Job: B,I Pgm: B,I REXX: B,I Exec

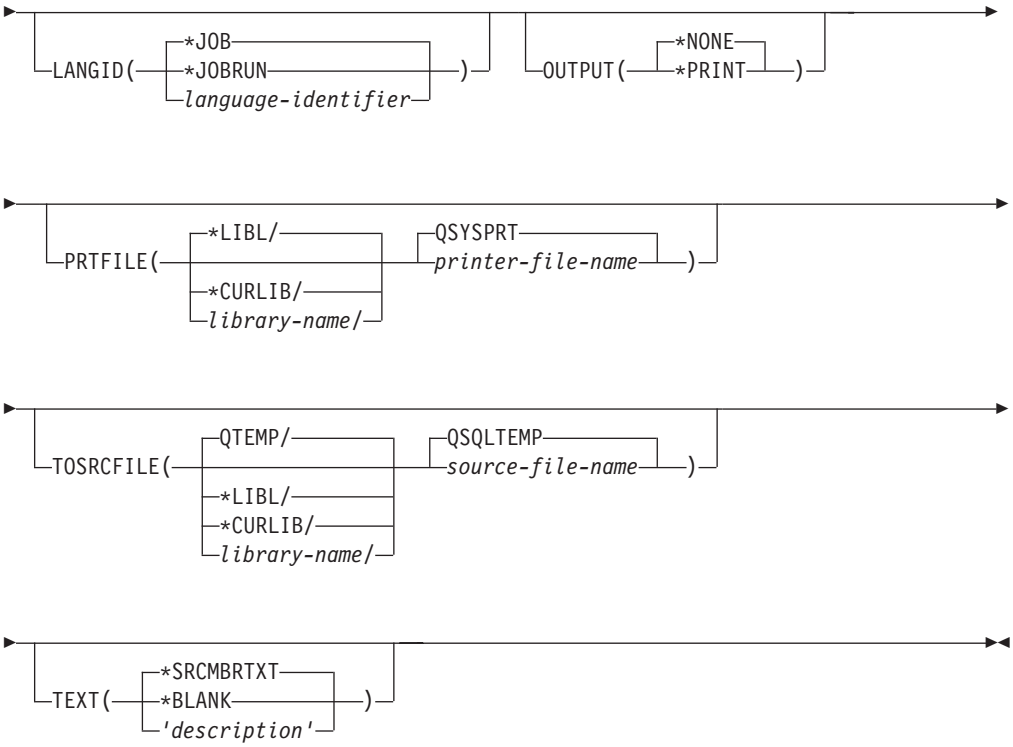


CRTSQLCBLI

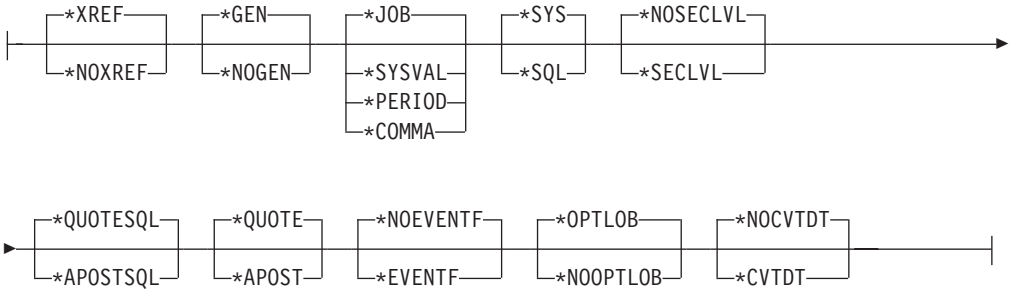

INCFILE(
*LIBL/
*CURLIB/
library-name/
*SRCFILE
source-file-name
)
COMMIT(
*UR
*CHG
*ALL
*RS
*CS
*NONE
*NC
*RR
)
CLOSQLCSR(
*ENDACTGRP
*ENDMOD
)
ALWCPYDTA(
*OPTIMIZE
*YES
*NO
)
ALWBLK(
*ALLREAD
*NONE
*READ
)
DLYPRP(
*NO
*YES
)
GENLVL(
10
severity-level
)
DATFMT(
*JOB
*USA
*ISO
*EUR
*JIS
*MDY
*DMY
*YMD
*JUL
)
DATSEP(
*JOB
'/'
'.'
','
'-'
' '
*BLANK
)
TIMFMT(
*HMS
*USA
*ISO
*EUR
*JIS
)
TIMSEP(
*JOB
':'
'.'
','
' '
*BLANK
)
REPLACE(
*YES
*NO
)
RDB(
*LOCAL
relational-database-name
*NONE
)


USER(*CURRENT user-name)
PASSWORD(*NONE password)
RDBCNNMTH(*DUW *RUW)
DFTRDBCOL(*NONE collection-name)
DYNDFTCOL(*NO *YES)
SQLPKG(*OBJLIB/ library-name/ *OBJ package-name)
SQLPATH(*NAMING *LIBL collection-name)
SQLCURRULE(*DB2 *STD)
SAAFLAG(*NOFLAG *FLAG)
FLAGSTD(*NONE *ANS)
DBGVIEW(*NONE *SOURCE)
USRPRF(*NAMING *OWNER *USER)
DYNUSRPRF(*USER *OWNER)
SRTSEQ(*JOB *JOBRUN *LANGIDUNQ *LANGIDSHR *HEX *LIBL/ *CURLIB/ library-name/ table-name)

CRTSQLCBLI



OPTION の詳細:



注:

1 これより前にあるパラメーターはすべて、定位置形式で指定されます。

目的:

SQL ILE COBOL オブジェクト作成 (CRTSQLCBLI) コマンドは構造化照会言語 (SQL) プリコンパイラーを呼び出すもので、このプリコンパイラーが SQL ステートメントを含む COBOL ソース・プログラムをプリコンパイルし、一時ソース・メンバーを作成してから、任意選択で ILE COBOL コンパイラーを呼び出してモジュール、プログラム、またはサービス・プログラムの作成を行います。

パラメーター:

OBJ

作成するオブジェクトの修飾名を指定します。

***CURLIB:** ジョブの現行ライブラリー内で新しいオブジェクトが作成されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: オブジェクトが作成されるライブラリーの名前を指定します。

object-name: 作成するオブジェクトの名前を指定します。

SRCFILE

SQL ステートメントとともに COBOL ソース・プログラムが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QCBLLESRC: ソース・ファイル名が指定されていない場合は、ソース・ファイル QCBLLESRC に COBOL ソース・プログラムが入ります。

source-file-name: COBOL ソース・プログラムが入っているソース・ファイルの名前を指定します。

SRCMBR

COBOL ソース・プログラムが入っているソース・ファイル・メンバーの名前を指定します。このパラメーターは、SRCFILE パラメーターのソース・ファイル名がデータベース・ファイルである場合にのみ指定します。このパラメーターを指定しないと、OBJ パラメーターに指定した OBJ 名が使用されます。

***OBJ:** OBJ パラメーターに指定した名前と同じ名前をもつソース・ファイルのメンバーに COBOL ソース・プログラムがあることを指定します。

source-file-member-name: COBOL ソースが入っているメンバーの名前を指定します。

OPTION

COBOL ソース・プログラムをプリコンパイルするときに次のオプションのうち 1 つ以上が使用されるかどうかを指定します。オプションが 2 度以上使用される場合、または 2 つのオプションが対立する場合は、指定された最後のオプションが使用されます。

要素 1: 相互参照オプション

***XREF:** プリコンパイラーは、プログラムの中の項目と、プログラムの中でそれらの項目を参照しているステートメントの番号との間の相互参照を行います。

***NOXREF:** プリコンパイラーは名前の相互参照を行いません。

要素 2: プログラム作成オプション

***GEN:** プリコンパイラーは、OBJTYPE パラメーターが指定するオブジェクトを作成します。

***NOGEN:** プリコンパイラーは ILE COBOL コンパイラーの呼び出しを行わないので、モジュール、プログラム、サービス・プログラム、または SQL パッケージは作成されません。

要素 3: 小数点オプション

***JOB:** SQL で数値定数の小数点として使用される値は、プリコンパイル時にジョブ用に指定されている小数点の表現になります。

***SYSVAL:** SQL ステートメントの中の数値定数に小数点として使用する値は QDECFMT システム値になります。

***PERIOD:** SQL ステートメントの中で数値定数の小数点として使用する値はピリオド (.) になります。

注: QDECFMT が、小数点として使用する値がコンマ (,) であると指定している場合、リストの中の (SELECT 文節、VALUE 文節のような) 数値定数は、いずれもコンマ (,) の後にブランク ( ) を置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオド (.) である VALUES(1.1,2.23,4.1) と同じものです。

***COMMA:** SQL ステートメントの中の数値定数に小数点として使用する値はコンマ (,) になります。

注: リストの中の (SELECT 文節、VALUES 文節などのように) 数値定数は、いずれもコンマ (,) の後にブランク ( ) を置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオド (.) である VALUES(1.1,2.23,4.1) と同じものです。

要素 4: 命名規則オプション

***SYS:** システム命名規則 (library-name/file-name) を使用します。

***SQL:** SQL の命名規則 (schema-name.table-name) を使用します。

iSeries 以外のリモート・データベース上でプログラムを作成するときは、命名規則として *SQL を指定しなければなりません。

要素 5: 2 次レベル・メッセージ・テキスト・オプション

***NOSECLVL:** 2 次レベルのテキスト記述はリストに追加されません。

***SECLVL:** リストのいずれかのメッセージについても、置換データを含む 2 次レベルのテキストが追加されます。

要素 6: 文字列区切り文字オプション

***QUOTESQL:** 2 重引用符 (") が SQL ステートメント内の文字列区切り文字になります。

***APOSTSQL:** アポストロフィ (') が SQL ステートメント内の文字列区切り文字になります。

要素 7: リテラル・オプション

***QUOTE:** COBOL ステートメントの中の数字リテラルおよびブール・リテラルでないリテラルに 2 重引用符 (") を使用します。

***APOST:** COBOL ステートメントの中の数字リテラルおよびブール・リテラルでないリテラルにアポストロフィ (') を使用します。

要素 8: イベント・ファイル作成

***NOEVENTF:** コンパイラーは、連携開発環境プログラム/400 (CODE/400) が使用するためのイベント・ファイルを作成しません。

***EVENTF:** コンパイラーは、連携開発環境プログラム/400 (CODE/400) が使用するイベント・ファイルを作成します。イベント・ファイルは、ソース・ライブラリー内のファイル EVFEVENT のメンバーとして作成されます。CODE/400 はこのファイルを使用して、CODE/400 編集プログラムに統合されたエラー・フィードバックを提供します。このオプションは、通常はユーザーの代わりに CODE/400 によって指定されます。

要素 9: DRDA に対するラージ・オブジェクトの最適化

***OPTLOB:** カーソルに対する最初の FETCH によって、これ以降のすべての FETCH での LOB (ラージ・オブジェクト) に対するカーソルの使用方法が決定されます。このオプションは、カーソルがクローズされるまで、引き続き有効です。

最初の FETCH が LOB ロケーターを使用して LOB 列にアクセスする場合、そのカーソルに対する後続の FETCH がその LOB 列を取り出して LOB ホスト変数に入れることはありません。

最初の FETCH が LOB 列を LOB ホスト変数に置いた場合、そのカーソルに対する後続の FETCH がその列に対して LOB ロケーターを使用することはできません。

***NOOPTLOB:** 列が検索されて LOB ロケーターまたは LOB ホスト変数に入れられるかどうかについての制約はありません。このオプションは、パフォーマンス低下の原因となることがあります。

要素 10: 日付変換

***NOCVTDT:** 外部記述のデータベース・ファイルから検索された日付、時刻、およびタイム・スタンプの各データ・タイプは、日付、時刻、およびタイム・スタンプのデータ・タイプを使用して処理されます。

***CVTDT:** 外部記述のデータベース・ファイルから検索された日付、時刻、およびタイム・スタンプの各データ・タイプは、固定長文字フィールドとして処理されます。

TGTRLS

作成中のオブジェクトを使用するオペレーティング・システムのリリース・レベルを指定します。

*CURRENT 値および *PRV 値の例では、VxRxMx 形式でリリースを指定する *release-level* 値が示されています。ここで Vx はバージョン、Rx はリリース、

Mx はモディフィケーション・レベルです。たとえば、V2R3M0 はバージョン 2、リリース 3、モディフィケーション・レベル 0 です。

***CURRENT:** オブジェクトは、ユーザーのシステムで現在稼働しているオペレーティング・システムのリリースで使用されます。たとえば、V2R3M5 がシステムで稼働している場合、*CURRENT はユーザーが V2R3M5 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

注: V2R3M5 がシステム上で稼働しており、V2R3M0 が導入されたシステムでオブジェクトが使用される場合、TGTRLS(*CURRENT) ではなく TGTRLS(V2R3M0) を指定してください。

***PRV:** オブジェクトはオペレーティング・システムのモディフィケーション・レベル 0 の前のリリースで使用されます。たとえば、 V2R3M5 がユーザーのシステムで稼働している場合、*PRV はユーザーが V2R2M0 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

release-level: VxRxMx 形式でリリースを指定してください。指定されたリリースのシステム上、あるいは以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することができます。

有効な値は、現行バージョン、リリース、モディフィケーション・レベルによります。また、有効な値は各新規リリースで変わります。コマンドがサポートする初期リリース・レベルよりもさらに前のリリース・レベルを指定した場合には、エラー・メッセージはこれがサポートする最も初期のリリース・レベルを示して送信されます。

OBJTYPE

作成するオブジェクトのタイプを指定します。

***PGM:** SQL プリコンパイラーは CRTBNDCBL コマンドを出してバインド・プログラムを作成します。

***MODULE:** SQL プリコンパイラーは CRTCBLMOD を出してモジュールを作成します。

***SRVPGM:** SQL プリコンパイラーは CRTCBLMOD コマンドと CRTSRVPGM コマンドを出してサービス・プログラムを作成します。

注:

1. OBJTYPE(*PGM) または OBJTYPE(*SRVPGM) を指定し、かつ RDB パラメーターも指定すると、プログラムの作成後に、SQL プリコンパイラーによって CRTSQLPKG コマンドが出されます。 OBJTYPE(*MODULE) を指定した場合は、SQL パッケージが作成されないので、CRTPGM コマンドまたは CRTSRVPGM コマンドでプログラムを作成した後でユーザーが CRTSQLPKG コマンドを出さなければなりません。
2. *NOGEN を指定した場合は、SQL 一時ソース・メンバーだけが作成され、モジュール、プログラム、サービス・プログラム、または SQL パッケージは作成されません。

INCFILE

SQL の INCLUDE ステートメントを用いてプログラムに組み込むメンバーが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

***SRCFILE:** SRCFILE パラメーターで指定した修飾されたソース・ファイルには、SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入ります。

source-file-name: SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入っているソース・ファイルの名前を指定します。ここに指定するソース・ファイルのレコード長は、SRCFILE パラメーターに指定したソース・ファイルのレコード長より小さくてはなりません。

COMMIT

コンパイル済みの単位内の SQL ステートメントがコミットメント制御下で実行されるかどうかを指定します。ホスト言語ソースの中で参照されているファイルは、このオプションによって影響を受けません。SQL ステートメントの中で参照される SQL テーブル、SQL ビュー、および SQL パッケージだけが影響されます。

***CHG または *UR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができます。

***ALL または *RS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。

***CS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。選択されたが、更新されていない行は、次の行が選択されるまでロックされます。他のジョブにおけるコミットされていない変更は見ることができません。

***NONE または *NC:** コミットメント制御が使用されないことを指定します。他のジョブにおけるコミットされていない変更は見ることができます。プログラムに SQL の DROP SCHEMA ステートメントが組み込まれている場合は、*NONE または *NC を使用しなければなりません。RDB パラメーターでリレ

ーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries 以外のシステム上にある場合は、*NONE または *NC を指定することはできません。

***RR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。
SELECT、UPDATE、DELETE、および INSERT ステートメントで参照されているすべてのテーブルは、作業単位 (トランザクション) の終わりまで排他的にロックされます。

CLOSQLCSR

SQL カーソルが暗黙にクローズされるとき、SQL 準備済みステートメントが暗黙に破棄され、LOCK TABLE のロックが解除されることを指定します。
CLOSE、COMMIT、または ROLLBACK (HOLD を指定しない) SQL ステートメントを発行すると、SQL カーソルが明示的にクローズされます。

***ENDACTGRP:** 活動化グループが終了した時点で、SQL カーソルがクローズされ、SQL 準備済みステートメントが暗黙に廃棄され、LOCK TABLE のロックが解除されます。

***ENDMOD:** モジュールが終了した時点で、SQL カーソルをクローズし、SQL 準備済みステートメントを暗黙に廃棄します。LOCK TABLE のロックは、活性化グループが終了した時点で解除します。

ALWCPYDTA

SELECT ステートメントでデータのコピーが使用できるかどうかを指定します。

***OPTIMIZE:** データベースから直接検索されたデータを使用するか、データのコピーを使用するかどうかは、システムが決定します。決定は、どの方式が最良のパフォーマンスを提供するかに基づいて行われます。 COMMIT が *CHG または *CS で ALWBLK が *ALLREAD でない場合、あるいは COMMIT が *ALL または *RR の場合、データのコピーが使用されるのは、照会を実行する必要があるときだけです。

***YES:** 必要な場合にデータのコピーが使用されます。

***NO:** データのコピーを使用することはできません。照会を実行するのにデータの一時コピーが必要な場合には、エラー・メッセージが返されます。

ALWBLK

データベース・マネージャーがレコードのブロック化を使用できるかどうか、および読み取り専用カーソルについてどの程度までブロック化が使用できるかを指定します。

***ALLREAD:** COMMIT パラメーターで *NONE または *CHG が指定されている場合は、行は読み取り専用カーソルに対してブロックされます。プログラム内で、明示的に更新することができないすべてのカーソルは、プログラム内に EXECUTE または EXECUTE IMMEDIATE ステートメントがある場合でも、読み取り専用にオープンされます。

*ALLREAD を指定すると、

- *READ の場合に許されるブロック化に加え、コミットメント制御レベル *CHG のもとでレコードのブロック化を行うことができます。
- プログラムの中のほとんどすべての読み取り専用カーソルのパフォーマンスを向上できますが、照会が次のように制限されます。
  - ロールバック (ROLLBACK) コマンド、ホスト言語で書いた ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、*ALLREAD の指定があるとき、読み取り専用カーソルの再位置決めはしません。
  - カーソル内の行を更新するために、位置付けされた UPDATE または DELETE 使用ステートメントの動的実行 (たとえば、EXECUTE IMMEDIATE による) をすることはできません。ただし、カーソルに関する DECLARE ステートメントに FOR UPDATE 文節が含まれている場合を除きます。

***NONE:** カーソルの対象となるデータを検索するとき、行がブロック化されません。

*NONE を指定すると、

- 検索されるデータが最新であることが保証されます。
- 照会の対象となるデータの最初の行を検索するときの所要時間が短縮します。
- 照会の最初の数行だけが検索されてから照会がクローズされるときは、プログラムによって使用されないデータ行のブロックの検索をデータベース・マネージャーに中止させます。
- 大量の行を検索する照会全体のパフォーマンスが低下する可能性があります。

***READ:** 次のとき、カーソルの対象となるデータを読み取り専用で検索するときレコードがブロック化されます。

- COMMIT パラメーターに *NONE の指定があるとき。これは、コミットメント制御が使用されないことを示します。
- カーソルが FOR READ ONLY 文節を使用して宣言されているか、あるいは位置指定の UPDATE または DELETE ステートメントをカーソルに対して実行できる動的ステートメントがないとき。

*READ を指定すると、上記条件を満足する照会の全体のパフォーマンスが向上し、大量のレコードを検索することができます。

DLYPRP

PREPARE ステートメントについての動的ステートメント妥当性検査を、OPEN、 EXECUTE、または DESCRIBE ステートメントが実行されるまで遅延させるかどうかを指定します。妥当性検査を遅延させると、余分な妥当性検査が除かれるため、パフォーマンスが向上します。

***NO:** 動的ステートメント妥当性検査を遅延させません。動的ステートメントが準備されるとき、アクセス・プランが妥当性検査されます。動的ステートメントが OPEN または EXECUTE ステートメントで使用される場合、アクセス・プランが再度妥当性検査されます。動的ステートメントによって参照されるオブジェクトの権限または存在は変化する場合があるので、OPEN または EXECUTE

ステートメントを出した後、SQLCODE または SQLSTATE を検査して、動的ステートメントがまだ有効であるか確かめる必要があります。

***YES:** 動的ステートメント妥当性検査を、動的ステートメントが OPEN、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅延させます。動的ステートメントが使用されたときは、その妥当性検査が行われて、アクセス・プランが作られます。このパラメーターで *YES を指定する場合は、OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後 SQLCODE と SQLSTATE を調べて、動的ステートメントが有効であるかどうかを確かめておく必要があります。

注: *YES を指定したときは、PREPARE ステートメントで INTO 文節が使用されている場合や、OPEN が動的ステートメントに対して出される前に DESCRIBE ステートメントがその動的ステートメントを使用した場合は、パフォーマンスは向上しません。

GENLVL

作成操作が失敗する時の重大度レベルを指定します。重大度レベルがこの値より大きいエラーが発生する場合は、操作が終了します。

10: 省略時の重大度レベルは 10 です。

severity-level: 0 から 40 までの範囲で重大度レベルの値を指定します。

DATFMT

日付結果列にアクセスするときに使用される形式を指定します。すべての出力日付フィールドは、指定された形式で返されます。入力日付文字列については、日付が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、*USA、*ISO、*EUR、または *JIS を指定しなければなりません。

***JOB:** ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***USA:** 米国の日付形式 (mm/dd/yyyy) が使用されます。

***ISO:** 国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR:** ヨーロッパの日付形式 (dd.mm.yyyy) が使用されます。

***JIS:** 日本工業規格の日付形式 (yyyy-mm-dd) が使用されます。

***MDY:** 日付形式 (mm/dd/yy) が使用されます。

***DMY:** 日付形式 (dd/mm/yy) が使用されます。

***YMD:** 日付形式 (yy/mm/dd) が使用されます。

***JUL:** 年間通算日の日付形式 (yy/ddd) が使用されます。

DATSEP
日付結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブに指定された日付区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

'/': スラッシュ (/) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

'-': ダッシュ (-) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

TIMFMT
時刻結果列にアクセスするときに使用される形式を指定します。入力時刻文字列については、時刻が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、時刻形式は、時刻区切り記号がコロンまたはピリオドである *USA、*ISO、*EUR、*JIS、または *HMS でなければなりません。

***HMS:** **hh:mm:ss** 形式が使用されます。

***USA:** 米国の時刻形式 **hh:mm xx** が使用されます。ただし、**xx** は AM か PM です。

***ISO:** 国際標準化機構 (ISO) の時刻形式 **hh.mm.ss** が使用されます。

***EUR:** ヨーロッパの時刻形式 **hh.mm.ss** が使用されます。

***JIS:** 日本工業規格の時刻形式 **hh:mm:ss** が使用されます。

TIMSEP
時刻結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブのために指定された時刻区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

':': コロン (:) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

REPLACE

同じライブラリー内に同じ名前およびタイプの既存の SQL モジュール、プログラム、サービス・プログラム、またはパッケージかある場合に、SQL モジュール、プログラム、サービス・プログラム、またはパッケージを作成するかどうかを指定します。このパラメーターの値は、CRTCBLMOD、CRTBNDCBL、CRTSRVPGM、および CRTSQLPKG コマンドに渡されます。

***YES:** 新しい SQL モジュール、プログラム、サービス・プログラム、またはパッケージが作成され、指定したライブラリーの中の同じ名前およびタイプの既存の SQL オブジェクトは QRPLOBJ に移されます。

***NO:** 指定されたライブラリーに同じ名前とタイプの SQL オブジェクトがすでに存在している場合は、新しい SQL モジュール、プログラム、サービス・プログラム、またはパッケージは作成されません。

RDB

SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。

***LOCAL:** プログラムは分散 SQL プログラムとして作成されます。 SQL ステートメントはローカル・データベースにアクセスします。プリコンパイル・プロセスの一部として SQL パッケージ・オブジェクトは作成されません。 SQL パッケージの作成 (CRTSQLPKG) コマンドを使用することができます。

relational-database-name: 新しい SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。ローカル・リレーショナル・データベースの名前を指定すると、作成されるプログラムは分散 SQL プログラムになります。 SQL ステートメントはローカル・データベースにアクセスします。

***NONE:** SQL パッケージ・オブジェクトは作成されません。プログラム・オブジェクトは分散プログラムではなく、 SQL パッケージの作成 (CRTSQLPKG) コマンドは使用できません。

USER

会話の開始時にリモート・システムに送られるユーザー名を指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***CURRENT:** 現在のジョブが実行されているユーザー・プロファイルが使用されます。

user-name: アプリケーション・サーバー・ジョブに使用されるユーザー名を指定します。

PASSWORD

リモート・システムで使用されるパスワードを指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***NONE:** パスワードは送られません。この値を指定する場合は、USER(*CURRENT) も指定しなければなりません。

password: USER パラメーターに指定したユーザー名のパスワードを指定します。

RDBCNNMTH

CONNECT ステートメントに使用する意味を指定します。詳細については、「*SQL 解説書*」の CONNECT (タイプ 1) および CONNECT (タイプ 2) を参照してください。

***DUW:** 分散作業単位をサポートするために CONNECT (タイプ 2) の意味が使用されます。追加のリレーショナル・データベースへの連続する CONNECT ステートメントによって、直前の接続が切断されることはありません。

***RUW:** リモート作業単位をサポートするのに CONNECT (タイプ 1) の意味が使用されます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

DFTRDBCOL

テーブル、ビュー、索引および SQL パッケージの非修飾名に使用されるスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用されます。

***NONE:** OPTION パラメーターで定義した命名規則が使用されます。

schema-name: スキーマ識別名を指定します。この値は、OPTION パラメーターで指定した命名規則の代わりに使用されます。

DYNDFTCOL

DFTRDBCOL パラメーター用に指定した省略時スキーマ名が動的ステートメントでも使用されるかどうかを指定します。

***NO:** 動的 SQL ステートメント用のテーブル、ビュー、索引、および SQL パッケージの非修飾名用の DFTRDBCOL パラメーターに指定した値は使用されません。OPTION パラメーターで指定した命名規則が使用されます。

***YES:** DFTRDBCOL パラメーターに指定したスキーマ名は、動的 SQL ステートメント内のテーブル、ビュー、索引、および SQL パッケージの非修飾名用に使用されます。

SQLPKG

このコマンドの RDB パラメーターで指定されたリレーショナル・データベースで作成される SQL パッケージの修飾名を指定します。

指定できるライブラリー値は次のとおりです。

***OBJLIB:** パッケージは、OBJ パラメーターで指定したライブラリーと同じ名前のライブラリー内に作成されます。

library-name: パッケージが作成されるライブラリーの名前を指定します。

***OBJ:** SQL パッケージの名前は、 OBJ パラメーターで指定したオブジェクト名と同じになります。

package-name: SQL パッケージの名前を指定します。リモート・システムが iSeries システムでないときは、8 文字を超える名前は指定できません。

SQLPATH

静的 SQL ステートメント内のプロシージャー、関数、およびユーザー定義タイプを検出するために使用するパスを指定します。

***NAMING:** 使用するパスは、OPTION パラメーターで指定した命名規則に従います。

*SYS 命名の場合、使用するパスは、*LIBL です (実行時の現行ライブラリー・リスト)。

*SQL 命名の場合、使用するパスは、″QSYS″、″QSYS2″、″userid″ です。ただし、″userid″ は、USER 特殊レジスターの値です。schema-name が DFTRDBCOL パラメーターに指定された場合、その schema-name がユーザー ID に置き替わります。

***LIBL:** 使用するパスは、実行時のライブラリー・リストです。

schema-name: 1 つ以上のスキーマ名のリストを指定します。最大 268 の個別スキーマを指定できます。

SQLCURRULE

SQL ステートメントに使用する意味を指定します。

***DB2:** SQL ステートメントのすべての意味が、DB2 について設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数が文字データとして扱われます。

***STD:** SQL ステートメントすべての意味が、ISO および ANSI の SQL 規格によって設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数がバイナリー・データとして扱われます。

SAAFLAG

IBM SQL フラグ付け機能を指定します。このパラメーターは、SQL ステートメントにフラグ付けし、SQL ステートメントが IBM SQL 構文に準拠しているかを検査します。IBM データベース・プロダクトの IBM SQL 構文に関する詳細は、「*DRDA IBM SQL Reference*」にあります。

***NOFLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行いません。

***FLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行います。

FLAGSTD

米国規格協会 (ANSI) のフラグ機能を指定します。このパラメーターは SQL ステートメントにフラグ付けし、ステートメントが次の標準に準拠するかどうかを検査します。

```
ANSI X3.135-1992 entry
ISO 9075-1992 entry
FIPS 127.2 entry
```

***NONE:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行いません。

***ANS:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行います。

DBGVIEW

SQL プリコンパイラーによって提供されるソース・デバッグ情報のタイプを指定します。

***NONE:** ソース・プログラム・ビューは生成されません。

***SOURCE:** SQL プリコンパイラーはルート・ステートメントと、必要であれば SQL INCLUDE ステートメントに関するソース・プログラム・ビューを提供します。ビューには、プリコンパイラーが生成したステートメントが入ります。

USRPRF

コンパイル済みプログラム・オブジェクトが実行されるときに使用されるユーザー・プロファイル (プログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトに対して所有する権限を含む) を指定します。プログラム・オブジェクトが使用できるオブジェクトの制御には、プログラム所有者またはプログラム・ユーザーのプロファイルが使用されます。

***NAMING:** ユーザー・プロファイルが命名規則によって判別されます。命名規則が *SQL である場合は、USRPRF(*OWNER) が使用されます。命名規則が *SYS である場合は、USRPRF(*USER) が使用されます。

***USER:** プログラム・オブジェクトを実行するユーザーのプロファイルが使用されます。

***OWNER:** プログラム所有者とプログラム・ユーザーの両方のユーザー・プロファイルがプログラムの実行時に使用されます。

DYNUSRPRF

動的 SQL ステートメントで使用するユーザー・プロファイルを指定します。

***USER:** ローカル・プログラムの場合、そのプログラムのユーザーのプロファイルの下で動的 SQL ステートメントが実行されます。分散プログラムの場合は、SQL パッケージのユーザーのプロファイルの下で動的 SQL ステートメントが実行されます。

***OWNER:** ローカル・プログラムの場合、そのプログラムの所有者のプロファイルの下で動的 SQL ステートメントが実行されます。分散プログラムの場合は、SQL パッケージの所有者のプロファイルの下で動的 SQL ステートメントが実行されます。

SRTSEQ

SQL ステートメントの中の文字列比較に使用される分類順序テーブルを指定します。

注: アプリケーション・サーバーが iSeries システム上にない分散アプリケーション、またはリリース・レベルが V2R3M0 より前の分散アプリケーションでは、このパラメーターに *HEX を指定する必要があります。

***JOB:** ジョブの SRTSEQ 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの SRTSEQ 値は、プログラム実行時に検索されます。分散アプリケーションの場合、SRTSEQ(*JOBRUN) が有効なのは、LANGID(*JOBRUN) も指定されている場合だけです。

***LANGIDUNQ:** パラメーターで指定した言語用の固有の分類テーブルが使用されます。

テーブル名は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

***LANGIDSHR:** 分類順序テーブルは複数の文字に同じ重みを使用するので、LANGID パラメーターで指定した言語に関連付けられた共用分類順序テーブルになります。

***HEX:** 分類順序は使用しません。分類順序を決定するには、文字の 16 進値を使用します。

table-name: 使用する分類順序テーブルの名前を指定します。

LANGID

SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されたときに使用される言語識別コードを指定します。

***JOB:** ジョブの LANGID 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの LANGID 値は、プログラム実行時に検索されます。分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

language-identifier: 言語識別コードを指定します。

OUTPUT

プリコンパイラーのリストを生成するかどうかを指定します。

***NONE:** プリコンパイラーのリストは生成されません。

***PRINT:** プリコンパイラーのリストが生成されます。

PRTFILE

プリコンパイラー印刷出力が送られる印刷装置ファイルの修飾名を指定します。ファイルの長さは 132 バイト以上でなければなりません。レコード長が 132 バイト未満のファイルを指定すると、情報が失われます。

印刷装置ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QSYSPRT: ファイル名の指定がない場合、プリコンパイラーの印刷出力は IBM 提供の印刷装置ファイル QSYSPRT に送られます。

printer-file-name: プリコンパイラー印刷出力が送られる印刷装置ファイルの名前を指定します。

TOSRCFILE

SQL プリコンパイラーによって処理された出力ソース・メンバーを含む、ソース・ファイルの修飾名を指定します。指定したソース・ファイルが見つからない場合、作成されます。出力メンバーの名前は、SRCMBR パラメーターに指定した名前と同じになります。

指定できるライブラリー値は次のとおりです。

QTEMP: ライブラリー QTEMP が使用されます。

***LIBL:** 指定したファイルがジョブのライブラリー・リストで検索されます。ライブラリー・リストに載せられているどのライブラリーにもファイルが見つからなければ、現行ライブラリー内にファイルが作成されます。

***CURLIB:** ジョブ用の現行ライブラリーが使用されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 出力ソース・ファイルを含むライブラリーの名前を指定します。

QSQLTEMP: ソース・ファイル QSQLTEMP が使用されます。

source-file-name: 出力ソース・メンバーを含むソース・ファイルの名前を指定します。

TEXT

印刷装置ファイルを簡単に記述するテキストを指定します。このパラメーターの詳細については、Information Center の「CL 解説書」の『TEXT パラメーター』を参照してください。

***SRCMBRTXT:** テキストは COBOL プログラムを作成するために使用されるソース・ファイル・メンバーから取られます。データベース・ソース・メンバーのテキストを追加または変更するには、原始ステートメント入力ユーティリティー開始 (STRSEU) コマンドを使用するか、あるいは物理ファイル・メンバー追加 (ADDPFM) コマンドまたは物理ファイル・メンバー変更 (CHGPFM) コマンドを使用します。ソース・ファイルがインライン・ファイルまたは装置ファイルの場合は、テキストはブランクになります。

***BLANK:** テキストは指定されません。

'description': 50 文字以下のテキストをアポストロフィで囲んで指定します。

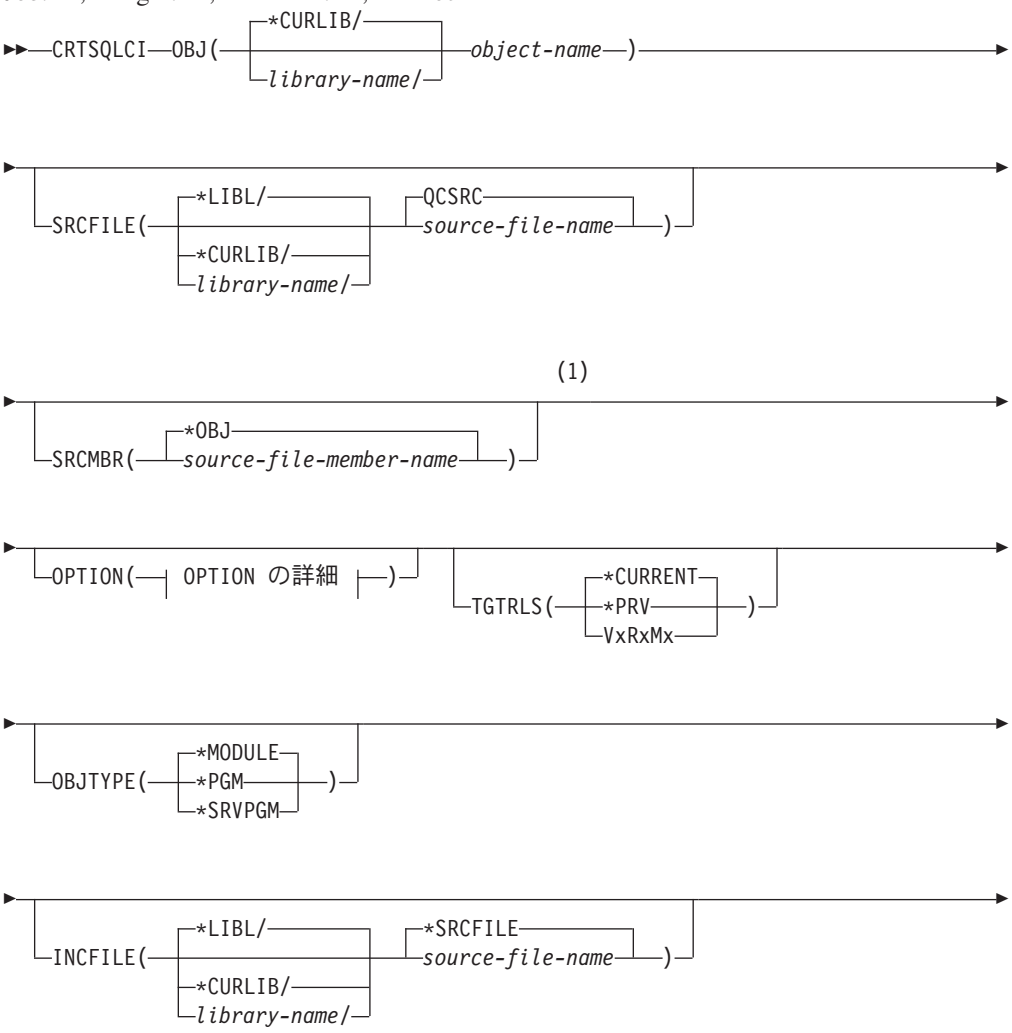
例:

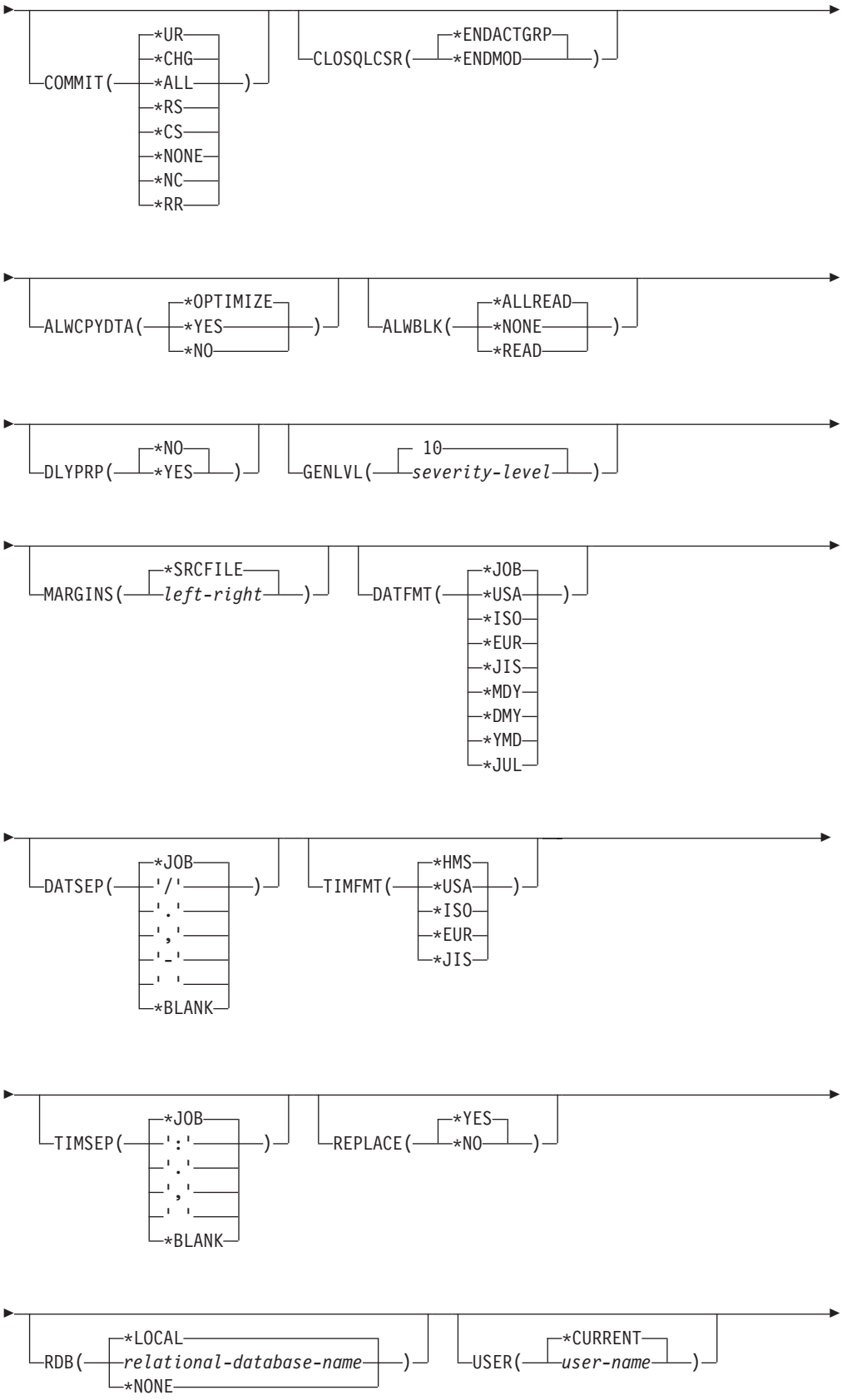
```
CRTSQLCBLI  PAYROLL OBJTYPE(*MODULE) TEXT('Payroll Program')
```

このコマンドは SQL プリコンパイラーを実行させるもので、そのプリコンパイラーはソース・プログラムをプリコンパイルし、変更したソース・プログラムをライブラリー QTEMP のファイル QSQLTEMP 内のメンバー PAYROLL の中に保管します。 SQL プリコンパイラーが作成したソース・メンバーを使用して現行ライブラリーの中でモジュール PAYROLL を作成するために ILE COBOL コンパイラーが呼び出されます。

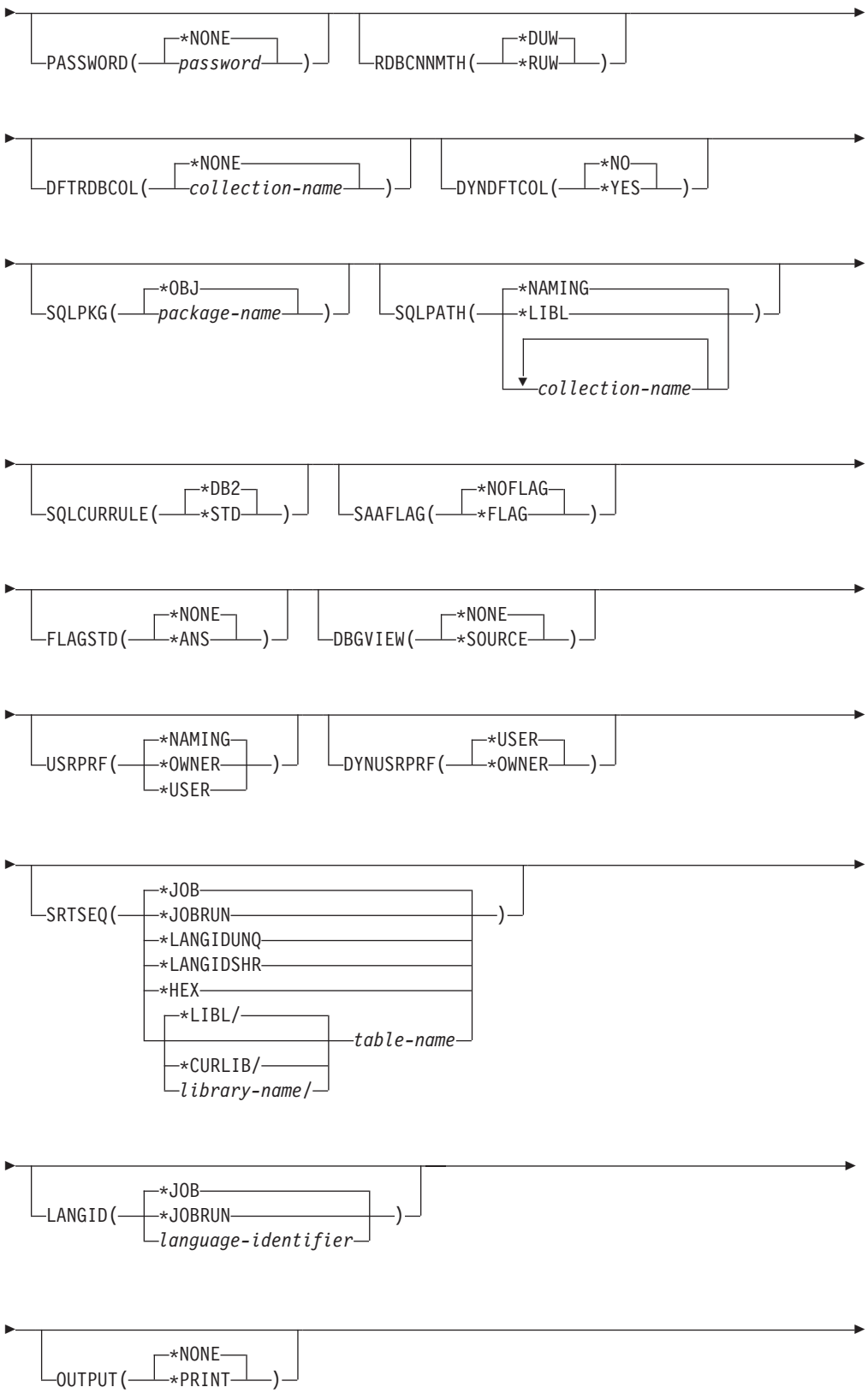
CRTSQLCI (SQL ILE C オブジェクト作成) コマンド

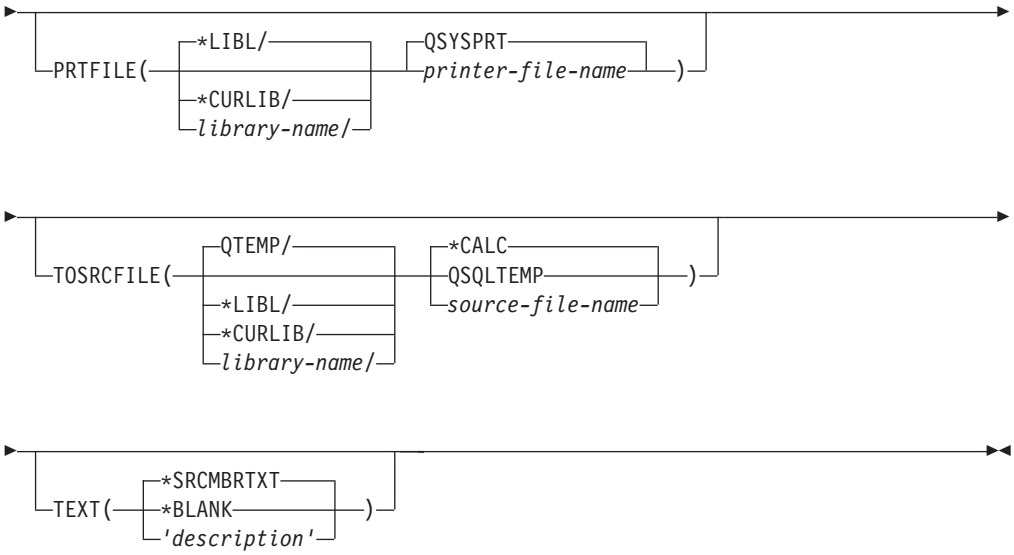
Job: B,I Pgm: B,I REXX: B,I Exec



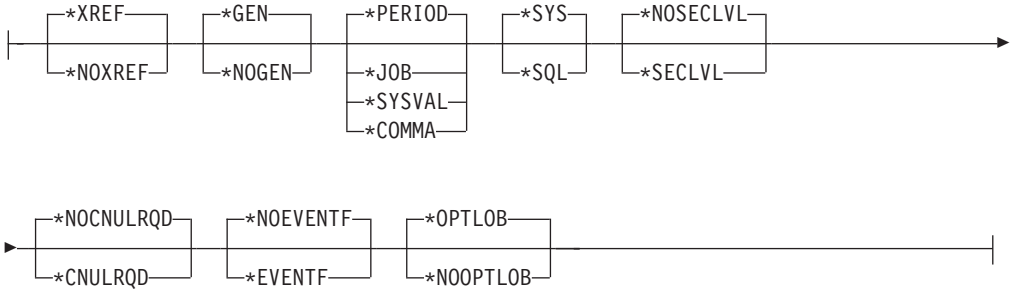

COMMIT(*UR *CHG *ALL *RS *CS *NONE *NC *RR)
CLOSQLCSR(*ENDACTGRP *ENDMOD)
ALWCPYDTA(*OPTIMIZE *YES *NO)
ALWBLK(*ALLREAD *NONE *READ)
DLYPRP(*NO *YES)
GENLVL(10 severity-level)
MARGINS(*SRCFILE left-right)
DATFMT(*JOB *USA *ISO *EUR *JIS *MDY *DMY *YMD *JUL)
DATSEP(*JOB '/' '.' ',' '-' ' ' *BLANK)
TIMFMT(*HMS *USA *ISO *EUR *JIS)
TIMSEP(*JOB ':' '.' ',' ' ' *BLANK)
REPLACE(*YES *NO)
RDB(*LOCAL relational-database-name *NONE)
USER(*CURRENT user-name)

CRTSQLCI


PASSWORD(*NONE password)
RDBCNNMTH(*DUW *RUW)
DFTRDBCOL(*NONE collection-name)
DYNDFTCOL(*NO *YES)
SQLPKG(*OBJ package-name)
SQLPATH(*NAMING *LIBL collection-name)
SQLCURRULE(*DB2 *STD)
SAAFLAG(*NOFLAG *FLAG)
FLAGSTD(*NONE *ANS)
DBGVIEW(*NONE *SOURCE)
USRPRF(*NAMING *OWNER *USER)
DYNUSRPRF(*USER *OWNER)
SRTSEQ(*JOB *JOBRUN *LANGIDUNQ *LANGIDSHR *HEX *LIBL/ *CURLIB/ library-name/ table-name)
LANGID(*JOB *JOBRUN language-identifier)
OUTPUT(*NONE *PRINT)



OPTION の詳細:



注:

1 これより前にあるパラメーターはすべて、定位置形式で指定されます。

目的:

SQL ILE C オブジェクト作成 (CRTSQLCI) コマンドは構造化照会言語 (SQL) プリコンパイラーを呼び出すもので、このプリコンパイラーが SQL ステートメントを含む C ソース・プログラムをプリコンパイルし、一時ソース・メンバーを作成してから、任意で ILE C コンパイラーを呼び出してモジュールの作成、プログラムの作成、またはサービス・プログラムの作成を行います。

パラメーター:

OBJ

作成するオブジェクトの修飾名を指定します。

オブジェクトの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***CURLIB:** ジョブの現行ライブラリー内でオブジェクトが作成されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: オブジェクトが作成されるライブラリーの名前を指定します。

object-name: 作成するオブジェクトの名前を指定します。

SRCFILE

SQL ステートメントとともに C ソース・プログラムが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QCSRC: ソース・ファイル名の指定がない場合、IBM 提供のソース・ファイル QCSRC には C ソース・プログラムが入ります。

source-file-name: C ソース・プログラムが入っているソース・ファイルの名前を指定します。

SRCMBR

C ソース・プログラムが入っているソース・ファイル・メンバーの名前を指定します。このパラメーターを指定するのは、SRCFILE パラメーター内のソース・ファイル名がデータベース・ファイルである場合だけです。このパラメーターを指定しないと、OBJ パラメーターに指定した OBJ 名が使用されます。

***OBJ:** OBJ パラメーターに指定した名前と同じ名前をもつソース・ファイルのメンバーに C ソース・プログラムがあることを指定します。

source-file-member-name: C ソースが入っているメンバーの名前を指定します。

OPTION

C ソース・プログラムをプリコンパイルするときに次のオプションのうち 1 つ以上が使用されるかどうかを指定します。オプションが 2 度以上使用される場合、または 2 つのオプションが対立する場合は、指定された最後のオプションが使用されます。

要素 1: 相互参照オプション

***XREF:** プリコンパイラーは、プログラムの中の項目と、プログラムの中でそれらの項目を参照しているステートメントの番号との間の相互参照を行います。

***NOXREF:** プリコンパイラーは名前の相互参照を行いません。

要素 2: プログラム作成オプション

***GEN:** プリコンパイラーは、OBJTYPE パラメーターが指定するオブジェクトを作成します。

***NOGEN:** プリコンパイラーは C コンパイラーを呼び出しません。モジュール、プログラム、サービス・プログラム、または SQL パッケージは作成されません。

要素 3: 小数点オプション

***PERIOD:** SQL ステートメントの中の数値定数に小数点として使用する値はピリオドになります。

***JOB:** SQL で数値定数の小数点として使用される値は、プリコンパイル時にジョブ用に指定されている小数点の表現になります。

***SYSVAL:** SQL ステートメントの中の数値定数に小数点として使用する値は QDECFMT システム値になります。

注: QDECFMT が小数点として使用する値をコンマにすることを指定している場合は、リストの中の (SELECT 文節、VALUES 文節などのように) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

***COMMA:** SQL ステートメントの中の数値定数に小数点として使用する値はコンマになります。

注: リストの中の (SELECT 文節、VALUES 文節のような) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

要素 4: 命名規則オプション

***SYS:** システム命名規則 (library-name/file-name) を使用します。

***SQL:** SQL の命名規則 (schema-name.table-name) を使用します。iSeries システム以外のリモート・データベースでパッケージを作成するときは、*SQL を命名規則として指定する必要があります。

要素 5: 2 次レベル・メッセージ・テキスト・オプション

***NOSECLVL:** 2 次レベルのテキスト記述はリストに追加されません。

***SECLVL:** リストのいずれかのメッセージについても、置換データを含む 2 次レベルのテキストが追加されます。

要素 6: NUL 必要オプション

***NOCNULRQD:** 出力文字ホスト変数および出力グラフィック・ホスト変数の場合は、ホスト変数の長さがデータと正確に同じ長さであるとき、NUL 終了文字は返されません。入力文字ホスト変数および出力グラフィック・ホスト変数には、NUL 終了文字は必要ありません。

***CNULRQD:** 出力文字ホスト変数および出力グラフィック・ホスト変数には必ず NUL 終了文字が入っています。 NUL 終了文字用の十分な空間がない場合は、データが切り捨てられ、NUL 終了文字が追加されます。入力文字ホスト変数および出力グラフィック・ホスト変数には、NUL 終了文字が必要です。

要素 7: イベント・ファイル作成

***NOEVENTF:** コンパイラーは、連携開発環境プログラム/400 (CODE/400) が使用するためのイベント・ファイルを作成しません。

***EVENTF:** コンパイラーは、連携開発環境プログラム/400 (CODE/400) が使用するイベント・ファイルを作成します。イベント・ファイルは、ソース・ライブラリー内のファイル EVFEVENT のメンバーとして作成されます。CODE/400 はこのファイルを使用して、CODE/400 編集プログラムに統合されたエラー・フィードバックを提供します。このオプションは、通常はユーザーの代わりに CODE/400 によって指定されます。

要素 8: DRDA に対するラージ・オブジェクトの最適化

***OPTLOB:** カーソルに対する最初の FETCH によって、これ以降のすべての FETCH での LOB (ラージ・オブジェクト) に対するカーソルの使用方法が決定されます。このオプションは、カーソルがクローズされるまで、引き続き有効です。

最初の FETCH が LOB ロケーターを使用して LOB 列にアクセスする場合、そのカーソルに対する後続の FETCH がその LOB 列を取り出して LOB ホスト変数に入れることはありません。

最初の FETCH が LOB 列を LOB ホスト変数に置いた場合、そのカーソルに対する後続の FETCH がその列に対して LOB ロケーターを使用することはできません。

***NOOPTLOB:** 列が検索されて LOB ロケーターまたは LOB ホスト変数に入れられるかどうかについての制約はありません。このオプションは、パフォーマンス低下の原因となることがあります。

TGTRLS

作成中のオブジェクトを使用するオペレーティング・システムのリリース・レベルを指定します。

*CURRENT 値および *PRV 値の例では、VxRxMx 形式でリリースを指定する *release-level* 値が示されています。ここで Vx はバージョン、Rx はリリース、Mx はモディフィケーション・レベルです。たとえば、V2R3M0 はバージョン 2、リリース 3、モディフィケーション・レベル 0 です。

***CURRENT:** オブジェクトは、ユーザーのシステムで現在稼働しているオペレーティング・システムのリリースで使用されます。たとえば、V2R3M5 がシステムで稼働している場合、*CURRENT はユーザーが V2R3M5 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

注: V2R3M5 がシステム上で稼働しており、V2R3M0 が導入されたシステムでオブジェクトが使用される場合、TGTRLS(*CURRENT) ではなく TGTRLS(V2R3M0) を指定してください。

***PRV:** オブジェクトはオペレーティング・システムのモディフィケーション・レベル 0 の前のリリースで使用されます。たとえば、V2R3M5 がユーザーのシ

ステムで稼働している場合、*PRV はユーザーが V2R2M0 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

release-level: VxRxMx 形式でリリースを指定してください。指定されたリリースのシステム上、あるいは以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することができます。

有効な値は、現行バージョン、リリース、モディフィケーション・レベルによります。また、有効な値は各新規リリースで変わります。コマンドがサポートする初期リリース・レベルよりもさらに前のリリース・レベルを指定した場合には、エラー・メッセージはこれがサポートする最も初期のリリース・レベルを示して送信されます。

OBJTYPE

作成するオブジェクトのタイプを指定します。

***MODULE:** SQL プリコンパイラーは、モジュールを作成するための CRTCMOD コマンドを出します。

***PGM:** SQL プリコンパイラーは、バインド・プログラムを作成するための CRTBNDC コマンドを出します。

***SRVPGM:** SQL プリコンパイラーは、サービス・プログラムを作成するための CRTCMOD コマンドおよび CRTSRVPGM コマンドを出します。

ユーザーは、OBJ パラメーターで指定した名前と同じ名前をもつソース・メンバーを QSRVSRC 内で作成する必要があります。ソース・メンバーには、モジュールへのエクスポート情報が入っている必要があります。エクスポート・ファイルについての詳しい情報は、「*ILE C OS/400 用 プログラマーの手引き*」にあります。

注:

1. OBJTYPE(*PGM) または OBJTYPE(*SRVPGM) を指定し、かつ RDB パラメーターも指定すると、プログラムの作成後に、SQL プリコンパイラーによって CRTSQLPKG コマンドが出されます。 OBJTYPE(*MODULE) を指定した場合は、SQL パッケージが作成されないので、CRTPGM または CRTSRVPGM コマンドでプログラムを作成した後でユーザーが CRTSQLPKG コマンドを出さなければなりません。
2. *NOGEN を指定すると、SQL 一時ソース・メンバーだけが生成され、モジュール、プログラム、サービス・プログラム、または SQL パッケージは作成されません。

INCFILE

SQL の INCLUDE ステートメントを用いてプログラムに組み込むメンバーが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

***SRCFILE:** SRCFILE パラメーターで指定した修飾されたソース・ファイルには、SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入ります。

source-file-name: SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入っているソース・ファイルの名前を指定します。ここに指定するソース・ファイルのレコード長は、SRCFILE パラメーターに指定したソース・ファイルのレコード長より小さくてはなりません。

COMMIT

コンパイル済みの単位内の SQL ステートメントがコミットメント制御下で実行されるかどうかを指定します。ホスト言語ソースの中で参照されているファイルは、このオプションによって影響を受けません。SQL ステートメントの中で参照される SQL テーブル、SQL ビュー、および SQL パッケージだけが影響されます。

***CHG または *UR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができます。

***ALL または *RS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。

***CS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。選択されたが、更新されていない行は、次の行が選択されるまでロックされます。他のジョブにおけるコミットされていない変更は見ることができません。

***NONE または *NC:** コミットメント制御が使用されないことを指定します。他のジョブにおけるコミットされていない変更は見ることができます。プログラムに SQL の DROP SCHEMA ステートメントが組み込まれている場合は、*NONE または *NC を使用しなければなりません。RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries 以外のシステム上にある場合は、*NONE または *NC を指定することはできません。

***RR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミ

ットされていない変更は見ることができません。
SELECT、UPDATE、DELETE、および INSERT ステートメントで参照されているすべてのテーブルは、作業単位 (トランザクション) の終わりまで排他的にロックされます。

CLOSQLCSR
SQL カーソルが暗黙にクローズされるとき、SQL 準備済みステートメントが暗黙に破棄され、LOCK TABLE のロックが解除されることを指定します。
CLOSE、COMMIT、または ROLLBACK (HOLD を指定しない) SQL ステートメントを発行すると、SQL カーソルが明示的にクローズされます。

***ENDACTGRP:** 活動化グループが終了した時点で、SQL カーソルがクローズされ、SQL 準備済みステートメントが暗黙に廃棄され、LOCK TABLE のロックが解除されます。

***ENDMOD:** モジュールが終了した時点で、SQL カーソルをクローズし、SQL 準備済みステートメントを暗黙に廃棄します。呼び出しスタック上の最初の SQL プログラムが終了すると、LOCK TABLE のロックが解除されます。

ALWCPYDTA
SELECT ステートメントでデータのコピーが使用できるかどうかを指定します。

***OPTIMIZE:** データベースから直接検索されたデータを使用するか、データのコピーを使用するかどうかは、システムが決定します。決定は、どの方式が最良のパフォーマンスを提供するかに基づいて行われます。 COMMIT が *CHG または *CS で ALWBLK が *ALLREAD でない場合、あるいは COMMIT が *ALL または *RR の場合、データのコピーが使用されるのは、照会を実行する必要があるときだけです。

***YES:** 必要な場合にデータのコピーが使用されます。

***NO:** データのコピーを使用することはできません。照会を実行するのにデータの一時コピーが必要な場合には、エラー・メッセージが返されます。

ALWBLK
データベース・マネージャーがレコードのブロック化を使用できるかどうか、および読み取り専用カーソルについてどの程度までブロック化が使用できるかを指定します。

***ALLREAD:** COMMIT パラメーターで *NONE または *CHG が指定されている場合は、行は読み取り専用カーソルに対してブロックされます。プログラム内で、明示的に更新することができないすべてのカーソルは、プログラム内に EXECUTE または EXECUTE IMMEDIATE ステートメントがある場合でも、読み取り専用にオープンされます。

*ALLREAD を指定すると、

- *READ の場合に許されるブロック化に加え、コミットメント制御レベル *CHG のもとでレコードのブロック化を行うことができます。
- プログラムの中のほとんどすべての読み取り専用カーソルのパフォーマンスを向上できますが、照会が次のように制限されます。

– ロールバック (ROLLBACK) コマンド、ホスト言語で書いた ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、*ALLREAD の指定があるとき、読み取り専用カーソルの再位置決めはしません。
– カーソル内の行を更新するために、位置付けされた UPDATE または DELETE 使用ステートメントの動的実行 (たとえば、EXECUTE IMMEDIATE による) をすることはできません。ただし、カーソルに関する DECLARE ステートメントに FOR UPDATE 文節が含まれている場合を除きます。

***NONE:** カーソルの対象となるデータを検索するとき、行がブロック化されません。

*NONE を指定すると、

- 検索されるデータが最新であることが保証されます。
- 照会の対象となるデータの最初の行を検索するときの所要時間が短縮します。
- 照会の最初の数行だけが検索されてから照会がクローズされるときは、プログラムによって使用されないデータ行のブロックの検索をデータベース・マネージャーに中止させます。
- 大量の行を検索する照会全体のパフォーマンスが低下する可能性があります。

***READ:** 次のとき、カーソルの対象となるデータを読み取り専用で検索するときレコードがブロック化されます。

- COMMIT パラメーターに *NONE の指定があるとき。これは、コミットメント制御が使用されないことを示します。
- カーソルが FOR READ ONLY 文節を使用して宣言されているか、あるいは位置指定の UPDATE または DELETE ステートメントをカーソルに対して実行できる動的ステートメントがないとき。

*READ を指定すると、上記条件を満足する照会の全体のパフォーマンスが向上し、大量のレコードを検索することができます。

DLYPRP

PREPARE ステートメントについての動的ステートメント妥当性検査を、OPEN、 EXECUTE、または DESCRIBE ステートメントが実行されるまで遅延させるかどうかを指定します。妥当性検査を遅延させると、余分な妥当性検査が除かれるため、パフォーマンスが向上します。

***NO:** 動的ステートメント妥当性検査を遅延させません。動的ステートメントが準備されるとき、アクセス・プランが妥当性検査されます。動的ステートメントが OPEN または EXECUTE ステートメントで使用される場合、アクセス・プランが再度妥当性検査されます。動的ステートメントによって参照されるオブジェクトの権限または存在は変化する場合があるので、OPEN または EXECUTE ステートメントを出した後、SQLCODE または SQLSTATE を検査して、動的ステートメントがまだ有効であるか確かめる必要があります。

***YES:** 動的ステートメント妥当性検査を、動的ステートメントが OPEN、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅延させ

ます。動的ステートメントが使用されたときは、その妥当性検査が行われて、アクセス・プランが作られます。このパラメーターで *YES を指定する場合は、OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後 SQLCODE と SQLSTATE を調べて、動的ステートメントが有効であるかどうかを確かめておく必要があります。

注: *YES を指定したときは、PREPARE ステートメントで INTO 文節が使用されている場合や、OPEN が動的ステートメントに対して出される前に DESCRIBE ステートメントがその動的ステートメントを使用した場合は、パフォーマンスは向上しません。

GENLVL

作成操作が失敗する時の重大度レベルを指定します。重大度レベルがこの値より大きいエラーが発生する場合は、操作が終了します。

10: 省略時の重大度レベルは 10 です。

severity-level: 0 から 40 までの範囲で重大度レベルの値を指定します。

MARGINS

プリコンパイラー入力レコードのうちソース・テキストが入っている部分を指定します。

***SRCFILE:** プリコンパイラーは、 SRCMBR パラメーターに指定したファイル・メンバーのマージン値を使用します。

要素 1: 左マージン

left: ステートメントの開始位置を指定します。有効な値は 1 から 32754 までです。

要素 2: 右マージン

right: ステートメントの終了位置を指定します。有効な値は 1 から 32754 までです。

DATFMT

日付結果列にアクセスするときに使用される形式を指定します。すべての出力日付フィールドは、指定された形式で返されます。入力日付文字列については、日付が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、*USA、*ISO、*EUR、または *JIS を指定しなければなりません。

***JOB:** ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***USA:** 米国の日付形式 (mm/dd/yyyy) が使用されます。

***ISO:** 国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR:** ヨーロッパの日付形式 (dd.mm.yyyy) が使用されます。

***JIS:** 日本工業規格の日付形式 (yyyy-mm-dd) が使用されます。

***MDY:** 日付形式 (mm/dd/yy) が使用されます。

***DMY:** 日付形式 (dd/mm/yy) が使用されます。

***YMD:** 日付形式 (yy/mm/dd) が使用されます。

***JUL:** 年間通算日の日付形式 (yy/ddd) が使用されます。

DATSEP
日付結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブに指定された日付区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

'/': スラッシュ (/) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

'-': ダッシュ (-) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

TIMFMT
時刻結果列にアクセスするときに使用される形式を指定します。入力時刻文字列については、時刻が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力時刻文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、時刻形式は、時刻区切り記号がコロンまたはピリオドである *USA、*ISO、*EUR、*JIS、または *HMS でなければなりません。

***HMS:** **hh:mm:ss** 形式が使用されます。

***USA:** 米国の時刻形式 **hh:mm xx** が使用されます。ただし、**xx** は AM か PM です。

***ISO:** 国際標準化機構 (ISO) の時刻形式 **hh.mm.ss** が使用されます。

***EUR:** ヨーロッパの時刻形式 **hh.mm.ss** が使用されます。

***JIS:** 日本工業規格の時刻形式 **hh:mm:ss** が使用されます。

TIMSEP

時刻結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブのために指定された時刻区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

':': コロン (:) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

REPLACE

同じライブラリー内に同じ名前およびタイプの既存の SQL モジュール、プログラム、サービス・プログラム、またはパッケージかある場合に、SQL モジュール、プログラム、サービス・プログラム、またはパッケージを作成するかどうかを指定します。このパラメーターの値は、CRTCMOD、CRTBNDC、CRTSRVPGM、および CRTSQLPKG の各コマンドに渡されます。

***YES:** 新規の SQL モジュール、プログラム、サービス・プログラム、またはパッケージが作成され、指定したライブラリー内にある同じ名前およびタイプの既存のオブジェクトはすべて QRPLOBJ に移されます。

***NO:** 指定されたライブラリーに同じ名前およびタイプのオブジェクトがすでに存在している場合は、新規の SQL モジュール、プログラム、サービス・プログラム、またはパッケージは作成されません。

RDB

SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。

***LOCAL:** プログラムは分散 SQL プログラムとして作成されます。 SQL ステートメントはローカル・データベースにアクセスします。プリコンパイル・プロセスの一部として SQL パッケージ・オブジェクトは作成されません。 SQL パッケージの作成 (CRTSQLPKG) コマンドを使用することができます。

relational-database-name: 新しい SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。ローカル・リレーショナル・データベースの名前を指定すると、作成されるプログラムは分散 SQL プログラムになります。 SQL ステートメントはローカル・データベースにアクセスします。

***NONE:** SQL パッケージ・オブジェクトは作成されません。プログラム・オブジェクトは分散プログラムではなく、 SQL パッケージの作成 (CRTSQLPKG) コマンドは使用できません。

USER

会話の開始時にリモート・システムに送られるユーザー名を指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***CURRENT:** 現在のジョブが実行されているユーザー・プロファイルが使用されます。

user-name: アプリケーション・サーバー・ジョブに使用されるユーザー名を指定します。

PASSWORD

リモート・システムで使用されるパスワードを指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***NONE:** パスワードは送られません。この値を指定する場合は、USER(*CURRENT) も指定しなければなりません。

password: USER パラメーターに指定したユーザー名のパスワードを指定します。

RDBCNNMTH

CONNECT ステートメントに使用する意味を指定します。詳細については、「SQL 解説書」を参照してください。

***DUW:** 分散作業単位をサポートするために CONNECT (タイプ 2) の意味が使用されます。追加のリレーショナル・データベースへの連続する CONNECT ステートメントによって、直前の接続が切断されることはありません。

***RUW:** リモート作業単位をサポートするのに CONNECT (タイプ 1) の意味が使用されます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

DFTRDBCOL

テーブル、ビュー、索引および SQL パッケージの非修飾名に使用されるスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用されます。

***NONE:** OPTION パラメーターで定義した命名規則が使用されます。

schema-name: スキーマ識別名を指定します。この値は、OPTION パラメーターで指定した命名規則の代わりに使用されます。

DYNDFTCOL

DFTRDBCOL パラメーター用に指定した省略時スキーマ名が動的ステートメントでも使用されるかどうかを指定します。

***NO:** 動的 SQL ステートメント用のテーブル、ビュー、索引、および SQL パッケージの非修飾名用の DFTRDBCOL パラメーターに指定した値は使用されません。OPTION パラメーターで指定した命名規則が使用されます。

***YES:** DFTRDBCOL パラメーターに指定したスキーマ名は、動的 SQL ステートメント内のテーブル、ビュー、索引、および SQL パッケージの非修飾名用に使用されます。

SQLPKG

このコマンドの RDB パラメーターで指定されたリレーショナル・データベースで作成される SQL パッケージの修飾名を指定します。

指定できるライブラリー値は次のとおりです。

***OBJLIB:** パッケージは、OBJ パラメーターで指定したライブラリーと同じ名前のライブラリー内に作成されます。

library-name: パッケージが作成されるライブラリーの名前を指定します。

***OBJ:** SQL パッケージの名前は、 OBJ パラメーターで指定したオブジェクト名と同じになります。

package-name: SQL パッケージの名前を指定します。リモート・システムが iSeries システムでないときは、8 文字を超える名前は指定できません。

SQLPATH

静的 SQL ステートメント内のプロシージャー、関数、およびユーザー定義タイプを検出するために使用するパスを指定します。

***NAMING:** 使用するパスは、OPTION パラメーターで指定した命名規則に従います。

*SYS 命名の場合、使用するパスは、*LIBL です (実行時の現行ライブラリー・リスト)。

*SQL 命名の場合、使用するパスは、″QSYS″、″QSYS2″、″userid″ です。ただし、″userid″ は、USER 特殊レジスターの値です。schema-name が DFTRDBCOL パラメーターに指定された場合、その schema-name がユーザー ID に置き替わります。

***LIBL:** 使用するパスは、実行時のライブラリー・リストです。

schema-name: 1 つ以上のスキーマ名のリストを指定します。最大 268 の個別スキーマを指定できます。

SQLCURRULE

SQL ステートメントに使用する意味を指定します。

***DB2:** SQL ステートメントのすべての意味が、DB2 について設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数が文字データとして扱われます。

***STD:** SQL ステートメントすべての意味が、ISO および ANSI の SQL 規格によって設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数がバイナリー・データとして扱われます。

SAAFLAG

IBM SQL フラグ付け機能を指定します。このパラメーターは、SQL ステートメントにフラグ付けし、SQL ステートメントが IBM SQL 構文に準拠しているかを検査します。IBM データベース・プロダクトの IBM SQL 構文に関する詳細は、「*DRDA IBM SQL Reference*」にあります。

***NOFLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行いません。

***FLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行います。

FLAGSTD

米国規格協会 (ANSI) のフラグ機能を指定します。このパラメーターは SQL ステートメントにフラグ付けし、ステートメントが次の標準に準拠するかどうかを検査します。

```
ANSI X3.135-1992 entry
ISO 9075-1992 entry
FIPS 127.2 entry
```

***NONE:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行いません。

***ANS:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行います。

DBGVIEW

このパラメーターは、SQL プリコンパイラーが提供する、ソース・デバッグ情報のタイプを指定します。

***NONE:** ソース・プログラム・ビューは生成されません。

***SOURCE:** SQL プリコンパイラーはルート・ステートメントと、必要であれば SQL INCLUDE ステートメントに関するソース・プログラム・ビューを提供します。ビューには、プリコンパイラーが生成したステートメントが入ります。

USRPRF

コンパイル済みプログラム・オブジェクトが実行されるときに使用されるユーザー・プロファイル (プログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトに対して所有する権限を含む) を指定します。プログラム・オブジェクトが使用できるオブジェクトの制御には、プログラム所有者またはプログラム・ユーザーのプロファイルが使用されます。

***NAMING:** ユーザー・プロファイルが命名規則によって判別されます。命名規則が *SQL である場合は、USRPRF(*OWNER) が使用されます。命名規則が *SYS である場合は、USRPRF(*USER) が使用されます。

***USER:** プログラム・オブジェクトを実行するユーザーのプロファイルが使用されます。

***OWNER:** プログラム所有者とプログラム・ユーザーの両方のユーザー・プロファイルがプログラムの実行時に使用されます。

DYNUSRPRF

動的 SQL ステートメントで使用するユーザー・プロファイルを指定します。

***USER:** ローカルな動的 SQL ステートメントは、プログラムのユーザーのプロファイルに基づいて実行されます。分散動的 SQL ステートメントは、SQL パッケージのユーザーのプロファイルに基づいて実行されます。

***OWNER:** ローカルな動的 SQL ステートメントは、プログラムの所有者のプロファイルに基づいて実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のプロファイルに基づいて実行されます。

SRTSEQ

SQL ステートメントの中の文字列比較に使用される分類順序テーブルを指定します。

注: アプリケーション・サーバーが iSeries システム上にない分散アプリケーション、またはリリース・レベルが V2R3M0 より前の分散アプリケーションでは、このパラメーターに *HEX を指定する必要があります。

***JOB:** ジョブの SRTSEQ 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの LANGID 値は、プログラム実行時に検索されます。分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

***HEX:** 分類順序テーブルは使用しません。分類順序を決定するには、文字の 16 進値を使用します。

***LANGIDSHR:** 分類順序テーブルは複数の文字に同じ重みを使用するので、LANGID パラメーターで指定した言語に関連付けられた共用分類順序テーブルになります。

***LANGIDUNQ:** パラメーターで指定した言語用の固有の分類テーブルが使用されます。

テーブル名は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

table-name: 使用する分類順序テーブルの名前を指定します。

LANGID

SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されたときに使用される言語識別コードを指定します。

***JOB:** ジョブの LANGID 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの LANGID 値は、プログラム実行時に検索されます。分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

language-identifier: 言語識別コードを指定します。

OUTPUT

プリコンパイラーのリストを生成するかどうかを指定します。

***NONE:** プリコンパイラーのリストは生成されません。

***PRINT:** プリコンパイラーのリストが生成されます。

PRTFILE

プリコンパイラー印刷出力が送られる印刷装置ファイルの修飾名を指定します。ファイルの長さは 132 バイト以上でなければなりません。レコード長が 132 バイト未満のファイルを指定すると、情報が失われます。

印刷装置ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QSYSPRT: ファイル名の指定がない場合、プリコンパイラーの印刷出力は IBM 提供の印刷装置ファイル QSYSPRT に送られます。

printer-file-name: プリコンパイラー印刷出力が送られる印刷装置ファイルの名前を指定します。

TOSRCFILE

SQL プリコンパイラーが処理した出力ソース・メンバーを含む、ソース・ファイルの修飾名を指定します。指定したソース・ファイルを検出できない場合、プリコンパイラーはそのファイルを作成します。出力メンバーの名前は、SRCMBR パラメーターに指定した名前と同じになります。

指定できるライブラリー値は次のとおりです。

QTEMP: ライブラリー QTEMP が使用されます。

***LIBL:** 指定したファイルがジョブのライブラリー・リストで検索されます。ライブラリー・リストに載せられているどのライブラリーにもファイルが見つからなければ、現行ライブラリー内にファイルが作成されます。

***CURLIB:** ジョブ用の現行ライブラリーが使用されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 出力ソース・ファイルを含むライブラリーの名前を指定します。

***CALC:** 出力ソース・ファイル名は、ソース・ファイルのマージンに基づいて生成されます。この名前は QSQLTxxxxx で、xxxxx はソース・ファイルの幅です。ソース・ファイルのレコード長が 92 以下の場合は、この名前は QSQLTEMP になります。

QSQLTEMP: ソース・ファイル QSQLTEMP が使用されます。

source-file-name: 出力ソース・メンバーを含むソース・ファイルの名前を指定します。

TEXT

プログラムおよびその機能を簡単に記述するテキストを指定します。このパラメーターの詳細については、Information Center の「CL 解説書」の『TEXT パラメーター』を参照してください。

***SRCMBRTXT:** テキストは C プログラムを作成するために使用されるソース・ファイル・メンバーから取られます。データベース・ソース・メンバーのテキストを追加または変更するには、原始ステートメント入力ユーティリティー開始 (STRSEU) コマンドを使用するか、物理ファイル・メンバー追加 (ADDPFM)

コマンドまたは物理ファイル・メンバー変更 (CHGPFM) コマンドのいずれかを使用します。ソース・ファイルがインライン・ファイルまたは装置ファイルの場合は、テキストはブランクになります。

***BLANK:** テキストは指定されません。

'description': 50 文字以下のテキストをアポストロフィで囲んで指定します。

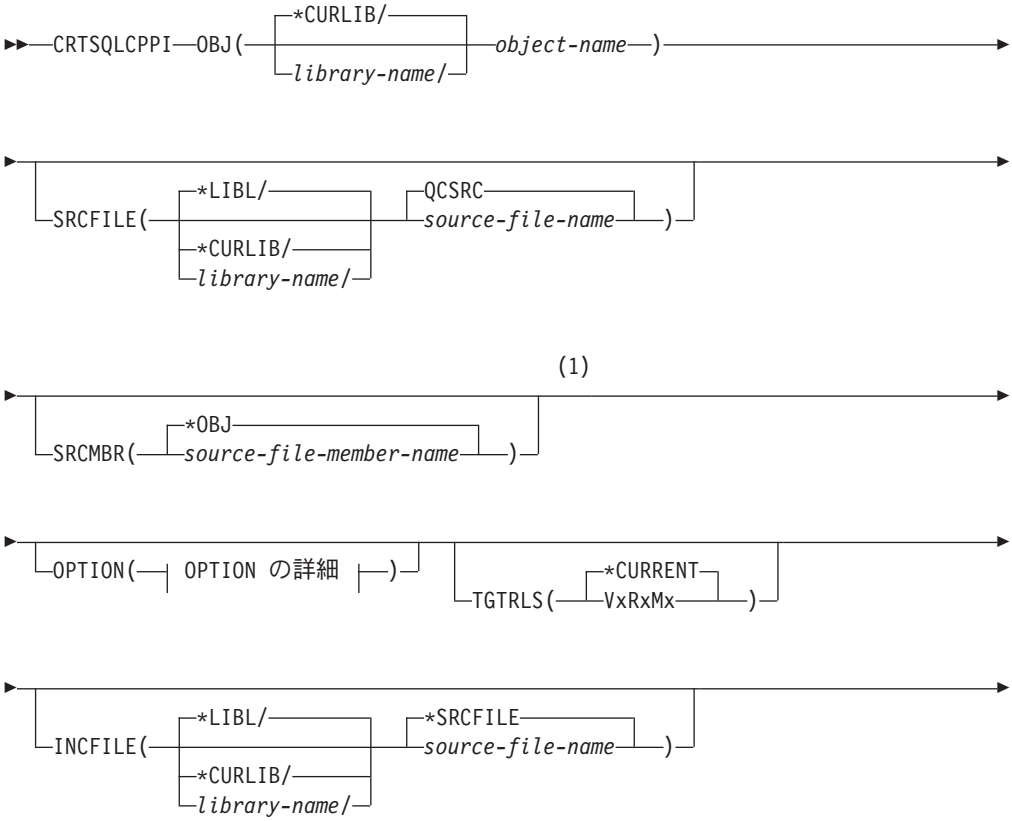
例:

```
CRTSQLCI  PAYROLL  OBJTYPE(*MODULE)
  TEXT('Payroll Program')
```

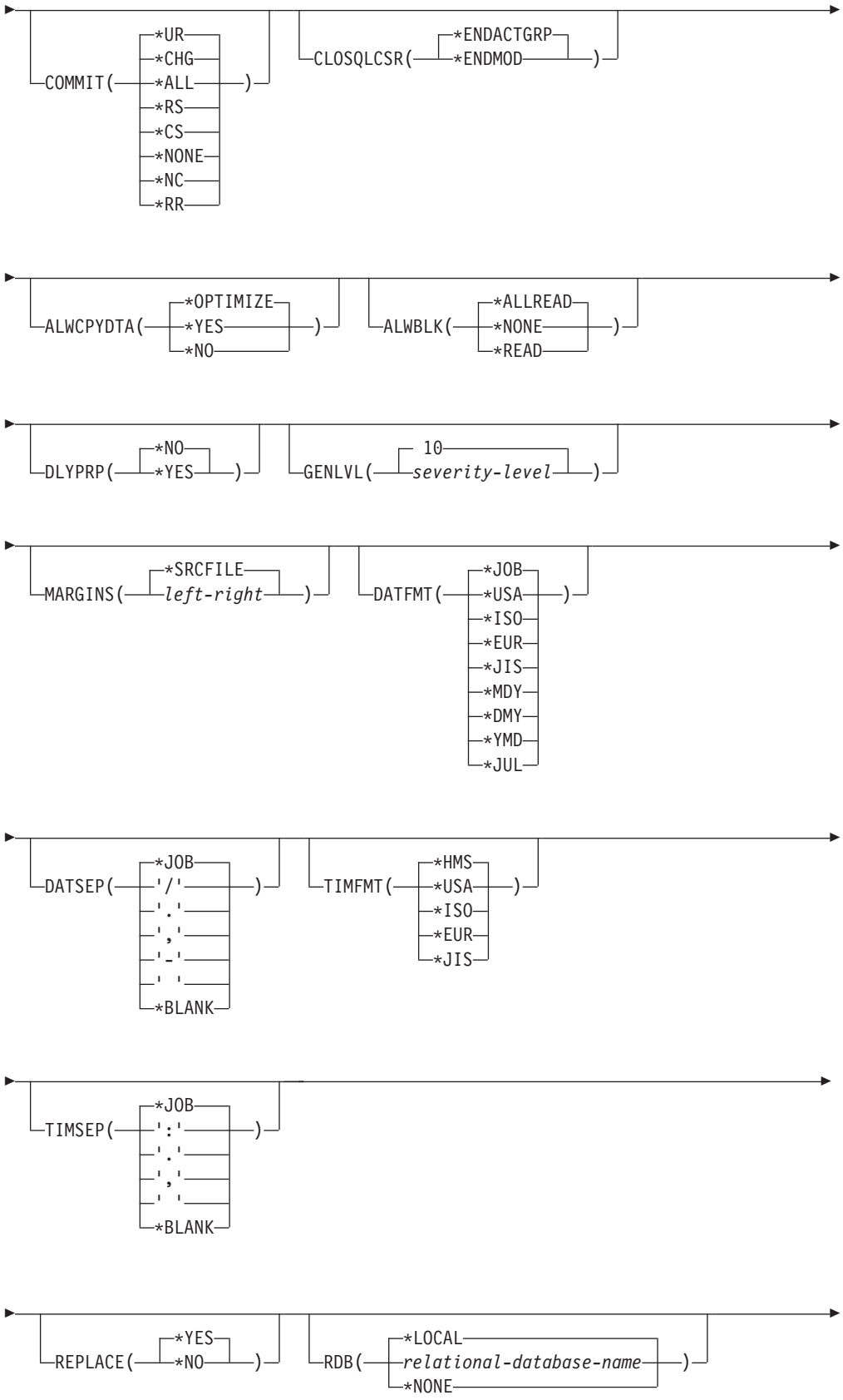
このコマンドは SQL プリコンパイラーを実行させるもので、そのプリコンパイラーはソース・プログラムをプリコンパイルし、変更したソース・プログラムをライブラリー QTEMP のファイル QSQLTEMP 内のメンバー PAYROLL の中に保管します。 SQL プリコンパイラーが作成したソース・メンバーを使用して現行ライブラリーの中でモジュール PAYROLL を作成するために ILE C for iSeries コンパイラーが呼び出されます。

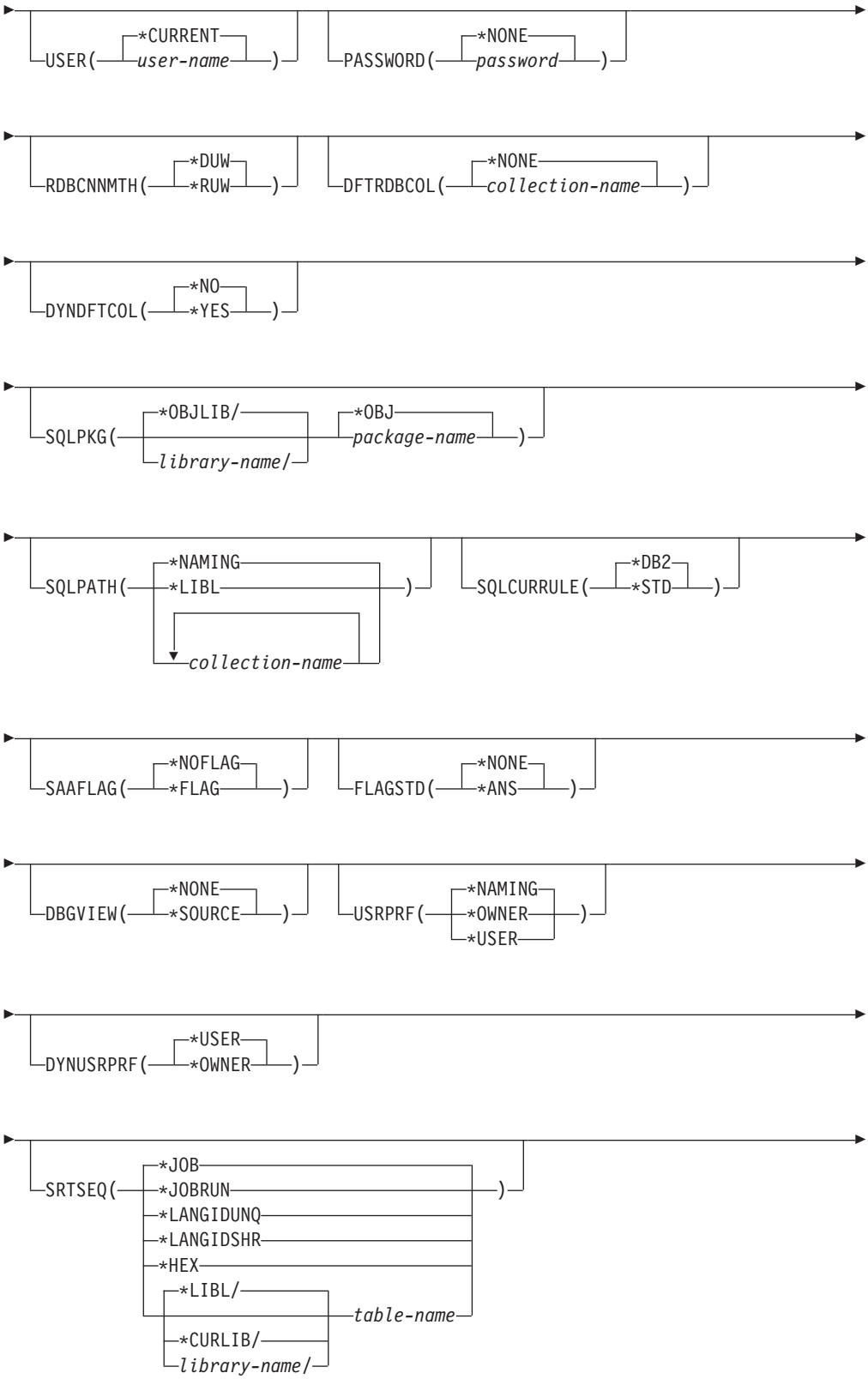
CRTSQLCPPI (SQL C++ オブジェクト作成) コマンド

Job: B,I Pgm: B,I REXX: B,I Exec

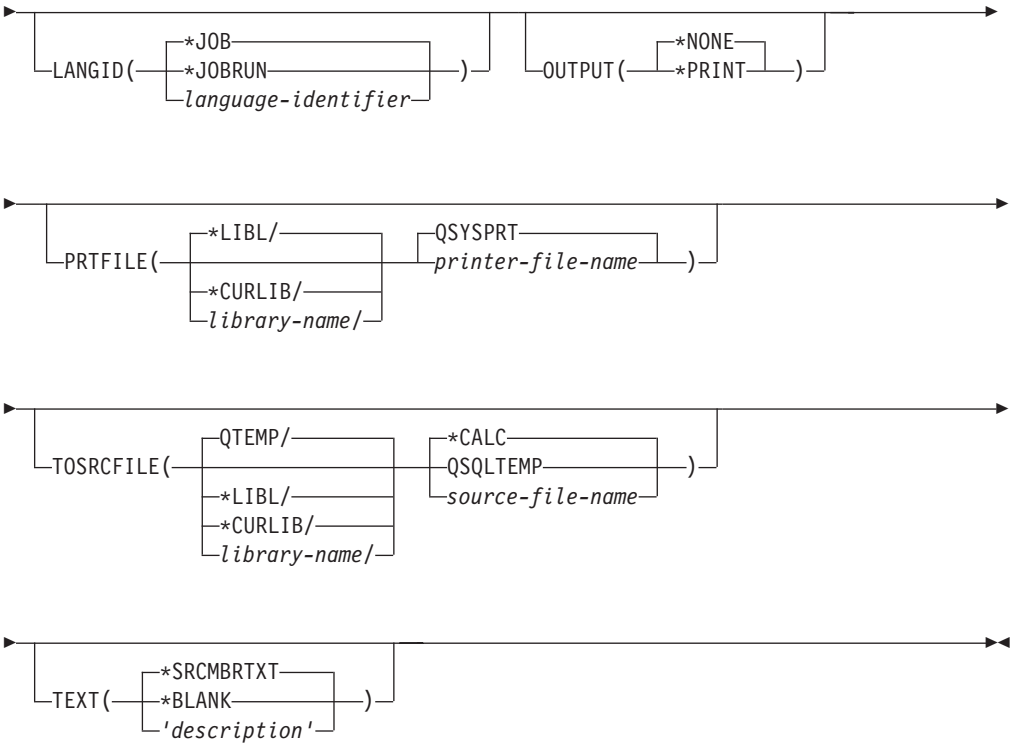


CRTSQLCPPI

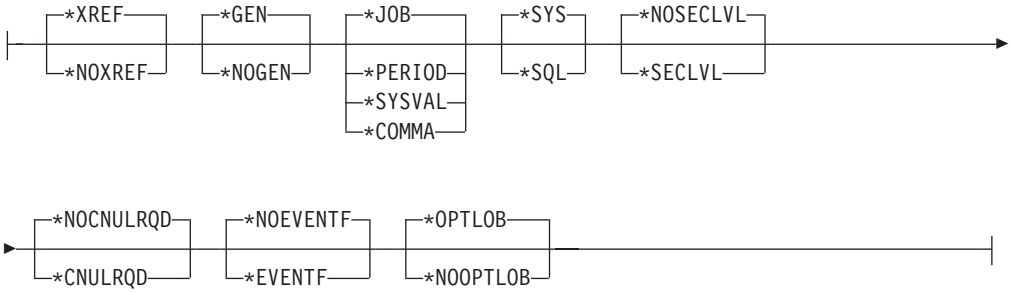

COMMIT(*UR *CHG *ALL *RS *CS *NONE *NC *RR)
CLOSQLCSR(*ENDACTGRP *ENDMOD)
ALWCPYDTA(*OPTIMIZE *YES *NO)
ALWBLK(*ALLREAD *NONE *READ)
DLYPRP(*NO *YES)
GENLVL(10 severity-level)
MARGINS(*SRCFILE left-right)
DATFMT(*JOB *USA *ISO *EUR *JIS *MDY *DMY *YMD *JUL)
DATSEP(*JOB '/' '.' ',' '-' ' ' *BLANK)
TIMFMT(*HMS *USA *ISO *EUR *JIS)
TIMSEP(*JOB ':' '.' ',' ' ' *BLANK)
REPLACE(*YES *NO)
RDB(*LOCAL relational-database-name *NONE)


USER(*CURRENT user-name)
PASSWORD(*NONE password)
RDBCNNMTH(*DUW *RUW)
DFTRDBCOL(*NONE collection-name)
DYNDFTCOL(*NO *YES)
SQLPKG(*OBJLIB/ library-name/ *OBJ package-name)
SQLPATH(*NAMING *LIBL collection-name)
SQLCURRULE(*DB2 *STD)
SAAFLAG(*NOFLAG *FLAG)
FLAGSTD(*NONE *ANS)
DBGVIEW(*NONE *SOURCE)
USRPRF(*NAMING *OWNER *USER)
DYNUSRPRF(*USER *OWNER)
SRTSEQ(*JOB *JOBRUN *LANGIDUNQ *LANGIDSHR *HEX *LIBL/ *CURLIB/ library-name/ table-name)

CRTSQLCPPI



OPTION の詳細:



注:

1 これより前にあるパラメーターはすべて、定位置形式で指定されます。

目的:

SQL C++ オブジェクト作成 (CRTSQLCPPI) コマンドは、構造化照会言語 (SQL) プリコンパイラーを呼び出すものです。この SQL プリコンパイラーは、SQL ステートメントを含む C++ ソースをプリコンパイルし、一時ソース・メンバーを作成し、それから任意で C++ コンパイラーを呼び出してモジュールを作成します。

パラメーター:

OBJ

プリコンパイラーが作成するオブジェクトの修飾名を指定します。

次のライブラリー値のいずれか 1 つで、オブジェクトの名前を修飾することができます。

***CURLIB** ジョブ用の現行ライブラリーでオブジェクトが作成されます。ジョブ用の現行ライブラリーとしてライブラリーを指定しない場合、プリコンパイラーは QGPL ライブラリーを使用します。

library-name: オブジェクトが作成されるライブラリーの名前を指定します。

object-name: プリコンパイラーが作成するオブジェクトの名前を指定します。

SRCFILE

SQL ステートメントとともに C++ ソース・プログラムが入っているソース・ファイルの修飾名を指定します。

次のライブラリー値のいずれか 1 つで、ソース・ファイルの名前を修飾することができます。

***LIBL:** プリコンパイラーは、一致が検出されるまで、ジョブのライブラリー・リストに含まれるすべてのライブラリーを検索します。

***CURLIB:** プリコンパイラーは、ジョブ用の現行ライブラリーを検索します。ジョブ用の現行ライブラリーとしてライブラリーを指定しない場合、プリコンパイラーは QGPL ライブラリーを使用します。

library-name: プリコンパイラーが検索するライブラリーの名前を指定します。

QCSRC: ソース・ファイル名の指定がないと、IBM 提供のソース・ファイル QCSRC には C++ ソース・プログラムが入ります。

source-file-name: C++ ソース・プログラムが入っているソース・ファイルの名前を指定します。

SRCMBR

C++ ソース・プログラムが入っているソース・ファイル・メンバーの名前を指定します。このパラメーターを指定するのは、SRCFILE パラメーター内のソース・ファイル名がデータベース・ファイルである場合だけです。このパラメーターを指定しない場合、プリコンパイラーは OBJ パラメーター上に指定した OBJ 名を使用します。

***OBJ:** OBJ パラメーターに指定したファイルと同じ名前をもつソース・ファイルのメンバーに C++ ソース・プログラムがあることを指定します。

ソース・ファイル・メンバー名: C++ ソースが入っているメンバーの名前を指定します。

OPTION

C++ ソース・プログラムをプリコンパイルするときに次のオプションのうち 1 つ以上が使用されるかどうかを指定します。オプションが 2 度以上使用される場合、または 2 つのオプションが対立する場合は、指定された最後のオプションが使用されます。

要素 1: 相互参照オプション

***XREF:** プリコンパイラーは、プログラムの中の項目と、プログラムの中でそれらの項目を参照しているステートメントの番号との間の相互参照を行います。

***NOXREF:** プリコンパイラーは名前の相互参照を行いません。

要素 2: プログラム作成オプション

***GEN:** プリコンパイラーは、モジュール・オブジェクトを作成します。

***NOGEN:** プリコンパイラーは、C++ コンパイラーを呼び出さず、モジュールも作成しません。

要素 3: 小数点オプション

***JOB:** SQL で数値定数の小数点として使用される値は、プリコンパイル時にジョブ用に指定されている小数点の表現になります。

注: 小数点としてコンマを使用することをジョブで指定する場合、リストの中の (SELECT 文節、VALUES 文節のような) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

***PERIOD:** SQL ステートメントの中の数値定数に小数点として使用する値はピリオドになります。

***COMMA:** SQL ステートメントの中の数値定数に小数点として使用する値はコンマになります。

注: リストの中の (SELECT 文節、VALUES 文節のような) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

要素 4: 命名規則オプション

***SYS:** システム命名規則 (library-name/file-name) を使用します。

***SQL:** SQL の命名規則 (schema-name.table-name) を使用します。 iSeries システム以外のリモート・データベースでパッケージを作成するときは、*SQL を命名規則として指定する必要があります。

要素 5: 2 次レベル・メッセージ・テキスト・オプション

***NOSECLVL:** 2 次レベルのテキスト記述はリストに追加されません。

***SECLVL:** リストのいずれかのメッセージについても、置換データを含む 2 次レベルのテキストが追加されます。

要素 6: NUL 必要オプション

***NOCNULRQD:** 出力文字ホスト変数および出力グラフィック・ホスト変数の場合は、ホスト変数の長さがデータと正確に同じ長さであるとき、 NUL 終了文字は返されません。入力文字ホスト変数および出力グラフィック・ホスト変数には、NUL 終了文字は必要ありません。

***CNULRQD:** 出力文字ホスト変数および出力グラフィック・ホスト変数には必ず NUL 終了文字が入っています。NUL 終了文字用の十分な空間がない場合は、データが切り捨てられ、NUL 終了文字が追加されます。入力文字ホスト変数および出力グラフィック・ホスト変数には、NUL 終了文字が必要です。

要素 7: イベント・ファイル作成

***NOEVENTF:** コンパイラーは、連携開発環境プログラム/400 (CODE/400) が使用するためのイベント・ファイルを作成しません。

***EVENTF:** コンパイラーは、連携開発環境プログラム/400 (CODE/400) が使用するイベント・ファイルを作成します。イベント・ファイルは、ソース・ライブラリー内のファイル EVFEVENT のメンバーとして作成されます。CODE/400 はこのファイルを使用して、CODE/400 編集プログラムに統合されたエラー・フィードバックを提供します。通常 CODE/400 は、ユーザーに代わってこのオプションを指定します。

要素 8: DRDA に対するラージ・オブジェクトの最適化

***OPTLOB:** カーソルに対する最初の FETCH によって、これ以降のすべての FETCH での LOB (ラージ・オブジェクト) に対するカーソルの使用方法が決定されます。このオプションは、カーソルがクローズされるまで、引き続き有効です。

最初の FETCH が LOB ロケーターを使用して LOB 列にアクセスする場合、そのカーソルに対する後続の FETCH がその LOB 列を取り出して LOB ホスト変数に入れることはありません。

最初の FETCH が LOB 列を LOB ホスト変数に置いた場合、そのカーソルに対する後続の FETCH がその列に対して LOB ロケーターを使用することはできません。

***NOOPTLOB:** 列が検索されて LOB ロケーターまたは LOB ホスト変数に入れられるかどうかについての制約はありません。このオプションは、パフォーマンス低下の原因となることがあります。

TGTRLS

作成中のオブジェクトを使用する予定のオペレーティング・システムのリリース・レベルを指定します。

*CURRENT 値の例では、*release-level* 値の場合と同様に、VxRxMx 形式を使用してリリースを指定します。この形式を使用する場合、Vx はバージョン、Rx はリリース、そして Mx はモディフィケーション・レベルを示します。たとえば、V2R3M0 はバージョン 2、リリース 3、モディフィケーション・レベル 0 です。

***CURRENT:** オブジェクトは、ユーザーのシステムで現在稼働しているオペレーティング・システムのリリースで使用されます。たとえば、システムで V2R3M5 が稼働している場合、*CURRENT は、V2R3M5 が導入されたシステムでオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

注: V2R3M5 がシステム上で稼働しており、V2R3M0 が導入されたシステムでオブジェクトが使用される場合、TGTRLS(*CURRENT) ではなく TGTRLS(V2R3M0) を指定してください。

release-level: VxRxMx 形式でリリースを指定してください。指定されたリリースのシステム上、あるいは以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することができます。

有効な値は、現行バージョン、リリース、モディフィケーション・レベルによります。また、有効な値は各新規リリースで変わります。コマンドがサポートする初期リリース・レベルよりもさらに前のリリース・レベルを指定した場合には、エラー・メッセージはこれがサポートする最も初期のリリース・レベルを示して送信されます。

INCFILE

SQL の INCLUDE ステートメントを用いてプログラムに組み込むメンバーが入っているソース・ファイルの修飾名を指定します。

次のライブラリー値のいずれか 1 つで、ソース・ファイルの名前を修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

***SRCFILE:** SRCFILE パラメーターで指定した修飾されたソース・ファイルには、SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入ります。

source-file-name: SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入っているソース・ファイルの名前を指定します。ここに指定するソース・ファイルのレコード長は、SRCFILE パラメーターに指定したソース・ファイルのレコード長より小さくてはなりません。

COMMIT

コンパイル済みの単位内の SQL ステートメントがコミットメント制御下で実行されるかどうかを指定します。ホスト言語ソースの中で参照されているファイルは、このオプションによって影響を受けません。SQL ステートメントの中で参照される SQL テーブル、SQL ビュー、および SQL パッケージだけが影響されます。

***CHG または *UR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができます。

***ALL または *RS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。

***CS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。選択されたが、更新されていない行は、次の行が選択されるまでロックされます。他のジョブにおけるコミットされていない変更は見ることができません。

***NONE または *NC:** コミットメント制御が使用されないことを指定します。他のジョブにおけるコミットされていない変更は見ることができます。プログラムに SQL の DROP SCHEMA ステートメントが組み込まれている場合は、*NONE または *NC を使用しなければなりません。RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries 以外のシステム上にある場合は、*NONE または *NC を指定することはできません。

***RR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。
SELECT、UPDATE、DELETE、および INSERT ステートメントで参照されているすべてのテーブルは、作業単位 (トランザクション) の終わりまで排他的にロックされます。

CLOSQLCSR

SQL カーソルが暗黙にクローズされるとき、SQL 準備済みステートメントが暗黙に破棄され、LOCK TABLE のロックが解除されることを指定します。
CLOSE、COMMIT、または ROLLBACK (HOLD を指定しない) SQL ステートメントを発行すると、SQL カーソルが明示的にクローズされます。

***ENDACTGRP:** 活動化グループが終了した時点で、SQL カーソルがクローズされ、SQL 準備済みステートメントが暗黙に廃棄され、LOCK TABLE のロックが解除されます。

***ENDMOD:** モジュールが終了した時点で、SQL カーソルをクローズし、SQL 準備済みステートメントを暗黙に廃棄します。呼び出しスタック上の最初の SQL プログラムが終了すると、LOCK TABLE のロックが解除されます。

ALWCPYDTA

SELECT ステートメントでデータのコピーが使用できるかどうかを指定します。

***OPTIMIZE:** データベースから直接検索されたデータを使用するか、データのコピーを使用するかどうかは、システムが決定します。決定は、どの方式が最良のパフォーマンスを提供するかに基づいて行われます。 COMMIT が *CHG または *CS で ALWBLK が *ALLREAD でない場合、あるいは COMMIT が *ALL または *RR の場合、データのコピーが使用されるのは、照会を実行する必要があるときだけです。

***YES:** 必要な場合にデータのコピーが使用されます。

***NO:** データのコピーを使用することはできません。照会を実行するのにデータの一時コピーが必要な場合には、エラー・メッセージが返されます。

ALWBLK

データベース・マネージャーがレコードのブロック化を使用できるかどうか、および読み取り専用カーソルについてどの程度までブロック化が使用できるかを指定します。

***ALLREAD:** COMMIT パラメーターで *NONE または *CHG が指定されている場合は、行は読み取り専用カーソルに対してブロックされます。プログラム内で、明示的に更新することができないすべてのカーソルは、プログラム内に EXECUTE または EXECUTE IMMEDIATE ステートメントがある場合でも、読み取り専用にオープンされます。

*ALLREAD を指定すると、

- *READ の場合に許されるブロック化に加え、コミットメント制御レベル *CHG のもとでレコードのブロック化を行うことができます。
- プログラムの中のほとんどすべての読み取り専用カーソルのパフォーマンスを向上できますが、照会が次のように制限されます。
  - ロールバック (ROLLBACK) コマンド、ホスト言語で書いた ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、*ALLREAD の指定があるとき、読み取り専用カーソルの再位置決めはしません。
  - カーソル内の行を更新するために、位置付けされた UPDATE または DELETE 使用ステートメントの動的実行 (たとえば、EXECUTE IMMEDIATE による) をすることはできません。ただし、カーソルに関する DECLARE ステートメントに FOR UPDATE 文節が含まれている場合を除きます。

***NONE:** カーソルの対象となるデータを検索するとき、行がブロック化されません。

*NONE を指定すると、

- 検索されるデータが最新であることが保証されます。
- 照会の対象となるデータの最初の行を検索するときの所要時間が短縮します。
- 照会の最初の数行だけが検索されてから照会がクローズされるときは、プログラムによって使用されないデータ行のブロックの検索をデータベース・マネージャーに中止させます。
- 大量の行を検索する照会全体のパフォーマンスが低下する可能性があります。

***READ:** 次のとき、カーソルの対象となるデータを読み取り専用で検索するときレコードがブロック化されます。

- COMMIT パラメーターに *NONE の指定があるとき。これは、コミットメント制御が使用されないことを示します。
- カーソルが FOR READ ONLY 文節を使用して宣言されているか、あるいは位置指定の UPDATE または DELETE ステートメントをカーソルに対して実行できる動的ステートメントがないとき。

*READ を指定すると、上記条件を満足する照会の全体のパフォーマンスが向上し、大量のレコードを検索することができます。

DLYPRP

PREPARE ステートメントについての動的ステートメント妥当性検査を、OPEN、EXECUTE、または DESCRIBE ステートメントが実行されるまで遅延させるかどうかを指定します。妥当性検査を遅延させると、余分な妥当性検査が除かれるため、パフォーマンスが向上します。

***NO:** 動的ステートメント妥当性検査を遅延させません。動的ステートメントが準備されるとき、アクセス・プランが妥当性検査されます。動的ステートメントが OPEN または EXECUTE ステートメントで使用される場合、アクセス・プランが再度妥当性検査されます。動的ステートメントによって参照されるオブジェクトの権限または存在は変化する場合があるので、OPEN または EXECUTE ステートメントを出した後、SQLCODE または SQLSTATE を検査して、動的ステートメントがまだ有効であるか確かめる必要があります。

***YES:** 動的ステートメント妥当性検査を、動的ステートメントが OPEN、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅延させます。動的ステートメントが使用されたときは、その妥当性検査が行われて、アクセス・プランが作られます。このパラメーターで *YES を指定する場合は、OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後 SQLCODE と SQLSTATE を調べて、動的ステートメントが有効であるかどうかを確かめておく必要があります。

注: *YES を指定したときは、PREPARE ステートメントで INTO 文節が使用されている場合や、OPEN が動的ステートメントに対して出される前に DESCRIBE ステートメントがその動的ステートメントを使用した場合は、パフォーマンスは向上しません。

GENLVL

作成操作が失敗する時の重大度レベルを指定します。重大度レベルがこの値より大きいエラーが発生する場合は、操作が終了します。

10: 省略時の重大度レベルは 10 です。

severity-level: 0 から 40 までの範囲で重大度レベルの値を指定します。

MARGINS

プリコンパイラー入力レコードのうちソース・テキストが入っている部分を指定します。

***SRCFILE:** SRCMBR パラメーターに指定したファイル・メンバーのマージン値が使用されます。

要素 1: 左マージン

left: ステートメントの開始位置を指定します。有効な値は 1 から 32754 までです。

要素 2: 右マージン

right: ステートメントの終了位置を指定します。有効な値は 1 から 32754 までです。

DATFMT

日付結果列にアクセスするときに使用される形式を指定します。すべての出力日

付フィールドは、指定された形式で返されます。入力日付文字列については、日付が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、*USA、*ISO、*EUR、または *JIS を指定しなければなりません。

***JOB:** ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***USA:** 米国の日付形式 (mm/dd/yyyy) が使用されます。

***ISO:** 国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR:** ヨーロッパの日付形式 (dd.mm.yyyy) が使用されます。

***JIS:** 日本工業規格の日付形式 (yyyy-mm-dd) が使用されます。

***MDY:** 日付形式 (mm/dd/yy) が使用されます。

***DMY:** 日付形式 (dd/mm/yy) が使用されます。

***YMD:** 日付形式 (yy/mm/dd) が使用されます。

***JUL:** 年間通算日の日付形式 (yy/ddd) が使用されます。

DATSEP
日付結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブに指定された日付区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

'/': スラッシュ (/) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

'-': ダッシュ (-) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

TIMFMT

時刻結果列にアクセスするときに使用される形式を指定します。入力時刻文字列については、時刻が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力時刻文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、時刻形式は、時刻区切り記号がコロンまたはピリオドである *USA、*ISO、*EUR、*JIS、または *HMS でなければなりません。

***HMS:** **hh:mm:ss** 形式が使用されます。

***USA:** 米国の時刻形式 **hh:mm xx** が使用されます。ただし、**xx** は AM か PM です。

***ISO:** 国際標準化機構 (ISO) の時刻形式 **hh.mm.ss** が使用されます。

***EUR:** ヨーロッパの時刻形式 **hh.mm.ss** が使用されます。

***JIS:** 日本工業規格の時刻形式 **hh:mm:ss** が使用されます。

TIMSEP

時刻結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブのために指定された時刻区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

':': コロン (:) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

REPLACE

同じライブラリーに同じ名前の既存の SQL モジュールがあるときに、 SQL モジュールが作成されるかどうかを指定します。このパラメーターの値は、CRTCPPMOD コマンドに渡されます。

***YES:** 新規の SQL モジュールが作成され、指定したライブラリー内にある同じ名前の既存のオブジェクトはすべて QRPLOBJ に移されます。

***NO:** 指定されたライブラリーに同じ名前のオブジェクトがすでに存在している場合は、新規の SQL モジュールは作成されません。

RDB

SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。

***LOCAL:** プログラムは分散 SQL プログラムとして作成されます。 SQL ステートメントはローカル・データベースにアクセスします。プリコンパイル・プロセスの一部として SQL パッケージ・オブジェクトは作成されません。 SQL パッケージの作成 (CRTSQLPKG) コマンドを使用することができます。

relational-database-name: 新しい SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。ローカル・リレーショナル・データベースの名前を指定すると、作成されるプログラムは分散 SQL プログラムになります。 SQL ステートメントはローカル・データベースにアクセスします。

***NONE:** SQL パッケージ・オブジェクトは作成されません。プログラム・オブジェクトは分散プログラムではなく、 SQL パッケージの作成 (CRTSQLPKG) コマンドは使用できません。

USER

会話の開始時にリモート・システムに送られるユーザー名を指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***CURRENT:** 現在のジョブが実行されているユーザー・プロファイルが使用されます。

user-name: アプリケーション・サーバー・ジョブに使用されるユーザー名を指定します。

PASSWORD

リモート・システムで使用されるパスワードを指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***NONE:** パスワードは送られません。この値を指定する場合は、USER(*CURRENT) も指定しなければなりません。

password: USER パラメーターに指定したユーザー名のパスワードを指定します。

RDBCNNMTH

SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。

***DUW:** 分散作業単位をサポートするために CONNECT (タイプ 2) の意味が使用されます。追加のリレーショナル・データベースへの連続する CONNECT ステートメントによって、直前の接続が切断されることはありません。

***RUW:** リモート作業単位をサポートするのに CONNECT (タイプ 1) の意味が使用されます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

DFTRDBCOL

テーブル、ビュー、索引および SQL パッケージの非修飾名に使用されるスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用されます。

***NONE:** OPTION パラメーターで定義した命名規則が使用されます。

schema-name: スキーマ識別名を指定します。この値は、OPTION パラメーターで指定した命名規則の代わりに使用されます。

DYNDFTCOL

DFTRDBCOL パラメーター用に指定した省略時スキーマ名が動的ステートメントでも使用されるかどうかを指定します。

***NO:** 動的 SQL ステートメント用のテーブル、ビュー、索引、および SQL パッケージの非修飾名用の DFTRDBCOL パラメーターに指定した値は使用されません。OPTION パラメーターで指定した命名規則が使用されます。

***YES:** DFTRDBCOL パラメーターに指定したスキーマ名は、動的 SQL ステートメント内のテーブル、ビュー、索引、および SQL パッケージの非修飾名用に使用されます。

SQLPKG

このコマンドの RDB パラメーターで指定されたリレーショナル・データベースで作成される SQL パッケージの修飾名を指定します。

指定できるライブラリー値は次のとおりです。

***OBJLIB:** パッケージは、OBJ パラメーターで指定したライブラリーと同じ名前のライブラリー内に作成されます。

library-name: パッケージが作成されるライブラリーの名前を指定します。

***OBJ:** SQL パッケージの名前は、 OBJ パラメーターで指定したオブジェクト名と同じになります。

package-name: SQL パッケージの名前を指定します。リモート・システムが iSeries システムでないときは、8 文字を超える名前は指定できません。

SQLPATH

静的 SQL ステートメント内のプロシージャー、関数、およびユーザー定義タイプを検出するために使用するパスを指定します。

***NAMING:** 使用するパスは、OPTION パラメーターで指定した命名規則に従います。

*SYS 命名の場合、使用するパスは、*LIBL です (実行時の現行ライブラリー・リスト)。

*SQL 命名の場合、使用するパスは、″QSYS″、″QSYS2″、″userid″ です。ただし、″userid″ は、USER 特殊レジスターの値です。schema-name が DFTRDBCOL パラメーターに指定された場合、その schema-name がユーザー ID に置き替わります。

***LIBL:** 使用するパスは、実行時のライブラリー・リストです。

schema-name: 1 つ以上のスキーマ名のリストを指定します。最大 268 の個別スキーマを指定できます。

SQLCURRULE

SQL ステートメントに使用する意味を指定します。

***DB2:** SQL ステートメントのすべての意味が、DB2 について設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数が文字データとして扱われます。

***STD:** SQL ステートメントすべての意味が、ISO および ANSI の SQL 規格によって設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数がバイナリー・データとして扱われます。

SAAFLAG

IBM SQL フラグ付け機能を指定します。このパラメーターは、SQL ステートメントにフラグ付けし、SQL ステートメントが IBM SQL 構文に準拠しているかを検査します。IBM データベース・プロダクトの IBM SQL 構文に関する詳細は、「*DRDA IBM SQL Reference*」にあります。

***NOFLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行いません。

***FLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行います。

FLAGSTD

米国規格協会 (ANSI) のフラグ機能を指定します。このパラメーターは SQL ステートメントにフラグ付けし、ステートメントが次の標準に準拠するかどうかを検査します。

```
ANSI X3.135-1992 entry
ISO 9075-1992 entry
FIPS 127.2 entry
```

***NONE:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行いません。

***ANS:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行います。

DBGVIEW

このパラメーターは、SQL プリコンパイラーが提供する、ソース・デバッグ情報のタイプを指定します。

***NONE:** ソース・プログラム・ビューは生成されません。

***SOURCE:** SQL プリコンパイラーはルート・ステートメントと、必要であれば SQL INCLUDE ステートメントに関するソース・プログラム・ビューを提供します。ビューには、プリコンパイラーが生成したステートメントが入ります。

USRPRF

コンパイル済みプログラム・オブジェクトが実行されるときに使用されるユーザー・プロファイル (プログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトに対して所有する権限を含む) を指定します。プログラム・オブジェクトが使用できるオブジェクトの制御には、プログラム所有者またはプログラム・ユーザーのプロファイルが使用されます。

***NAMING:** ユーザー・プロファイルが命名規則によって判別されます。命名規則が *SQL である場合は、USRPRF(*OWNER) が使用されます。命名規則が *SYS である場合は、USRPRF(*USER) が使用されます。

***USER:** プログラム・オブジェクトを実行するユーザーのプロファイルが使用されます。

***OWNER:** プログラム所有者とプログラム・ユーザーの両方のユーザー・プロファイルがプログラムの実行時に使用されます。

DYNUSRPRF
動的 SQL ステートメントで使用するユーザー・プロファイルを指定します。

***USER:** ローカルな動的 SQL ステートメントは、プログラムのユーザーのプロファイルに基づいて実行されます。分散動的 SQL ステートメントは、SQL パッケージのユーザーのプロファイルに基づいて実行されます。

***OWNER:** ローカルな動的 SQL ステートメントは、プログラムの所有者のプロファイルに基づいて実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のプロファイルに基づいて実行されます。

SRTSEQ
SQL ステートメントの中の文字列比較に使用される分類順序テーブルを指定します。

注: アプリケーション・サーバーが iSeries システム上にない分散アプリケーション、またはリリース・レベルが V2R3M0 より前の分散アプリケーションでは、このパラメーターに *HEX を指定する必要があります。

***JOB:** ジョブの SRTSEQ 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの SRTSEQ 値は、プログラム実行時に検索されます。分散アプリケーションの場合、SRTSEQ(*JOBRUN) が有効なのは、LANGID(*JOBRUN) も指定されている場合だけです。

***HEX:** 分類順序テーブルは使用しません。分類順序を決定するには、文字の 16 進値を使用します。

***LANGIDSHR:** 分類順序テーブルは、各文字に固有の重みを使用するので、LANGID パラメーターで指定した言語用の固有の分類テーブルになります。

***LANGIDUNQ:** LANGID パラメーターで指定した言語用の固有の分類テーブルが使用されます。

テーブル名は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

table-name: テーブル名を指定します。

LANGID
SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されたときに使用される言語識別コードを指定します。

***JOB:** ジョブの LANGID 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの LANGID 値は、プログラム実行時に検索されます。分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

language-identifier: 言語識別コードを指定します。

OUTPUT

プリコンパイラーのリストを生成するかどうかを指定します。

***NONE:** プリコンパイラーのリストは生成されません。

***PRINT:** プリコンパイラーのリストが生成されます。

PRTFILE

プリコンパイラー印刷出力が送られる印刷装置ファイルの修飾名を指定します。ファイルの長さは 132 バイト以上でなければなりません。レコード長が 132 バイト未満のファイルを指定すると、情報が失われます。

印刷装置ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QSYSPRT: ファイル名の指定がない場合、プリコンパイラーの印刷出力は IBM 提供の印刷装置ファイル QSYSPRT に送られます。

printer-file-name: プリコンパイラー印刷出力が送られる印刷装置ファイルの名前を指定します。

TOSRCFILE

SQL プリコンパイラーによって処理された出力ソース・メンバーを含む、ソース・ファイルの修飾名を指定します。指定したソース・ファイルが見つからない場合、作成されます。出力メンバーの名前は、SRCMBR パラメーターに指定した名前と同じになります。

指定できるライブラリー値は次のとおりです。

QTEMP: ライブラリー QTEMP が使用されます。

***LIBL:** 指定したファイルがジョブのライブラリー・リストで検索されます。ライブラリー・リストに載せられているどのライブラリーにもファイルが見つからなければ、現行ライブラリー内にファイルが作成されます。

***CURLIB:** ジョブ用の現行ライブラリーが使用されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 出力ソース・ファイルを含むライブラリーの名前を指定します。

***CALC:** 出力ソース・ファイル名は、ソース・ファイルのマージンに基づいて生成されます。この名前は QSQLTxxxxx で、xxxxx はソース・ファイルの幅です。ソース・ファイルのレコード長が 92 以下の場合は、この名前は QSQLTEMP になります。

QSQLTEMP: ソース・ファイル QSQLTEMP が使用されます。

source-file-name: 出力ソース・メンバーを含むソース・ファイルの名前を指定します。

TEXT

プログラムおよびその機能を簡単に記述するテキストを指定します。このパラメーターの詳細については、Information Center の「CL 解説書」の『TEXT パラメーター』を参照してください。

***SRCMBRTXT:** テキストは C++ プログラムを作成するために使用されるソース・ファイル・メンバーから取られます。データベース・ソース・メンバーのテキストを追加または変更するときには、原始ステートメント入力ユーティリティー開始 (STRSEU) コマンドを使用できます。さらに、物理ファイル・メンバー追加 (ADDPFM) または物理ファイル・メンバー変更 (CHGPFM) コマンドを使用することもできます。ソース・ファイルがインライン・ファイルまたは装置ファイルの場合は、テキストはブランクになります。

***BLANK:** テキストは指定されません。

'description': 50 文字以下のテキストをアポストロフィで囲んで指定します。

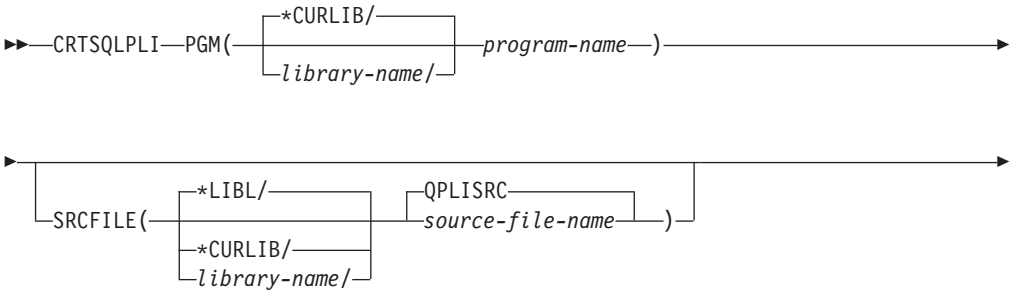
例:

```
CRTSQLCPPI  PAYROLL  OBJTYPE(*MODULE)
  TEXT('Payroll Program')
```

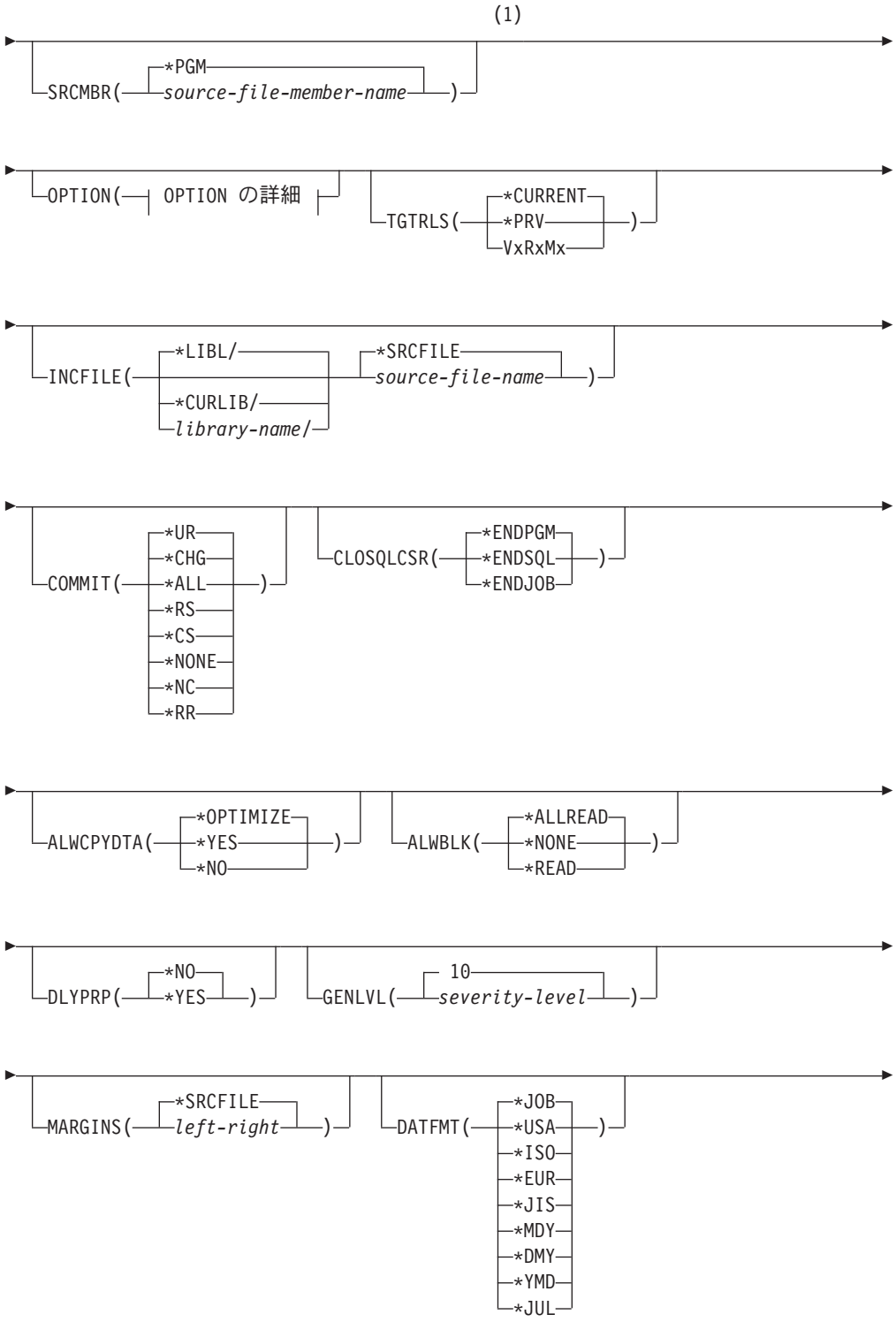
このコマンドは SQL プリコンパイラーを実行させるもので、そのプリコンパイラーはソース・プログラムをプリコンパイルし、変更したソース・プログラムをライブラリー QTEMP のファイル QSQLTEMP 内のメンバー PAYROLL の中に保管します。このコマンドは、ILE C++ コンパイラーを呼び出してから、 SQL プリコンパイラーが作成したソース・メンバーを使用することにより、現行ライブラリーの中でモジュール PAYROLL を作成します。

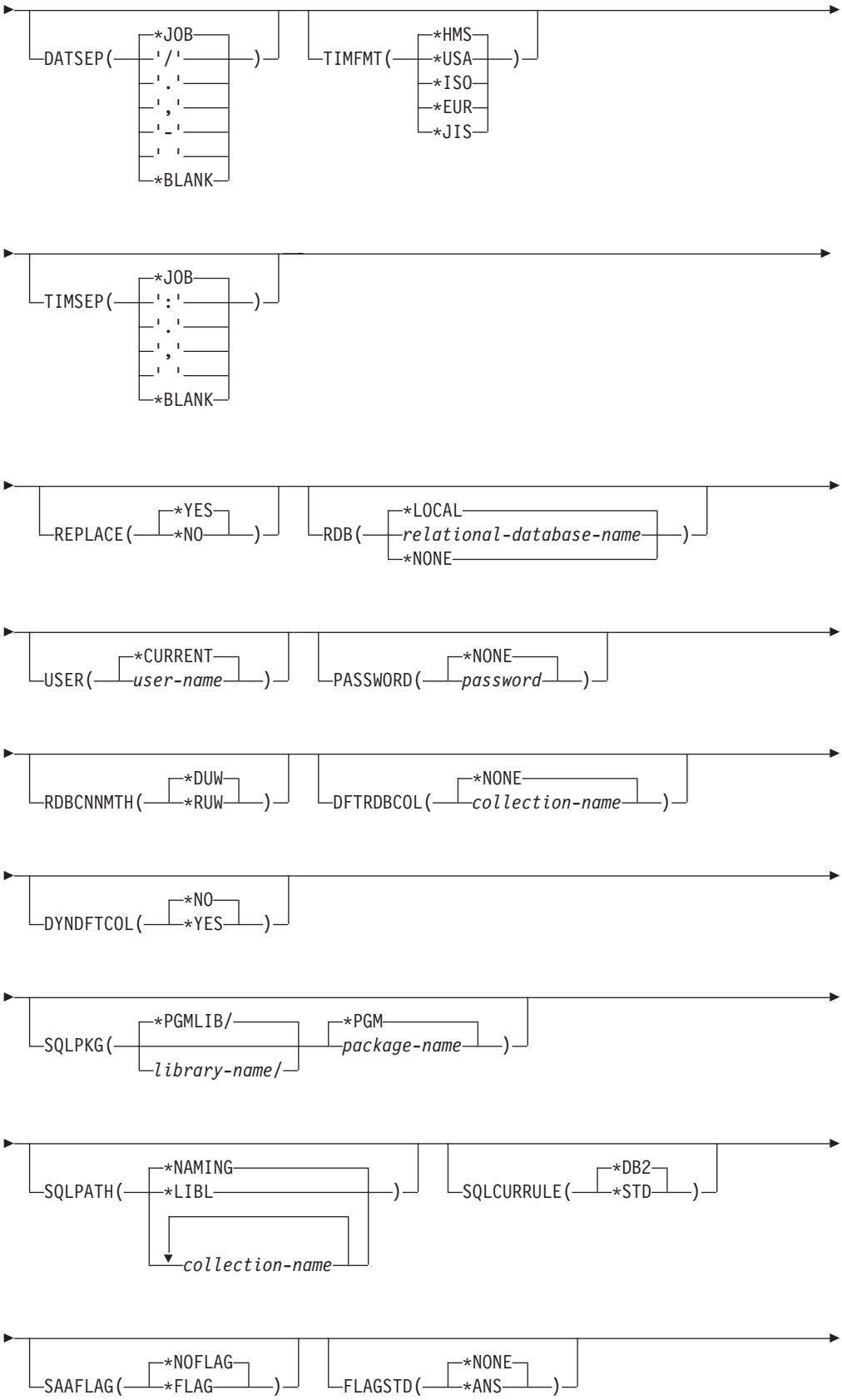
CRTSQLPLI (SQL PL/I 作成) コマンド

Job: B,I Pgm: B,I REXX: B,I Exec

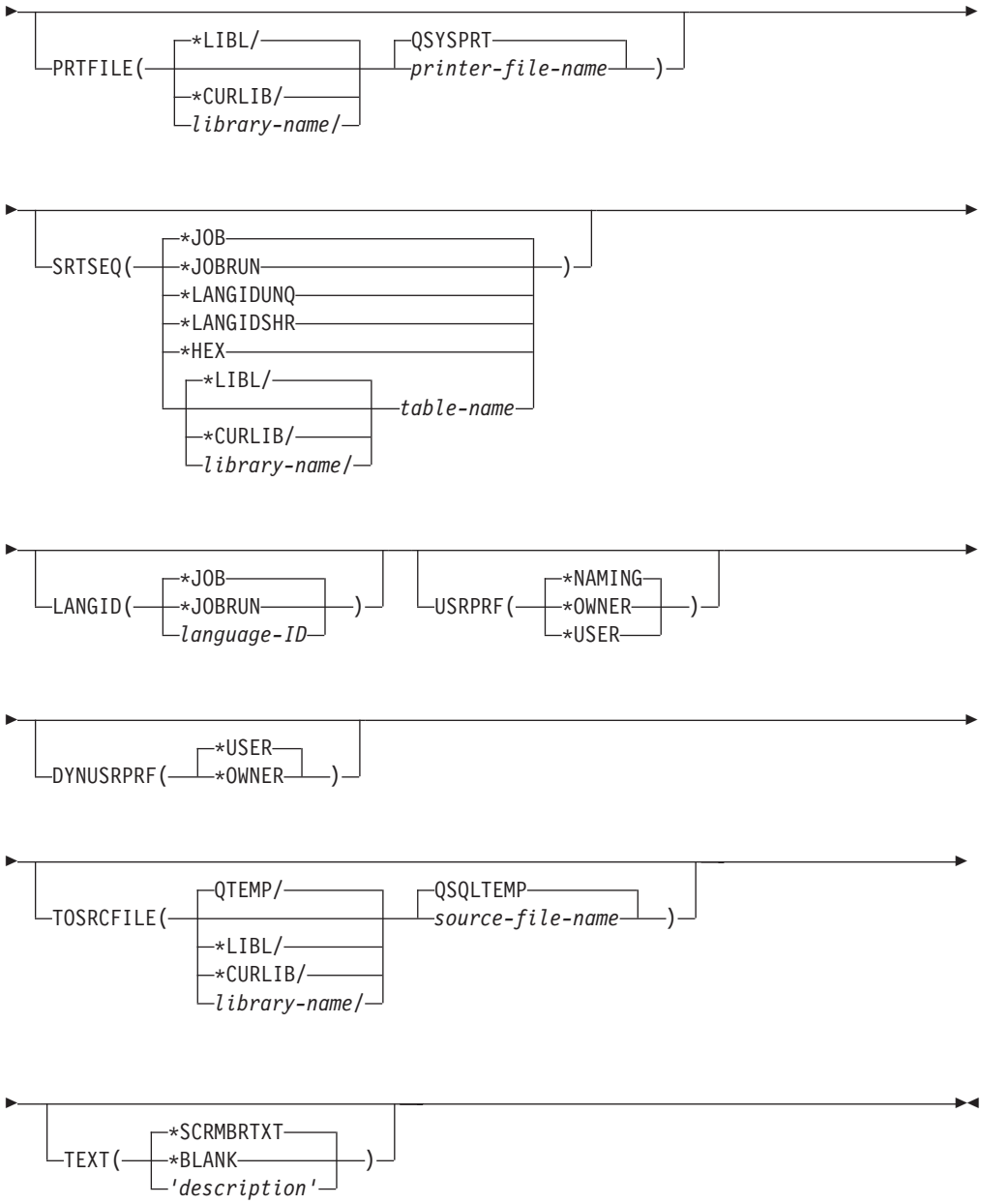


CRTSQLPLI

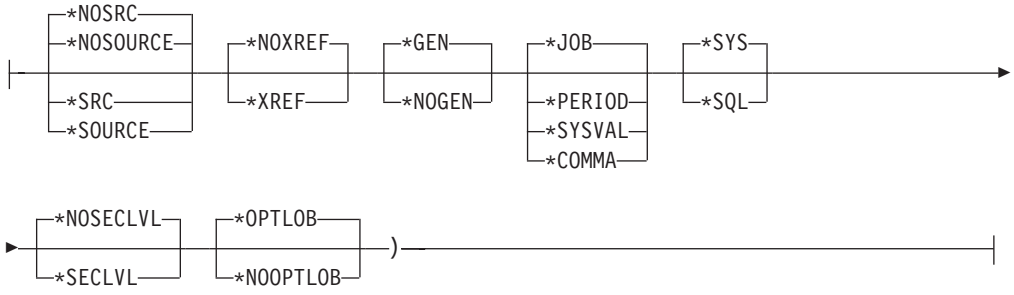

(1)
SRCMBR(
*PGM
source-file-member-name
)
OPTION(
OPTION の詳細
TGTRLS(
*CURRENT
*PRV
VxRxMx
)
INCFILE(
*LIBL/
*CURLIB/
library-name/
*SRCFILE
source-file-name
)
COMMIT(
*UR
*CHG
*ALL
*RS
*CS
*NONE
*NC
*RR
)
CLOSQLCSR(
*ENDPGM
*ENDSQL
*ENDJOB
)
ALWCPYDTA(
*OPTIMIZE
*YES
*NO
)
ALWBLK(
*ALLREAD
*NONE
*READ
)
DLYPRP(
*NO
*YES
)
GENLVL(
10
severity-level
)
MARGINS(
*SRCFILE
left-right
)
DATFMT(
*JOB
*USA
*ISO
*EUR
*JIS
*MDY
*DMY
*YMD
*JUL
)


DATSEP(*JOB '/' '.' ',' '-' ' ' *BLANK)
TIMFMT(*HMS *USA *ISO *EUR *JIS)
TIMSEP(*JOB ':' '.' ',' ' ' *BLANK)
REPLACE(*YES *NO)
RDB(*LOCAL relational-database-name *NONE)
USER(*CURRENT user-name)
PASSWORD(*NONE password)
RDBCNNMTH(*DUW *RUW)
DFTRDBCOL(*NONE collection-name)
DYNDFTCOL(*NO *YES)
SQLPKG(*PGMLIB/ library-name/ *PGM package-name)
SQLPATH(*NAMING *LIBL collection-name)
SQLCURRULE(*DB2 *STD)
SAAFLAG(*NOFLAG *FLAG)
FLAGSTD(*NONE *ANS)

CRTSQLPLI



OPTION の詳細:



注:

1 これより前にあるパラメーターはすべて、定位置形式で指定されます。

目的:

SQL PL/I 作成 (CRTSQLPLI) コマンドは構造化照会言語 (SQL) プリコンパイラーを呼び出すもので、このプリコンパイラーが SQL ステートメントを含む PL/I ソース・プログラムをプリコンパイルし、一時ソース・メンバーを作成してから、任意選択で PL/I コンパイラーを呼び出してプログラムのコンパイルを行います。

パラメーター:

PGM

コンパイルしたプログラムの修飾名を指定します。

コンパイルした PL/I プログラムの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: コンパイルした PL/I プログラムが作成されるライブラリーの名前を指定します。

program-name: コンパイルしたプログラムの名前を指定します。

SRCFILE

SQL ステートメントとともに PL/I ソース・プログラムが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QPLISRC: ソース・ファイル名の指定がないときは、IBM 提供のソース・ファイル QPLISRC には PL/I ソース・プログラムが入ります。

source-file-name: PL/I ソース・プログラムが入っているソース・ファイルの名前を指定します。

SRCMBR

PL/I ソース・プログラムが入っているソース・ファイル・メンバーの名前を指定します。このパラメーターは、SRCFILE パラメーターのソース・ファイル名がデータベース・ファイルである場合にのみ指定します。このパラメーターが指定されない場合は、PGM パラメーターに指定された PGM 名が使用されます。

***PGM:** PGM パラメーターに指定した名前と同じ名前をもつソース・ファイルのメンバーに PL/I ソース・プログラムがあることを指定します。

source-file-member-name: PL/I ソースが入っているメンバーの名前を指定します。

OPTION

PL/I ソース・プログラムをプリコンパイルするときに次のオプションのうち 1 つ以上が使用されるかどうかを指定します。オプションが 2 度以上使用される場合、または 2 つのオプションが対立する場合は、指定された最後のオプションが使用されます。

要素 1: ソース・リスト・オプション

***NOSOURCE:** または ***NOSRC:** プリコンパイルまたはパッケージ作成時にエラーが検出されなければ、プリコンパイラーによるソース印刷出力は行われません。

***SOURCE または *SRC:** プリコンパイラーは、PL/I ソース入力から成るソース印刷出力を作成します。

要素 2: 相互参照オプション

***NOXREF:** プリコンパイラーは名前の相互参照を行いません。

***XREF:** プリコンパイラーは、プログラムの中の項目と、プログラムの中でそれらの項目を参照しているステートメントの番号との間の相互参照を行います。

要素 3: プログラム作成オプション

***GEN:** コンパイラーはコンパイル後に実行できるプログラムを作成します。リレーショナル・データベース名が RDB パラメーターで指定されている場合、SQL パッケージ・オブジェクトが作成されます。

***NOGEN:** プリコンパイラーは C コンパイラーを呼び出さないので、プログラムと SQL パッケージは作成されません。

要素 4: 小数点オプション

***JOB:** SQL で数値定数の小数点として使用される値は、プリコンパイル時にジョブ用に指定されている小数点の表現になります。

***PERIOD:** SQL ステートメントの中の数値定数に小数点として使用する値はピリオドになります。

***SYSVAL:** SQL ステートメントの中の数値定数に小数点として使用する値は QDECFMT システム値になります。

注: QDECFMT が小数点として使用する値をコンマにすることを指定している場合は、リストの中の (SELECT 文節、VALUES 文節などのように) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

***COMMA:** SQL ステートメントの中の数値定数に小数点として使用する値はコンマになります。

注: リストの中の (SELECT 文節、VALUES 文節のような) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

要素 5: 命名規則オプション

***SYS:** システム命名規則 (library-name/file-name) を使用します。

***SQL:** SQL の命名規則 (schema-name.table-name) を使用します。iSeries 以外のリモート・データベース上でプログラムを作成するときは、命名規則として *SQL を指定しなければなりません。

要素 6: 2 次レベル・メッセージ・テキスト・オプション

***NOSECLVL:** 2 次レベルのテキスト記述はリストに追加されません。

***SECLVL:** リストのいずれかのメッセージについても、置換データを含む 2 次レベルのテキストが印刷出力に追加されます。

要素 7: DRDA オプションに対するラージ・オブジェクトの最適化

***OPTLOB:** カーソルに対する最初の FETCH によって、これ以降のすべての FETCH での LOB (ラージ・オブジェクト) に対するカーソルの使用方法が決定されます。このオプションは、カーソルがクローズされるまで、引き続き有効です。

最初の FETCH が LOB ロケーターを使用して LOB 列にアクセスする場合、そのカーソルに対する後続の FETCH がその LOB 列を取り出して LOB ホスト変数に入れることはありません。

最初の FETCH が LOB 列を LOB ホスト変数に置いた場合、そのカーソルに対する後続の FETCH がその列に対して LOB ロケーターを使用することはできません。

***NOOPTLOB:** 列が検索されて LOB ロケーターまたは LOB ホスト変数に入れられるかどうかについての制約はありません。このオプションは、パフォーマンス低下の原因となることがあります。

TGTRLS

作成中のオブジェクトを使用するオペレーティング・システムのリリース・レベルを指定します。

*CURRENT 値および *PRV 値の例では、VxRxMx 形式でリリースを指定する *release-level* 値が示されています。ここで Vx はバージョン、Rx はリリース、Mx はモディフィケーション・レベルです。たとえば、V2R3M0 はバージョン 2、リリース 3、モディフィケーション・レベル 0 です。

***CURRENT:** オブジェクトは、ユーザーのシステムで現在稼働しているオペレーティング・システムのリリースで使用されます。たとえば、V2R3M5 がシステムで稼働している場合、*CURRENT はユーザーが V2R3M5 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

注: V2R3M5 がシステム上で稼働しており、V2R3M0 が導入されたシステムでオブジェクトが使用される場合、TGTRLS(*CURRENT) ではなく TGTRLS(V2R3M0) を指定してください。

***PRV:** オブジェクトはオペレーティング・システムのモディフィケーション・レベル 0 の前のリリースで使用されます。たとえば、 V2R3M5 がユーザーのシステムで稼働している場合、*PRV はユーザーが V2R2M0 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

release-level: VxRxMx 形式でリリースを指定してください。指定されたリリースのシステム上、あるいは以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することができます。

有効な値は、現行バージョン、リリース、モディフィケーション・レベルによります。また、有効な値は各新規リリースで変わります。コマンドがサポートする初期リリース・レベルよりもさらに前のリリース・レベルを指定した場合には、エラー・メッセージはこれがサポートする最も初期のリリース・レベルを示して送信されます。

INCFILE

SQL の INCLUDE ステートメントを用いてプログラムに組み込むメンバーが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

***SRCFILE:** SRCFILE パラメーターで指定した修飾されたソース・ファイルには、SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入ります。

source-file-name: SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入っているソース・ファイルの名前を指定します。指定するソース・ファイルのレコード長は、SRCFILE パラメーターに指定したソース・ファイルのレコード長より小さくてはなりません。

COMMIT

コンパイル済みプログラム内の SQL ステートメントがコミットメント制御のもとで実行されるかどうかを指定します。ホスト言語ソースの中で参照されているファイルは、このオプションによって影響を受けません。SQL ステートメントの中で参照される SQL テーブル、SQL ビュー、および SQL パッケージだけが影響されます。

***CHG または *UR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、 RENAME、および REVOKE ステートメントで参照されるオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができます。

***ALL または *RS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、 RENAME、および REVOKE ステートメントで参照されるオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。

***CS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。選択されたが、更新されていない行は、次の行が選択されるまでロックされます。他のジョブにおけるコミットされていない変更は見ることができません。

***NONE または *NC:** コミットメント制御が使用されないことを指定します。他のジョブにおけるコミットされていない変更は見ることができます。プログラムに SQL の DROP SCHEMA ステートメントが組み込まれている場合は、*NONE または *NC を使用しなければなりません。RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries 以外のシステム上にある場合は、*NONE または *NC を指定することはできません。

***RR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、 RENAME、および REVOKE ステートメントで参照されているオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。
SELECT、UPDATE、DELETE、および INSERT ステートメントで参照されているすべてのテーブルは、作業単位 (トランザクション) の終わりまで排他的にロックされます。

CLOSQLCSR

SQL カーソルが暗黙にクローズされるとき、SQL 準備済みステートメントが暗黙に破棄され、LOCK TABLE のロックが解除されることを指定します。
CLOSE、COMMIT、または ROLLBACK (HOLD を指定しない) SQL ステートメントを発行すると、SQL カーソルが明示的にクローズされます。

***ENDPGM:** プログラムが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが破棄されます。呼び出しスタック上の最初の SQL プログラムが終了すると、LOCK TABLE のロックが解除されます。

***ENDSQL:** SQL カーソルは呼び出しと次の呼び出しの間はオープンしたままで、別の SQL OPEN を実行せずに取り出すことができます。呼び出しスタック上でより高い位置にあるプログラムの 1 つは、少なくとも 1 つの SQL ステートメントを実行しているはずです。呼び出しスタック上の最初の SQL プログラムが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが廃棄され、LOCK TABLE のロックが解除されます。呼び出された最初の SQL プログラム (呼び出しスタック上の最初の SQL プログラム) であるプログラムに *ENDSQL が指定される場合、プログラムは *ENDPGM が指定されたかのように扱われます。

***ENDJOB:** SQL カーソルは呼び出しと次の呼び出しの間はオープンしたままで、別の SQL OPEN を実行せずに取り出すことができます。呼び出しスタック上でより高い位置にあるプログラムは、SQL ステートメントを実行させる必要

がありません。呼び出しスタック上の最初の SQL プログラムが終了するときに、SQL カーソルはオープンのままになり、SQL 準備済みステートメントが保持され、LOCK TABLE のロックが保留されます。ジョブが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが廃棄され、LOCK TABLE のロックが解除されます。

ALWCPYDTA

SELECT ステートメントでデータのコピーが使用できるかどうかを指定します。

***OPTIMIZE:** データベースから直接検索されたデータを使用するか、データのコピーを使用するかどうかは、システムが決定します。決定は、どの方式が最良のパフォーマンスを提供するかに基づいて行われます。 COMMIT が *CHG または *CS で ALWBLK が *ALLREAD でない場合、あるいは COMMIT が *ALL または *RR の場合、データのコピーが使用されるのは、照会を実行する必要があるときだけです。

***YES:** 必要な場合にデータのコピーが使用されます。

***NO:** データのコピーを使用することはできません。照会を実行するのにデータの一時コピーが必要な場合には、エラー・メッセージが返されます。

ALWBLK

データベース・マネージャーがレコードのブロック化を使用できるかどうか、および読み取り専用カーソルについてどの程度までブロック化が使用できるかを指定します。

***ALLREAD:** COMMIT パラメーターで *NONE または *CHG が指定されている場合は、行は読み取り専用カーソルに対してブロックされます。プログラム内で、明示的に更新することができないすべてのカーソルは、プログラム内に EXECUTE または EXECUTE IMMEDIATE ステートメントがある場合でも、読み取り専用にオープンされます。

*ALLREAD を指定すると、

- *READ の場合に許されるブロック化に加え、コミットメント制御レベル *CHG のもとでレコードのブロック化を行うことができます。
- プログラムの中のほとんどすべての読み取り専用カーソルのパフォーマンスを向上できますが、照会が次のように制限されます。
  - ロールバック (ROLLBACK) コマンド、ホスト言語で書いた ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、*ALLREAD の指定があるとき、読み取り専用カーソルの再位置決めはしません。
  - カーソル内の行を更新するために、位置付けされた UPDATE または DELETE 使用ステートメントの動的実行 (たとえば、EXECUTE IMMEDIATE による) をすることはできません。ただし、カーソルに関する DECLARE ステートメントに FOR UPDATE 文節が含まれている場合を除きます。

***NONE:** カーソルの対象となるデータを検索するとき、行がブロック化されません。

*NONE を指定すると、

- 検索されるデータが最新であることが保証されます。
- 照会の対象となるデータの最初の行を検索するときの所要時間が短縮します。
- 照会の最初の数行だけが検索されてから照会がクローズされるときは、プログラムによって使用されないデータ行のブロックの検索をデータベース・マネージャーに中止させます。
- 大量の行を検索する照会全体のパフォーマンスが低下する可能性があります。

***READ:** 次のとき、カーソルの対象となるデータを読み取り専用で検索するときレコードがブロック化されます。

- COMMIT パラメーターに *NONE の指定があるとき。これは、コミットメント制御が使用されないことを示します。
- カーソルが FOR READ ONLY 文節を使用して宣言されているか、あるいは位置指定の UPDATE または DELETE ステートメントをカーソルに対して実行できる動的ステートメントがないとき。

*READ を指定すると、上記条件を満足する照会の全体のパフォーマンスが向上し、大量のレコードを検索することができます。

DLYPRP

PREPARE ステートメントについての動的ステートメント妥当性検査を、OPEN、EXECUTE、または DESCRIBE ステートメントが実行されるまで遅延させるどうかを指定します。妥当性検査を遅延させると、余分な妥当性検査が除かれるため、パフォーマンスが向上します。

***NO:** 動的ステートメント妥当性検査を遅延させません。動的ステートメントが準備されるとき、アクセス・プランが妥当性検査されます。動的ステートメントが OPEN または EXECUTE ステートメントで使用される場合、アクセス・プランが再度妥当性検査されます。動的ステートメントによって参照されるオブジェクトの権限または存在は変化する場合があるので、OPEN または EXECUTE ステートメントを出した後、SQLCODE または SQLSTATE を検査して、動的ステートメントがまだ有効であるか確かめる必要があります。

***YES:** 動的ステートメント妥当性検査を、動的ステートメントが OPEN、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅延させます。動的ステートメントが使用されたときは、その妥当性検査が行われて、アクセス・プランが作られます。このパラメーターで *YES を指定する場合は、OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後 SQLCODE と SQLSTATE を調べて、動的ステートメントが有効であるかどうかを確かめておく必要があります。

注: *YES を指定したときは、PREPARE ステートメントで INTO 文節が使用されている場合や、OPEN が動的ステートメントに対して出される前に DESCRIBE ステートメントがその動的ステートメントを使用した場合は、パフォーマンスは向上しません。

GENLVL

プログラムが変更されないエラーの重大度レベルを指定します。プログラム内で

SQL ステートメントを準備しているときに、このパラメーターで指定された値と等しいかまたはより大きい重大度レベルのエラーが発生すると、プログラムは変更されません。

10: 省略時の重大度レベルは 10 です。

severity-level: 0 から 40 までの範囲で重大度レベルの値を指定します。

MARGINS

プリコンパイラー入力レコードのうちソース・テキストが入っている部分を指定します。

***SRCFILE:** SRCMBR パラメーターに指定したファイル・メンバーのマージン値が使用されます。メンバーが SQLPLI ソース・タイプのときは、マージン値は SEU サービス画面で指定した値になります。メンバーが別のソース・タイプである場合、マージン値は省略時値である 2 と 72 になります。

要素 1: 左マージン

left: ステートメントの開始位置を指定します。有効な値は 1 から 80 までです。

要素 2: 右マージン

right: ステートメントの終了位置を指定します。有効な値は 1 から 80 までです。

DATFMT

日付結果列にアクセスするときに使用される形式を指定します。すべての出力日付フィールドは、指定された形式で返されます。入力日付文字列については、日付が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、*USA、*ISO、*EUR、または *JIS を指定しなければなりません。

***JOB:** ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***USA:** 米国の日付形式 (mm/dd/yyyy) が使用されます。

***ISO:** 国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR:** ヨーロッパの日付形式 (dd.mm.yyyy) が使用されます。

***JIS:** 日本工業規格の日付形式 (yyyy-mm-dd) が使用されます。

***MDY:** 日付形式 (mm/dd/yy) が使用されます。

***DMY:** 日付形式 (dd/mm/yy) が使用されます。

***YMD:** 日付形式 (yy/mm/dd) が使用されます。

***JUL:** 年間通算日の日付形式 (yy/ddd) が使用されます。

DATSEP
日付結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブに指定された日付区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

'/': スラッシュ (/) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

'-': ダッシュ (-) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

TIMFMT
時刻結果列にアクセスするときに使用される形式を指定します。入力時刻文字列については、時刻が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、時刻形式は、時刻区切り記号がコロンまたはピリオドである *USA、*ISO、*EUR、*JIS、または *HMS でなければなりません。

***HMS:** (hh:mm:ss) 形式が使用されます。

***USA:** 米国の時刻形式 (hh:mm *xx*) が使用されます。ただし、*xx* は AM か PM です。

***ISO:** 国際標準化機構 (ISO) の時刻形式 (hh.mm.ss) が使用されます。

***EUR:** ヨーロッパの時刻形式 (hh.mm.ss) が使用されます。

***JIS:** 日本工業規格の時刻形式 (hh:mm:ss) が使用されます。

TIMSEP
時刻結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブのために指定された時刻区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

':': コロン (:) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

REPLACE

同じライブラリーに同じ名前のプログラムまたは SQL パッケージが存在するときに新しいプログラムまたは SQL パッケージが作成されるかどうかを指定します。このパラメーターの値は、CRTPLIPGM コマンドに渡されます。このパラメーターについての詳細は、「CL 解説書」の『付録 A. 拡張パラメーター記述』にあります。

***YES:** 新しいプログラムまたは SQL パッケージが作成され、指定したライブラリーの中の、同じ名前とタイプを持つ既存のプログラムまたは SQL パッケージは QRPLOBJ に移されます。

***NO:** 指定されたライブラリーに同じ名前およびタイプのオブジェクトがすでに存在している場合は、新規のプログラムまたは SQL パッケージは作成されません。

RDB

SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。

***LOCAL:** プログラムは分散 SQL プログラムとして作成されます。 SQL ステートメントはローカル・データベースにアクセスします。プリコンパイル・プロセスの一部として SQL パッケージ・オブジェクトは作成されません。 SQL パッケージの作成 (CRTSQLPKG) コマンドを使用することができます。

relational-database-name: 新しい SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。ローカル・リレーショナル・データベースの名前を指定すると、作成されるプログラムは分散 SQL プログラムになります。 SQL ステートメントはローカル・データベースにアクセスします。

***NONE:** SQL パッケージ・オブジェクトは作成されません。プログラム・オブジェクトは分散プログラムではなく、 SQL パッケージの作成 (CRTSQLPKG) コマンドは使用できません。

USER

会話の開始時にリモート・システムに送られるユーザー名を指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***CURRENT:** 現在のジョブが実行されているユーザー・プロファイルが使用されます。

user-name: アプリケーション・サーバー・ジョブに使用されるユーザー名を指定します。

PASSWORD

リモート・システムで使用されるパスワードを指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***NONE:** パスワードは送られません。この値を指定する場合は、USER(*CURRENT) も指定しなければなりません。

password: USER パラメーターに指定したユーザー名のパスワードを指定します。

RDBCNNMTH

CONNECT ステートメントに使用する意味を指定します。詳細については、「*SQL 解説書*」の CONNECT (タイプ 1) および CONNECT (タイプ 2) を参照してください。

***DUW:** 分散作業単位をサポートするために CONNECT (タイプ 2) の意味が使用されます。追加のリレーショナル・データベースへの連続する CONNECT ステートメントによって、直前の接続が切断されることはありません。

***RUW:** リレーショナル・データベースへは 1 つの接続だけが使用できます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

DFTRDBCOL

テーブル、ビュー、索引および SQL パッケージの非修飾名に使用されるスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用されます。

***NONE:** OPTION パラメーターで定義した命名規則が使用されます。

schema-name: スキーマ識別名を指定します。この値は、OPTION パラメーターで指定した命名規則の代わりに使用されます。

DYNDFTCOL

DFTRDBCOL パラメーター用に指定した省略時スキーマ名が動的ステートメントでも使用されるかどうかを指定します。

***NO:** 動的 SQL ステートメント用のテーブル、ビュー、索引、および SQL パッケージの非修飾名用の DFTRDBCOL パラメーターに指定した値は使用されません。OPTION パラメーターで指定した命名規則が使用されます。

***YES:** DFTRDBCOL パラメーターに指定したスキーマ名は、動的 SQL ステートメント内のテーブル、ビュー、索引、および SQL パッケージの非修飾名用に使用されます。

SQLPKG

このコマンドの RDB パラメーターで指定されたリレーショナル・データベースで作成される SQL パッケージの修飾名を指定します。

指定できるライブラリー値は次のとおりです。

***PGMLIB:** パッケージは、プログラムが置かれているライブラリーと同じ名前をもつライブラリーの中に作成されます。

library-name: パッケージが作成されるライブラリーの名前を指定します。

***PGM:** パッケージ名はプログラム名と同じ名前になります。

package-name: RDBNAME パラメーターに指定したリモート・データベース上で作成されたパッケージの名前を指定します。

SQLPATH
静的 SQL ステートメント内のプロシージャー、関数、およびユーザー定義タイプを検出するために使用するパスを指定します。

***NAMING:** 使用するパスは、OPTION パラメーターで指定した命名規則に従います。

*SYS 命名の場合、使用するパスは、*LIBL です (実行時の現行ライブラリー・リスト)。

*SQL 命名の場合、使用するパスは、″QSYS″、″QSYS2″、″userid″ です。ただし、″userid″ は、USER 特殊レジスターの値です。schema-name が DFTRDBCOL パラメーターに指定された場合、その schema-name がユーザー ID に置き替わります。

***LIBL:** 使用するパスは、実行時のライブラリー・リストです。

schema-name: 1 つ以上のスキーマ名のリストを指定します。最大 268 の個別スキーマを指定できます。

SQLCURRULE
SQL ステートメントに使用する意味を指定します。

***DB2:** SQL ステートメントのすべての意味が、DB2 について設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数が文字データとして扱われます。

***STD:** SQL ステートメントすべての意味が、ISO および ANSI の SQL 規格によって設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数がバイナリー・データとして扱われます。

SAAFLAG
IBM SQL フラグ付け機能を指定します。このパラメーターは、SQL ステートメントにフラグ付けし、SQL ステートメントが IBM SQL 構文に準拠しているかを検査します。 IBM データベース・プロダクトの IBM SQL 構文に関する詳細は、「*DRDA IBM SQL Reference*」にあります。

***NOFLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行いません。

***FLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行います。

FLAGSTD
米国規格協会 (ANSI) のフラグ機能を指定します。このパラメーターは SQL ステートメントにフラグ付けし、ステートメントが次の標準に準拠するかどうかを検査します。

```
ANSI X3.135-1992 entry
ISO 9075-1992 entry
FIPS 127.2 entry
```

米国規格協会 (ANSI) のフラグ機能を指定します。このパラメーターは SQL ステートメントにフラグ付けし、ステートメントが次の標準に準拠するかどうかを検査します。

```
ANSI X3.135-1992 entry
ISO 9075-1992 entry
FIPS 127.2 entry
```

***NONE:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行いません。

***ANS:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行います。

PRTFILE

リストが送られる先の印刷装置ファイルの修飾名を指定します。ファイルは 132 バイト以上のレコード長をもっている必要があります。そうでない場合は情報が失われます。

印刷装置ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QSYSPRT: ファイル名の指定がない場合、プリコンパイラーの印刷出力は IBM 提供の印刷装置ファイル QSYSPRT に送られます。

printer-file-name: プリコンパイラー印刷出力が送られる印刷装置ファイルの名前を指定します。

SRTSEQ

SQL ステートメントの中の文字列比較に使用される分類順序テーブルを指定します。

注: アプリケーション・サーバーが iSeries システム上にない分散アプリケーション、またはリリース・レベルが V2R3M0 より前の分散アプリケーションでは、このパラメーターに *HEX を指定する必要があります。

***JOB:** ジョブの SRTSEQ 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの SRTSEQ 値は、プログラム実行時に検索されます。分散アプリケーションの場合、SRTSEQ(*JOBRUN) が有効なのは、LANGID(*JOBRUN) も指定されている場合だけです。

***LANGIDUNQ:** この分類順序テーブルは、コード・ページの各文字に、固有の分類順序を割り当てます。

***LANGIDSHR:** LANGID パラメーターで指定した言語用の共用分類テーブルが使用されます。

***HEX:** 分類順序テーブルは使用しません。分類順序を決定するには、文字の 16 進値を使用します。

分類順序テーブルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

table-name: 使用する分類順序テーブルの名前を指定します。

LANGID

SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されたときに使用される言語識別コードを指定します。

***JOB:** ジョブの LANGID 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの LANGID 値は、プログラム実行時に検索されます。分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

language-id: プログラムが使用する言語識別コードを指定します。

USRPRF

コンパイル済みプログラム・オブジェクトが実行されるときに使用されるユーザー・プロファイル (プログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトに対して所有する権限を含む) を指定します。プログラム・オブジェクトが使用できるオブジェクトの制御には、プログラム所有者またはプログラム・ユーザーのプロファイルが使用されます。

***NAMING:** ユーザー・プロファイルが命名規則によって判別されます。命名規則が *SQL である場合は、USRPRF(*OWNER) が使用されます。命名規則が *SYS である場合は、USRPRF(*USER) が使用されます。

***USER:** プログラム・オブジェクトを実行するユーザーのプロファイルが使用されます。

***OWNER:** プログラム所有者とプログラム・ユーザーの両方のユーザー・プロファイルがプログラムの実行時に使用されます。

DYNUSRPRF

動的 SQL ステートメントに使用されるユーザー・プロファイルを指定します。

***USER:** プログラムは、プログラムのユーザーのユーザー・プロファイルに基づいて実行されます。

***OWNER:** ローカルな動的 SQL ステートメントは、プログラムの所有者のユーザー・プロファイルに基づいて実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のユーザー・プロファイルに基づいて実行されます。

TOSRCFILE

SQL プリコンパイラーによって処理された出力ソース・メンバーを含む、ソー

ス・ファイルの修飾名を指定します。指定したソース・ファイルが見つからない場合、作成されます。出力メンバーの名前は、SRCMBR パラメーターに指定した名前と同じになります。

指定できるライブラリー値は次のとおりです。

QTEMP: ライブラリー QTEMP が使用されます。

***LIBL:** 指定したファイルがジョブのライブラリー・リストで検索されます。ライブラリー・リストに載せられているどのライブラリーにもファイルが見つからなければ、現行ライブラリー内にファイルが作成されます。

***CURLIB:** ジョブ用の現行ライブラリーが使用されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 出力ソース・ファイルを含むライブラリーの名前を指定します。

QSQLTEMP: ソース・ファイル QSQLTEMP が使用されます。

source-file-name: 出力ソース・メンバーを含むソース・ファイルの名前を指定します。

TEXT

プログラムおよびその機能を簡単に記述するテキストを指定します。このパラメーターの詳細については、Information Center の「CL 解説書」の『TEXT パラメーター』を参照してください。

***SCRMBRTXT:** テキストは PL/I プログラムを作成するために使用されるソース・ファイル・メンバーから取られます。データベース・ソース・メンバーのテキストを追加または変更するには、原始ステートメント入力ユーティリティー開始 (STRSEU) コマンドを使用するか、物理ファイル・メンバー追加 (ADDPFM) または物理ファイル・メンバー変更 (CHGPFM) コマンドを使用します。ソース・ファイルがインライン・ファイルまたは装置ファイルの場合は、テキストはブランクになります。

***BLANK:** テキストは指定されません。

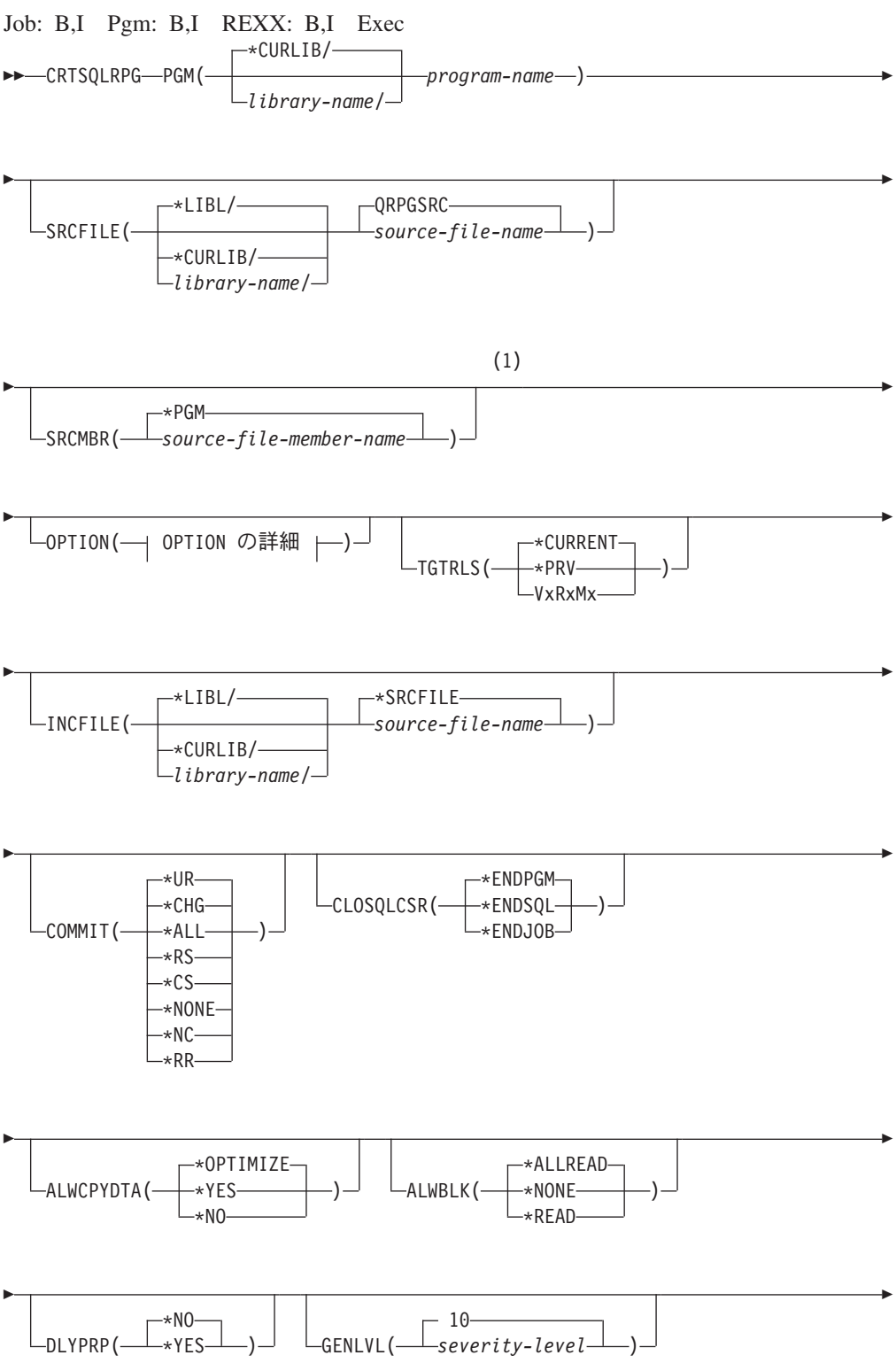
'description': 50 文字以下のテキストをアポストロフィで囲んで指定します。

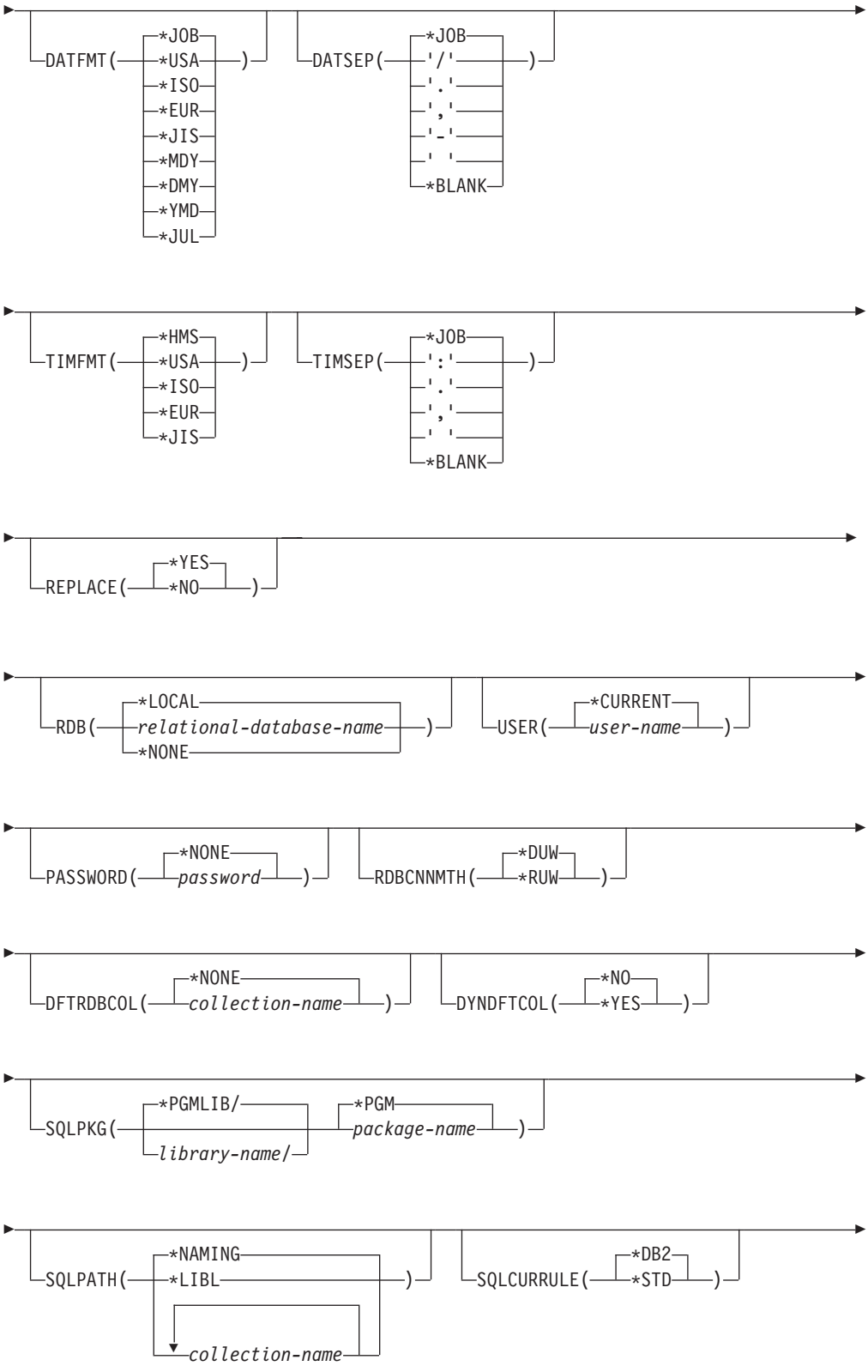
例:

```
CRTSQLPLI   PAYROLL  TEXT('Payroll Program')
```

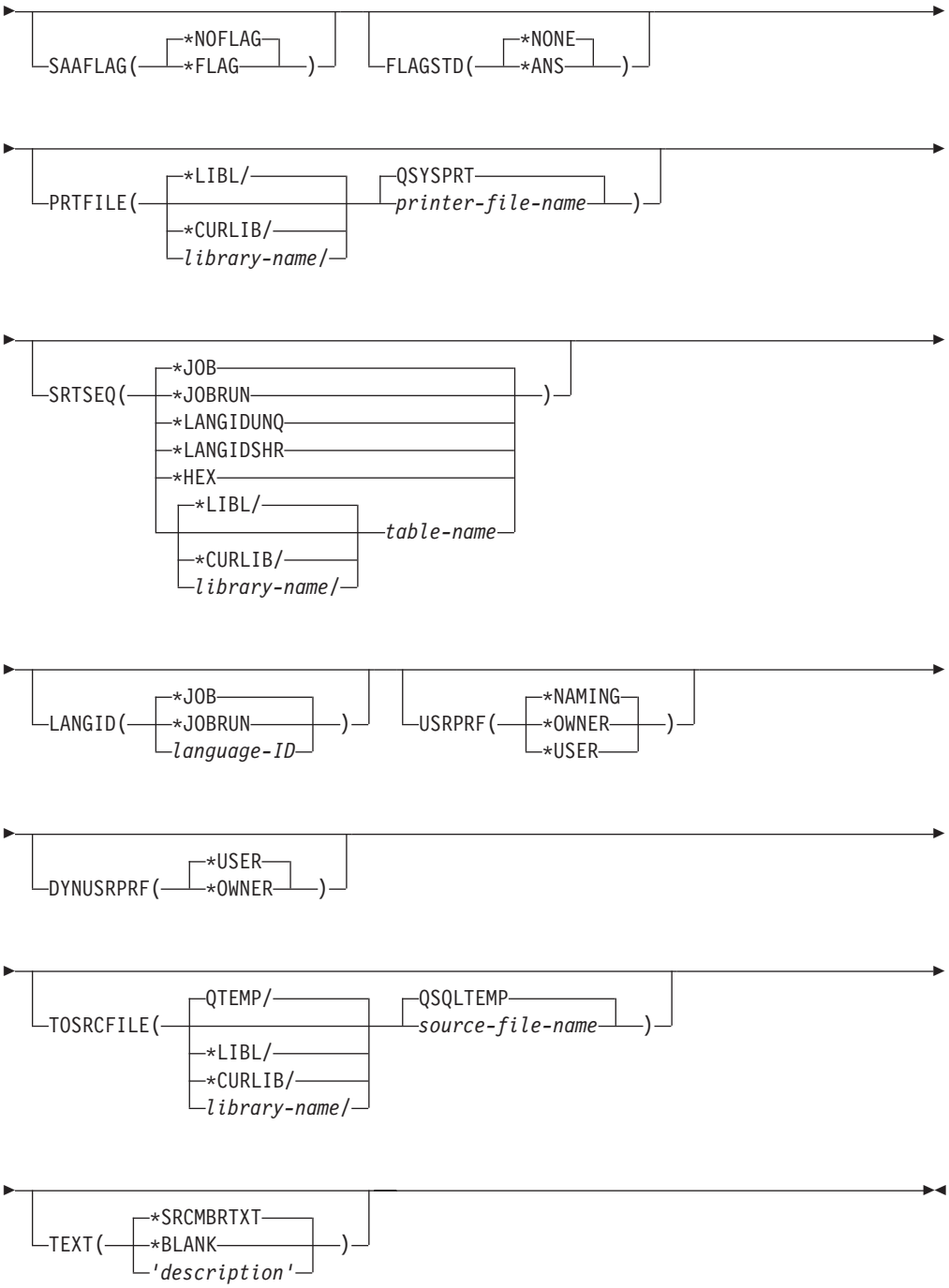
このコマンドは SQL プリコンパイラーを実行させるもので、そのプリコンパイラーはソース・プログラムをプリコンパイルし、変更したソース・プログラムをライブラリー QTEMP のファイル QSQLTEMP 内のメンバー PAYROLL の中に保管します。SQL プリコンパイラーが作成したソース・メンバーを使用して現行ライブラリーの中でプログラム PAYROLL を作成するために、PL/I コンパイラーが呼び出されます。

CRTSQLRPG (SQL RPG 作成) コマンド

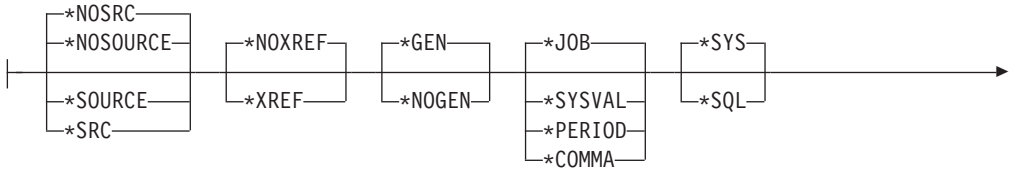



DATFMT(*JOB *USA *ISO *EUR *JIS *MDY *DMY *YMD *JUL)
DATSEP(*JOB '/' '.' ',' '-' ' ' *BLANK)
TIMFMT(*HMS *USA *ISO *EUR *JIS)
TIMSEP(*JOB ':' '.' ',' ' ' *BLANK)
REPLACE(*YES *NO)
RDB(*LOCAL relational-database-name *NONE)
USER(*CURRENT user-name)
PASSWORD(*NONE password)
RDBCNNMTH(*DUW *RUW)
DFTRDBCOL(*NONE collection-name)
DYNDFTCOL(*NO *YES)
SQLPKG(*PGMLIB/ library-name/ *PGM package-name)
SQLPATH(*NAMING *LIBL collection-name)
SQLCURRULE(*DB2 *STD)

CRTSQLRPG



OPTION の詳細:



注:

1 これより前にあるパラメーターはすべて、定位置形式で指定されます。

目的:

SQL RPG 作成 (CRTSQLRPG) コマンドは構造化照会言語 (SQL) プリコンパイラーを呼び出すもので、このプリコンパイラーが SQL ステートメントを含む RPG ソース・プログラムをプリコンパイルし、一時ソース・メンバーを作成してから、任意選択で RPG コンパイラーを呼び出してプログラムのコンパイルを行います。

パラメーター:

PGM

コンパイルしたプログラムの修飾名を指定します。

コンパイルした RPG プログラムの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***CURLIB:** ジョブ用の現行ライブラリーでコンパイルした RPG プログラムが作成されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: コンパイルした RPG プログラムが作成されるライブラリーの名前を指定します。

program-name: コンパイルしたプログラムの名前を指定します。

SRCFILE

SQL ステートメントとともに RPG ソース・プログラムが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QRPGSRC: ソース・ファイル名の指定がないときは、IBM 提供のソース・ファイル QRPGSRC には RPG ソース・プログラムが入ります。

source-file-name: RPG ソース・プログラムが入っているソース・ファイルの名前を指定します。

SRCMBR

RPG ソース・プログラムが入っているソース・ファイル・メンバーの名前を指定します。このパラメーターは、SRCFILE パラメーターのソース・ファイル名

がデータベース・ファイルである場合にのみ指定します。このパラメーターが指定されない場合は、PGM パラメーターに指定された PGM 名が使用されます。

***PGM:** PGM パラメーターに指定した名前と同じ名前をもつソース・ファイルのメンバーに RPG ソース・プログラムがあることを指定します。

source-file-member-name: RPG ソースが入っているメンバーの名前を指定します。

OPTION

RPG ソース・プログラムをプリコンパイルするときに次のオプションのうち 1 つ以上が使用されるかどうかを指定します。オプションが 2 度以上使用される場合、または 2 つのオプションが対立する場合は、指定された最後のオプションが使用されます。

要素 1: ソース・リスト・オプション

***NOSOURCE** または ***NOSRC:** プリコンパイルまたはパッケージ作成時にエラーが検出されなければ、プリコンパイラーによるソース印刷出力は行われません。

***SOURCE または *SRC:** プリコンパイラーは、RPG ソース入力から成るソース印刷出力を作成します。

要素 2: 相互参照オプション

***NOXREF:** プリコンパイラーは名前の相互参照を行いません。

***XREF:** プリコンパイラーは、プログラムの中の項目と、プログラムの中でそれらの項目を参照しているステートメントの番号との間の相互参照を行います。

要素 3: プログラム作成オプション

***GEN:** コンパイラーはコンパイル後に実行できるプログラムを作成します。リレーショナル・データベース名が RDB パラメーターで指定されている場合、SQL パッケージ・オブジェクトが作成されます。

***NOGEN:** プリコンパイラーは RPG コンパイラーを呼び出さないので、プログラムと SQL パッケージは作成されません。

要素 4: 小数点オプション

***JOB:** SQL で数値定数の小数点として使用される値は、プリコンパイル時にジョブ用に指定されている小数点の表現になります。

***SYSVAL:** SQL ステートメントの中の数値定数に小数点として使用する値は QDECFMT システム値になります。

注: QDECFMT が小数点として使用する値をコンマにすることを指定している場合は、リストの中の (SELECT 文節、VALUES 文節などのように) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

***PERIOD:** SQL ステートメントの中の数値定数に小数点として使用する値はピリオドになります。

***COMMA:** SQL ステートメントの中の数値定数に小数点として使用する値はコンマになります。

注: リストの中の (SELECT 文節、VALUES 文節などのような) 数値定数はどれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

要素 5: 命名規則オプション

***SYS:** システム命名規則 (library-name/file-name) を使用します。

***SQL:** SQL の命名規則 (schema-name.table-name) を使用します。iSeries 以外のリモート・データベース上でプログラムを作成するときは、命名規則として *SQL を指定しなければなりません。

要素 6: 2 次レベル・メッセージ・テキスト・オプション

***NOSECLVL:** 2 次レベルのテキスト記述はリストに追加されません。

***SECLVL:** リストのいずれかのメッセージについても、置換データを含む 2 次レベルのテキストが追加されます。

要素 7: ソース順序番号オプション

***NOSEQSRC:** QSQLTEMP 内の新しいソース・メンバーの作成時に、入力ソース・ファイルからのソース順序番号が使用されます。

***SEQSRC:** QSQLTEMP 内の新しいソース・メンバーに書き込まれるソース・レコードは、000001 から番号が付けられます。

要素 8: デバッグ・リスト・ビュー・オプション

***NOLSTDBG:** エラーとデバッグの情報が生成されません。

***LSTDBG:** SQL プリコンパイラーはリスト・ビュー、およびこのビューに必要なエラーとデバッグの情報を生成します。 *LSTDBG を使用できるのは、CODE/400 プロダクトを使用してユーザーのプログラムをコンパイルしている場合に限られます。

TGTRLS

作成中のオブジェクトを使用するオペレーティング・システムのリリース・レベルを指定します。

*CURRENT 値および *PRV 値の例では、VxRxMx 形式でリリースを指定する *release-level* 値が示されています。ここで Vx はバージョン、Rx はリリース、Mx はモディフィケーション・レベルです。たとえば、V2R3M0 はバージョン 2、リリース 3、モディフィケーション・レベル 0 です。

***CURRENT:** オブジェクトは、ユーザーのシステムで現在稼働しているオペレーティング・システムのリリースで使用されます。たとえば、V2R3M5 がシステムで稼働している場合、*CURRENT はユーザーが V2R3M5 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

注: V2R3M5 がシステム上で稼働しており、V2R3M0 が導入されたシステムでオブジェクトが使用される場合、TGTRLS(*CURRENT) ではなく TGTRLS(V2R3M0) を指定してください。

***PRV:** オブジェクトはオペレーティング・システムのモディフィケーション・レベル 0 の前のリリースで使用されます。たとえば、 V2R3M5 がユーザーのシステムで稼働している場合、*PRV はユーザーが V2R2M0 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

release-level: VxRxMx 形式でリリースを指定してください。指定されたリリースのシステム上、あるいは以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することができます。

有効な値は、現行バージョン、リリース、モディフィケーション・レベルによります。また、有効な値は各新規リリースで変わります。コマンドがサポートする初期リリース・レベルよりもさらに前のリリース・レベルを指定した場合には、エラー・メッセージはこれがサポートする最も初期のリリース・レベルを示して送信されます。

INCFILE

SQL の INCLUDE ステートメントを用いてプログラムに組み込むメンバーが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

***SRCFILE:** SRCFILE パラメーターで指定した修飾されたソース・ファイルには、SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入ります。

source-file-name: SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入っているソース・ファイルの名前を指定します。ここに指定するソース・ファイルのレコード長は、SRCFILE パラメーターに指定したソース・ファイルのレコード長より小さくてはなりません。

COMMIT

コンパイル済みプログラム内の SQL ステートメントがコミットメント制御のもとで実行されるかどうかを指定します。ホスト言語ソースの中で参照されているファイルは、このオプションによって影響を受けません。SQL ステートメントの中で参照される SQL テーブル、SQL ビュー、および SQL パッケージだけが影響されます。

注: RPG ソース・プログラムで参照されているファイルは、このオプションの影響を受けません。

***CHG または *UR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができます。

***ALL または *RS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。

***CS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。選択されたが、更新されていない行は、次の行が選択されるまでロックされます。他のジョブにおけるコミットされていない変更は見ることができません。

***NONE または *NC:** コミットメント制御が使用されないことを指定します。他のジョブにおけるコミットされていない変更は見ることができます。プログラムに SQL の DROP SCHEMA ステートメントが組み込まれている場合は、*NONE または *NC を使用しなければなりません。RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries 以外のシステム上にある場合は、*NONE または *NC を指定することはできません。

***RR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。
SELECT、UPDATE、DELETE、および INSERT ステートメントで参照されているすべてのテーブルは、作業単位 (トランザクション) の終わりまで排他的にロックされます。

CLOSQLCSR

SQL カーソルが暗黙にクローズされるとき、SQL 準備済みステートメントが暗黙に破棄され、LOCK TABLE のロックが解除されることを指定します。
CLOSE、COMMIT、または ROLLBACK (HOLD を指定しない) SQL ステートメントを発行すると、SQL カーソルが明示的にクローズされます。

***ENDPGM:** プログラムが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが破棄されます。呼び出しスタック上の最初の SQL プログラムが終了すると、LOCK TABLE のロックが解除されます。

***ENDSQL:** SQL カーソルは呼び出しと次の呼び出しの間はオープンしたままで、別の SQL OPEN を実行せずに取り出すことができます。呼び出しスタック上でより高い位置にあるプログラムの 1 つは、少なくとも 1 つの SQL ステートメントを実行しているはずです。呼び出しスタック上の最初の SQL プログラムが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメント

が廃棄され、LOCK TABLE のロックが解除されます。呼び出された最初の SQL プログラム (呼び出しスタック上の最初の SQL プログラム) であるプログラムに *ENDSQL が指定される場合、プログラムは *ENDPGM が指定されたかのように扱われます。

***ENDJOB:** SQL カーソルは呼び出しと次の呼び出しの間はオープンしたままで、別の SQL OPEN を実行せずに取り出すことができます。呼び出しスタック上でより高い位置にあるプログラムの 1 つは、少なくとも 1 つの SQL ステートメントを実行しているはずです。呼び出しスタック上の最初の SQL プログラムが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが廃棄され、LOCK TABLE のロックが解除されます。呼び出された最初の SQL プログラム (呼び出しスタック上の最初の SQL プログラム) であるプログラムに *ENDSQL が指定される場合、プログラムは *ENDPGM が指定されたかのように扱われます。

ALWCPYDTA

SELECT ステートメントでデータのコピーが使用できるかどうかを指定します。

***OPTIMIZE:** データベースから直接検索されたデータを使用するか、データのコピーを使用するかどうかは、システムが決定します。決定は、どの方式が最良のパフォーマンスを提供するかに基づいて行われます。 COMMIT が *CHG または *CS で ALWBLK が *ALLREAD でない場合、あるいは COMMIT が *ALL または *RR の場合、データのコピーが使用されるのは、照会を実行する必要があるときだけです。

***YES:** 必要な場合にデータのコピーが使用されます。

***NO:** データのコピーを使用することはできません。照会を実行するのにデータの一時コピーが必要な場合には、エラー・メッセージが返されます。

ALWBLK

データベース・マネージャーがレコードのブロック化を使用できるかどうか、および読み取り専用カーソルについてどの程度までブロック化が使用できるかを指定します。

***ALLREAD:** COMMIT パラメーターで *NONE または *CHG が指定されている場合は、行は読み取り専用カーソルに対してブロックされます。プログラム内で、明示的に更新することができないすべてのカーソルは、プログラム内に EXECUTE または EXECUTE IMMEDIATE ステートメントがある場合でも、読み取り専用にオープンされます。

*ALLREAD を指定すると、

- *READ の場合に許されるブロック化に加え、コミットメント制御レベル *CHG のもとでレコードのブロック化を行うことができます。
- プログラムの中のほとんどすべての読み取り専用カーソルのパフォーマンスを向上できますが、照会が次のように制限されます。
  - ロールバック (ROLLBACK) コマンド、ホスト言語で書いた ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、*ALLREAD の指定があるとき、読み取り専用カーソルの再位置決めはしません。

– カーソル内の行を更新するために、位置付けされた UPDATE または DELETE 使用ステートメントの動的実行 (たとえば、EXECUTE IMMEDIATE による) をすることはできません。ただし、カーソルに関する DECLARE ステートメントに FOR UPDATE 文節が含まれている場合を除きます。

***NONE:** カーソルの対象となるデータを検索するとき、行がブロック化されません。

*NONE を指定すると、

- 検索されるデータが最新であることが保証されます。
- 照会の対象となるデータの最初の行を検索するときの所要時間が短縮します。
- 照会の最初の数行だけが検索されてから照会がクローズされるときは、プログラムによって使用されないデータ行のブロックの検索をデータベース・マネージャーに中止させます。
- 大量の行を検索する照会全体のパフォーマンスが低下する可能性があります。

***READ:** 次のとき、カーソルの対象となるデータを読み取り専用で検索するときレコードがブロック化されます。

- COMMIT パラメーターに *NONE の指定があるとき。これは、コミットメント制御が使用されないことを示します。
- カーソルが FOR READ ONLY 文節を使用して宣言されているか、あるいは位置指定の UPDATE または DELETE ステートメントをカーソルに対して実行できる動的ステートメントがないとき。

*READ を指定すると、上記条件を満足する照会の全体のパフォーマンスが向上し、大量のレコードを検索することができます。

DLYPRP

PREPARE ステートメントについての動的ステートメント妥当性検査を、OPEN、EXECUTE、または DESCRIBE ステートメントが実行されるまで遅延させるかどうかを指定します。妥当性検査を遅延させると、余分な妥当性検査が除かれるため、パフォーマンスが向上します。

***NO:** 動的ステートメント妥当性検査を遅延させません。動的ステートメントが準備されるとき、アクセス・プランが妥当性検査されます。動的ステートメントが OPEN または EXECUTE ステートメントで使用される場合、アクセス・プランが再度妥当性検査されます。動的ステートメントによって参照されるオブジェクトの権限または存在は変化する場合があるので、OPEN または EXECUTE ステートメントを出した後、SQLCODE または SQLSTATE を検査して、動的ステートメントがまだ有効であるか確かめる必要があります。

***YES:** 動的ステートメント妥当性検査を、動的ステートメントが OPEN、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅延させます。動的ステートメントが使用されたときは、その妥当性検査が行われて、アクセス・プランが作られます。このパラメーターで *YES を指定する場合は、

OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後 SQLCODE と SQLSTATE を調べて、動的ステートメントが有効であるかどうかを確かめておく必要があります。

注: *YES を指定したときは、PREPARE ステートメントで INTO 文節が使用されている場合や、OPEN が動的ステートメントに対して出される前に DESCRIBE ステートメントがその動的ステートメントを使用した場合は、パフォーマンスは向上しません。

GENLVL
作成操作が失敗する時の重大度レベルを指定します。この値と等しいかまたはより大きい重大度レベルのエラーが発生すると、操作が終了します。

10: 省略時の重大度レベルは 10 です。

severity-level: 0 から 40 までの範囲で重大度レベルの値を指定します。

DATFMT
日付結果列にアクセスするときに使用される形式を指定します。すべての出力日付フィールドは、指定された形式で返されます。入力日付文字列については、日付が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、*USA、*ISO、*EUR、または *JIS を指定しなければなりません。

***JOB:** ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***USA:** 米国の日付形式 (mm/dd/yyyy) が使用されます。

***ISO:** 国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR:** ヨーロッパの日付形式 (dd.mm.yyyy) が使用されます。

***JIS:** 日本工業規格の日付形式 (yyyy-mm-dd) が使用されます。

***MDY:** 日付形式 (mm/dd/yy) が使用されます。

***DMY:** 日付形式 (dd/mm/yy) が使用されます。

***YMD:** 日付形式 (yy/mm/dd) が使用されます。

***JUL:** 年間通算日の日付形式 (yy/ddd) が使用されます。

DATSEP
日付結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブに指定された日付区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

'/': スラッシュ (/) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

'-': ダッシュ (-) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

TIMFMT

時刻結果列にアクセスするときに使用される形式を指定します。入力時刻文字列については、時刻が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、時刻形式は、時刻区切り記号がコロンまたはピリオドである *USA、*ISO、*EUR、*JIS、または *HMS でなければなりません。

***HMS:** (hh:mm:ss) 形式が使用されます。

***USA:** 米国の時刻形式 (hh:mm *xx*) が使用されます。ただし、*xx* は AM か PM です。

***ISO:** 国際標準化機構 (ISO) の時刻形式 (hh.mm.ss) が使用されます。

***EUR:** ヨーロッパの時刻形式 (hh.mm.ss) が使用されます。

***JIS:** 日本工業規格の時刻形式 (hh:mm:ss) が使用されます。

TIMSEP

時刻結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブのために指定された時刻区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

':': コロン (:) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK**: ブランク ( ) が使用されます。

REPLACE

同じライブラリーに同じ名前の既存の SQL パッケージがあるときに、新しい SQL パッケージが作成されるかどうかを指定します。このパラメーターの値は、C コマンドに渡されます。このパラメーターについての詳細は、「CL 解説書」の『付録 A. 拡張パラメーター記述』にあります。

***YES**: 新しいプログラムまたは SQL パッケージが作成され、指定したライブラリーの中の、同じ名前とタイプを持つ既存のプログラムまたは SQL パッケージは QRPLOBJ に移されます。

***NO**: 指定されたライブラリーに同じ名前およびタイプのオブジェクトがすでに存在している場合は、新規のプログラムまたは SQL パッケージは作成されません。

RDB

SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。

***LOCAL**: プログラムは分散 SQL プログラムとして作成されます。 SQL ステートメントはローカル・データベースにアクセスします。プリコンパイル・プロセスの一部として SQL パッケージ・オブジェクトは作成されません。 SQL パッケージの作成 (CRTSQLPKG) コマンドを使用することができます。

relational-database-name: 新しい SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。ローカル・リレーショナル・データベースの名前を指定すると、作成されるプログラムは分散 SQL プログラムになります。 SQL ステートメントはローカル・データベースにアクセスします。

***NONE**: SQL パッケージ・オブジェクトは作成されません。プログラム・オブジェクトは分散プログラムではなく、 SQL パッケージの作成 (CRTSQLPKG) コマンドは使用できません。

USER

会話の開始時にリモート・システムに送られるユーザー名を指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***CURRENT**: 現在のジョブが実行されているユーザー・プロファイルが使用されます。

user-name: アプリケーション要求元ジョブで使用しているユーザー名を指定します。

PASSWORD

リモート・システムで使用されるパスワードを指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***NONE**: パスワードは送られません。この値を指定する場合は、USER(*CURRENT) も指定しなければなりません。

password: USER パラメーターに指定したユーザー名のパスワードを指定します。

RDBCNNMTH

CONNECT ステートメントに使用する意味を指定します。詳細については、「SQL 解説書」を参照してください。

***DUW:** 分散作業単位をサポートするために CONNECT (タイプ 2) の意味が使用されます。追加のリレーショナル・データベースへの連続する CONNECT ステートメントによって、直前の接続が切断されることはありません。

***RUW:** リモート作業単位をサポートするのに CONNECT (タイプ 1) の意味が使用されます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

DFTRDBCOL

テーブル、ビュー、索引および SQL パッケージの非修飾名に使用されるスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用されます。

***NONE:** OPTION パラメーターで定義した命名規則が使用されます。

schema-name: スキーマ識別名を指定します。この値は、OPTION パラメーターで指定した命名規則の代わりに使用されます。

DYNDFTCOL

DFTRDBCOL パラメーター用に指定した省略時スキーマ名が動的ステートメントでも使用されるかどうかを指定します。

***NO:** 動的 SQL ステートメント用のテーブル、ビュー、索引、および SQL パッケージの非修飾名用の DFTRDBCOL パラメーターに指定した値は使用されません。OPTION パラメーターで指定した命名規則が使用されます。

***YES:** DFTRDBCOL パラメーターに指定したスキーマ名は、動的 SQL ステートメント内のテーブル、ビュー、索引、および SQL パッケージの非修飾名用に使用されます。

SQLPKG

このコマンドの RDB パラメーターで指定されたリレーショナル・データベースで作成される SQL パッケージの修飾名を指定します。

指定できるライブラリー値は次のとおりです。

***PGMLIB:** パッケージは、プログラムが置かれているライブラリーと同じ名前をもつライブラリーの中に作成されます。

library-name: パッケージが作成されるライブラリーの名前を指定します。

***PGM:** パッケージ名はプログラム名と同じ名前になります。

package-name: RDBNAME パラメーターに指定したリモート・データベース上で作成されたパッケージの名前を指定します。

SQLPATH

静的 SQL ステートメント内のプロシージャー、関数、およびユーザー定義タイプを検出するために使用するパスを指定します。

***NAMING:** 使用するパスは、OPTION パラメーターで指定した命名規則に従います。

*SYS 命名の場合、使用するパスは、*LIBL です (実行時の現行ライブラリー・リスト)。

*SQL 命名の場合、使用するパスは、″QSYS″、″QSYS2″、″userid″ です。ただし、″userid″ は、USER 特殊レジスターの値です。schema-name が DFTRDBCOL パラメーターに指定された場合、その schema-name がユーザー ID に置き替わります。

***LIBL:** 使用するパスは、実行時のライブラリー・リストです。

schema-name: 1 つ以上のスキーマ名のリストを指定します。最大 268 の個別スキーマを指定できます。

SQLCURRULE

SQL ステートメントに使用する意味を指定します。

***DB2:** SQL ステートメントのすべての意味が、DB2 について設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数が文字データとして扱われます。

***STD:** SQL ステートメントすべての意味が、ISO および ANSI の SQL 規格によって設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数がバイナリー・データとして扱われます。

SAAFLAG

IBM SQL フラグ付け機能を指定します。このパラメーターは、SQL ステートメントにフラグ付けし、SQL ステートメントが IBM SQL 構文に準拠しているかを検査します。IBM データベース・プロダクトの IBM SQL 構文に関する詳細は、「*DRDA IBM SQL Reference*」にあります。

***NOFLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行いません。

***FLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行います。

FLAGSTD

米国規格協会 (ANSI) のフラグ機能を指定します。このパラメーターは SQL ステートメントにフラグ付けし、ステートメントが次の標準に準拠するかどうかを検査します。

```
ANSI X3.135-1992 entry
ISO 9075-1992 entry
FIPS 127.2 entry
```

***NONE:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行いません。

***ANS:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行います。

PRTFILE

リストが送られる先の印刷装置ファイルの修飾名を指定します。ファイルは 132 バイト以上のレコード長をもっている必要があります。そうでない場合は情報が失われます。

印刷装置ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: コンパイラー印刷出力が送られる印刷装置ファイルの名前を指定します。

QSYSPRT: ファイル名の指定がない場合、プリコンパイラーの印刷出力は IBM 提供の印刷装置ファイル QSYSPRT に送られます。

printer-file-name: コンパイラー印刷出力が送られる印刷装置ファイルの名前を指定します。

SRTSEQ

SQL ステートメントの中の文字列比較に使用される分類順序テーブルを指定します。

注: アプリケーション・サーバーが iSeries システム上にない分散アプリケーション、またはリリース・レベルが V2R3M0 より前の分散アプリケーションでは、このパラメーターに *HEX を指定する必要があります。

***JOB:** ジョブの SRTSEQ 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの SRTSEQ 値は、プログラム実行時に検索されます。分散アプリケーションの場合、SRTSEQ(*JOBRUN) が有効なのは、LANGID(*JOBRUN) も指定されている場合だけです。

***LANGIDUNQ:** LANGID パラメーターで指定した言語用の固有の分類テーブルが使用されます。

***LANGIDSHR:** LANGID パラメーターで指定した言語用の共用分類テーブルが使用されます。

***HEX:** 分類順序テーブルは使用しません。分類順序を決定するには、文字の 16 進値を使用します。

分類順序テーブルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

table-name: 使用する分類順序テーブルの名前を指定します。

LANGID

SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されたときに使用される言語識別コードを指定します。

***JOB:** ジョブの LANGID 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの LANGID 値は、プログラム実行時に検索されます。分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

language-id: プログラムが使用する言語識別コードを指定します。

USRPRF

コンパイル済みプログラム・オブジェクトが実行されるときに使用されるユーザー・プロファイル (プログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトに対して所有する権限を含む) を指定します。プログラム・オブジェクトが使用できるオブジェクトの制御には、プログラム所有者またはプログラム・ユーザーのプロファイルが使用されます。

***NAMING:** ユーザー・プロファイルが命名規則によって判別されます。命名規則が *SQL である場合は、USRPRF(*OWNER) が使用されます。命名規則が *SYS である場合は、USRPRF(*USER) が使用されます。

***USER:** プログラム・オブジェクトを実行するユーザーのプロファイルが使用されます。

***OWNER:** プログラム所有者とプログラム・ユーザーの両方のユーザー・プロファイルがプログラムの実行時に使用されます。

DYNUSRPRF

動的 SQL ステートメントに使用されるユーザー・プロファイルを指定します。

***USER:** ローカルな動的 SQL ステートメントは、ジョブのユーザー・プロファイルに基づいて実行されます。分散動的 SQL ステートメントはアプリケーション・サーバー・ジョブのユーザー・プロファイルに基づいて実行されます。

***OWNER:** ローカルな動的 SQL ステートメントは、プログラムの所有者のユーザー・プロファイルに基づいて実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のユーザー・プロファイルに基づいて実行されます。

TOSRCFILE

SQL プリコンパイラーによって処理された出力ソース・メンバーを含む、ソース・ファイルの修飾名を指定します。指定したソース・ファイルが見つからない場合、作成されます。出力メンバーの名前は、SRCMBR パラメーターに指定した名前と同じになります。

指定できるライブラリー値は次のとおりです。

QTEMP: ライブラリー QTEMP が使用されます。

***LIBL:** 指定したファイルがジョブのライブラリー・リストで検索されます。ライブラリー・リストに載せられているどのライブラリーにもファイルが見つからなければ、現行ライブラリー内にファイルが作成されます。

***CURLIB:** ジョブ用の現行ライブラリーが使用されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 出力ソース・ファイルを含むライブラリーの名前を指定します。

QSQLTEMP: ソース・ファイル QSQLTEMP が使用されます。

source-file-name: 出力ソース・メンバーを含むソース・ファイルの名前を指定します。

TEXT

プログラムおよびその機能を簡単に記述したテキストを指定します。このパラメーターの詳細については、Information Center の「CL 解説書」の『TEXT パラメーター』を参照してください。

***SRCMBRTXT:** テキストは RPG プログラムを作成するために使用されるソース・ファイル・メンバーから取られます。データベース・ソース・メンバーのテキストを追加または変更するには、原始ステートメント入力ユーティリティー開始 (STRSEU) コマンドを使用するか、物理ファイル・メンバー追加 (ADDPFM) コマンドまたは物理ファイル・メンバー変更 (CHGPFM) コマンドを使用します。ソース・ファイルがインライン・ファイルまたは装置ファイルの場合は、テキストはブランクになります。

***BLANK:** テキストは指定されません。

'description': 50 文字以下のテキストをアポストロフィで囲んで指定します。

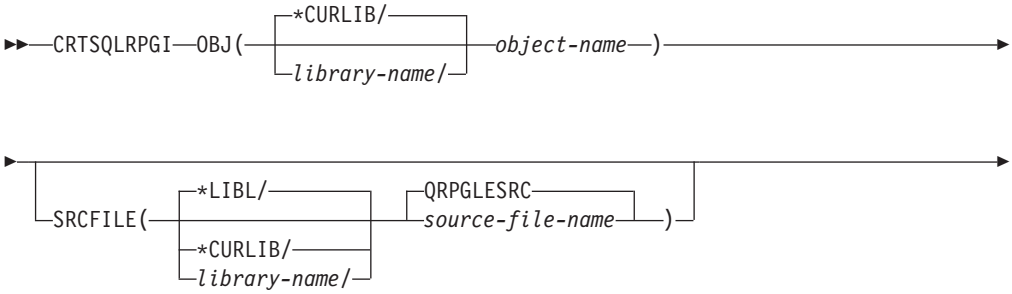
例:

```
CRTSQLRPG   PGM(JONES/ARBR5)
  TEXT('Accounts Receivable Branch 5')
```

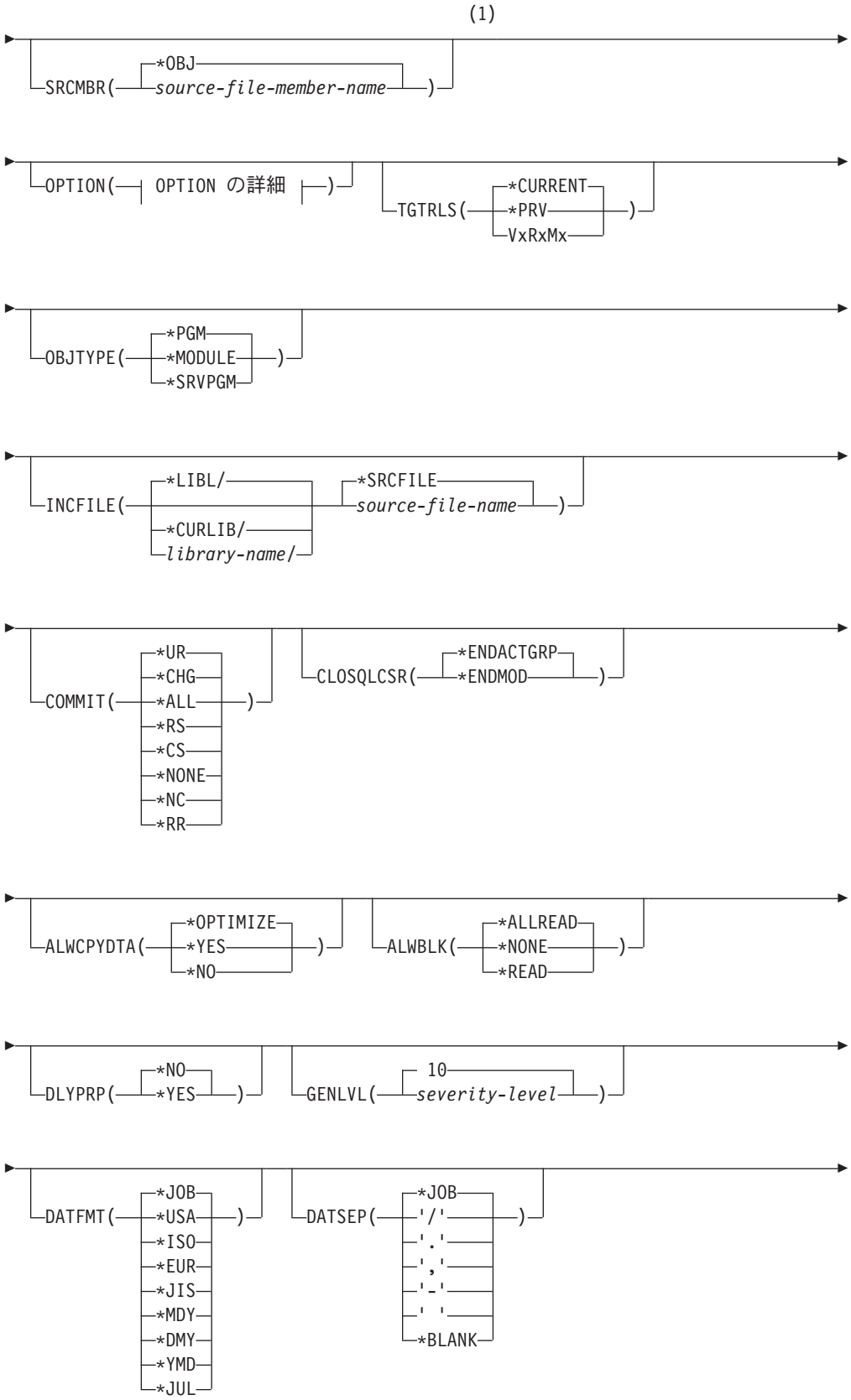
このコマンドは SQL プリコンパイラーを実行させるもので、そのプリコンパイラーはソース・プログラムをプリコンパイルし、変更したソース・プログラムをライブラリー QTEMP のファイル QSQLTEMP 内のメンバー ARBR5 の中に保管します。 SQL プリコンパイラーが作成したソース・メンバーを使用してライブラリー JONES の中でプログラム ARBR5 を作成するために RPG コンパイラーが呼び出されます。

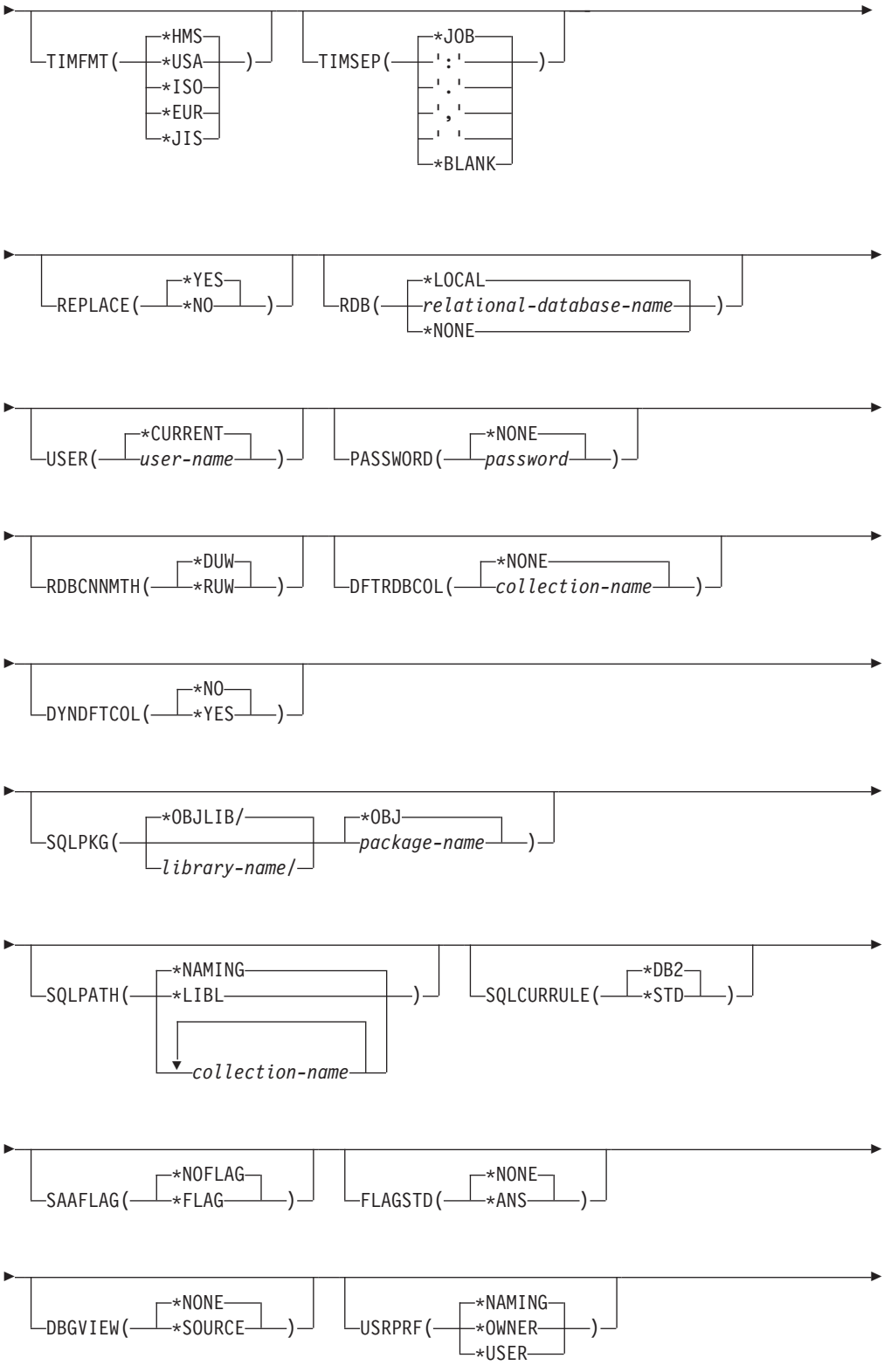
CRTSQLRPGI (SQL ILE RPG オブジェクト作成) コマンド

Job: B,I Pgm: B,I REXX: B,I Exec

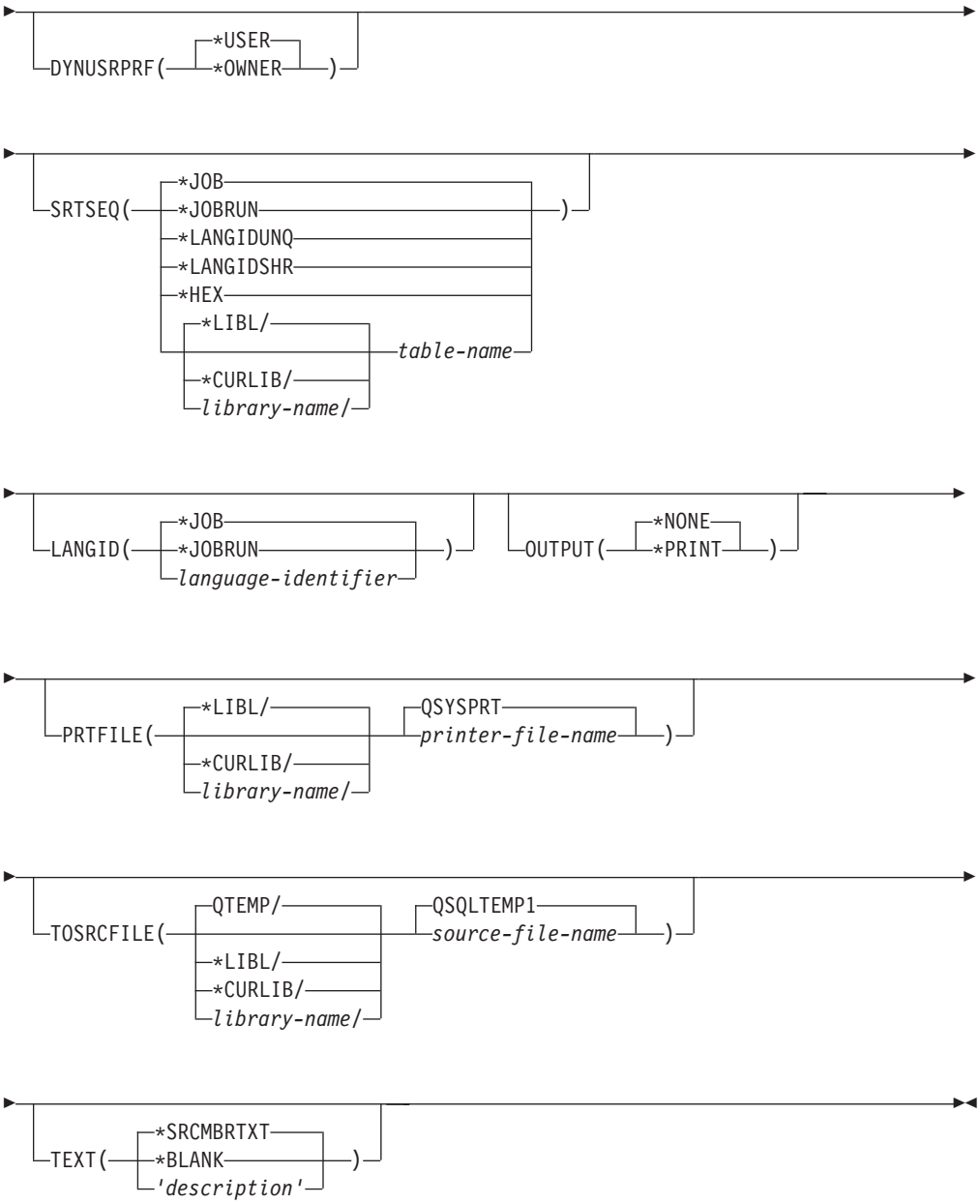


CRTSQLRPGI

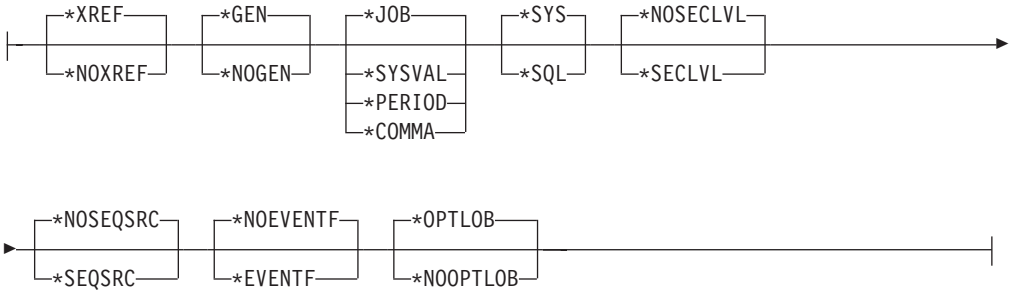

(1)
SRCMBR(
*OBJ
source-file-member-name
)
OPTION(
OPTION の詳細
)
TGTRLS(
*CURRENT
*PRV
VxRxMx
)
OBJTYPE(
*PGM
*MODULE
*SRVPGM
)
INCFILE(
*LIBL/
*CURLIB/
library-name/
*SRCFILE
source-file-name
)
COMMIT(
*UR
*CHG
*ALL
*RS
*CS
*NONE
*NC
*RR
)
CLOSQLCSR(
*ENDACTGRP
*ENDMOD
)
ALWCPYDTA(
*OPTIMIZE
*YES
*NO
)
ALWBLK(
*ALLREAD
*NONE
*READ
)
DLYPRP(
*NO
*YES
)
GENLVL(
10
severity-level
)
DATFMT(
*JOB
*USA
*ISO
*EUR
*JIS
*MDY
*DMY
*YMD
*JUL
)
DATSEP(
*JOB
'/'
'.'
','
'-'
' '
*BLANK
)


TIMFMT(*HMS *USA *ISO *EUR *JIS)
TIMSEP(*JOB ':' '.' ',' ' ' *BLANK)
REPLACE(*YES *NO)
RDB(*LOCAL relational-database-name *NONE)
USER(*CURRENT user-name)
PASSWORD(*NONE password)
RDBCNNMTH(*DUW *RUW)
DFTRDBCOL(*NONE collection-name)
DYNDFTCOL(*NO *YES)
SQLPKG(*OBJLIB/ library-name/ *OBJ package-name)
SQLPATH(*NAMING *LIBL collection-name)
SQLCURRULE(*DB2 *STD)
SAAFLAG(*NOFLAG *FLAG)
FLAGSTD(*NONE *ANS)
DBGVIEW(*NONE *SOURCE)
USRPRF(*NAMING *OWNER *USER)

CRTSQLRPGI



OPTION の詳細:



注:

1 これより前にあるパラメーターはすべて、定位置形式で指定されます。

目的:

SQL ILE RPG オブジェクト作成 (CRTSQLRPGI) コマンドは構造化照会言語 (SQL) プリコンパイラーを呼び出すもので、このプリコンパイラーが SQL ステートメントを含む RPG ソース・プログラムをプリコンパイルし、一時ソース・メンバーを作成してから、任意選択で ILE RPG コンパイラーを呼び出してモジュールの作成、プログラムの作成、またはサービス・プログラムの作成を行います。

パラメーター:

OBJ

作成するオブジェクトの修飾名を指定します。

***CURLIB:** ジョブの現行ライブラリー内で新しいオブジェクトが作成されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: オブジェクトが作成されるライブラリーの名前を指定します。

object-name: 作成するオブジェクトの名前を指定します。

SRCFILE

SQL ステートメントとともに RPG ソース・プログラムが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QRPGLESRC: ソース・ファイル名の指定がないときは、IBM 提供のソース・ファイル QRPGLESRC に RPG ソース・プログラムが入ります。

source-file-name: RPG ソース・プログラムが入っているソース・ファイルの名前を指定します。

SRCMBR

RPG ソース・プログラムが入っているソース・ファイル・メンバーの名前を指定します。このパラメーターは、SRCFILE パラメーターのソース・ファイル名がデータベース・ファイルである場合にのみ指定します。このパラメーターの指定がない場合、OBJ パラメーターで指定した PGM 名が使用されます。

***OBJ:** OBJ パラメーターに指定した名前と同じ名前をもつソース・ファイルのメンバーに RPG ソース・プログラムがあることを指定します。

source-file-member-name: RPG ソースが入っているメンバーの名前を指定します。

OPTION

RPG ソース・プログラムをプリコンパイルするときに次のオプションのうち 1 つ以上が使用されるかどうかを指定します。オプションが 2 度以上使用される場合、または 2 つのオプションが対立する場合は、指定された最後のオプションが使用されます。

要素 1: 相互参照オプション

***XREF:** プリコンパイラーは、プログラムの中の項目と、プログラムの中でそれらの項目を参照しているステートメントの番号との間の相互参照を行います。

***NOXREF:** プリコンパイラーは名前の相互参照を行いません。

要素 2: プログラム作成オプション

***GEN:** プリコンパイラーは、OBJTYPE パラメーターが指定するオブジェクトを作成します。

***NOGEN:** プリコンパイラーは RPG コンパイラーを呼び出しません。また、モジュール、プログラム、サービス・プログラム、および SQL パッケージは作成されません。

要素 3: 小数点オプション

***JOB:** SQL で数値定数の小数点として使用される値は、プリコンパイル時にジョブ用に指定されている小数点の表現になります。

***SYSVAL:** SQL ステートメントの中の数値定数に小数点として使用する値は QDECFMT システム値になります。

注: QDECFMT が、小数点として使用する値がコンマ (,) であると指定している場合、リストの中の (SELECT 文節、VALUE 文節のような) 数値定数は、いずれもコンマ (,) の後にブランク ( ) を置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオド (.) である VALUES(1.1,2.23,4.1) と同じものです。

PERIOD:** SQL ステートメントの中で数値定数の小数点として使用する値はピリオド (.) になります。COMMA:** SQL ステートメントの中の数値定数に小数点として使用する値はコンマ (,) になります。

注: リストの中の (SELECT 文節、VALUES 文節などのように) 数値定数は、いずれもコンマ (,) の後にブランク ( ) を置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオド (.) である VALUES(1.1,2.23,4.1) と同じものです。

要素 4: 命名規則オプション

***SYS:** システム命名規則 (library-name/file-name) を使用します。

***SQL:** SQL の命名規則 (schema-name.table-name) を使用します。iSeries 以外のリモート・データベース上でプログラムを作成するときは、命名規則として *SQL を指定しなければなりません。

要素 5: 2 次レベル・メッセージ・テキスト・オプション

***NOSECLVL:** 2 次レベルのテキスト記述はリストに追加されません。

***SECLVL:** リストのいずれかのメッセージについても、置換データを含む 2 次レベルのテキストが追加されます。

要素 6: ソース順序番号

***NOSEQSRC:** QSQLTEMP1 内に作成されたソース・ファイル・メンバーには、プリコンパイラーが読み取った元のソース・ファイルと同じ順序番号が付きます。

***SEQSRC:** QSQLTEMP1 内に作成されたソース・ファイル・メンバーには、000001 から始まり、000001 ずつ増加する順序番号が入ります。

要素 7: イベント・ファイル作成

***NOEVENTF:** コンパイラーは、連携開発環境プログラム/400 (CODE/400) が使用するイベント・ファイルを作成しません。

***EVENTF:** コンパイラーは、連携開発環境プログラム/400 (CODE/400) が使用するイベント・ファイルを作成します。イベント・ファイルは、ソース・ライブラリー内のファイル EVFEVENT のメンバーとして作成されます。 CODE/400 はこのファイルを使用して、CODE/400 編集プログラムに統合されたエラー・フィードバックを提供します。このオプションは、通常はユーザーの代わりに CODE/400 によって指定されます。

要素 8: 日付変換

***NOCVTDT:** 外部記述のファイルから検索された日付、時刻、およびタイム・スタンプの各データ・タイプは、固有の RPG 言語を使用して処理されます。

***CVTDT:** 外部記述のファイルから検索された日付、時刻、およびタイム・スタンプの各データ・タイプは、固定長文字として処理されます。

要素 9: DRDA に対するラージ・オブジェクトの最適化

***OPTLOB:** カーソルに対する最初の FETCH によって、これ以降のすべての FETCH での LOB (ラージ・オブジェクト) に対するカーソルの使用方法が決定されます。このオプションは、カーソルがクローズされるまで、引き続き有効です。

最初の FETCH が LOB ロケーターを使用して LOB 列にアクセスする場合、そのカーソルに対する後続の FETCH がその LOB 列を取り出して LOB ホスト変数に入れることはありません。

最初の FETCH が LOB 列を LOB ホスト変数に置いた場合、そのカーソルに対する後続の FETCH がその列に対して LOB ロケーターを使用することはできません。

***NOOPTLOB:** 列が検索されて LOB ロケーターまたは LOB ホスト変数に入れられるかどうかについての制約はありません。このオプションは、パフォーマンス低下の原因となることがあります。

TGTRLS

作成中のオブジェクトを使用するオペレーティング・システムのリリース・レベルを指定します。

*CURRENT 値および *PRV 値の例では、VxRxMx 形式でリリースを指定する *release-level* 値が示されています。ここで Vx はバージョン、Rx はリリース、Mx はモディフィケーション・レベルです。たとえば、V2R3M0 はバージョン 2、リリース 3、モディフィケーション・レベル 0 です。

***CURRENT:** オブジェクトは、ユーザーのシステムで現在稼働しているオペレーティング・システムのリリースで使用されます。たとえば、V2R3M5 がシステムで稼働している場合、*CURRENT はユーザーが V2R3M5 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

注: V2R3M5 がシステム上で稼働しており、V2R3M0 が導入されたシステムでオブジェクトが使用される場合、TGTRLS(*CURRENT) ではなく TGTRLS(V2R3M0) を指定してください。

***PRV:** オブジェクトはオペレーティング・システムのモディフィケーション・レベル 0 の前のリリースで使用されます。たとえば、 V2R3M5 がユーザーのシステムで稼働している場合、*PRV はユーザーが V2R2M0 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

release-level: VxRxMx 形式でリリースを指定してください。指定されたリリースのシステム上、あるいは以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することができます。

有効な値は、現行バージョン、リリース、モディフィケーション・レベルによります。また、有効な値は各新規リリースで変わります。コマンドがサポートする初期リリース・レベルよりもさらに前のリリース・レベルを指定した場合には、エラー・メッセージはこれがサポートする最も初期のリリース・レベルを示して送信されます。

OBJTYPE

作成するオブジェクトのタイプを指定します。

***PGM:** SQL プリコンパイラーは、バインド・プログラムを作成するための CRTBNDRPG コマンドを出します。

***MODULE:** SQL プリコンパイラーは、モジュールを作成するための CRTRPGMOD コマンドを出します。

***SRVPGM:** SQL プリコンパイラーは、サービス・プログラムを作成するための CRTRPGMOD コマンドと CRTSRVPGM コマンドを出します。

注:

1. OBJTYPE(*PGM) または OBJTYPE(*SRVPGM) を指定し、かつ RDB パラメーターも指定すると、プログラムの作成後に、SQL プリコンパイラーによって CRTSQLPKG コマンドが出されます。 OBJTYPE(*MODULE) を指定した場合は、SQL パッケージが作成されないので、CRTPGM コマンドまた

は CRTSRVPGM コマンドでプログラムを作成した後でユーザーが CRTSQLPKG コマンドを出さなければなりません。

2. *NOGEN の指定がないときは、SQL 一時ソース・メンバーだけが生成され、モジュール、プログラム、サービス・プログラム、および SQL パッケージは作成されません。

INCFILE

SQL の INCLUDE ステートメントを用いてプログラムに組み込むメンバーが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

***SRCFILE:** SRCFILE パラメーターで指定した修飾されたソース・ファイルには、SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入ります。

source-file-name: SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入っているソース・ファイルの名前を指定します。ここに指定するソース・ファイルのレコード長は、SRCFILE パラメーターに指定したソース・ファイルのレコード長より小さくてはなりません。

COMMIT

コンパイル済みの単位内の SQL ステートメントがコミットメント制御下で実行されるかどうかを指定します。ホスト言語ソースの中で参照されているファイルは、このオプションによって影響を受けません。SQL ステートメントの中で参照される SQL テーブル、SQL ビュー、および SQL パッケージだけが影響されます。

***CHG または *UR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができます。

***ALL または *RS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。

***CS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。選択されたが、更新されてい

ない行は、次の行が選択されるまでロックされます。他のジョブにおけるコミットされていない変更は見ることができません。

***NONE または *NC:** コミットメント制御が使用されないことを指定します。他のジョブにおけるコミットされていない変更は見ることができます。プログラムに SQL の DROP SCHEMA ステートメントが組み込まれている場合は、*NONE または *NC を使用しなければなりません。RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries 以外のシステム上にある場合は、*NONE または *NC を指定することはできません。

***RR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。
SELECT、UPDATE、DELETE、および INSERT ステートメントで参照されているすべてのテーブルは、作業単位 (トランザクション) の終わりまで排他的にロックされます。

CLOSQLCSR

SQL カーソルが暗黙にクローズされるとき、SQL 準備済みステートメントが暗黙に破棄され、LOCK TABLE のロックが解除されることを指定します。
CLOSE、COMMIT、または ROLLBACK (HOLD を指定しない) SQL ステートメントを発行すると、SQL カーソルが明示的にクローズされます。

***ENDACTGRP:** 活動化グループが終了した時点で、SQL カーソルがクローズされ、SQL 準備済みステートメントが暗黙に廃棄され、LOCK TABLE のロックが解除されます。

***ENDMOD:** モジュールが終了した時点で、SQL カーソルをクローズし、SQL 準備済みステートメントを暗黙に廃棄します。呼び出しスタック上の最初の SQL プログラムが終了すると、LOCK TABLE のロックが解除されます。

ALWCPYDTA

SELECT ステートメントでデータのコピーが使用できるかどうかを指定します。

***OPTIMIZE:** データベースから直接検索されたデータを使用するか、データのコピーを使用するかどうかは、システムが決定します。決定は、どの方式が最良のパフォーマンスを提供するかに基づいて行われます。 COMMIT が *CHG または *CS で ALWBLK が *ALLREAD でない場合、あるいは COMMIT が *ALL または *RR の場合、データのコピーが使用されるのは、照会を実行する必要があるときだけです。

***YES:** 必要な場合にデータのコピーが使用されます。

***NO:** データのコピーを使用することはできません。照会を実行するのにデータの一時コピーが必要な場合には、エラー・メッセージが返されます。

ALWBLK

データベース・マネージャーがレコードのブロック化を使用できるかどうか、および読み取り専用カーソルについてどの程度までブロック化が使用できるかを指定します。

***ALLREAD:** COMMIT パラメーターで *NONE または *CHG が指定されている場合は、行は読み取り専用カーソルに対してブロックされます。プログラム内で、明示的に更新することができないすべてのカーソルは、プログラム内に EXECUTE または EXECUTE IMMEDIATE ステートメントがある場合でも、読み取り専用にオープンされます。

*ALLREAD を指定すると、

- *READ の場合に許されるブロック化に加え、コミットメント制御レベル *CHG のもとでレコードのブロック化を行うことができます。
- プログラムの中のほとんどすべての読み取り専用カーソルのパフォーマンスを向上できますが、照会が次のように制限されます。
  - ロールバック (ROLLBACK) コマンド、ホスト言語で書いた ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、*ALLREAD の指定があるとき、読み取り専用カーソルの再位置決めはしません。
  - カーソル内の行を更新するために、位置付けされた UPDATE または DELETE 使用ステートメントの動的実行 (たとえば、EXECUTE IMMEDIATE による) をすることはできません。ただし、カーソルに関する DECLARE ステートメントに FOR UPDATE 文節が含まれている場合を除きます。

***NONE:** カーソルの対象となるデータを検索するとき、行がブロック化されません。

*NONE を指定すると、

- 検索されるデータが最新であることが保証されます。
- 照会の対象となるデータの最初の行を検索するときの所要時間が短縮します。
- 照会の最初の数行だけが検索されてから照会がクローズされるときは、プログラムによって使用されないデータ行のブロックの検索をデータベース・マネージャーに中止させます。
- 大量の行を検索する照会全体のパフォーマンスが低下する可能性があります。

***READ:** 次のとき、カーソルの対象となるデータを読み取り専用で検索するときレコードがブロック化されます。

- COMMIT パラメーターに *NONE の指定があるとき。これは、コミットメント制御が使用されないことを示します。
- カーソルが FOR READ ONLY 文節を使用して宣言されているか、あるいは位置指定の UPDATE または DELETE ステートメントをカーソルに対して実行できる動的ステートメントがないとき。

*READ を指定すると、上記条件を満足する照会の全体のパフォーマンスが向上し、大量のレコードを検索することができます。

DLYPRP

PREPARE ステートメントについての動的ステートメント妥当性検査を、OPEN、EXECUTE、または DESCRIBE ステートメントが実行されるまで遅延さ

せるかどうかを指定します。妥当性検査を遅延させると、余分な妥当性検査が除かれるため、パフォーマンスが向上します。

***NO:** 動的ステートメント妥当性検査を遅延させません。動的ステートメントが準備されるとき、アクセス・プランが妥当性検査されます。動的ステートメントが OPEN または EXECUTE ステートメントで使用される場合、アクセス・プランが再度妥当性検査されます。動的ステートメントによって参照されるオブジェクトの権限または存在は変化する場合があるので、OPEN または EXECUTE ステートメントを出した後、SQLCODE または SQLSTATE を検査して、動的ステートメントがまだ有効であるか確かめる必要があります。

***YES:** 動的ステートメント妥当性検査を、動的ステートメントが OPEN、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅延させます。動的ステートメントが使用されたときは、その妥当性検査が行われて、アクセス・プランが作られます。このパラメーターで *YES を指定する場合は、OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後 SQLCODE と SQLSTATE を調べて、動的ステートメントが有効であるかどうかを確かめておく必要があります。

注: *YES を指定したときは、PREPARE ステートメントで INTO 文節が使用されている場合や、OPEN が動的ステートメントに対して出される前に DESCRIBE ステートメントがその動的ステートメントを使用した場合は、パフォーマンスは向上しません。

GENLVL

作成操作が失敗する時の重大度レベルを指定します。重大度レベルがこの値より大きいエラーが発生する場合は、操作が終了します。

10: 省略時の重大度レベルは 10 です。

severity-level: 0 から 40 までの範囲で重大度レベルの値を指定します。

DATFMT

日付結果列にアクセスするときに使用される形式を指定します。すべての出力日付フィールドは、指定された形式で返されます。入力日付文字列については、日付が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、*USA、*ISO、*EUR、または *JIS を指定しなければなりません。

***JOB:** ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***USA:** 米国の日付形式 (mm/dd/yyyy) が使用されます。

***ISO:** 国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR:** ヨーロッパの日付形式 (dd.mm.yyyy) が使用されます。

***JIS:** 日本工業規格の日付形式 (yyyy-mm-dd) が使用されます。

***MDY:** 日付形式 (mm/dd/yy) が使用されます。

***DMY:** 日付形式 (dd/mm/yy) が使用されます。

***YMD:** 日付形式 (yy/mm/dd) が使用されます。

***JUL:** 年間通算日の日付形式 (yy/ddd) が使用されます。

DATSEP
日付結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブに指定された日付区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

'/': スラッシュ (/) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

'-': ダッシュ (-) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

TIMFMT
時刻結果列にアクセスするときに使用される形式を指定します。入力時刻文字列については、時刻が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力時刻文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries システム以外のシステムにある場合は、時刻形式は、時刻区切り記号がコロンまたはピリオドである *USA、*ISO、*EUR、*JIS、または *HMS でなければなりません。

***HMS:** **hh:mm:ss** 形式が使用されます。

***USA:** 米国の時刻形式 **hh:mm xx** が使用されます。ただし、**xx** は AM か PM です。

***ISO:** 国際標準化機構 (ISO) の時刻形式 **hh.mm.ss** が使用されます。

***EUR:** ヨーロッパの時刻形式 **hh.mm.ss** が使用されます。

***JIS:** 日本工業規格の時刻形式 **hh:mm:ss** が使用されます。

TIMSEP

時刻結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブのために指定された時刻区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

':': コロン (:) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

REPLACE

同じライブラリー内に同じ名前およびタイプの既存の SQL モジュール、プログラム、サービス・プログラム、またはパッケージかある場合に、SQL モジュール、プログラム、サービス・プログラム、またはパッケージを作成するかどうかを指定します。このパラメーターの値は、CRTRPGMOD、CRTBNDRPG、CRTSRVPGM、および CRTSQLPKG コマンドに渡されます。

***YES:** 新しい SQL モジュール、プログラム、サービス・プログラム、またはパッケージが作成され、指定したライブラリーの中の同じ名前およびタイプの既存の SQL オブジェクトは QRPLOBJ に移されます。

***NO:** 指定されたライブラリーに同じ名前とタイプの SQL オブジェクトがすでに存在している場合は、新しい SQL モジュール、プログラム、サービス・プログラム、またはパッケージは作成されません。

RDB

SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。

***LOCAL:** プログラムは分散 SQL プログラムとして作成されます。 SQL ステートメントはローカル・データベースにアクセスします。プリコンパイル・プロセスの一部として SQL パッケージ・オブジェクトは作成されません。 SQL パッケージの作成 (CRTSQLPKG) コマンドを使用することができます。

relational-database-name: 新しい SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。ローカル・リレーショナル・データベースの名前を指定すると、作成されるプログラムは分散 SQL プログラムになります。 SQL ステートメントはローカル・データベースにアクセスします。

***NONE:** SQL パッケージ・オブジェクトは作成されません。プログラム・オブジェクトは分散プログラムではなく、 SQL パッケージの作成 (CRTSQLPKG) コマンドは使用できません。

USER

会話の開始時にリモート・システムに送られるユーザー名を指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***CURRENT:** 現在のジョブが実行されているユーザー・プロファイルが使用されます。

user-name: アプリケーション・サーバー・ジョブに使用されるユーザー名を指定します。

PASSWORD

リモート・システムで使用されるパスワードを指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***NONE:** パスワードは送られません。この値を指定する場合は、USER(*CURRENT) も指定しなければなりません。

password: USER パラメーターに指定したユーザー名のパスワードを指定します。

RDBCNNMTH

CONNECT ステートメントに使用する意味を指定します。詳細については、「SQL 解説書」を参照してください。

***DUW:** 分散作業単位をサポートするために CONNECT (タイプ 2) の意味が使用されます。追加のリレーショナル・データベースへの連続する CONNECT ステートメントによって、直前の接続が切断されることはありません。

***RUW:** リモート作業単位をサポートするのに CONNECT (タイプ 1) の意味が使用されます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

DFTRDBCOL

テーブル、ビュー、索引および SQL パッケージの非修飾名に使用されるスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用されます。

***NONE:** OPTION パラメーターで定義した命名規則が使用されます。

schema-name: スキーマ識別名を指定します。この値は、OPTION パラメーターで指定した命名規則の代わりに使用されます。

DYNDFTCOL

DFTRDBCOL パラメーター用に指定した省略時スキーマ名が動的ステートメントでも使用されるかどうかを指定します。

***NO:** 動的 SQL ステートメント用のテーブル、ビュー、索引、および SQL パッケージの非修飾名用の DFTRDBCOL パラメーターに指定した値は使用されません。OPTION パラメーターで指定した命名規則が使用されます。

***YES:** DFTRDBCOL パラメーターに指定したスキーマ名は、動的 SQL ステートメント内のテーブル、ビュー、索引、および SQL パッケージの非修飾名用に使用されます。

SQLPKG

このコマンドの RDB パラメーターで指定されたリレーショナル・データベースで作成される SQL パッケージの修飾名を指定します。

指定できるライブラリー値は次のとおりです。

***OBJLIB:** パッケージは、OBJ パラメーターで指定したライブラリーと同じ名前のライブラリー内に作成されます。

library-name: パッケージが作成されるライブラリーの名前を指定します。

***OBJ:** SQL パッケージの名前は、 OBJ パラメーターで指定したオブジェクト名と同じになります。

package-name: SQL パッケージの名前を指定します。リモート・システムが iSeries システムでないときは、8 文字を超える名前は指定できません。

SQLPATH

静的 SQL ステートメント内のプロシージャー、関数、およびユーザー定義タイプを検出するために使用するパスを指定します。

***NAMING:** 使用するパスは、OPTION パラメーターで指定した命名規則に従います。

*SYS 命名の場合、使用するパスは、*LIBL です (実行時の現行ライブラリー・リスト)。

*SQL 命名の場合、使用するパスは、″QSYS″、″QSYS2″、″userid″ です。ただし、″userid″ は、USER 特殊レジスターの値です。schema-name が DFTRDBCOL パラメーターに指定された場合、その schema-name がユーザー ID に置き替わります。

***LIBL:** 使用するパスは、実行時のライブラリー・リストです。

schema-name: 1 つ以上のスキーマ名のリストを指定します。最大 268 の個別スキーマを指定できます。

SQLCURRULE

SQL ステートメントに使用する意味を指定します。

***DB2:** SQL ステートメントのすべての意味が、DB2 について設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数が文字データとして扱われます。

***STD:** SQL ステートメントすべての意味が、ISO および ANSI の SQL 規格によって設定されている規則の省略時値になります。このオプションによって次の意味が制御されます。

- 16 進定数がバイナリー・データとして扱われます。

SAAFLAG

IBM SQL フラグ付け機能を指定します。このパラメーターは、SQL ステートメントにフラグ付けし、SQL ステートメントが IBM SQL 構文に準拠しているかを検査します。 IBM データベース・プロダクトの IBM SQL 構文に関する詳細は、「*DRDA IBM SQL Reference*」にあります。

***NOFLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行いません。

***FLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行います。

FLAGSTD

米国規格協会 (ANSI) のフラグ機能を指定します。このパラメーターは SQL ステートメントにフラグ付けし、ステートメントが次の標準に準拠するかどうかを検査します。

```
ANSI X3.135-1992 entry
ISO 9075-1992 entry
FIPS 127.2 entry
```

***NONE:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行いません。

***ANS:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行います。

DBGVIEW

SQL プリコンパイラーによって提供されるソース・デバッグ情報のタイプを指定します。

***NONE:** ソース・プログラム・ビューは生成されません。

***SOURCE:** SQL プリコンパイラーはルート・ステートメントと、必要であれば SQL INCLUDE ステートメントに関するソース・プログラム・ビューを提供します。ビューには、プリコンパイラーが生成したステートメントが入ります。

USRPRF

コンパイル済みプログラム・オブジェクトが実行されるときに使用されるユーザー・プロファイル (プログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトに対して所有する権限を含む) を指定します。プログラム・オブジェクトが使用できるオブジェクトの制御には、プログラム所有者またはプログラム・ユーザーのプロファイルが使用されます。

***NAMING:** ユーザー・プロファイルが命名規則によって判別されます。命名規則が *SQL である場合は、USRPRF(*OWNER) が使用されます。命名規則が *SYS である場合は、USRPRF(*USER) が使用されます。

***USER:** プログラム・オブジェクトを実行するユーザーのプロファイルが使用されます。

***OWNER:** プログラム所有者とプログラム・ユーザーの両方のユーザー・プロファイルがプログラムの実行時に使用されます。

DYNUSRPRF

動的 SQL ステートメントで使用するユーザー・プロファイルを指定します。

***USER:** ローカル動的 SQL ステートメントは、プログラムのユーザー・プロファイルの下で実行されます。分散動的 SQL ステートメントは、SQL パッケージのユーザー・プロファイルの下で実行されます。

***OWNER:** ローカル動的 SQL ステートメントは、プログラムの所有者のプロファイルの下で実行されます。分散動的 SQL ステートメントは、 SQL パッケージの所有者のプロファイルの下で実行されます。

SRTSEQ

SQL ステートメントの中の文字列比較に使用される分類順序テーブルを指定します。

注: アプリケーション・サーバーが iSeries システム上にない分散アプリケーション、またはリリース・レベルが V2R3M0 より前の分散アプリケーションでは、このパラメーターに *HEX を指定する必要があります。

***JOB:** ジョブの SRTSEQ 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの SRTSEQ 値は、プログラム実行時に検索されます。分散アプリケーションの場合、SRTSEQ(*JOBRUN) が有効なのは、LANGID(*JOBRUN) も指定されている場合だけです。

***LANGIDUNQ:** LANGID パラメーターで指定した言語用の固有の分類テーブルが使用されます。

***LANGIDSHR:** 分類順序テーブルは複数の文字に同じ重みを使用するので、LANGID パラメーターで指定した言語に関連付けられた共用分類順序テーブルになります。

***HEX:** 分類順序テーブルは使用しません。分類順序を決定するには、文字の 16 進値を使用します。

分類順序テーブルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

table-name: 使用する分類順序テーブルの名前を指定します。

LANGID

SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されたときに使用される言語識別コードを指定します。

***JOB:** ジョブの LANGID 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの LANGID 値は、プログラム実行時に検索されます。分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

language-identifier: 言語識別コードを指定します。

OUTPUT

プリコンパイラーのリストを生成するかどうかを指定します。

***NONE:** プリコンパイラーのリストは生成されません。

***PRINT:** プリコンパイラーのリストが生成されます。

PRTFILE

プリコンパイラー印刷出力が送られる印刷装置ファイルの修飾名を指定します。ファイルの長さは 132 バイト以上でなければなりません。レコード長が 132 バイト未満のファイルを指定すると、情報が失われます。

印刷装置ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QSYSPRT: ファイル名の指定がない場合、プリコンパイラーの印刷出力は IBM 提供の印刷装置ファイル QSYSPRT に送られます。

printer-file-name: プリコンパイラー印刷出力が送られる印刷装置ファイルの名前を指定します。

TOSRCFILE

SQL プリコンパイラーによって処理された出力ソース・メンバーを含む、ソース・ファイルの修飾名を指定します。指定したソース・ファイルが見つからない場合、作成されます。出力メンバーの名前は、SRCMBR パラメーターに指定した名前と同じになります。

指定できるライブラリー値は次のとおりです。

QTEMP: ライブラリー QTEMP が使用されます。

***LIBL:** 指定したファイルがジョブのライブラリー・リストで検索されます。ライブラリー・リストに載せられているどのライブラリーにもファイルが見つからなければ、現行ライブラリー内にファイルが作成されます。

***CURLIB:** ジョブ用の現行ライブラリーが使用されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 出力ソース・ファイルを含むライブラリーの名前を指定します。

QSQLTEMP1: ソース・ファイル QSQLTEMP1 が使用されます。

source-file-name: 出力ソース・メンバーを含むソース・ファイルの名前を指定します。

TEXT

機能を簡単に記述するテキストを指定します。このパラメーターの詳細については、Information Center の「CL 解説書」の『TEXT パラメーター』を参照してください。

***SRCMBRTXT:** テキストは RPG プログラムを作成するために使用されるソース・ファイル・メンバーから取られます。データベース・ソース・メンバーのテキストを追加または変更するには、原始ステートメント入力ユーティリティー開始 (STRSEU) コマンドを使用するか、あるいは物理ファイル・メンバー追加 (ADDPFM) コマンドまたは物理ファイル・メンバー変更 (CHGPFM) コマンドを使用します。ソース・ファイルがインライン・ファイルまたは装置ファイルの場合は、テキストはブランクになります。

***BLANK:** テキストは指定されません。

'description': 50 文字以下のテキストをアポストロフィで囲んで指定します。

例:

```
CRTSQLRPGI  PAYROLL OBJTYPE(*PGM) TEXT('Payroll Program')
```

このコマンドは SQL プリコンパイラーを実行させるもので、そのプリコンパイラーはソース・プログラムをプリコンパイルし、変更したソース・プログラムをライブラリー QTEMP のファイル QSQLTEMP1 内のメンバー PAYROLL の中に保管します。 SQL プリコンパイラーが作成したソース・メンバーを使用して現行ライブラリーの中でプログラム PAYROLL を作成するために、ILE RPG コンパイラーが呼び出されます。

付録 C. FORTRAN for iSeries プリコンパイラーの使い方

この付録には、FORTRAN for iSeries プリコンパイラーの構文図が記載されています。ただし、このコンパイラーは、iSeries では今後サポートされません。 347 ページの『付録 D. FORTRAN アプリケーションでの SQL ステートメントのコーディング』では、SQL ステートメントを FORTRAN/400 プログラムに組み込む場合に固有のアプリケーションおよびコーディング上の要件について説明します。

詳細については、『FORTRAN/400 プリコンパイラーの使い方』を参照してください。

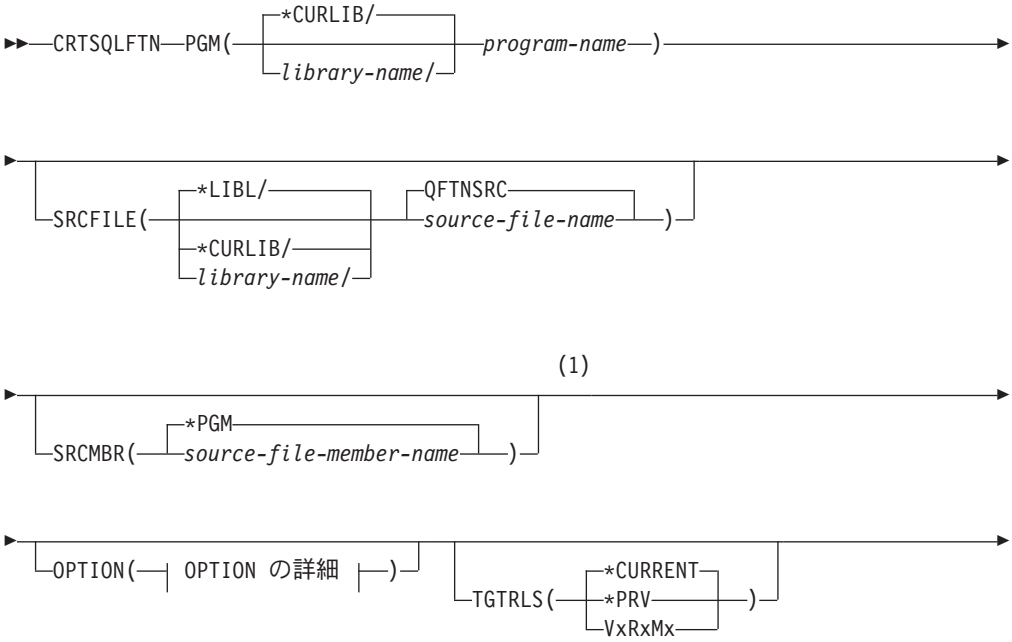
注: コード例についての詳細は、 viii ページの『コードについての特記事項』を参照してください。

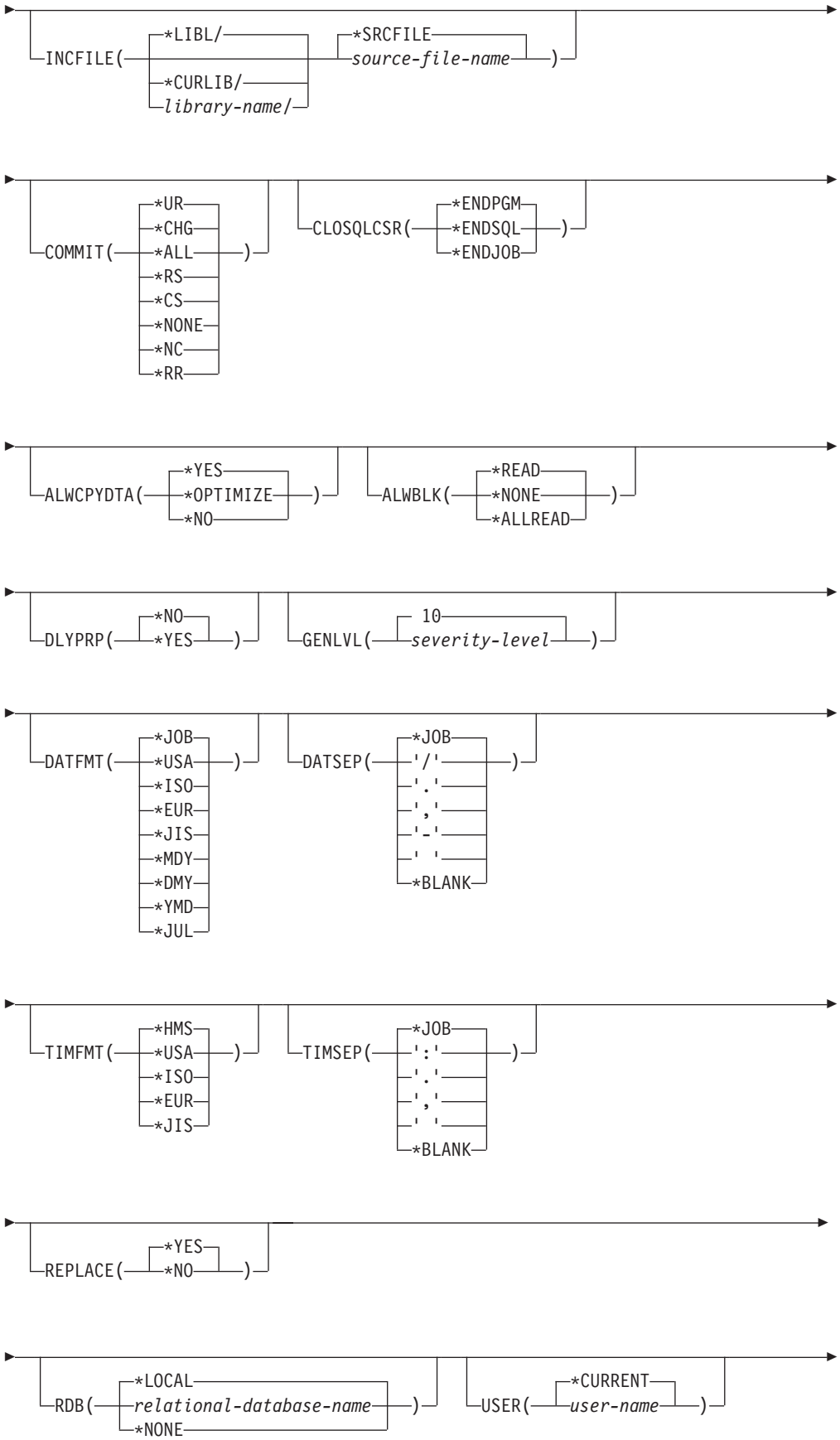
FORTRAN/400 プリコンパイラーの使い方

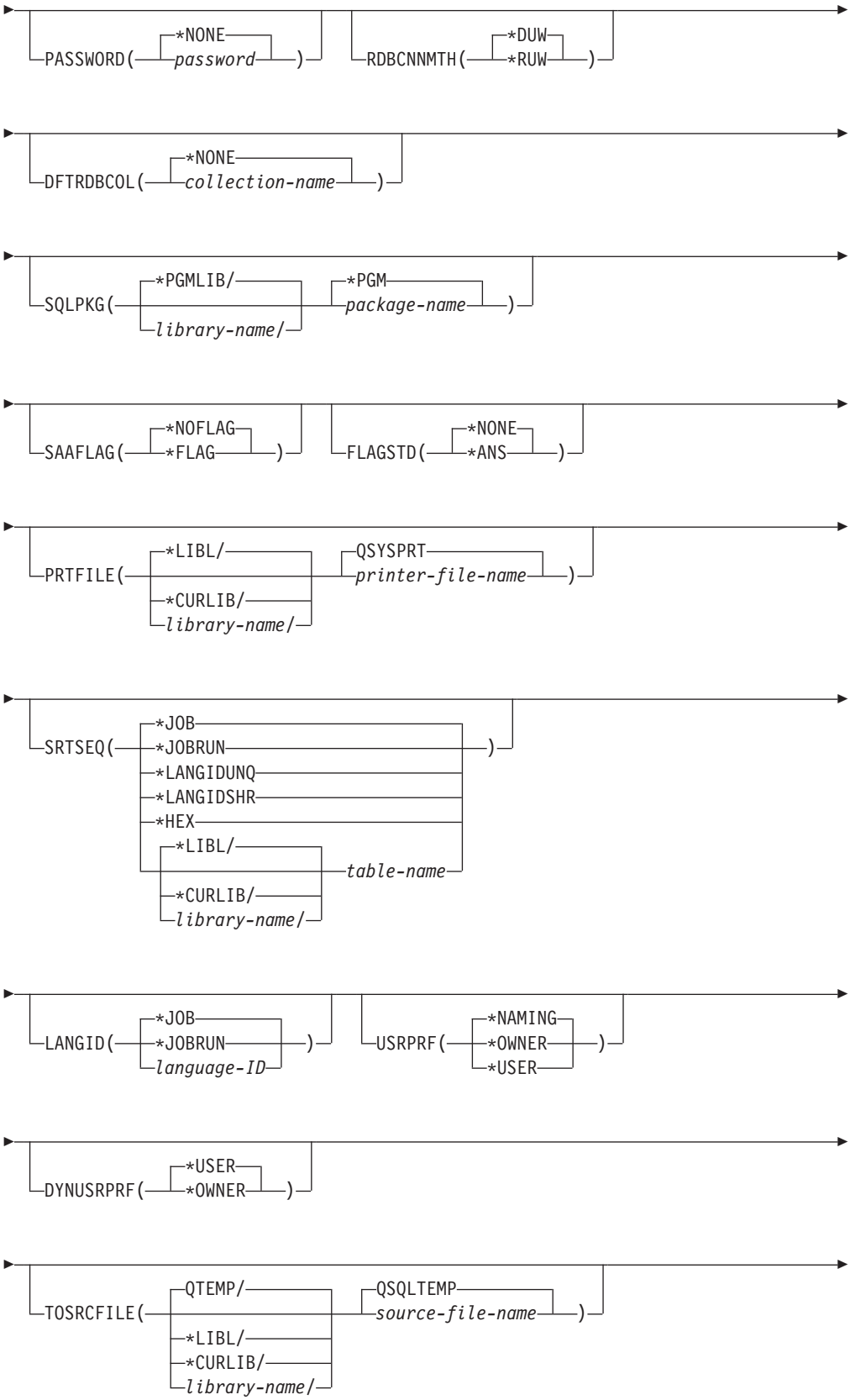
FORTRAN/400 は、今後、iSeries システム用のコンパイラーとしてはサポートされません。この付録の目的は、SQL FORTRAN プリコンパイラーと IBM 以外のその他の FORTRAN コンパイラーを併用しているお客さまを支援することです。FORTRAN プリコンパイラーの使い方の説明については、『CRTSQLFTN (SQL FORTRAN 作成) コマンド』を参照してください。詳細については、『CRTSQLFTN (SQL FORTRAN 作成) コマンド』を参照してください。

CRTSQLFTN (SQL FORTRAN 作成) コマンド

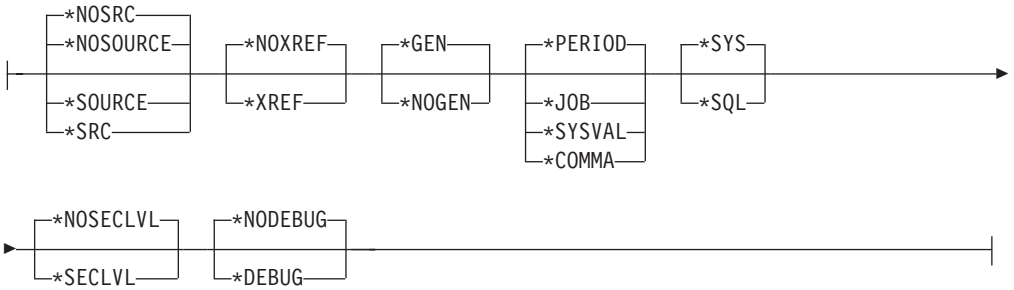
Job: B,I Pgm: B,I REXX: B,I Exec




INCFILE(
*LIBL/
*CURLIB/
library-name/
*SRCFILE
source-file-name
)
COMMIT(
*UR
*CHG
*ALL
*RS
*CS
*NONE
*NC
*RR
)
CLOSQLCSR(
*ENDPGM
*ENDSQL
*ENDJOB
)
ALWCPYDTA(
*YES
*OPTIMIZE
*NO
)
ALWBLK(
*READ
*NONE
*ALLREAD
)
DLYPRP(
*NO
*YES
)
GENLVL(
10
severity-level
)
DATFMT(
*JOB
*USA
*ISO
*EUR
*JIS
*MDY
*DMY
*YMD
*JUL
)
DATSEP(
*JOB
'/'
'.'
','
'-'
' '
*BLANK
)
TIMFMT(
*HMS
*USA
*ISO
*EUR
*JIS
)
TIMSEP(
*JOB
':'
'.'
','
' '
*BLANK
)
REPLACE(
*YES
*NO
)
RDB(
*LOCAL
relational-database-name
*NONE
)
USER(
*CURRENT
user-name
)


PASSWORD(*NONE password)
RDBCNNMTH(*DUW *RUW)
DFTRDBCOL(*NONE collection-name)
SQLPKG(*PGMLIB/ library-name/ *PGM package-name)
SAAFLAG(*NOFLAG *FLAG)
FLAGSTD(*NONE *ANS)
PRTFILE(*LIBL/ *CURLIB/ library-name/ QSYSPRT printer-file-name)
SRTSEQ(*JOB *JOBRUN *LANGIDUNQ *LANGIDSHR *HEX *LIBL/ *CURLIB/ library-name/ table-name)
LANGID(*JOB *JOBRUN language-ID)
USRPRF(*NAMING *OWNER *USER)
DYNUSRPRF(*USER *OWNER)
TOSRCFILE(QTEMP/ *LIBL/ *CURLIB/ library-name/ QSQLTEMP source-file-name)

OPTION の詳細:



注:

1 これより前にあるパラメーターはすべて、定位置形式で指定されます。

CRTSQLFTN コマンドの目的

SQL FORTRAN 作成 (CRTSQLFTN) コマンドは構造化照会言語 (SQL) プリコンパイラーを呼び出すもので、このプリコンパイラーが SQL ステートメントを含む FORTRAN ソース・プログラムをプリコンパイルし、一時ソース・メンバーを作成してから、任意選択で FORTRAN コンパイラーを呼び出してプログラムをコンパイルします。

CRTSQLFTN コマンドのパラメーター

PGM

コンパイルしたプログラムの修飾名を指定します。

コンパイルした FORTRAN プログラムの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***CURLIB:** ジョブ用の現行ライブラリーでコンパイルした FORTRAN プログラムが作成されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: コンパイルした FORTRAN プログラムが生成されるライブラリーの名前を指定します。

program-name: コンパイルした FORTRAN プログラムの名前を指定します。

SRCFILE

SQL ステートメントとともに FORTRAN ソース・プログラムが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QFTNSRC: ソース・ファイル名の指定がないと、IBM 提供のソース・ファイル QFTNSRC には FORTRAN ソース・プログラムが入ります。

source-file-name: FORTRAN ソース・プログラムが入っているソース・ファイルの名前を指定します。

SRCMBR

FORTRAN ソースが入っているソース・ファイル・メンバーの名前を指定します。このパラメーターは、SRCFILE パラメーターのソース・ファイル名がデータベース・ファイルである場合にのみ指定します。このパラメーターが指定されない場合は、PGM パラメーターに指定された PGM 名が使用されます。

***PGM:** PGM パラメーターに指定した名前と同じ名前をもつソース・ファイルのメンバーに FORTRAN ソース・プログラムがあることを指定します。

source-file-member-name: FORTRAN ソースが入っているメンバーの名前を指定します。

OPTION

FORTRAN ソース・プログラムをプリコンパイルするときに次のオプションのうち 1 つ以上が使用されるかどうかを指定します。オプションが 2 度以上使用される場合、または 2 つのオプションが対立する場合は、指定された最後のオプションが使用されます。

要素 1: ソース・リスト・オプション

***NOSOURCE:** または ***NOSRC:** プリコンパイルまたはパッケージ作成時にエラーが検出されなければ、プリコンパイラーによるソース印刷出力は行われません。

***SOURCE または *SRC:** プリコンパイラーは、FORTRAN ソース入力から成るソース印刷出力を作成します。

要素 2: 相互参照オプション

***NOXREF:** プリコンパイラーは名前の相互参照を行いません。

***XREF:** プリコンパイラーは、プログラムの中の項目と、プログラムの中でそれらの項目を参照しているステートメントの番号との間の相互参照を行います。

要素 3: プログラム作成オプション

***GEN:**

***NOGEN:** プリコンパイラーは FORTRAN コンパイラーを呼び出さないので、プログラムと SQL パッケージは作成されません。

要素 4: 小数点オプション

***PERIOD:** SQL ステートメントの中の数値定数に小数点として使用する値はピリオドになります。

***JOB** SQL で数値定数の小数点として使用される値は、プリコンパイル時にジョブ用に指定されている小数点の表現になります。

***SYSVAL:** SQL ステートメントの中の数値定数に小数点として使用する値は QDECFMT システム値になります。

注: QDECFMT が小数点として使用する値をコンマにすることを指定している場合は、リストの中の (SELECT 文節、VALUES 文節などのように) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

***COMMA:** SQL ステートメントの中の数値定数に小数点として使用する値はコンマになります。

注: リストの中の (SELECT 文節、VALUES 文節のような) 数値定数は、いずれもコンマの後にブランクを置くことによって区切らなければなりません。たとえば、VALUES(1,1, 2,23, 4,1) は、小数点がピリオドである VALUES(1.1,2.23,4.1) と同じものです。

要素 5: 命名規則オプション

***SYS:** システム命名規則 (library-name/file-name) を使用します。

***SQL:** SQL の命名規則 (schema-name.table-name) を使用します。iSeries サーバー以外のリモート・データベース上でプログラムを作成するときは、命名規則として *SQL を指定しなければなりません。

要素 6: 2 次レベル・メッセージ・テキスト・オプション

***NOSECLVL:** 2 次レベルのテキスト記述はリストに追加されません。

***SECLVL:** リストのいずれかのメッセージについても、置換データを含む 2 次レベルのテキストが追加されます。

要素 7: デバッグ・オプション

***NODEBUG:** 記号拡張プログラム・モデル (EPM) のデバッグ情報はプログラムとともに保管されません。このオプションはコンパイラーに渡されるものなので、SQL プリコンパイラーには影響しません。

***DEBUG:** 記号 EPM のデバッグ情報はプログラムとともに保管されます。このオプションはコンパイラーに渡されるものなので、SQL プリコンパイラーには影響しません。

TGTRLS

作成中のオブジェクトを使用するオペレーティング・システムのリリース・レベルを指定します。

*CURRENT 値および *PRV 値の例では、VxRxMx 形式でリリースを指定する *release-level* 値が示されています。ここで Vx はバージョン、Rx はリリース、Mx はモディフィケーション・レベルです。たとえば、V2R3M0 はバージョン 2、リリース 3、モディフィケーション・レベル 0 です。

***CURRENT:** オブジェクトは、ユーザーのシステムで現在稼働しているオペレーティング・システムのリリースで使用されます。たとえば、V2R3M5 がシス

テムで稼働している場合、*CURRENT はユーザーが V2R3M5 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

注: V2R3M5 がシステム上で稼働しており、V2R3M0 が導入されたシステムでオブジェクトが使用される場合、TGTRLS(*CURRENT) ではなく TGTRLS(V2R3M0) を指定してください。

***PRV:** オブジェクトはオペレーティング・システムのモディフィケーション・レベル 0 の前のリリースで使用されます。たとえば、 V2R3M5 がユーザーのシステムで稼働している場合、*PRV はユーザーが V2R2M0 が導入されたシステム上でオブジェクトを使用する予定であることを意味します。また、ユーザーは、以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することもできます。

release-level: VxRxMx 形式でリリースを指定してください。指定されたリリースのシステム上、あるいは以降のリリースのオペレーティング・システムを導入したシステム上でオブジェクトを使用することができます。

有効な値は、現行バージョン、リリース、モディフィケーション・レベルによります。また、有効な値は各新規リリースで変わります。コマンドがサポートする初期リリース・レベルよりもさらに前のリリース・レベルを指定した場合には、エラー・メッセージはこれがサポートする最も初期のリリース・レベルを示して送信されます。

INCFILE

SQL の INCLUDE ステートメントを用いてプログラムに組み込むメンバーが入っているソース・ファイルの修飾名を指定します。

ソース・ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

***SRCFILE:** SRCFILE パラメーターで指定した修飾されたソース・ファイルには、SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入ります。

source-file-name: SQL の INCLUDE ステートメントで指定されたソース・ファイル・メンバーが入っているソース・ファイルの名前を指定します。ここに指定するソース・ファイルのレコード長は、SRCFILE パラメーターに指定したソース・ファイルのレコード長より小さくてはなりません。

COMMIT

コンパイル済みプログラム内の SQL ステートメントがコミットメント制御のもとで実行されるかどうかを指定します。ホスト言語ソースの中で参照されている

ファイルは、このオプションによって影響を受けません。SQL ステートメントの中で参照される SQL テーブル、SQL ビュー、および SQL パッケージだけが影響されます。

***CHG または *UR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができます。

***ALL または *RS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されるオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。

***CS:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。選択されたが、更新されていない行は、次の行が選択されるまでロックされます。他のジョブにおけるコミットされていない変更は見ることができません。

***NONE または *NC:** コミットメント制御が使用されないことを指定します。他のジョブにおけるコミットされていない変更は見ることができます。プログラムに SQL の DROP SCHEMA ステートメントが組み込まれている場合は、*NONE または *NC を使用しなければなりません。RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries 以外のシステム上にある場合は、*NONE または *NC を指定することはできません。

***RR:** SQL の ALTER、CALL、COMMENT ON、CREATE、DROP、GRANT、LABEL ON、RENAME、および REVOKE ステートメントで参照されているオブジェクトと選択、更新、削除、および挿入される行が、作業単位 (トランザクション) の終わりまでロックされることを指定します。他のジョブにおけるコミットされていない変更は見ることができません。
SELECT、UPDATE、DELETE、および INSERT ステートメントで参照されているすべてのテーブルは、作業単位 (トランザクション) の終わりまで排他的にロックされます。

CLOSQLCSR

SQL カーソルが暗黙にクローズされるとき、SQL 準備済みステートメントが暗黙に破棄され、LOCK TABLE のロックが解除されることを指定します。
CLOSE、COMMIT、または ROLLBACK (HOLD を指定しない) SQL ステートメントを発行すると、SQL カーソルが明示的にクローズされます。

***ENDPGM:** プログラムが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが破棄されます。呼び出しスタック上の最初の SQL プログラムが終了すると、LOCK TABLE のロックが解除されます。

***ENDSQL:** SQL カーソルは呼び出しと次の呼び出しの間はオープンしたままで、別の SQL OPEN を実行せずに取り出すことができます。呼び出しスタック上でより高い位置にあるプログラムの 1 つは、少なくとも 1 つの SQL ステー

トメントを実行しているはずです。呼び出しスタック上の最初の SQL プログラムが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが廃棄され、LOCK TABLE のロックが解除されます。呼び出された最初の SQL プログラム (呼び出しスタック上の最初の SQL プログラム) であるプログラムに *ENDSQL が指定される場合、プログラムは *ENDPGM が指定されたかのように扱われます。

***ENDJOB:** SQL カーソルは呼び出しと次の呼び出しの間はオープンしたままで、別の SQL OPEN を実行せずに取り出すことができます。呼び出しスタック上でより高い位置にあるプログラムは、SQL ステートメントを実行させる必要がありません。呼び出しスタック上の最初の SQL プログラムが終了するときに、SQL カーソルはオープンのままになり、SQL 準備済みステートメントが保持され、LOCK TABLE のロックが保留されます。ジョブが終了すると、SQL カーソルがクローズされ、SQL 準備済みステートメントが廃棄され、LOCK TABLE のロックが解除されます。

ALWCPYDTA

SELECT ステートメントでデータのコピーが使用できるかどうかを指定します。

***OPTIMIZE:** データベースから直接検索されたデータを使用するか、データのコピーを使用するかどうかは、システムが決定します。決定は、どの方式が最良のパフォーマンスを提供するかに基づいて行われます。 COMMIT が *CHG または *CS で ALWBLK が *ALLREAD でない場合、あるいは COMMIT が *ALL または *RR の場合、データのコピーが使用されるのは、照会を実行する必要があるときだけです。

***YES:** 必要な場合にデータのコピーが使用されます。

***NO:** データのコピーを使用することはできません。照会を実行するのにデータの一時コピーが必要な場合には、エラー・メッセージが返されます。

ALWBLK

データベース・マネージャーがレコードのブロック化を使用できるかどうか、および読み取り専用カーソルについてどの程度までブロック化が使用できるかを指定します。

***ALLREAD:** COMMIT パラメーターで *NONE または *CHG が指定されている場合は、行は読み取り専用カーソルに対してブロックされます。プログラム内で、明示的に更新することができないすべてのカーソルは、プログラム内に EXECUTE または EXECUTE IMMEDIATE ステートメントがある場合でも、読み取り専用にオープンされます。

*ALLREAD を指定すると、

- *READ の場合に許されるブロック化に加え、コミットメント制御レベル *CHG のもとでレコードのブロック化を行うことができます。
- プログラムの中のほとんどすべての読み取り専用カーソルのパフォーマンスを向上できますが、照会が次のように制限されます。
  - ロールバック (ROLLBACK) コマンド、ホスト言語で書いた ROLLBACK ステートメント、または ROLLBACK HOLD SQL ステートメントは、*ALLREAD の指定があるとき、読み取り専用カーソルの再位置決めはしません。

– カーソル内の行を更新するために、位置付けされた UPDATE または DELETE 使用ステートメントの動的実行 (たとえば、EXECUTE IMMEDIATE による) をすることはできません。ただし、カーソルに関する DECLARE ステートメントに FOR UPDATE 文節が含まれている場合を除きます。

***NONE:** カーソルの対象となるデータを検索するとき、行がブロック化されません。

*NONE を指定すると、

- 検索されるデータが最新であることが保証されます。
- 照会の対象となるデータの最初の行を検索するときの所要時間が短縮します。
- 照会の最初の数行だけが検索されてから照会がクローズされるときは、プログラムによって使用されないデータ行のブロックの検索をデータベース・マネージャーに中止させます。
- 大量の行を検索する照会全体のパフォーマンスが低下する可能性があります。

***READ:** 次のとき、カーソルの対象となるデータを読み取り専用で検索するときレコードがブロック化されます。

- COMMIT パラメーターに *NONE の指定があるとき。これは、コミットメント制御が使用されないことを示します。
- カーソルが FOR FETCH ONLY 文節を使用して宣言されているか、あるいは位置指定の UPDATE または DELETE ステートメントをカーソルに対して実行できる動的ステートメントがないとき。

*READ を指定すると、上記条件を満足する照会の全体のパフォーマンスが向上し、大量のレコードを検索することができます。

DLYPRP

PREPARE ステートメントについての動的ステートメント妥当性検査を、OPEN、EXECUTE、または DESCRIBE ステートメントが実行されるまで遅延させるかどうかを指定します。妥当性検査を遅延させると、余分な妥当性検査が除かれるため、パフォーマンスが向上します。

***NO:** 動的ステートメント妥当性検査を遅延させません。動的ステートメントが準備されるとき、アクセス・プランが妥当性検査されます。動的ステートメントが OPEN または EXECUTE ステートメントで使用される場合、アクセス・プランが再度妥当性検査されます。動的ステートメントによって参照されるオブジェクトの権限または存在は変化する場合があるので、OPEN または EXECUTE ステートメントを出した後、SQLCODE または SQLSTATE を検査して、動的ステートメントがまだ有効であるか確かめる必要があります。

***YES:** 動的ステートメント妥当性検査を、動的ステートメントが OPEN、EXECUTE、または DESCRIBE SQL ステートメントで使用されるまで遅延させます。動的ステートメントが使用されたときは、その妥当性検査が行われて、アクセス・プランが作られます。このパラメーターで *YES を指定する場合は、

OPEN、EXECUTE、または DESCRIBE ステートメントを実行した後 SQLCODE と SQLSTATE を調べて、動的ステートメントが有効であるかどうかを確かめておく必要があります。

注: *YES を指定したときは、PREPARE ステートメントで INTO 文節が使用されている場合や、OPEN が動的ステートメントに対して出される前に DESCRIBE ステートメントがその動的ステートメントを使用した場合は、パフォーマンスは向上しません。

GENLVL

作成操作が失敗する時の重大度レベルを指定します。この値と等しいかまたはより大きい重大度レベルのエラーが発生すると、操作が終了します。

10: 省略時の重大度レベルは 10 です。

severity-level: 0 から 40 までの範囲で重大度レベルの値を指定します。

DATFMT

日付結果列にアクセスするときに使用される形式を指定します。すべての出力日付フィールドは、指定された形式で返されます。入力日付文字列については、日付が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries サーバー・システム以外のシステムにある場合は、*USA、*ISO 、*EUR、または *JIS を指定しなければなりません。

***JOB:** ジョブに指定された形式が使用されます。ジョブの現行日付形式を決定するには、ジョブ表示 (DSPJOB) コマンドを使用してください。

***USA:** 米国の日付形式 (mm/dd/yyyy) が使用されます。

***ISO:** 国際標準化機構 (ISO) の日付形式 (yyyy-mm-dd) が使用されます。

***EUR:** ヨーロッパの日付形式 (dd.mm.yyyy) が使用されます。

***JIS:** 日本工業規格の日付形式 (yyyy-mm-dd) が使用されます。

***MDY:** 日付形式 (mm/dd/yy) が使用されます。

***DMY:** 日付形式 (dd/mm/yy) が使用されます。

***YMD:** 日付形式 (yy/mm/dd) が使用されます。

***JUL:** 年間通算日の日付形式 (yy/ddd) が使用されます。

DATSEP

日付結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、*JOB、*MDY、*DMY、*YMD、または *JUL が DATFMT パラメーターで指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブに指定された日付区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

'/': スラッシュ (/) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

'-': ダッシュ (-) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK:** ブランク ( ) が使用されます。

TIMFMT
時刻結果列にアクセスするときに使用される形式を指定します。入力時刻文字列については、時刻が有効な形式で指定されているかどうかを判別するために、指定された値が使用されます。

注: *USA、*ISO、*EUR、または *JIS の形式を使用する入力日付文字列は常に有効です。

RDB パラメーターでリレーショナル・データベースを指定していて、そのリレーショナル・データベースが iSeries サーバー・システム以外のシステムにある場合、時刻形式は、時刻区切り記号がコロンまたはピリオドである *USA、*ISO、*EUR、*JIS、または *HMS でなければなりません。

***HMS:** (hh:mm:ss) 形式が使用されます。

***USA:** 米国の時刻形式 (hh:mm *xx*) が使用されます。ただし、*xx* は AM か PM です。

***ISO:** 国際標準化機構 (ISO) の時刻形式 (hh.mm.ss) が使用されます。

***EUR:** ヨーロッパの時刻形式 (hh.mm.ss) が使用されます。

***JIS:** 日本工業規格の時刻形式 (hh:mm:ss) が使用されます。

TIMSEP
時刻結果列にアクセスするときに使用される区切り記号を指定します。

注: このパラメーターは、TIMFMT パラメーターで *HMS が指定されたときだけ適用されます。

***JOB:** プリコンパイル時にジョブのために指定された時刻区切り記号が使用されます。ジョブ表示 (DSPJOB) コマンドを使用すると、ジョブの現在の値を確かめることができます。

':': コロン (:) が使用されます。

'.': ピリオド (.) が使用されます。

',': コンマ (,) が使用されます。

' ': ブランク ( ) が使用されます。

***BLANK**: ブランク ( ) が使用されます。

REPLACE

同じライブラリーに同じ名前のプログラムまたは SQL パッケージが存在するときに新しいプログラムまたは SQL パッケージが作成されるかどうかを指定します。このパラメーターの値は、CRTFTNPGM コマンドに渡されます。このパラメーターの詳細については、Information Center の「CL 解説書」の『REPLACE パラメーター』を参照してください。

***YES**: 新しいプログラムまたは SQL パッケージが作成され、指定したライブラリーの中の、同じ名前とタイプを持つ既存のプログラムまたは SQL パッケージは QRPLOBJ に移されます。

***NO**: 指定されたライブラリーに同じ名前およびタイプのオブジェクトがすでに存在している場合は、新規のプログラムまたは SQL パッケージは作成されません。

RDB

SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。

***LOCAL**: プログラムは分散 SQL プログラムとして作成されます。 SQL ステートメントはローカル・データベースにアクセスします。プリコンパイル・プロセスの一部として SQL パッケージ・オブジェクトは作成されません。 SQL パッケージの作成 (CRTSQLPKG) コマンドを使用することができます。

relational-database-name: 新しい SQL パッケージ・オブジェクトが作成されるリレーショナル・データベースの名前を指定します。ローカル・リレーショナル・データベースの名前を指定すると、作成されるプログラムは分散 SQL プログラムになります。 SQL ステートメントはローカル・データベースにアクセスします。

***NONE**: SQL パッケージ・オブジェクトは作成されません。プログラム・オブジェクトは分散プログラムではなく、 SQL パッケージの作成 (CRTSQLPKG) コマンドは使用できません。

USER

会話の開始時にリモート・システムに送られるユーザー名を指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***CURRENT**: 現在のジョブが実行されているユーザー・プロファイルが使用されます。

user-name: アプリケーション・サーバー・ジョブに使用されるユーザー名を指定します。

PASSWORD

リモート・システムで使用されるパスワードを指定します。このパラメーターは、RDB が指定されている場合にのみ有効です。

***NONE**: パスワードは送られません。この値を指定する場合は、USER(*CURRENT) も指定しなければなりません。

password: USER パラメーターに指定したユーザー名のパスワードを指定します。

RDBCNNMTH

CONNECT ステートメントに使用する意味を指定します。詳細については、「*SQL 解説書*」の CONNECT (タイプ 1) および CONNECT (タイプ 2) を参照してください。

***DUW:** 分散作業単位をサポートするために CONNECT (タイプ 2) の意味が使用されます。追加のリレーショナル・データベースへの連続する CONNECT ステートメントによって、直前の接続が切断されることはありません。

***RUW:** リモート作業単位をサポートするのに CONNECT (タイプ 1) の意味が使用されます。連続する CONNECT ステートメントによって、新しい接続が確立される前に直前の接続が切断されることになります。

DFTRDBCOL

テーブル、ビュー、索引および SQL パッケージの非修飾名に使用されるスキーマ名を指定します。このパラメーターは、静的 SQL ステートメントにのみ適用されます。

***NONE:** OPTION パラメーターで定義した命名規則が使用されます。

schema-name: スキーマ識別名を指定します。この値は、OPTION パラメーターで指定した命名規則の代わりに使用されます。

SQLPKG

このコマンドの RDB パラメーターで指定されたリレーショナル・データベースで作成される SQL パッケージの修飾名を指定します。

指定できるライブラリー値は次のとおりです。

***PGMLIB:** パッケージは、プログラムが置かれているライブラリーと同じ名前をもつライブラリーの中に作成されます。

library-name: パッケージが作成されるライブラリーの名前を指定します。

***PGM:** パッケージ名はプログラム名と同じ名前になります。

package-name: RDB パラメーターで指定したリモート・データベース上で作成されるパッケージの名前を指定します。

SAAFLAG

IBM SQL フラグ付け機能を指定します。このパラメーターは、SQL ステートメントにフラグ付けし、SQL ステートメントが IBM SQL 構文に適合しているか検査します。 IBM データベース・プロダクトの IBM SQL 構文に関する詳細は、「*DRDA IBM SQL Reference*」にあります。

***NOFLAG:** プリコンパイラーは、 SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行いません。

***FLAG:** プリコンパイラーは、SQL ステートメントが IBM SQL 構文に準拠しているかどうかの検査を行います。

FLAGSTD

米国規格協会 (ANSI) のフラグ機能を指定します。このパラメーターは SQL ステートメントにフラグ付けし、ステートメントが次の標準に準拠するかどうかを検査します。

```
ANSI X3.135-1992 entry
ISO 9075-1992 entry
FIPS 127.2 entry
```

***NONE:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行いません。

***ANS:** プリコンパイラーは、SQL ステートメントが ANSI 規格に準拠しているかどうかの検査を行います。

PRTFILE

リストが送られる先の印刷装置ファイルの修飾名を指定します。ファイルは 132 バイト以上のレコード長をもっている必要があります。そうでない場合は情報が失われます。

印刷装置ファイルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

QSYSPRT: ファイル名の指定がない場合、プリコンパイラーの印刷出力は IBM 提供の印刷装置ファイル QSYSPRT に送られます。

printer-file-name: プリコンパイラー印刷出力が送られる印刷装置ファイルの名前を指定します。

SRTSEQ

SQL ステートメントの中の文字列比較に使用される分類順序テーブルを指定します。

注: アプリケーション・サーバーが iSeries サーバー・システム上にない分散アプリケーション、またはリリース・レベルが V2R3M0 より前の分散アプリケーションでは、このパラメーターに *HEX を指定する必要があります。

***JOB:** ジョブの SRTSEQ 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの SRTSEQ 値は、プログラム実行時に検索されます。分散アプリケーションの場合、SRTSEQ(*JOBRUN) が有効なのは、LANGID(*JOBRUN) も指定されている場合だけです。

***LANGIDUNQ:** パラメーターで指定した言語用の固有の分類テーブルが使用されます。

***LANGIDSHR:** LANGID パラメーターで指定した言語用の共用分類テーブルが使用されます。

***HEX:** 分類順序テーブルは使用しません。分類順序を決定するには、文字の 16 進値を使用します。

分類順序テーブルの名前は、次のライブラリー値のいずれか 1 つで修飾することができます。

***LIBL:** 最初の一致が見つかるまで、ジョブのライブラリー・リスト内のすべてのライブラリーが探索されます。

***CURLIB:** ジョブ用の現行ライブラリーが探索されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 探索するライブラリーの名前を指定します。

table-name: 使用する分類順序テーブルの名前を指定します。

LANGID

SRTSEQ(*LANGIDUNQ) または SRTSEQ(*LANGIDSHR) が指定されたときに使用される言語識別コードを指定します。

***JOB:** ジョブの LANGID 値はプリコンパイル時に検索されます。

***JOBRUN:** ジョブの LANGID 値は、プログラム実行時に検索されます。分散アプリケーションの場合、LANGID(*JOBRUN) が有効なのは、SRTSEQ(*JOBRUN) も指定されている場合だけです。

language-id: プログラムが使用する言語識別コードを指定します。

USRPRF

コンパイル済みプログラム・オブジェクトが実行されるときに使用されるユーザー・プロファイル (プログラム・オブジェクトが静的 SQL ステートメント内の各オブジェクトに対して所有する権限を含む) を指定します。プログラム・オブジェクトが使用できるオブジェクトの制御には、プログラム所有者またはプログラム・ユーザーのプロファイルが使用されます。

***NAMING:** ユーザー・プロファイルが命名規則によって判別されます。命名規則が *SQL である場合は、USRPRF(*OWNER) が使用されます。命名規則が *SYS である場合は、USRPRF(*USER) が使用されます。

***USER:** プログラム・オブジェクトを実行するユーザーのプロファイルが使用されます。

***OWNER:** プログラム所有者とプログラム・ユーザーの両方のユーザー・プロファイルがプログラムの実行時に使用されます。

DYNUSRPRF

動的 SQL ステートメントに使用されるユーザー・プロファイルを指定します。

***USER:** ローカルな動的 SQL ステートメントは、ジョブのユーザー・プロファイルに基づいて実行されます。分散動的 SQL ステートメントはアプリケーション・サーバー・ジョブのユーザー・プロファイルに基づいて実行されます。

***OWNER:** ローカルな動的 SQL ステートメントは、プログラムの所有者のユーザー・プロファイルに基づいて実行されます。分散動的 SQL ステートメントは、SQL パッケージの所有者のユーザー・プロファイルに基づいて実行されます。

TOSRCFILE

SQL プリコンパイラーによって処理された出力ソース・メンバーを含む、ソー

ス・ファイルの修飾名を指定します。指定したソース・ファイルが見つからない場合、作成されます。出力メンバーの名前は、SRCMBR パラメーターに指定した名前と同じになります。

指定できるライブラリー値は次のとおりです。

QTEMP: ライブラリー QTEMP が使用されます。

***LIBL:** 指定したファイルがジョブのライブラリー・リストで検索されます。ライブラリー・リストに載せられているどのライブラリーにもファイルが見つからなければ、現行ライブラリー内にファイルが作成されます。

***CURLIB:** ジョブ用の現行ライブラリーが使用されます。ジョブ用の現行ライブラリーとして指定されたライブラリーがない場合は、QGPL ライブラリーが使用されます。

library-name: 出力ソース・ファイルを含むライブラリーの名前を指定します。

QSQLTEMP: ソース・ファイル QSQLTEMP が使用されます。

source-file-name: 出力ソース・メンバーを含むソース・ファイルの名前を指定します。

TEXT

LANGID を簡単に記述するテキストを指定します。このパラメーターの詳細については、Information Center の「CL 解説書」の『TEXT パラメーター』を参照してください。

***SRCMBRTXT:** テキストは FORTRAN プログラムを作成するために使用されるソース・ファイル・メンバーから取られます。データベース・ソース・メンバーのテキストを追加または変更するには、原始ステートメント入力ユーティリティー開始 (STRSEU) コマンドを使用するか、あるいは物理ファイル・メンバー追加 (ADDPFM) コマンドまたは物理ファイル・メンバー変更 (CHGPFM) コマンドを使用します。ソース・ファイルがインライン・ファイルまたは装置ファイルの場合は、テキストはブランクになります。

***BLANK:** テキストは指定されません。

'description': 50 文字以下のテキストをアポストロフィで囲んで指定します。

CRTSQLFTN コマンドの例

```
CRTSQLFTN   PAYROLL  TEXT('Payroll Program')
```

このコマンドは SQL プリコンパイラーを実行させるもので、そのプリコンパイラーはソース・プログラムをプリコンパイルし、変更したソース・プログラムをライブラリー QTEMP のファイル QSQLTEMP 内のメンバー PAYROLL の中に保管します。SQL プリコンパイラーが作成したソース・メンバーを使用して現行ライブラリーの中でプログラム PAYROLL を作成するために、FORTRAN コンパイラーが呼び出されます。

付録 D. FORTRAN アプリケーションでの SQL ステートメントのコーディング

この付録では、SQL ステートメントを FORTRAN プログラムに組み込む場合の、固有のアプリケーションおよびコーディング上の要件について説明します。ホスト変数に必要な要件についても説明します。

詳細については、以下のセクションを参照してください。

- 『FORTRAN アプリケーションでの SQL 連絡域の定義』
- 349 ページの『FORTRAN アプリケーションでの SQL 記述子域の定義』
- 349 ページの『FORTRAN アプリケーションでの SQL ステートメントの組み込み』
- 352 ページの『FORTRAN アプリケーションでのホスト変数の使用』
- 354 ページの『SQL データ・タイプと FORTRAN データ・タイプの対応関係の判別』
- 355 ページの『FORTRAN アプリケーションでの標識変数の使用』

注: コード例についての詳細は、viii ページの『コードについての特記事項』を参照してください。

FORTRAN アプリケーションでの SQL 連絡域の定義

SQL ステートメントを含んでいる FORTRAN プログラムには、次のいずれかまたは両方を持っている必要があります。

- INTEGER として宣言されている SQLCOD 変数
- CHARACTER*5 として宣言している SQLSTA (または SQLSTATE) 変数

または、

- SQLCA (SQLCOD および SQLSTA 変数が入っている)

SQLCOD および SQLSTA (または SQLSTATE) 値は、各 SQL ステートメントが実行された後、データベース・マネージャーによってセットされます。アプリケーションは、SQLCOD 値または SQLSTA (または SQLSTATE) 値を調べて、最後の SQL ステートメントが正しく実行されたかどうかを判定することができます。

SQLCA は、直後または、SQL の INCLUDE ステートメントの使用によって、FORTRAN プログラムの中にコーディングすることができます。 SQL の INCLUDE ステートメントを使用するときは、次のような標準の宣言を含める必要があります。

```
EXEC SQL INCLUDE SQLCA
```

SQLCA を組み込んだ FORTRAN ソース・ステートメントは次のとおりです。

```
*
*    The SQL communications area
*
```

```
      CHARACTER SQLCA(136)
      CHARACTER SQLCAID*8
      INTEGER*4 SQLCABC
      INTEGER*4 SQLCODE
      INTEGER*2 SQLERRML
      CHARACTER SQLERRMC*70
      CHARACTER SQLERRP*8
      INTEGER*4 SQLERRD(6)
      CHARACTER SQLWARN*11
      CHARACTER SQLSTATE*5
      EQUIVALENCE (SQLCA( 1), SQLCAID)
      EQUIVALENCE (SQLCA(  9), SQLCABC)
      EQUIVALENCE (SQLCA( 13), SQLCODE)
      EQUIVALENCE (SQLCA( 17), SQLERRML)
      EQUIVALENCE (SQLCA( 19), SQLERRMC)
      EQUIVALENCE (SQLCA( 89), SQLERRP)
      EQUIVALENCE (SQLCA( 97), SQLERRD)
      EQUIVALENCE (SQLCA(121), SQLWARN)
      EQUIVALENCE (SQLCA(132), SQLSTATE)

*
      INTEGER*4 SQLCOD
     C          SQLERR(6)
      INTEGER*2 SQLTXL
      CHARACTER SQLERP*8,
     C          SQLWRN(0:7)*1,
     C          SQLWRX(1:3)*1,
     C          SQLTXT*70,
     C          SQLSTT*5,
     C          SQLWRNWK*8,
     C          SQLWRXWK*3,
     C          SQLERRWK*24,
     C          SQLERRDWK*24
      EQUIVALENCE (SQLWRN(1), SQLWRNWK)
      EQUIVALENCE (SQLWRX(1), SQLWRXWK)
      EQUIVALENCE (SQLCA(97), SQLERRDWK)
      EQUIVALENCE (SQLERR(1), SQLERRWK)
      COMMON /SQLCA1/SQLCOD,SQLERR,SQLTXTL
      COMMON /SQLCA2/SQLERP,SQLWRN,SQLTXT,SQLWRX,SQLSTT
```

SQLSTATE の宣言がプログラムの中で検出され、コンパイラーが SQLCA を提供している場合は、SQLSTATE は SQLSTOTE に置き換えられます。他の IBM SQL のインプリメンテーションとの互換性を第一に考慮しなくてもよい場合は、プログラムに FORTRAN 変数の SQLCOD、SQLSTA、または SQLSTATE をコーディングすることによって、SQLCA を組み込むことをお勧めします。これにより、パフォーマンスが向上しますが、互換性のある SQLCA は生成されません。

SQLCA の詳細については、「SQL 解説書」の『SQL 連絡域』を参照してください。

SQLCOD、SQLSTA、SQLSTATE、および SQLCA の各変数は、最初の実行可能な SQL ステートメントの前に置かなければなりません。プログラムで実行可能なすべての SQL ステートメントは、SQLCOD、SQLSTA、SQLSTATE、および SQLCA の各変数の宣言の有効範囲内になければなりません。

プログラムで実行可能なすべての SQL ステートメントは、SQLCOD 変数または SQLCA 変数の宣言の有効範囲内になければなりません。

FORTRAN アプリケーションでの SQL 記述子域の定義

SQLDA を必要とするステートメントには、次のものがあります。

EXECUTE...USING DESCRIPTOR *記述子名*

FETCH...USING DESCRIPTOR *記述子名*

OPEN...USING DESCRIPTOR *記述子名*

CALL...USING DESCRIPTOR *記述子名*

DESCRIBE *ステートメント名* INTO *記述子名*

DESCRIBE TABLE *ホスト変数* INTO *記述子名*

PREPARE *ステートメント名* INTO *記述子名*

SQLCA と異なり、SQLDA を 2 つ以上プログラムの中に置くことができ、また SQLDA の名前は有効であれば、どの名前でも使えます。

動的 SQL は高度なプログラミング技法です。これについては、「*DB2 UDB for iSeries SQL プログラミング概念*」の『動的 SQL アプリケーション』で説明します。動的 SQL を使用すると、ユーザーのプログラムはその実行と平行して SQL ステートメントを作成し、実行させることができます。動的に実行される変数 SELECT リスト (すなわち、照会の一部として返されるデータのリスト) を指定する SELECT ステートメントには、SQL 記述子域 (SQLDA) が必要です。これは、SELECT の結果を受け入れるために割り振るべき変数の数とタイプが事前に予測できないからです。SQLDA ではポインター変数を使用しますが、これは FORTRAN ではサポートされないので、INCLUDE SQLDA ステートメントは FORTRAN プログラムでは指定できません。SQLDA を C、COBOL、PL/I、または ILE RPG の各プログラムでセットアップして FORTRAN プログラムに渡さない限り、SQLDA を使用することはできません。

SQLDA の詳細については、「*SQL 解説書*」の『SQL 記述子域』を参照してください。

行記憶域を使用して複数行用 FETCH ステートメントに SQLDA をコーディングすれば、各 FETCH ステートメントごとに複数の行を取り出すことができます。この方法は、多くの行がアプリケーションによって読み取られる場合には、そのアプリケーションのパフォーマンスを向上させます。 FETCH ステートメントの使用に関する詳細については、「SQL 解説書」を参照してください。

FORTRAN アプリケーションでの SQL ステートメントの組み込み

SQL ステートメントは、実行可能なステートメントが置かれる個所ならば、FORTRAN プログラム内のどこにでもコーディングすることができます。 SQL ステートメントが IF ステートメント内にある場合は、必要とされる THEN ステートメントと END IF ステートメントが生成されます。

FORTRAN プログラムの中の SQL ステートメントはいずれも、EXEC SQL で始まっていなければなりません。 EXEC SQL キーワードはいずれも 1 行に置かなければなりませんが、ステートメントの残りの部分はキーワードと同じ行およびそれ以降の行に置くことができます。

例:

FORTRAN プログラムの中にコーディングされる UPDATE ステートメントをコーディングすると、次のようになります。

```
      EXEC SQL
     C  UPDATE DEPARTMENT
     C  SET MGRNO = :MGRNUM
     C  WHERE DEPTNO = :INTDEPT
```

どの SQL ステートメントの場合も、同じ行に別の SQL ステートメントや FORTRAN ステートメントを続けることはできません。

FORTRAN では、ステートメント内の語を区切るためにブランクの使用は必須ではありませんが、SQL 言語では必須です。組み込み SQL の場合の規則は SQL 構文の規則に従うので、区切り文字として 1 つ以上のブランクを使用する必要があります。

詳細については、以下のセクションを参照してください。

- 『SQL を使用する FORTRAN アプリケーションでの注記』
- 『SQL を使用する FORTRAN アプリケーションでのデバッグ行』
- 351 ページの『SQL を使用する FORTRAN アプリケーションでの SQL ステートメントの継続』
- 351 ページの『SQL を使用する FORTRAN アプリケーションでのコードの組み込み』
- 351 ページの『SQL を使用する FORTRAN アプリケーションでのマージン』
- 351 ページの『SQL を使用する FORTRAN アプリケーションでの名前』
- 352 ページの『SQL を使用する FORTRAN アプリケーションでのステートメント・ラベル』
- 352 ページの『SQL を使用する FORTRAN アプリケーションでの WHENEVER ステートメント』
- 352 ページの『SQL プリコンパイラーでの FORTRAN コンパイル時オプション』

SQL を使用する FORTRAN アプリケーションでの注記

SQL の注記 (--) の他に、FORTRAN の注記は、組み込み SQL ステートメント内のブランクが許される場所にはどこにでも入れることができます。ただし、キーワードの EXEC と SQL の間には入れられません。

注記はその行の終わりまで書き込むことができます。注記行は、継続する SQL ステートメントの行と行の間に置くことができます。文字 (!) は、文字文脈または 6 桁目に現れる場合を除いて、注記を表します。

SQL を使用する FORTRAN アプリケーションでのデバッグ行

デバッグ・ステートメントを含む行 (1 桁目が 'D' または 'd' の行) は、プリコンパイラーによって注記行として扱われます。

SQL を使用する FORTRAN アプリケーションでの SQL ステートメントの継続

SQL ステートメントの行の継続に関する規則は、EXEC SQL を 1 行以内に指定する点を除けば、他の FORTRAN ステートメントの場合と同じです。

DBCS データを含む定数は、継続される行の 73 桁目にシフトイン文字を入れ、継続行の 6 桁目にシフトアウト文字を入れることによって、複数行にわたって継続させることができます。

この SQL ステートメントの G'<AABBCCDDEEFFGGHHIIJJKK>' はグラフィック定数として有効です。

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

SQL を使用する FORTRAN アプリケーションでのコードの組み込み

SQL ステートメントまたは FORTRAN ステートメントは、ステートメントを組み込もうとするソース・コード内の個所に次の SQL ステートメントを組み込むことによって、挿入することができます。

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

FORTRAN INCLUDE コンパイラー・ディレクティブは、SQL ステートメントまたは SQL ステートメントの中で使用される FORTRAN ホスト変数宣言を組み込むためには使用できません。

SQL を使用する FORTRAN アプリケーションでのマージン

SQL ステートメント (EXEC SQL で始まる) は 7 桁目から 72 桁目にコーディングしてください。

SQL を使用する FORTRAN アプリケーションでの名前

有効な FORTRAN 変数名であれば、どの名前でもホスト変数に使えますが、次のような制約を受けます。

'SQ'、'SQL'、'RDI'、または 'DSN' で始まるホスト変数名や外部入り口名は、使用してはなりません。これらの名前はデータベース・マネージャー用に予約されています。

ホスト変数を識別するために次のキーワードは使用してはなりません。

FUNCTION
IMPLICIT
PROGRAM
SUBROUTINE

SQL を使用する FORTRAN アプリケーションでのステートメント・ラベル

実行可能な SQL ステートメントにステートメント番号を付けて、それを 1 桁目から 5 桁目までに指定することができます。ただし、プログラムの作成時にラベル付きの SQL ステートメントを使用すると、コードがそのステートメントを実行する前にそのラベルが付いた CONTINUE ステートメントが生成されます。ラベル付きの SQL ステートメントを DO ループの最後のステートメントにしてはなりません。 CONTINUE ステートメントは実行可能なステートメントであるので、FORTRAN プログラムで実行可能な最初のステートメントよりも前に置かれている SQL ステートメント (たとえば、INCLUDE や BEGIN DECLARE SECTION) には、ラベルを付けないようにしてください。

SQL を使用する FORTRAN アプリケーションでの WHENEVER ステートメント

SQL の WHENEVER ステートメントの GOTO 文節の対象となるものは、FORTRAN ソース・コード内のラベルであって、同じサブプログラム内のステートメントを参照していなければなりません。 WHENEVER ステートメントは同じサブプログラムの中の SQL ステートメントにだけ適用されます。

SQL プリコンパイラーでの FORTRAN コンパイル時オプション

FORTRAN PROCESS ステートメントを使用すると、FORTRAN コンパイラーのコンパイル時オプションを指定することができます。 PROCESS ステートメントは、プログラムを作成するためにプリコンパイラーによって呼び出されるとき、FORTRAN コンパイラーによって認識されますが、SQL プリコンパイラー自体は PROCESS ステートメントを認識しません。

FORTRAN アプリケーションでのホスト変数の使用

SQL ステートメントの中で使用するホスト変数はいずれも明示的に宣言しなければなりません。省略時のタイプまたは IMPLICIT ステートメントによるホスト変数の暗黙の宣言はサポートされません。 SQL ステートメントの中で使用するホスト変数は、SQL ステートメントの中でホスト変数を初めて使用する前に、宣言しておかなければなりません。

ホスト変数を定義するために使用される FORTRAN ステートメントは、その前に BEGIN DECLARE SECTION ステートメントを置き、その後に END DECLARE SECTION ステートメントを置く必要があります。 BEGIN DECLARE SECTION と END DECLARE SECTION を指定した場合、SQL ステートメントで使用するすべてのホスト変数宣言は、BEGIN DECLARE SECTION ステートメントと END DECLARE SECTION ステートメントの間になければなりません。注: LOB ホスト変数および ROWID ホスト変数は、FORTRAN ではサポートされません。

SQL ステートメントの中のホスト変数はいずれも、その前にコロン (:) を付けなければなりません。

ホスト変数の名前は、ホスト変数がそれぞれ異なるブロックやプロシージャー内にある場合であっても、プログラム内では固有になっていなければなりません。

文字のホスト変数の宣言では、その文字変数の長さを定義するために式を使用してはいけません。文字のホスト変数の宣言では、長さが未定義になっていてはなりません (たとえば、CHARACTER(*))。

ホスト変数を使用する SQL ステートメントは、その変数が宣言されたステートメントの有効範囲内になければなりません。

ホスト変数はスカラー変数でなければなりません。これらは、配列の要素 (添え字付き変数) にすることはできません。

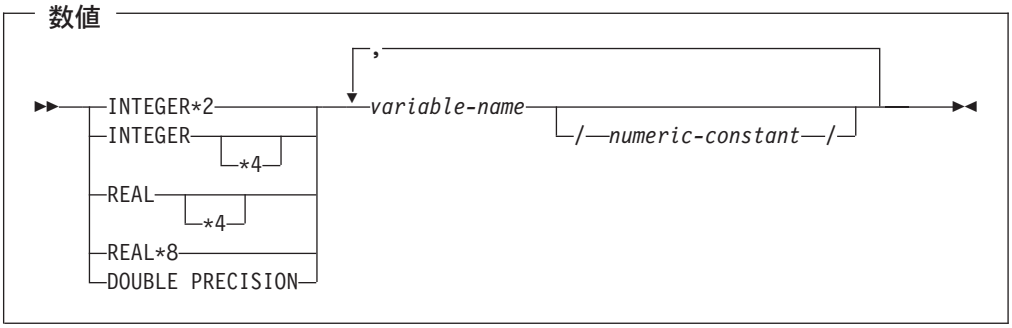
詳細については、『FORTRAN アプリケーションでのホスト変数の宣言』を参照してください。

FORTRAN アプリケーションでのホスト変数の宣言

FORTRAN プリコンパイラーは、有効な FORTRAN 宣言のサブセットだけを有効なホスト変数宣言として認識します。

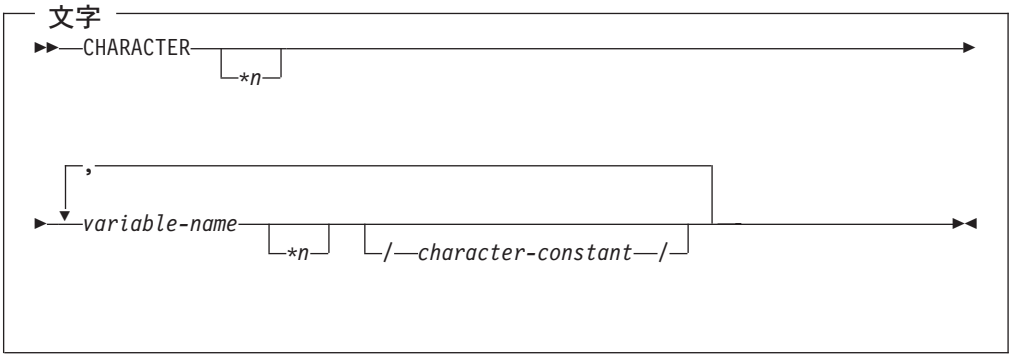
FORTRAN アプリケーションでの数値ホスト変数

下図は、有効な数値ホスト変数宣言の構文を示しています。



FORTRAN アプリケーションでの文字ホスト変数

下図は、有効な文字ホスト変数宣言の構文を示しています。



注: n は 32766 より大きくない定数でなければなりません。

SQL データ・タイプと FORTRAN データ・タイプの対応関係の判別

プリコンパイラーは、次の表に基づいて、ホスト変数のベース SQLTYPE とベース SQLLEN を判断します。ホスト変数が標識変数と一緒に記載されているときは、その SQLTYPE はベース SQLTYPE に 1 を加えたものです。

表 12. FORTRAN 宣言と代表的 SQL データ・タイプとの対応関係

FORTRAN データ・タイプ	ホスト変数の SQLTYPE	ホスト変数の SQLLEN	SQL データ・タイプ
INTEGER*2	500	2	SMALLINT
INTEGER*4	496	4	INTEGER
REAL*4	480	4	FLOAT (単精度)
REAL*8	480	8	FLOAT (倍精度)
CHARACTER*n	452	n	CHAR(n)

下表を参照すると、各 SQL データ・タイプに対応する FORTRAN データ・タイプを判別することができます。

表 13. SQL データ・タイプと代表的な FORTRAN 宣言との対応関係

SQL データ・タイプ	対応する FORTRAN 宣言	説明と注
SMALLINT	INTEGER*2	
INTEGER	INTEGER*4	
BIGINT	正確に対応するものなし	REAL*8 を使用。
DECIMAL(p,s) または NUMERIC(p,s)	正確に対応するものなし	REAL*8 を使用。
FLOAT (単精度)	REAL*4	
FLOAT (倍精度)	REAL*8	
CHAR(n)	CHARACTER*n	n は 1 から 32766 までの正の整数です。
VARCHAR(n)	正確に対応するものなし	予想される最大の VARCHAR 値が収まるだけの大きさの文字ホスト変数を使用してください。
GRAPHIC(n)	サポートなし	サポートなし
VARGRAPHIC(n)	サポートなし	サポートなし
DATE	CHARACTER*n	形式が *USA、*JIS、*EUR、または *ISO のときは、n は少なくとも 10 文字にする必要があります。形式が *YMD、*DMY、または *MDY のときは、n は少なくとも 8 文字にする必要があります。形式が *JUL のときは、n は少なくとも 6 文字にする必要があります。

表 13. SQL データ・タイプと代表的な FORTRAN 宣言との対応関係 (続き)

SQL データ・タイプ	対応する FORTRAN 宣言	説明と注
TIME	CHARACTER*n	*n* は少なくとも 6 文字に、秒を含める場合は、 *n* は少なくとも 8 文字にする必要があります。
TIMESTAMP	CHARACTER*n	n は少なくとも 19 が必要。マイクロ秒を全桁の精度で含める場合は、n は少なくとも 26 が必要。n が 26 未満のときは、マイクロ秒部分で切り捨てが起こる。

詳細については、『FORTRAN 変数宣言と使用法に関する注意事項』を参照してください。

FORTRAN 変数宣言と使用法に関する注意事項

FORTRAN では、小数点を含む数字の文字列は、実定数として解釈されます。 SQL ステートメントでは、このような数字の文字列は 10 進定数として解釈されます。そのために、SQL ステートメントで実 (浮動小数点) 定数を指定するときは、指数表記を使用します。

FORTRAN では、長さが 8 バイトの実 (浮動小数点) 定数には、指数標識に D を使用します (たとえば、3.14159D+04)。 SQL ステートメントでの 8 バイト浮動小数点定数には E を使用します (たとえば、3.14159E+04)。

FORTRAN アプリケーションでの標識変数の使用

標識変数は 2 バイトの整数です (INTEGER*2)。検索されるとき、標識変数はその対応するホスト変数にヌル値が割り当てられているかどうかを示すために使用されます。列に割り当てるときには、ヌル値を割り当てるべきであることを示すために負の標識変数が使用されます。

詳細については、「SQL 解説書」の『標識変数』を参照してください。

標識変数の宣言の仕方は、ホスト変数の場合と同じです。これらの 2 つの変数の宣言は、プログラマーに適切と思われる方法で組み合わせることができます。

例:

次のステートメントがあるとします。

```
    EXEC SQL FETCH CLS_CURSOR INTO :CLS_CD,
   C                              :DAY :DAY_IND,
   C                              :BGN :BGN_IND,
   C                              :ENDCLS :ENDCLS_IND
```

変数は次のように宣言することができます。

```
EXEC SQL BEGIN DECLARE SECTION
 CHARACTER*7 CLS_CD
 INTEGER*2   DAY
 CHARACTER*8 BGN, ENDCLS
 INTEGER*2   DAY_IND, BGN_IND, ENDCLS_IND
EXEC SQL END DECLARE SECTION
```

索引

日本語, 数字, 英字, 特殊文字の順に配列されています。なお, 濁音と半濁音は清音と同等に扱われています。

[ア行]

[カ行]

[サ行]

[タ行]

[ナ行]

[ハ行]

［マ行］

[ラ行]

[ワ行]

[数字]

B

C

D

E

F

I

L

M

N

P

Q

R

S

T

U

V

W

[特殊文字]

Printed in Japan

日本アイ・ビー・エム株式会社
〒106-8711　東京都港区六本木3-2-12