This file contains a depth-first tree traversal of the BNF
for the language done at about 27-AUG-1992 11:03:41.64.
The specific version of the BNF included here is: ANSI-only, SQL2-only.

<SQL terminal character> ::=
<SQL language character>
| <SQL embedded language character>

<SQL language character> ::=
<simple Latin letter>
| <digit>
| <SQL special character>

<simple Latin letter> ::=
<simple Latin upper case letter>
| <simple Latin lower case letter>

<simple Latin upper case letter> ::=
A | B | C | D | E | F | G | H | I | J | K | L | M | N | O
| P | Q | R | S | T | U | V | W | X | Y | Z

<simple Latin lower case letter> ::=
a | b | c | d | e | f | g | h | i | j | k | l | m | n | o
| p | q | r | s | t | u | v | w | x | y | z

<digit> ::=
0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<SQL special character> ::=
<space>
| <double quote>
| <percent>
| <ampersand>
| <quote>
| <left paren>
| <right paren>
| <asterisk>
| <plus sign>
| <comma>
| <minus sign>
| <period>
| <solidus>
| <colon>
| <semicolon>
| <less than operator>
| <equals operator>
| <greater than operator>
| <question mark>
| <underscore>
| <vertical bar>

<space> ::= !! <EMPHASIS>(space character in character set in use)

<double quote> ::= "

<percent> ::= %

<ampersand> ::= &

<quote> ::= '

<left paren> ::= (

<right paren> ::=)

<asterisk> ::= *

<plus sign> ::= +

<comma> ::= ,

<minus sign> ::= -

<period> ::= .

<solidus> ::= /

<colon> ::= :

<semicolon> ::= ;

<less than operator> ::= <

<equals operator> ::= =

<greater than operator> ::= >

<question mark> ::= ?

<underscore> ::= _

<vertical bar> ::= |

<SQL embedded language character> ::=
<left bracket>
| <right bracket>

<left bracket> ::= [

<right bracket> ::=]

<token> ::=
<nondelimiter token>
| <delimiter token>

<nondelimiter token> ::=
<regular identifier>
| <key word>
| <unsigned numeric literal>
| <national character string literal>
| <bit string literal>
| <hex string literal>

<regular identifier> ::= <identifier body>

<identifier body> ::=
<identifier start> [{ <underscore> | <identifier part> }...]

<identifier start> ::= <EMPHASIS>(!! See the Syntax Rules)

<identifier part> ::=
<identifier start>
| <digit>

<key word> ::=
<reserved word>
| <non-reserved word>

<reserved word> ::=
ABSOLUTE | ACTION | ADD | ALL
| ALLOCATE | ALTER | AND
| ANY | ARE
| AS | ASC
| ASSERTION | AT
| AUTHORIZATION | AVG
| BEGIN | BETWEEN | BIT | BIT_LENGTH
| BOTH | BY
| CASCADE | CASCADED | CASE | CAST
| CATALOG
| CHAR | CHARACTER | CHAR_LENGTH
| CHARACTER_LENGTH | CHECK | CLOSE | COALESCE
| COLLATE | COLLATION
| COLUMN | COMMIT
| CONNECT
| CONNECTION | CONSTRAINT
| CONSTRAINTS | CONTINUE
| CONVERT | CORRESPONDING | COUNT | CREATE | CROSS
| CURRENT
| CURRENT_DATE | CURRENT_TIME
| CURRENT_TIMESTAMP | CURRENT_USER | CURSOR
| DATE | DAY | DEALLOCATE | DEC
| DECIMAL | DECLARE | DEFAULT | DEFERRABLE
| DEFERRED | DELETE | DESC | DESCRIBE | DESCRIPTOR
| DIAGNOSTICS
| DISCONNECT | DISTINCT | DOMAIN | DOUBLE | DROP
| ELSE | END | END-EXEC | ESCAPE
| EXCEPT | EXCEPTION
| EXEC | EXECUTE | EXISTS
| EXTERNAL | EXTRACT
| FALSE | FETCH | FIRST | FLOAT | FOR
| FOREIGN | FOUND | FROM | FULL
| GET | GLOBAL | GO | GOTO
| GRANT | GROUP

| HAVING | HOUR
| IDENTITY | IMMEDIATE | IN | INDICATOR
| INITIALLY | INNER | INPUT
| INSENSITIVE | INSERT | INT | INTEGER | INTERSECT
| INTERVAL | INTO | IS
| ISOLATION
| JOIN
| KEY
| LANGUAGE | LAST | LEADING | LEFT
| LEVEL | LIKE | LOCAL | LOWER
| MATCH | MAX | MIN | MINUTE | MODULE
| MONTH
| NAMES | NATIONAL | NATURAL | NCHAR | NEXT | NO
| NOT | NULL
| NULLIF | NUMERIC
| OCTET_LENGTH | OF
| ON | ONLY | OPEN | OPTION | OR
| ORDER | OUTER
| OUTPUT | OVERLAPS
| PAD | PARTIAL | POSITION | PRECISION | PREPARE
| PRESERVE | PRIMARY
| PRIOR | PRIVILEGES | PROCEDURE | PUBLIC
| READ | REAL | REFERENCES | RELATIVE | RESTRICT
| REVOKE | RIGHT
| ROLLBACK | ROWS
| SCHEMA | SCROLL | SECOND | SECTION
| SELECT
| SESSION | SESSION_USER | SET
| SIZE | SMALLINT | SOME | SPACE | SQL | SQLCODE
| SQLERROR | SQLSTATE
| SUBSTRING | SUM | SYSTEM_USER
| TABLE | TEMPORARY
| THEN | TIME | TIMESTAMP
| TIMEZONE_HOUR | TIMEZONE_MINUTE
| TO | TRAILING | TRANSACTION
| TRANSLATE | TRANSLATION | TRIM | TRUE
| UNION | UNIQUE | UNKNOWN | UPDATE | UPPER | USAGE
| USER | USING
| VALUE | VALUES | VARCHAR | VARYING | VIEW
| WHEN | WHENEVER | WHERE | WITH | WORK | WRITE
| YEAR
| ZONE

<non-reserved word> ::=

ADA
| C | CATALOG_NAME
| CHARACTER_SET_CATALOG | CHARACTER_SET_NAME
| CHARACTER_SET_SCHEMA | CLASS_ORIGIN | COBOL |
COLLATION_CATALOG
| COLLATION_NAME | COLLATION_SCHEMA | COLUMN_NAME |
COMMAND_FUNCTION
| COMMITTED
| CONDITION_NUMBER | CONNECTION_NAME | CONSTRAINT_CATALOG
| CONSTRAINT_NAME
| CONSTRAINT_SCHEMA | CURSOR_NAME
| DATA | DATETIME_INTERVAL_CODE
| DATETIME_INTERVAL_PRECISION | DYNAMIC_FUNCTION
| FORTRAN
| LENGTH
| MESSAGE_LENGTH | MESSAGE_OCTET_LENGTH | MESSAGE_TEXT |
MORE | MUMPS
| NAME | NULLABLE | NUMBER
| PASCAL | PLI
| REPEATABLE | RETURNED_LENGTH | RETURNED_OCTET_LENGTH |
RETURNED_SQLSTATE
| ROW_COUNT
| SCALE | SCHEMA_NAME | SERIALIZABLE | SERVER_NAME |
SUBCLASS_ORIGIN
| TABLE_NAME | TYPE
| UNCOMMITTED | UNNAMED

<unsigned numeric literal> ::=
<exact numeric literal>
| <approximate numeric literal>

<exact numeric literal> ::=
<unsigned integer> [<period> [<unsigned integer>]]
| <period> <unsigned integer>

<unsigned integer> ::= <digit>...

<approximate numeric literal> ::= <mantissa> E <exponent>

<mantissa> ::= <exact numeric literal>

<exponent> ::= <signed integer>

<signed integer> ::= [<sign>] <unsigned integer>

<sign> ::= <plus sign> | <minus sign>

<national character string literal> ::=
N <quote> [<character representation>...] <quote>
[{ <separator>... <quote> [<character representation>...] <quote> }...]

<character representation> ::=
<nonquote character>
| <quote symbol>

<nonquote character> ::= !! <EMPHASIS>(See the Syntax Rules.)

<quote symbol> ::= <quote><quote>

<separator> ::= { <comment> | <space> | <newline> }...

<comment> ::=
<comment introducer> [<comment character>...] <newline>

<comment introducer> ::= <minus sign><minus sign>[<minus sign>...]

<comment character> ::=
<nonquote character>
| <quote>

<newline> ::= !! <EMPHASIS>(implementation-defined end-of-line indicator)

<bit string literal> ::=
B <quote> [<bit>...] <quote>
[{ <separator>... <quote> [<bit>...] <quote> }...]

<bit> ::= 0 | 1

<hex string literal> ::=
X <quote> [<hexit>...] <quote>
[{ <separator>... <quote> [<hexit>...] <quote> }...]

<hexit> ::= <digit> | A | B | C | D | E | F | a | b | c | d | e | f

<delimiter token> ::=
<character string literal>
| <date string>
| <time string>
| <timestamp string>
| <interval string>
| <delimited identifier>
| <SQL special character>
| <not equals operator>
| <greater than or equals operator>
| <less than or equals operator>
| <concatenation operator>
| <double period>
| <left bracket>
| <right bracket>

<character string literal> ::=
[<introducer><character set specification>]
<quote> [<character representation>...] <quote>
[{ <separator>... <quote> [<character representation>...] <quote> }...]

<introducer> ::= <underscore>

<character set specification> ::=
<standard character repertoire name>
| <implementation-defined character repertoire name>
| <user-defined character repertoire name>
| <standard universal character form-of-use name>
| <implementation-defined universal character form-of-use name>

<standard character repertoire name> ::= <character set name>

<character set name> ::= [<schema name> <period>]
<SQL language identifier>

```
<schema name> ::=
    [ <catalog name> <period> ] <unqualified schema name>

<catalog name> ::= <identifier>

<identifier> ::=
    [ <introducer><character set specification> ] <actual identifier>

<actual identifier> ::=
      <regular identifier>
    | <delimited identifier>

<delimited identifier> ::=
    <double quote> <delimited identifier body> <double quote>

<delimited identifier body> ::= <delimited identifier part>...

<delimited identifier part> ::=
      <nondoublequote character>
    | <doublequote symbol>

<nondoublequote character> ::= <EMPHASIS>(!! See the Syntax Rules)

<doublequote symbol> ::= <double quote><double quote>

<unqualified schema name> ::= <identifier>

<SQL language identifier> ::=
    <SQL language identifier start>
      [ { <underscore> | <SQL language identifier part> }... ]

<SQL language identifier start> ::= <simple Latin letter>

<SQL language identifier part> ::=
      <simple Latin letter>
    | <digit>

<implementation-defined character repertoire name> ::=
    <character set name>

<user-defined character repertoire name> ::= <character set name>

<standard universal character form-of-use name> ::=
    <character set name>

<implementation-defined universal character form-of-use name> ::=
    <character set name>

<date string> ::=
    <quote> <date value> <quote>

<date value> ::=
    <years value> <minus sign> <months value>
      <minus sign> <days value>

<years value> ::= <datetime value>

<datetime value> ::= <unsigned integer>

<months value> ::= <datetime value>

<days value> ::= <datetime value>

<time string> ::=
    <quote> <time value> [ <time zone interval> ] <quote>

<time value> ::=
    <hours value> <colon> <minutes value> <colon> <seconds value>

<hours value> ::= <datetime value>

<minutes value> ::= <datetime value>

<seconds value> ::=
      <seconds integer value> [ <period> [ <seconds fraction> ] ]

<seconds integer value> ::= <unsigned integer>

<seconds fraction> ::= <unsigned integer>

<time zone interval> ::=
    <sign> <hours value> <colon> <minutes value>

<timestamp string> ::=
    <quote> <date value> <space> <time value>
      [ <time zone interval> ] <quote>

<interval string> ::=
    <quote> { <year-month literal> | <day-time literal> } <quote>

<year-month literal> ::=
      <years value>
    | [ <years value> <minus sign> ] <months value>

<day-time literal> ::=
      <day-time interval>
    | <time interval>

<day-time interval> ::=
    <days value>
      [ <space> <hours value> [ <colon> <minutes value>
        [ <colon> <seconds value> ] ] ]

<time interval> ::=
      <hours value> [ <colon> <minutes value> [ <colon> <seconds value> ] ]
    | <minutes value> [ <colon> <seconds value> ]
    | <seconds value>

<not equals operator> ::= <>

<greater than or equals operator> ::= >=

<less than or equals operator> ::= <=

<concatenation operator> ::= ||

<double period> ::= ..

<module> ::=
    <module name clause>
    <language clause>
    <module authorization clause>
    [ <temporary table declaration>... ]
    <module contents>...

<module name clause> ::=
    MODULE [ <module name> ]
      [ <module character set specification> ]

<module name> ::= <identifier>

<module character set specification> ::=
    NAMES ARE <character set specification>

<language clause> ::=
    LANGUAGE <language name>

<language name> ::=
    ADA | C | COBOL | FORTRAN | MUMPS | PASCAL | PLI

<module authorization clause> ::=
      SCHEMA <schema name>
    | AUTHORIZATION <module authorization identifier>
    | SCHEMA <schema name>
          AUTHORIZATION <module authorization identifier>

<module authorization identifier> ::=
    <authorization identifier>

<authorization identifier> ::= <identifier>

<temporary table declaration> ::=
    DECLARE LOCAL TEMPORARY TABLE
      <qualified local table name>
      <table element list>
      [ ON COMMIT { PRESERVE | DELETE } ROWS ]

<qualified local table name> ::=
    MODULE <period> <local table name>

<local table name> ::= <qualified identifier>

<qualified identifier> ::= <identifier>
```

```
<table element list> ::=
        <left paren> <table element> [ { <comma> <table element> }... ] <right
paren>

<table element> ::=
        <column definition>
      | <table constraint definition>

<column definition> ::=
      <column name> { <data type> | <domain name> }
      [ <default clause> ]
      [ <column constraint definition>... ]
      [ <collate clause> ]

<column name> ::= <identifier>

<data type> ::=
        <character string type>
              [ CHARACTER SET <character set specification> ]
      | <national character string type>
      | <bit string type>
      | <numeric type>
      | <datetime type>
      | <interval type>

<character string type> ::=
        CHARACTER [ <left paren> <length> <right paren> ]
      | CHAR [ <left paren> <length> <right paren> ]
      | CHARACTER VARYING <left paren> <length> <right paren>
      | CHAR VARYING <left paren> <length> <right paren>
      | VARCHAR <left paren> <length> <right paren>

<length> ::= <unsigned integer>

<national character string type> ::=
        NATIONAL CHARACTER [ <left paren> <length> <right paren> ]
      | NATIONAL CHAR [ <left paren> <length> <right paren> ]
      | NCHAR [ <left paren> <length> <right paren> ]
      | NATIONAL CHARACTER VARYING <left paren> <length> <right paren>
      | NATIONAL CHAR VARYING <left paren> <length> <right paren>
      | NCHAR VARYING <left paren> <length> <right paren>

<bit string type> ::=
        BIT [ <left paren> <length> <right paren> ]
      | BIT VARYING <left paren> <length> <right paren>

<numeric type> ::=
        <exact numeric type>
      | <approximate numeric type>

<exact numeric type> ::=
        NUMERIC [ <left paren> <precision> [ <comma> <scale> ] <right paren> ]
      | DECIMAL [ <left paren> <precision> [ <comma> <scale> ] <right paren> ]
      | DEC [ <left paren> <precision> [ <comma> <scale> ] <right paren> ]
      | INTEGER
      | INT
      | SMALLINT

<precision> ::= <unsigned integer>

<scale> ::= <unsigned integer>

<approximate numeric type> ::=
        FLOAT [ <left paren> <precision> <right paren> ]
      | REAL
      | DOUBLE PRECISION

<datetime type> ::=
        DATE
      | TIME [ <left paren> <time precision> <right paren> ]
              [ WITH TIME ZONE ]
      | TIMESTAMP [ <left paren> <timestamp precision> <right paren> ]
              [ WITH TIME ZONE ]

<time precision> ::= <time fractional seconds precision>

<time fractional seconds precision> ::= <unsigned integer>

<timestamp precision> ::= <time fractional seconds precision>

<interval type> ::= INTERVAL <interval qualifier>

<interval qualifier> ::=
        <start field> TO <end field>
      | <single datetime field>

<start field> ::=
      <non-second datetime field>
            [ <left paren> <interval leading field precision> <right paren> ]

<non-second datetime field> ::= YEAR | MONTH | DAY | HOUR
      | MINUTE

<interval leading field precision> ::= <unsigned integer>

<end field> ::=
        <non-second datetime field>
      | SECOND [ <left paren> <interval fractional seconds precision> <right
paren> ]

<interval fractional seconds precision> ::= <unsigned integer>

<single datetime field> ::=
        <non-second datetime field>
              [ <left paren> <interval leading field precision> <right paren> ]
      | SECOND [ <left paren> <interval leading field precision>
              [ <comma> <interval fractional seconds precision> ] <right paren> ]

<domain name> ::= <qualified name>

<qualified name> ::=
      [ <schema name> <period> ] <qualified identifier>

<default clause> ::=
        DEFAULT <default option>

<default option> ::=
        <literal>
      | <datetime value function>
      | USER
      | CURRENT_USER
      | SESSION_USER
      | SYSTEM_USER
      | NULL

<literal> ::=
        <signed numeric literal>
      | <general literal>

<signed numeric literal> ::=
      [ <sign> ] <unsigned numeric literal>

<general literal> ::=
        <character string literal>
      | <national character string literal>
      | <bit string literal>
      | <hex string literal>
      | <datetime literal>
      | <interval literal>

<datetime literal> ::=
        <date literal>
      | <time literal>
      | <timestamp literal>

<date literal> ::=
      DATE <date string>

<time literal> ::=
      TIME <time string>

<timestamp literal> ::=
      TIMESTAMP <timestamp string>

<interval literal> ::=
      INTERVAL [ <sign> ] <interval string> <interval qualifier>

<datetime value function> ::=
        <current date value function>
      | <current time value function>
      | <current timestamp value function>

<current date value function> ::= CURRENT_DATE
```

```
<current time value function> ::=
        CURRENT_TIME [ <left paren> <time precision> <right paren> ]

<current timestamp value function> ::=
        CURRENT_TIMESTAMP [ <left paren> <timestamp precision> <right
paren> ]

<column constraint definition> ::=
      [ <constraint name definition> ]
      <column constraint>
        [ <constraint attributes> ]

<constraint name definition> ::= CONSTRAINT <constraint name>

<constraint name> ::= <qualified name>

<column constraint> ::=
        NOT NULL
      | <unique specification>
      | <references specification>
      | <check constraint definition>

<unique specification> ::=
      UNIQUE | PRIMARY KEY

<references specification> ::=
      REFERENCES <referenced table and columns>
        [ MATCH <match type> ]
        [ <referential triggered action> ]

<referenced table and columns> ::=
        <table name> [ <left paren> <reference column list> <right paren> ]

<table name> ::=
        <qualified name>
      | <qualified local table name>

<reference column list> ::= <column name list>

<column name list> ::=
      <column name> [ { <comma> <column name> }... ]

<match type> ::=
        FULL
      | PARTIAL

<referential triggered action> ::=
        <update rule> [ <delete rule> ]
      | <delete rule> [ <update rule> ]

<update rule> ::= ON UPDATE <referential action>

<referential action> ::=
        CASCADE
      | SET NULL
      | SET DEFAULT
      | NO ACTION

<delete rule> ::= ON DELETE <referential action>

<check constraint definition> ::=
      CHECK
          <left paren> <search condition> <right paren>

<search condition> ::=
        <boolean term>
      | <search condition> OR <boolean term>

<boolean term> ::=
        <boolean factor>
      | <boolean term> AND <boolean factor>

<boolean factor> ::=
      [ NOT ] <boolean test>

<boolean test> ::=
      <boolean primary> [ IS [ NOT ]
            <truth value> ]

<boolean primary> ::=
        <predicate>
      | <left paren> <search condition> <right paren>

<predicate> ::=
          <comparison predicate>
      | <between predicate>
      | <in predicate>
      | <like predicate>
      | <null predicate>
      | <quantified comparison predicate>
      | <exists predicate>
      | <unique predicate>
      | <match predicate>
      | <overlaps predicate>

<comparison predicate> ::=
      <row value constructor> <comp op>
          <row value constructor>

<row value constructor> ::=
          <row value constructor element>
      | <left paren> <row value constructor list> <right paren>
      | <row subquery>

<row value constructor element> ::=
          <value expression>
      | <null specification>
      | <default specification>

<value expression> ::=
          <numeric value expression>
      | <string value expression>
      | <datetime value expression>
      | <interval value expression>

<numeric value expression> ::=
          <term>
      | <numeric value expression> <plus sign> <term>
      | <numeric value expression> <minus sign> <term>

<term> ::=
          <factor>
      | <term> <asterisk> <factor>
      | <term> <solidus> <factor>

<factor> ::=
      [ <sign> ] <numeric primary>

<numeric primary> ::=
          <value expression primary>
      | <numeric value function>

<value expression primary> ::=
          <unsigned value specification>
      | <column reference>
      | <set function specification>
      | <scalar subquery>
      | <case expression>
      | <left paren> <value expression> <right paren>
      | <cast specification>

<unsigned value specification> ::=
          <unsigned literal>
      | <general value specification>

<unsigned literal> ::=
          <unsigned numeric literal>
      | <general literal>

<general value specification> ::=
          <parameter specification>
      | <dynamic parameter specification>
      | <variable specification>
      | USER
      | CURRENT_USER
      | SESSION_USER
      | SYSTEM_USER
      | VALUE

<parameter specification> ::=
      <parameter name> [ <indicator parameter> ]

<parameter name> ::= <colon> <identifier>
```

```
<indicator parameter> ::=
    [ INDICATOR ] <parameter name>

<dynamic parameter specification> ::= <question mark>

<variable specification> ::=
    <embedded variable name> [ <indicator variable> ]

<embedded variable name> ::=
    <colon><host identifier>

<host identifier> ::=
      <Ada host identifier>
    | <C host identifier>
    | <COBOL host identifier>
    | <Fortran host identifier>
    | <MUMPS host identifier>
    | <Pascal host identifier>
    | <PL/I host identifier>

<Ada host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<C host identifier> ::=
    !! <EMPHASIS>(See the Syntax Rules.)

<COBOL host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<Fortran host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<MUMPS host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<Pascal host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<PL/I host identifier> ::= !! <EMPHASIS>(See the Syntax Rules.)

<indicator variable> ::=
    [ INDICATOR ] <embedded variable name>

<column reference> ::= [ <qualifier> <period> ] <column name>

<qualifier> ::=
      <table name>
    | <correlation name>

<correlation name> ::= <identifier>

<set function specification> ::=
      COUNT <left paren> <asterisk> <right paren>
    | <general set function>

<general set function> ::=
      <set function type>
          <left paren> [ <set quantifier> ] <value expression> <right paren>

<set function type> ::=
    AVG | MAX | MIN | SUM | COUNT

<set quantifier> ::= DISTINCT | ALL

<scalar subquery> ::= <subquery>

<subquery> ::= <left paren> <query expression> <right paren>

<query expression> ::=
      <non-join query expression>
    | <joined table>

<non-join query expression> ::=
      <non-join query term>
    | <query expression> UNION   [ ALL ]
          [ <corresponding spec> ] <query term>
    | <query expression> EXCEPT [ ALL ]
          [ <corresponding spec> ] <query term>

<non-join query term> ::=
      <non-join query primary>
    | <query term> INTERSECT [ ALL ]
          [ <corresponding spec> ] <query primary>

<non-join query primary> ::=
      <simple table>
    | <left paren> <non-join query expression> <right paren>

<simple table> ::=
      <query specification>
    | <table value constructor>
    | <explicit table>

<query specification> ::=
    SELECT [ <set quantifier> ] <select list> <table expression>

<select list> ::=
      <asterisk>
    | <select sublist> [ { <comma> <select sublist> }... ]

<select sublist> ::=
      <derived column>
    | <qualifier> <period> <asterisk>

<derived column> ::= <value expression> [ <as clause> ]

<as clause> ::= [ AS ] <column name>

<table expression> ::=
    <from clause>
    [ <where clause> ]
    [ <group by clause> ]
    [ <having clause> ]

<from clause> ::= FROM <table reference>
    [ { <comma> <table reference> }... ]

<table reference> ::=
      <table name> [ [ AS ] <correlation name>
          [ <left paren> <derived column list> <right paren> ] ]
    | <derived table> [ AS ] <correlation name>
          [ <left paren> <derived column list> <right paren> ]
    | <joined table>

<derived column list> ::= <column name list>

<derived table> ::= <table subquery>

<table subquery> ::= <subquery>

<joined table> ::=
      <cross join>
    | <qualified join>
    | <left paren> <joined table> <right paren>

<cross join> ::=
    <table reference> CROSS JOIN <table reference>

<qualified join> ::=
    <table reference> [ NATURAL ] [ <join type> ] JOIN
      <table reference> [ <join specification> ]

<join type> ::=
      INNER
    | <outer join type> [ OUTER ]
    | UNION

<outer join type> ::=
      LEFT
    | RIGHT
    | FULL

<join specification> ::=
      <join condition>
    | <named columns join>

<join condition> ::= ON <search condition>

<named columns join> ::=
    USING <left paren> <join column list> <right paren>

<join column list> ::= <column name list>

<where clause> ::= WHERE <search condition>

<group by clause> ::=
    GROUP BY <grouping column reference list>

<grouping column reference list> ::=
```

```
        <grouping column reference>
            [ { <comma> <grouping column reference> }... ]

<grouping column reference> ::=
        <column reference> [ <collate clause> ]

<collate clause> ::= COLLATE <collation name>

<collation name> ::= <qualified name>

<having clause> ::= HAVING <search condition>

<table value constructor> ::=
        VALUES <table value constructor list>

<table value constructor list> ::=
        <row value constructor> [ { <comma> <row value constructor> }... ]

<explicit table> ::= TABLE <table name>

<query term> ::=
            <non-join query term>
        | <joined table>

<corresponding spec> ::=
        CORRESPONDING [ BY <left paren> <corresponding column list> <right
paren> ]

<corresponding column list> ::= <column name list>

<query primary> ::=
            <non-join query primary>
        | <joined table>

<case expression> ::=
            <case abbreviation>
        | <case specification>

<case abbreviation> ::=
            NULLIF <left paren> <value expression> <comma>
                    <value expression> <right paren>
        | COALESCE <left paren> <value expression>
                    { <comma> <value expression> }... <right paren>

<case specification> ::=
            <simple case>
        | <searched case>

<simple case> ::=
        CASE <case operand>
            <simple when clause>...
            [ <else clause> ]
        END

<case operand> ::= <value expression>

<simple when clause> ::= WHEN <when operand> THEN <result>

<when operand> ::= <value expression>

<result> ::= <result expression> | NULL

<result expression> ::= <value expression>

<else clause> ::= ELSE <result>

<searched case> ::=
        CASE
            <searched when clause>...
            [ <else clause> ]
        END

<searched when clause> ::= WHEN <search condition> THEN <result>

<cast specification> ::=
        CAST <left paren> <cast operand> AS
            <cast target> <right paren>

<cast operand> ::=
            <value expression>
        | NULL

<cast target> ::=
            <domain name>
        | <data type>

<numeric value function> ::=
            <position expression>
        | <extract expression>
        | <length expression>

<position expression> ::=
        POSITION <left paren> <character value expression>
                IN <character value expression> <right paren>

<character value expression> ::=
            <concatenation>
        | <character factor>

<concatenation> ::=
        <character value expression> <concatenation operator>
                <character factor>

<character factor> ::=
        <character primary> [ <collate clause> ]

<character primary> ::=
            <value expression primary>
        | <string value function>

<string value function> ::=
            <character value function>
        | <bit value function>

<character value function> ::=
            <character substring function>
        | <fold>
        | <form-of-use conversion>
        | <character translation>
        | <trim function>

<character substring function> ::=
        SUBSTRING <left paren> <character value expression> FROM <start
position>
                        [ FOR <string length> ] <right paren>

<start position> ::= <numeric value expression>

<string length> ::= <numeric value expression>

<fold> ::= { UPPER | LOWER }
            <left paren> <character value expression> <right paren>

<form-of-use conversion> ::=
        CONVERT <left paren> <character value expression>
                USING <form-of-use conversion name> <right paren>

<form-of-use conversion name> ::= <qualified name>

<character translation> ::=
        TRANSLATE <left paren> <character value expression>
                USING <translation name> <right paren>

<translation name> ::= <qualified name>

<trim function> ::=
        TRIM <left paren> <trim operands> <right paren>

<trim operands> ::=
        [ [ <trim specification> ] [ <trim character> ] FROM ] <trim source>

<trim specification> ::=
            LEADING
        | TRAILING
        | BOTH

<trim character> ::= <character value expression>

<trim source> ::= <character value expression>

<bit value function> ::=
        <bit substring function>

<bit substring function> ::=
```

```
      SUBSTRING <left paren> <bit value expression> FROM <start position>
          [ FOR <string length> ] <right paren>

<bit value expression> ::=
          <bit concatenation>
      | <bit factor>

<bit concatenation> ::=
      <bit value expression> <concatenation operator> <bit factor>

<bit factor> ::= <bit primary>

<bit primary> ::=
          <value expression primary>
      | <string value function>

<extract expression> ::=
      EXTRACT <left paren> <extract field>
          FROM <extract source> <right paren>

<extract field> ::=
          <datetime field>
      | <time zone field>

<datetime field> ::=
          <non-second datetime field>
      | SECOND

<time zone field> ::=
          TIMEZONE_HOUR
      | TIMEZONE_MINUTE

<extract source> ::=
          <datetime value expression>
      | <interval value expression>

<datetime value expression> ::=
          <datetime term>
      | <interval value expression> <plus sign> <datetime term>
      | <datetime value expression> <plus sign> <interval term>
      | <datetime value expression> <minus sign> <interval term>

<interval term> ::=
          <interval factor>
      | <interval term 2> <asterisk> <factor>
      | <interval term 2> <solidus> <factor>
      | <term> <asterisk> <interval factor>

<interval factor> ::=
      [ <sign> ] <interval primary>

<interval primary> ::=
          <value expression primary> [ <interval qualifier> ]

<interval term 2> ::= <interval term>

<interval value expression> ::=
          <interval term>
      | <interval value expression 1> <plus sign> <interval term 1>
      | <interval value expression 1> <minus sign> <interval term 1>
      | <left paren> <datetime value expression> <minus sign>
              <datetime term> <right paren> <interval qualifier>

<interval value expression 1> ::= <interval value expression>

<interval term 1> ::= <interval term>

<datetime term> ::=
          <datetime factor>

<datetime factor> ::=
          <datetime primary> [ <time zone> ]

<datetime primary> ::=
          <value expression primary>
      | <datetime value function>

<time zone> ::=
      AT <time zone specifier>

<time zone specifier> ::=
          LOCAL
      | TIME ZONE <interval value expression>

<length expression> ::=
          <char length expression>
      | <octet length expression>
      | <bit length expression>

<char length expression> ::=
      { CHAR_LENGTH | CHARACTER_LENGTH }
          <left paren> <string value expression> <right paren>

<string value expression> ::=
          <character value expression>
      | <bit value expression>

<octet length expression> ::=
      OCTET_LENGTH <left paren> <string value expression> <right paren>

<bit length expression> ::=
      BIT_LENGTH <left paren> <string value expression> <right paren>

<null specification> ::=
      NULL

<default specification> ::=
      DEFAULT

<row value constructor list> ::=
      <row value constructor element>
          [ { <comma> <row value constructor element> }... ]

<row subquery> ::= <subquery>

<comp op> ::=
          <equals operator>
      | <not equals operator>
      | <less than operator>
      | <greater than operator>
      | <less than or equals operator>
      | <greater than or equals operator>

<between predicate> ::=
      <row value constructor> [ NOT ] BETWEEN
          <row value constructor> AND <row value constructor>

<in predicate> ::=
      <row value constructor>
          [ NOT ] IN <in predicate value>

<in predicate value> ::=
          <table subquery>
      | <left paren> <in value list> <right paren>

<in value list> ::=
      <value expression> { <comma> <value expression> }...

<like predicate> ::=
      <match value> [ NOT ] LIKE <pattern>
          [ ESCAPE <escape character> ]

<match value> ::= <character value expression>

<pattern> ::= <character value expression>

<escape character> ::= <character value expression>

<null predicate> ::= <row value constructor>
      IS [ NOT ] NULL

<quantified comparison predicate> ::=
      <row value constructor> <comp op> <quantifier> <table subquery>

<quantifier> ::= <all> | <some>

<all> ::= ALL

<some> ::= SOME | ANY

<exists predicate> ::= EXISTS <table subquery>

<unique predicate> ::= UNIQUE <table subquery>
```

```
<match predicate> ::=
    <row value constructor> MATCH [ UNIQUE ]
        [ PARTIAL | FULL ] <table subquery>

<overlaps predicate> ::=
    <row value constructor 1> OVERLAPS <row value constructor 2>

<row value constructor 1> ::= <row value constructor>

<row value constructor 2> ::= <row value constructor>

<truth value> ::=
        TRUE
    | FALSE
    | UNKNOWN

<constraint attributes> ::=
        <constraint check time> [ [ NOT ] DEFERRABLE ]
    | [ NOT ] DEFERRABLE [ <constraint check time> ]

<constraint check time> ::=
        INITIALLY DEFERRED
    | INITIALLY IMMEDIATE

<table constraint definition> ::=
    [ <constraint name definition> ]
    <table constraint> [ <constraint attributes> ]

<table constraint> ::=
        <unique constraint definition>
    | <referential constraint definition>
    | <check constraint definition>

<unique constraint definition> ::=
                <unique specification> even in SQL3)
    <unique specification>
        <left paren> <unique column list> <right paren>

<unique column list> ::= <column name list>

<referential constraint definition> ::=
    FOREIGN KEY
            <left paren> <referencing columns> <right paren>
        <references specification>

<referencing columns> ::=
    <reference column list>

<module contents> ::=
        <declare cursor>
    | <dynamic declare cursor>
    | <procedure>

<declare cursor> ::=
    DECLARE <cursor name> [ INSENSITIVE ] [ SCROLL ] CURSOR
        FOR <cursor specification>

<cursor name> ::= <identifier>

<cursor specification> ::=
    <query expression> [ <order by clause> ]
        [ <updatability clause> ]

<order by clause> ::=
    ORDER BY <sort specification list>

<sort specification list> ::=
    <sort specification> [ { <comma> <sort specification> }... ]

<sort specification> ::=
    <sort key> [ <collate clause> ] [ <ordering specification> ]

<sort key> ::=
        <column name>
    | <unsigned integer>

<ordering specification> ::= ASC | DESC

<updatability clause> ::=
    FOR
        { READ ONLY |
            UPDATE [ OF <column name list> ] }

<dynamic declare cursor> ::=
    DECLARE <cursor name> [ INSENSITIVE ] [ SCROLL ] CURSOR
        FOR <statement name>

<statement name> ::= <identifier>

<procedure> ::=
    PROCEDURE <procedure name>
            <parameter declaration list> <semicolon>
        <SQL procedure statement> <semicolon>

<procedure name> ::= <identifier>

<parameter declaration list> ::=
            <left paren> <parameter declaration>
                [ { <comma> <parameter declaration> }... ] <right paren>
    | <parameter declaration>...

<parameter declaration> ::=
            <parameter name> <data type>
    | <status parameter>

<status parameter> ::=
    SQLCODE | SQLSTATE

<SQL procedure statement> ::=
            <SQL schema statement>
    | <SQL data statement>
    | <SQL transaction statement>
    | <SQL connection statement>
    | <SQL session statement>
    | <SQL dynamic statement>
    | <SQL diagnostics statement>

<SQL schema statement> ::=
            <SQL schema definition statement>
    | <SQL schema manipulation statement>

<SQL schema definition statement> ::=
            <schema definition>
    | <table definition>
    | <view definition>
    | <grant statement>
    | <domain definition>
    | <character set definition>
    | <collation definition>
    | <translation definition>
    | <assertion definition>

<schema definition> ::=
    CREATE SCHEMA <schema name clause>
        [ <schema character set specification> ]
        [ <schema element>... ]

<schema name clause> ::=
            <schema name>
    | AUTHORIZATION <schema authorization identifier>
    | <schema name> AUTHORIZATION
                <schema authorization identifier>

<schema authorization identifier> ::=
    <authorization identifier>

<schema character set specification> ::=
    DEFAULT CHARACTER
            SET <character set specification>

<schema element> ::=
            <domain definition>
    | <table definition>
    | <view definition>
    | <grant statement>
    | <assertion definition>
    | <character set definition>
    | <collation definition>
    | <translation definition>

<domain definition> ::=
    CREATE DOMAIN <domain name>
            [ AS ] <data type>
        [ <default clause> ]
```

```
        [ <domain constraint>... ]
        [ <collate clause> ]

<domain constraint> ::=
      [ <constraint name definition> ]
      <check constraint definition> [ <constraint attributes> ]

<table definition> ::=
      CREATE [ { GLOBAL | LOCAL } TEMPORARY ] TABLE
            <table name>
          <table element list>
          [ ON COMMIT { DELETE | PRESERVE } ROWS ]

<view definition> ::=
      CREATE VIEW <table name> [ <left paren> <view column list>
                                                <right paren> ]
          AS <query expression>
          [ WITH [ <levels clause> ] CHECK OPTION ]

<view column list> ::= <column name list>

<levels clause> ::=
      CASCADED | LOCAL

<grant statement> ::=
     GRANT <privileges> ON <object name>
        TO <grantee> [ { <comma> <grantee> }... ]
           [ WITH GRANT OPTION ]

<privileges> ::=
          ALL PRIVILEGES
       | <action list>

<action list> ::= <action> [ { <comma> <action> }... ]

<action> ::=
          SELECT
       | DELETE
       | INSERT [ <left paren> <privilege column list> <right paren> ]
       | UPDATE [ <left paren> <privilege column list> <right paren> ]
       | REFERENCES [ <left paren> <privilege column list> <right paren> ]
       | USAGE

<privilege column list> ::= <column name list>

<object name> ::=
          [ TABLE ] <table name>
       | DOMAIN <domain name>
       | COLLATION <collation name>
       | CHARACTER SET <character set name>
       | TRANSLATION <translation name>

<grantee> ::=
          PUBLIC
       | <authorization identifier>

<assertion definition> ::=
      CREATE ASSERTION <constraint name> <assertion check>
          [ <constraint attributes> ]

<assertion check> ::=
      CHECK
            <left paren> <search condition> <right paren>

<character set definition> ::=
      CREATE CHARACTER SET <character set name>
            [ AS ]
          <character set source>
          [ <collate clause> | <limited collation definition> ]

<character set source> ::=
          GET <existing character set name>

<existing character set name> ::=
          <standard character repertoire name>
       | <implementation-defined character repertoire name>
       | <schema character set name>

<schema character set name> ::= <character set name>

<limited collation definition> ::=
      COLLATION FROM <collation source>

<collation source> ::=
          <collating sequence definition>
       | <translation collation>

<collating sequence definition> ::=
          <external collation>
       | <schema collation name>
       | DESC <left paren> <collation name> <right paren>
       | DEFAULT

<external collation> ::=
      EXTERNAL <left paren> <quote> <external collation name> <quote> <right
paren>

<external collation name> ::=
          <standard collation name>
       | <implementation-defined collation name>

<standard collation name> ::= <collation name>

<implementation-defined collation name> ::= <collation name>

<schema collation name> ::= <collation name>

<translation collation> ::=
      TRANSLATION <translation name>
            [ THEN COLLATION <collation name> ]

<collation definition> ::=
      CREATE COLLATION <collation name> FOR
            <character set specification>
          FROM <collation source>
            [ <pad attribute> ]

<pad attribute> ::=
          NO PAD
       | PAD SPACE

<translation definition> ::=
      CREATE TRANSLATION <translation name>
          FOR <source character set specification>
             TO <target character set specification>
          FROM <translation source>

<source character set specification> ::= <character set specification>

<target character set specification> ::= <character set specification>

<translation source> ::=
          <translation specification>

<translation specification> ::=
          <external translation>
       | IDENTITY
       | <schema translation name>

<external translation> ::=
      EXTERNAL <left paren> <quote> <external translation name> <quote> <right
paren>

<external translation name> ::=
          <standard translation name>
       | <implementation-defined translation name>

<standard translation name> ::= <translation name>

<implementation-defined translation name> ::= <translation name>

<schema translation name> ::= <translation name>

<SQL schema manipulation statement> ::=
          <drop schema statement>
       | <alter table statement>
       | <drop table statement>
       | <drop view statement>
       | <revoke statement>
       | <alter domain statement>
       | <drop domain statement>
       | <drop character set statement>
       | <drop collation statement>
       | <drop translation statement>
```

```
    | <drop assertion statement>

<drop schema statement> ::=
    DROP SCHEMA <schema name> <drop behavior>

<drop behavior> ::= CASCADE | RESTRICT

<alter table statement> ::=
    ALTER TABLE <table name> <alter table action>

<alter table action> ::=
        <add column definition>
    | <alter column definition>
    | <drop column definition>
    | <add table constraint definition>
    | <drop table constraint definition>

<add column definition> ::=
    ADD [ COLUMN ] <column definition>

<alter column definition> ::=
    ALTER [ COLUMN ] <column name> <alter column action>

<alter column action> ::=
        <set column default clause>
    | <drop column default clause>

<set column default clause> ::=
    SET <default clause>

<drop column default clause> ::=
    DROP DEFAULT

<drop column definition> ::=
    DROP [ COLUMN ] <column name> <drop behavior>

<add table constraint definition> ::=
    ADD <table constraint definition>

<drop table constraint definition> ::=
    DROP CONSTRAINT <constraint name> <drop behavior>

<drop table statement> ::=
    DROP TABLE <table name> <drop behavior>

<drop view statement> ::=
    DROP VIEW <table name> <drop behavior>

<revoke statement> ::=
    REVOKE [ GRANT OPTION FOR ]
        <privileges>
        ON <object name>
      FROM <grantee> [ { <comma> <grantee> }... ] <drop behavior>

<alter domain statement> ::=
    ALTER DOMAIN <domain name> <alter domain action>

<alter domain action> ::=
        <set domain default clause>
    | <drop domain default clause>
    | <add domain constraint definition>
    | <drop domain constraint definition>

<set domain default clause> ::= SET <default clause>

<drop domain default clause> ::= DROP DEFAULT

<add domain constraint definition> ::=
    ADD <domain constraint>

<drop domain constraint definition> ::=
    DROP CONSTRAINT <constraint name>

<drop domain statement> ::=
    DROP DOMAIN <domain name> <drop behavior>

<drop character set statement> ::=
    DROP CHARACTER SET <character set name>

<drop collation statement> ::=
    DROP COLLATION <collation name>

<drop translation statement> ::=
    DROP TRANSLATION <translation name>

<drop assertion statement> ::=
    DROP ASSERTION <constraint name>

<SQL data statement> ::=
        <open statement>
    | <fetch statement>
    | <close statement>
    | <select statement: single row>
    | <SQL data change statement>

<open statement> ::=
    OPEN <cursor name>

<fetch statement> ::=
    FETCH [ [ <fetch orientation> ] FROM ]
      <cursor name> INTO <fetch target list>

<fetch orientation> ::=
        NEXT
    | PRIOR
    | FIRST
    | LAST
    | { ABSOLUTE | RELATIVE } <simple value specification>

<simple value specification> ::=
        <parameter name>
    | <embedded variable name>
    | <literal>

<fetch target list> ::=
    <target specification> [ { <comma> <target specification> }... ]

<target specification> ::=
        <parameter specification>
    | <variable specification>

<close statement> ::=
    CLOSE <cursor name>

<select statement: single row> ::=
    SELECT [ <set quantifier> ] <select list>
      INTO <select target list>
        <table expression>

<select target list> ::=
    <target specification> [ { <comma> <target specification> }... ]

<SQL data change statement> ::=
        <delete statement: positioned>
    | <delete statement: searched>
    | <insert statement>
    | <update statement: positioned>
    | <update statement: searched>

<delete statement: positioned> ::=
    DELETE FROM <table name>
      WHERE CURRENT OF <cursor name>

<delete statement: searched> ::=
    DELETE FROM <table name>
      [ WHERE <search condition> ]

<insert statement> ::=
    INSERT INTO <table name>
      <insert columns and source>

<insert columns and source> ::=
        [ <left paren> <insert column list> <right paren> ]
              <query expression>
    | DEFAULT VALUES

<insert column list> ::= <column name list>

<update statement: positioned> ::=
    UPDATE <table name>
      SET <set clause list>
        WHERE CURRENT OF <cursor name>

<set clause list> ::=
```

```
    <set clause> [ { <comma> <set clause> }... ]

<set clause> ::=
    <object column> <equals operator> <update source>

<object column> ::= <column name>

<update source> ::=
        <value expression>
    | <null specification>
    | DEFAULT

<update statement: searched> ::=
    UPDATE <table name>
      SET <set clause list>
      [ WHERE <search condition> ]

<SQL transaction statement> ::=
        <set transaction statement>
    | <set constraints mode statement>
    | <commit statement>
    | <rollback statement>

<set transaction statement> ::=
    SET TRANSACTION <transaction mode>
        [ { <comma> <transaction mode> }... ]

<transaction mode> ::=
        <isolation level>
    | <transaction access mode>
    | <diagnostics size>

<isolation level> ::=
    ISOLATION LEVEL <level of isolation>

<level of isolation> ::=
        READ UNCOMMITTED
    | READ COMMITTED
    | REPEATABLE READ
    | SERIALIZABLE

<transaction access mode> ::=
        READ ONLY
    | READ WRITE

<diagnostics size> ::=
    DIAGNOSTICS SIZE <number of conditions>

<number of conditions> ::= <simple value specification>

<set constraints mode statement> ::=
    SET CONSTRAINTS <constraint name list>
        { DEFERRED | IMMEDIATE }

<constraint name list> ::=
        ALL
    | <constraint name> [ { <comma> <constraint name> }... ]

<commit statement> ::=
    COMMIT [ WORK ]

<rollback statement> ::=
    ROLLBACK [ WORK ]

<SQL connection statement> ::=
        <connect statement>
    | <set connection statement>
    | <disconnect statement>

<connect statement> ::=
    CONNECT TO <connection target>

<connection target> ::=
        <SQL-server name>
        [ AS <connection name> ]
            correspondence with Tony Gordon)
        [ USER <user name> ]
    | DEFAULT

<SQL-server name> ::= <simple value specification>

<connection name> ::= <simple value specification>

<user name> ::= <simple value specification>

<set connection statement> ::=
    SET CONNECTION <connection object>

<connection object> ::=
        DEFAULT
    | <connection name>

<disconnect statement> ::=
    DISCONNECT <disconnect object>

<disconnect object> ::=
        <connection object>
    | ALL
    | CURRENT

<SQL session statement> ::=
        <set catalog statement>
    | <set schema statement>
    | <set names statement>
    | <set session authorization identifier statement>
    | <set local time zone statement>

<set catalog statement> ::=
    SET CATALOG <value specification>

<value specification> ::=
        <literal>
    | <general value specification>

<set schema statement> ::=
    SET SCHEMA <value specification>

<set names statement> ::=
    SET NAMES <value specification>

<set session authorization identifier statement> ::=
    SET SESSION AUTHORIZATION
        <value specification>

<set local time zone statement> ::=
    SET TIME ZONE
        <set time zone value>

<set time zone value> ::=
        <interval value expression>
    | LOCAL

<SQL dynamic statement> ::=
        <system descriptor statement>
    | <prepare statement>
    | <deallocate prepared statement>
    | <describe statement>
    | <execute statement>
    | <execute immediate statement>
    | <SQL dynamic data statement>

<system descriptor statement> ::=
        <allocate descriptor statement>
    | <deallocate descriptor statement>
    | <set descriptor statement>
    | <get descriptor statement>

<allocate descriptor statement> ::=
    ALLOCATE DESCRIPTOR <descriptor name>
        [ WITH MAX <occurrences> ]

<descriptor name> ::=
    [ <scope option> ] <simple value specification>

<scope option> ::=
        GLOBAL
    | LOCAL

<occurrences> ::= <simple value specification>

<deallocate descriptor statement> ::=
    DEALLOCATE DESCRIPTOR <descriptor name>

<set descriptor statement> ::=
```

```
    SET DESCRIPTOR <descriptor name>
        <set descriptor information>

<set descriptor information> ::=
        <set count>
    | VALUE <item number>
        <set item information> [ { <comma> <set item information> }... ]

<set count> ::=
    COUNT <equals operator> <simple value specification 1>

<simple value specification 1> ::= <simple value specification>

<item number> ::= <simple value specification>

<set item information> ::=
    <descriptor item name> <equals operator> <simple value specification 2>

<descriptor item name> ::=
        TYPE
    | LENGTH
    | OCTET_LENGTH
    | RETURNED_LENGTH
    | RETURNED_OCTET_LENGTH
    | PRECISION
    | SCALE
    | DATETIME_INTERVAL_CODE
    | DATETIME_INTERVAL_PRECISION
    | NULLABLE
    | INDICATOR
    | DATA
    | NAME
    | UNNAMED
    | COLLATION_CATALOG
    | COLLATION_SCHEMA
    | COLLATION_NAME
    | CHARACTER_SET_CATALOG
    | CHARACTER_SET_SCHEMA
    | CHARACTER_SET_NAME

<simple value specification 2> ::= <simple value specification>

<item number> ::= <simple value specification>

<get descriptor statement> ::=
    GET DESCRIPTOR <descriptor name> <get descriptor information>

<get descriptor information> ::=
        <get count>
    | VALUE <item number>
        <get item information> [ { <comma> <get item information> }... ]

<get count> ::=
    <simple target specification 1> <equals operator>
        COUNT

<simple target specification 1> ::= <simple target specification>

<simple target specification> ::=
        <parameter name>
    | <embedded variable name>

<get item information> ::=
    <simple target specification 2> <equals operator> <descriptor item name>>

<simple target specification 2> ::= <simple target specification>

<prepare statement> ::=
    PREPARE <SQL statement name> FROM <SQL statement variable>

<SQL statement name> ::=
        <statement name>
    | <extended statement name>

<extended statement name> ::=
    [ <scope option> ] <simple value specification>

<SQL statement variable> ::= <simple value specification>

<deallocate prepared statement> ::=
    DEALLOCATE PREPARE <SQL statement name>

<describe statement> ::=
        <describe input statement>
    | <describe output statement>

<describe input statement> ::=
    DESCRIBE INPUT <SQL statement name> <using descriptor>

<using descriptor> ::=
    { USING | INTO } SQL DESCRIPTOR <descriptor name>

<describe output statement> ::=
    DESCRIBE [ OUTPUT ] <SQL statement name> <using descriptor>

<execute statement> ::=
    EXECUTE <SQL statement name>
      [ <result using clause> ]
      [ <parameter using clause> ]

<result using clause> ::= <using clause>

<using clause> ::=
        <using arguments>
    | <using descriptor>

<using arguments> ::=
    { USING | INTO } <argument> [ { <comma> <argument> }... ]

<argument> ::= <target specification>

<parameter using clause> ::= <using clause>

<execute immediate statement> ::=
    EXECUTE IMMEDIATE <SQL statement variable>

<SQL dynamic data statement> ::=
        <allocate cursor statement>
    | <dynamic open statement>
    | <dynamic fetch statement>
    | <dynamic close statement>
    | <dynamic delete statement: positioned>
    | <dynamic update statement: positioned>

<allocate cursor statement> ::=
    ALLOCATE <extended cursor name> [ INSENSITIVE ]
        [ SCROLL ] CURSOR
      FOR <extended statement name>

<extended cursor name> ::=
    [ <scope option> ] <simple value specification>

<dynamic open statement> ::=
    OPEN <dynamic cursor name> [ <using clause> ]

<dynamic cursor name> ::=
        <cursor name>
    | <extended cursor name>

<dynamic fetch statement> ::=
    FETCH [ [ <fetch orientation> ] FROM ] <dynamic cursor name>
        <using clause>

<dynamic close statement> ::=
    CLOSE <dynamic cursor name>

<dynamic delete statement: positioned> ::=
    DELETE FROM <table name>
      WHERE CURRENT OF
          <dynamic cursor name>

<dynamic update statement: positioned> ::=
    UPDATE <table name>
      SET <set clause>
          [ { <comma> <set clause> }... ]
        WHERE CURRENT OF
            <dynamic cursor name>

<SQL diagnostics statement> ::=
    <get diagnostics statement>

<get diagnostics statement> ::=
    GET DIAGNOSTICS <sql diagnostics information>
```

```
<sql diagnostics information> ::=
        <statement information>
      | <condition information>

<statement information> ::=
      <statement information item> [ { <comma> <statement information item> }... ]

<statement information item> ::=
      <simple target specification> <equals operator> <statement information item
name>

<statement information item name> ::=
        NUMBER
      | MORE
      | COMMAND_FUNCTION
      | DYNAMIC_FUNCTION
      | ROW_COUNT

<condition information> ::=
      EXCEPTION <condition number>
        <condition information item> [ { <comma> <condition information item> }... ]

<condition number> ::= <simple value specification>

<condition information item> ::=
      <simple target specification> <equals operator> <condition information item
name>

<condition information item name> ::=
        CONDITION_NUMBER
      | RETURNED_SQLSTATE
      | CLASS_ORIGIN
      | SUBCLASS_ORIGIN
      | SERVER_NAME
      | CONNECTION_NAME
      | CONSTRAINT_CATALOG
      | CONSTRAINT_SCHEMA
      | CONSTRAINT_NAME
      | CATALOG_NAME
      | SCHEMA_NAME
      | TABLE_NAME
      | COLUMN_NAME
      | CURSOR_NAME
      | MESSAGE_TEXT
      | MESSAGE_LENGTH
      | MESSAGE_OCTET_LENGTH

<embedded SQL host program> ::=
        <embedded SQL Ada program>
      | <embedded SQL C program>
      | <embedded SQL COBOL program>
      | <embedded SQL Fortran program>
      | <embedded SQL MUMPS program>
      | <embedded SQL Pascal program>
      | <embedded SQL PL/I program>

<embedded SQL Ada program> ::= !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL C program> ::=
        !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL COBOL program> ::= !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL Fortran program> ::=
      !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL MUMPS program> ::= !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL Pascal program> ::=
      !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL PL/I program> ::= !! <EMPHASIS>(See the Syntax Rules.)

<embedded SQL declare section> ::=
        <embedded SQL begin declare>
          [ <embedded character set declaration> ]
          [ <host variable definition>... ]
        <embedded SQL end declare>
      | <embedded SQL MUMPS declare>

<embedded SQL begin declare> ::=
      <SQL prefix> BEGIN DECLARE SECTION
          [ <SQL terminator> ]

<SQL prefix> ::=
        EXEC SQL
      | <ampersand>SQL<left paren>

<SQL terminator> ::=
        END-EXEC
      | <semicolon>
      | <right paren>

<embedded character set declaration> ::=
      SQL NAMES ARE <character set specification>

<host variable definition> ::=
        <Ada variable definition>
      | <C variable definition>
      | <COBOL variable definition>
      | <Fortran variable definition>
      | <MUMPS variable definition>
      | <Pascal variable definition>
      | <PL/I variable definition>

<Ada variable definition> ::=
      <Ada host identifier> [ { <comma> <Ada host identifier> }... ] :
      <Ada type specification> [ <Ada initial value> ]

<Ada type specification> ::=
        <Ada qualified type specification>
      | <Ada unqualified type specification>

<Ada qualified type specification> ::=
        SQL_STANDARD.CHAR [ CHARACTER SET
            [ IS ] <character set specification> ]
              <left paren> 1 <double period> <length> <right paren>
      | SQL_STANDARD.BIT
              <left paren> 1 <double period> <length> <right paren>
      | SQL_STANDARD.SMALLINT
      | SQL_STANDARD.INT
      | SQL_STANDARD.REAL
      | SQL_STANDARD.DOUBLE_PRECISION
      | SQL_STANDARD.SQLCODE_TYPE
      | SQL_STANDARD.SQLSTATE_TYPE
      | SQL_STANDARD.INDICATOR_TYPE

<Ada unqualified type specification> ::=
        CHAR
              <left paren> 1 <double period> <length> <right paren>
      | BIT
              <left paren> 1 <double period> <length> <right paren>
      | SMALLINT
      | INT
      | REAL
      | DOUBLE_PRECISION
      | SQLCODE_TYPE
      | SQLSTATE_TYPE
      | INDICATOR_TYPE

<Ada initial value> ::=
      <Ada assignment operator> <character representation>...

<Ada assignment operator> ::= <colon><equals operator>

<C variable definition> ::=
        [ <C storage class> ]
        [ <C class modifier> ]
        <C variable specification>
      <semicolon>

<C storage class> ::=
        auto
      | extern
      | static

<C class modifier> ::= const | volatile

<C variable specification> ::=
        <C numeric variable>
      | <C character variable>
      | <C derived variable>

<C numeric variable> ::=
```

```
    { long | short | float | double }
      <C host identifier> [ <C initial value> ]
            [ { <comma> <C host identifier> [ <C initial value> ] }... ]

<C initial value> ::=
    <equals operator> <character representation>...

<C character variable> ::=
    char [ CHARACTER SET
            [ IS ] <character set specification> ]
      <C host identifier>
        <C array specification> [ <C initial value> ]
        [ { <comma> <C host identifier>
          <C array specification>
              [ <C initial value> ] }... ]

<C array specification> ::=
    <left bracket> <length> <right bracket>

<C derived variable> ::=
      <C VARCHAR variable>
    | <C bit variable>

<C VARCHAR variable> ::=
    VARCHAR [ CHARACTER SET [ IS ]
        <character set specification> ]
        <C host identifier>
          <C array specification> [ <C initial value> ]
        [ { <comma> <C host identifier>
            <C array specification>
                [ <C initial value> ] }... ]

<C bit variable> ::=
    BIT <C host identifier>
        <C array specification> [ <C initial value> ]
      [ { <comma> <C host identifier>
        <C array specification>
            [ <C initial value> ] }... ]

<COBOL variable definition> ::=
    {01|77} <COBOL host identifier> <COBOL type specification>
      [ <character representation>... ] <period>

<COBOL type specification> ::=
      <COBOL character type>
    | <COBOL bit type>
    | <COBOL numeric type>
    | <COBOL integer type>

<COBOL character type> ::=
    [ CHARACTER SET [ IS ]
        <character set specification> ]
    { PIC | PICTURE } [ IS ] { X [ <left paren> <length> <right paren> ] }...

<COBOL bit type> ::=
    { PIC | PICTURE } [ IS ]
      { B [ <left paren> <length> <right paren> ] }...

<COBOL numeric type> ::=
    { PIC | PICTURE } [ IS ]
      S <COBOL nines specification>
    [ USAGE [ IS ] ] DISPLAY SIGN LEADING SEPARATE

<COBOL nines specification> ::=
      <COBOL nines> [ V [ <COBOL nines> ] ]
    | V <COBOL nines>

<COBOL nines> ::= { 9 [ <left paren> <length> <right paren> ] }...

<COBOL integer type> ::=
      <COBOL computational integer>
    | <COBOL binary integer>

<COBOL computational integer> ::=
    { PIC | PICTURE } [ IS ] S<COBOL nines>
      [ USAGE [ IS ] ] { COMP | COMPUTATIONAL }

<COBOL binary integer> ::=
    { PIC | PICTURE } [ IS ] S<COBOL nines>
      [ USAGE [ IS ] ] BINARY

<Fortran variable definition> ::=
    <Fortran type specification>
    <Fortran host identifier>
        [ { <comma> <Fortran host identifier> }... ]

<Fortran type specification> ::=
      CHARACTER [ <asterisk> <length> ]
          [ CHARACTER SET [ IS ]
              <character set specification> ]
    | BIT [ <asterisk> <length> ]
    | INTEGER
    | REAL
    | DOUBLE PRECISION

<MUMPS variable definition> ::=
    { <MUMPS numeric variable> | <MUMPS character variable> }
        <semicolon>

<MUMPS numeric variable> ::=
    <MUMPS type specification>
      <MUMPS host identifier> [ { <comma> <MUMPS host identifier> }... ]

<MUMPS type specification> ::=
      INT
    | DEC
          [ <left paren> <precision> [ <comma> <scale> ] <right paren> ]
    | REAL

<MUMPS character variable> ::=
    VARCHAR <MUMPS host identifier> <MUMPS length specification>
      [ { <comma> <MUMPS host identifier> <MUMPS length specification> }... ]

<MUMPS length specification> ::=
    <left paren> <length> <right paren>

<Pascal variable definition> ::=
    <Pascal host identifier> [ { <comma> <Pascal host identifier> }... ] <colon>
      <Pascal type specification> <semicolon>

<Pascal type specification> ::=
      PACKED ARRAY
          <left bracket> 1 <double period> <length> <right bracket>
        OF CHAR
          [ CHARACTER SET [ IS ]
              <character set specification> ]
    | PACKED ARRAY
          <left bracket> 1 <double period> <length> <right bracket>
        OF BIT
    | INTEGER
    | REAL
    | CHAR [ CHARACTER SET
                                    [ IS ] <character set specification> ]
    | BIT

<PL/I variable definition> ::=
    {DCL | DECLARE}
        {   <PL/I host identifier>
          | <left paren> <PL/I host identifier>
              [ { <comma> <PL/I host identifier> }... ] <right paren> }
    <PL/I type specification>
    [ <character representation>... ] <semicolon>

<PL/I type specification> ::=
      { CHAR | CHARACTER } [ VARYING ]
          <left paren><length><right paren>
          [ CHARACTER SET
              [ IS ] <character set specification> ]
    | BIT [ VARYING ] <left paren><length><right paren>
    | <PL/I type fixed decimal> <left paren> <precision>
          [ <comma> <scale> ] <right paren>
    | <PL/I type fixed binary> [ <left paren> <precision> <right paren> ]
    | <PL/I type float binary> <left paren> <precision> <right paren>

<PL/I type fixed decimal> ::=
      { DEC | DECIMAL } FIXED
    | FIXED { DEC | DECIMAL }

<PL/I type fixed binary> ::=
      { BIN | BINARY } FIXED
    | FIXED { BIN | BINARY }

<PL/I type float binary> ::=
      { BIN | BINARY } FLOAT
```

```
    | FLOAT { BIN | BINARY }

<embedded SQL end declare> ::=
    <SQL prefix> END DECLARE SECTION
        [ <SQL terminator> ]

<embedded SQL MUMPS declare> ::=
    <SQL prefix>
      BEGIN DECLARE SECTION
        [ <embedded character set declaration> ]
        [ <host variable definition>... ]
      END DECLARE SECTION
    <SQL terminator>

<embedded SQL statement> ::=
    <SQL prefix>
      <statement or declaration>
    [ <SQL terminator> ]

<statement or declaration> ::=
        <declare cursor>
    | <dynamic declare cursor>
    | <temporary table declaration>
    | <embedded exception declaration>
    | <SQL procedure statement>

<embedded exception declaration> ::=
    WHENEVER <condition> <condition action>

<condition> ::=
    SQLERROR | NOT FOUND

<condition action> ::=
    CONTINUE | <go to>

<go to> ::=
    { GOTO | GO TO } <goto target>

<goto target> ::=
        <host label identifier>
    | <unsigned integer>
    | <host PL/I label variable>

<host label identifier> ::= !!<EMPHASIS>(See the Syntax Rules.)

<host PL/I label variable> ::= !!<EMPHASIS>(See the Syntax Rules.)

<preparable statement> ::=
        <preparable SQL data statement>
    | <preparable SQL schema statement>
    | <preparable SQL transaction statement>
    | <preparable SQL session statement>
    | <preparable implementation-defined statement>

<preparable SQL data statement> ::=
        <delete statement: searched>
    | <dynamic single row select statement>
    | <insert statement>
    | <dynamic select statement>
    | <update statement: searched>
    | <preparable dynamic delete statement: positioned>
    | <preparable dynamic update statement: positioned>

<dynamic single row select statement> ::= <query specification>

<dynamic select statement> ::= <cursor specification>

<preparable dynamic delete statement: positioned> ::=
    DELETE [ FROM <table name> ]
        WHERE CURRENT OF <cursor name>

<preparable dynamic update statement: positioned> ::=
    UPDATE [ <table name> ]
        SET <set clause list>
        WHERE CURRENT OF <cursor name>

<preparable SQL schema statement> ::=
        <SQL schema statement>

<preparable SQL transaction statement> ::=
        <SQL transaction statement>

<preparable SQL session statement> ::=
        <SQL session statement>

<preparable implementation-defined statement> ::=
    !! <EMPHASIS>(See the Syntax Rules.)

<direct SQL statement> ::=
    <directly executable statement> <semicolon>

<directly executable statement> ::=
        <direct SQL data statement>
    | <SQL schema statement>
    | <SQL transaction statement>
    | <SQL connection statement>
    | <SQL session statement>
    | <direct implementation-defined statement>

<direct SQL data statement> ::=
        <delete statement: searched>
    | <direct select statement: multiple rows>
    | <insert statement>
    | <update statement: searched>
    | <temporary table declaration>

<direct select statement: multiple rows> ::=
    <query expression> [ <order by clause> ]

<direct implementation-defined statement> ::=
    !!<EMPHASIS>(See the Syntax Rules)

<SQL object identifier> ::=
    <SQL provenance> <SQL variant>

<SQL provenance> ::= <arc1> <arc2> <arc3>

<arc1> ::= iso | 1 | iso <left paren> 1 <right paren>

<arc2> ::= standard | 0 | standard <left paren> 0 <right paren>

<arc3> ::= 9075

<SQL variant> ::= <SQL edition> <SQL conformance>

<SQL edition> ::= <1987> | <1989> | <1992>

<1987> ::= 0 | edition1987 <left paren> 0 <right paren>

<1989> ::= <1989 base> <1989 package>

<1989 base> ::= 1 | edition1989 <left paren> 1 <right paren>

<1989 package> ::= <integrity no> | <integrity yes>

<integrity no> ::= 0 | IntegrityNo <left paren> 0 <right paren>

<integrity yes> ::= 1 | IntegrityYes <left paren> 1 <right paren>

<1992> ::= 2 | edition1992 <left paren> 2 <right paren>

<SQL conformance> ::= <low> | <intermediate> | <high>

<low> ::= 0 | Low <left paren> 0 <right paren>

<intermediate> ::= 1 | Intermediate <left paren> 1 <right paren>

<high> ::= 2 | High <left paren> 2 <right paren>
```