SQL2 Syntax

SESSIONS, CONNECTIONS AND TRANSACTIONS

```
connect
    ::=   CONNECT TO { DEFAULT
                     | lit-param-or-var [ AS lit-param-or-var ]
                                        [ USER lit-param-or-var ] }

set-connection
    ::=   SET CONNECTION { DEFAULT | lit-param or var

disconnect
    ::=   DISCONNECT { DEFAULT | CURRENT | ALL | lit-param or var )

set-catalog
    ::=   SET CATALOG { lit-param or var | user-function-ref }

set-schema
    ::=   SET SCHEMA { lit-param or var | user-function-ref }

set-names
    ::=   SET NAMES { lit-param or var | user-function-ref }

set-authorization
    ::=   SET SESSION AUTHORISATION
                    { lit-param-or-var | user-function-ref }

set-time-zone
    ::=   SET TIME ZONE { interval-exp | LOCAL }

commit
    ::=   COMMIT [ WORK ]

rollback
    ::=   ROLLBACK [ WORK ]

set-transaction
    ::=   SET TRANSACTION
            [ READ ONLY | READ WRITE ]
            [ DIAGNOSTICS SIZE integer ]
            [ ISOLATION LEVEL { READ UNCOMMITTED
                              | READ COMMITTED
                              | REPEATABLE READ
                              | SERIALIZABLE } ]
```

DATA DEFINITION

```
schema-def
    ::=   CREATE SCHEMA [ schema ] [ AUTHORISATION user ]
               [ DEFAULT CHARACTER SET character-set ]
               [ schema-element-list ]

schema-element
    ::=   domain-def
        | base-table-def
        | view-def
        | authorization-def
        | general-constraint-def
        | character-set-def
        | collation-def
        | translation-def

domain-def
    ::=   CREATE DOMAIN domain [ AS ] data-type
               [ default-def ]
               [ domain-constraint-def ]

default-def
    ::=   DEFAULT { literal | niladic-function-ref | NULL }

base-table-def
    ::=   CREATE [ [ GLOBAL | LOCAL ] TEMPORARY ] TABLE base-table
                  ( base-table-element-commalist )
                  [ ON COMMIT { DELETE | PRESERVE } ROWS ]

base-table-element
    ::=   column-def | base-table-constraint-def

column-def
    ::=   column { data-type | domain }
                  [ default-def ]
                  [ column-constraint-def-list ]

view-def
    ::=   CREATE VIEW view [ ( column-commalist ) ]
              AS table-exp
                     [ WITH [ CASCADED | LOCAL ] CHECK OPTION ]

authorization-def
    ::=   GRANT { privilege-commalist | ALL PRIVILEGES }
                   ON accessible-object TO grantee-commalist
                                        [ WITH GRANT OPTION ]

general-constraint-def
    ::=   CREATE ASSERTION constraint CHECK ( cond-exp )
                                        [ deferrability ]

deferrability
    ::=   INITIALLY { DEFERRED | IMMEDIATE }
          [ NOT ] DEFERRABLE

privilege
    ::=   SELECT
        | INSERT [ ( column-commalist ) ]
        | UPDATE [ ( column-commalist ) ]
        | DELETE
        | REFERENCES [ ( column-commalist ) ]
        | USAGE
```

```
accessible-object
    ::=   DOMAIN domain
        | [ TABLE ] table
        | CHARACTER SET character-set
        | COLLATION collation
        | TRANSLATION translation

grantee
    ::=   user | PUBLIC

character-set-def
    ::=   CREATE CHARACTER SET character-set [ AS ]
             GET character-set
           [ COLLATE collation | COLLATION FROM collation-source ]

collation-source
    ::=   EXTERNAL ( 'collation' )
        | collation
        | DESC ( collation )
        | DEFAULT
        | TRANSLATION translation [ THEN COLLATION collation ]

collation-def
    ::=   CREATE COLLATION collation
                  FOR character-set
                  FROM collation-source

translation-def
    ::=   CREATE TRANSLATION translation
                  FOR character-set
                  TO character-set
                  FROM translation-source

translation-source
    ::=   EXTERNAL ( 'translation' )
        | IDENTITY
        | translation

domain-alteration
    ::=   ALTER DOMAIN domain domain-alteration-action

domain-alteration-action
    ::=   domain-default-alteration-action
        | domain-constraint-alteration-action

domain-default-alteration-action
    ::=   SET default-def
        | DROP DEFAULT

domain-constraint-alteration-action
    ::=   ADD domain-constraint-def
        | DROP CONSTRAINT constraint

base-table-alteration
    ::=   ALTER TABLE base-table base-table-alteration-action

base-table-alteration-action
    ::=   column-alteration-action
        | base-table-constraint-alteration-action

column-alteration-action
    ::=   ADD [ COLUMN ] column-def
        | ALTER [ COLUMN ] column
                 { SET default-def | DROP DEFAULT }
        | DROP [ COLUMN ] column { RESTRICT | CASCADE }
```

```
base-table-constraint-alteration-action
    ::=   ADD base-table-constraint-def
        | DROP CONSTRAINT constraint { RESTRICT | CASCADE }

schema-drop
    ::=   DROP SCHEMA schema { RESTRICT | CASCADE }

domain-drop
    ::=   DROP DOMAIN domain { RESTRICT | CASCADE }

base-table-drop
    ::=   DROP TABLE base-table { RESTRICT | CASCADE }

view-drop
    ::=   DROP VIEW view { RESTRICT | CASCADE }

authorization-drop
    ::=   REVOKE [ GRANT OPTION FOR ] priviloge-commalist
                  ON accessible-object FROM grantee-commalist
                                        { RESTRICT | CASCADE }

general-constraint-drop
    ::=   DROP ASSERTION constraint

character-set-drop
    ::=   DROP CHARACTER SET character-set

collation-drop
    ::=   DROP COLLATION collation

translation-drop
    ::=   DROP TRANSLATION translation
```

MODULES

```
module-def
    ::=   MODULE [ module ] [ NAMES ARE character-set ]
          LANGUAGE { ADA | C | COBOL | FORTRAN | MUMPS
                   | PASCAL | PLI }
        [ SCHEMA schema ] [ AUTHORISATION user ]
        [ temporary-table-def-list ]
          module-element-list

temporary-table-def
    ::=   DECLARE LOCAL TEMPORARY TABLE MODULE . base-table
                ( base-table-element-commalist )
                [ ON COMMIT { PRESERVE | DELETE } ROWS ]

module-element
    ::=   cursor-def
        | dynamic-cursor-def
        | procedure-def

procedure-def
    ::=   PROCEDURE procedure
           { parameter-def-list | ( parameter-def-commalist ) } ;
            SQL-statement ;

parameter-def
    ::=   parameter data-type
        | SQLCODE
        | SQLSTATE
```

DATA MANIPULATION

```
single-row-select
    ::=   SELECT [ ALL | DISTINCT ] select-item-commalist
          INTO target-commalist
          FROM table-ref-commalist
        [ WHERE cond-exp ]
        [ GROUP BY column-ref-commalist ]
        [ HAVING cond-exp ]

insert
    ::=   INSERT INTO table
          { [ ( column-commalist ) ] table-exp | DEFAULT VALUES }

searched-update
    ::=   UPDATE table
          SET update-assignment-commalist
        [ WHERE cond-exp ]

update-assignment
    ::=   column = { scalar-exp | DEFAULT | NULL }

searched-delete
    ::=   DELETE
          FROM table
        [ WHERE cond-exp ]

cursor-def
    ::=   DECLARE cursor [ INSENSITIVE ] [ SCROLL ] CURSOR FOR
                  table-exp
                [ ORDER BY order-item-commalist ]
          [ FOR { READ ONLY | UPDATE [ OF column-commalist ] } ]

order-item
    ::=   { column | integer } [ ASC | DESC ]

open
    ::=   OPEN cursor

fetch
    ::=   FETCH [ [ row-selector ] FROM ] cursor
                                    INTO target-commalist

row-selector
    ::=   NEXT | PRIOR | FIRST | LAST
                  | ABSOLUTE number | RELATIVE number

positioned-update
    ::=   UPDATE table
          SET update-assignment-commalist
          WHERE CURRENT OF cursor

positioned-delete
    ::=   DELETE
          FROM table
          WHERE CURRENT OF cursor

close
    ::=   CLOSE cursor
```

TABLE EXPRESSIONS

```
table-exp
    ::=   join-table-exp | nonjoin-table-exp

join-table-exp
    ::=   table-ref [ NATURAL ] [ join-type ] JOIN table-ref
                    [ ON cond-exp | USING ( column-commalist ) ]
        | table-ref CROSS JOIN table-ref
        | ( join-table-exp )

table-ref
    ::=   table [ [ AS ] range-variable
                  [ ( column-commalist ) ] ]
        | ( table-exp ) [ AS ] range-variable
                        [ ( column-commalist ) ]
        | join-table-exp

join-type
    ::=   INNER
        | LEFT [ OUTER ]
        | RIGHT [ OUTER ]
        | FULL [ OUTER ]
        | UNION

nonjoin-table-exp
    ::=   nonjoin-table-term
        | table-exp { UNION | EXCEPT } [ ALL ]
                [ CORRESPONDING [ BY ( column-commalist ) ] ]
                      table-term

nonjoin-table-term
    ::=   nonjoin-table-primary
        | table-term INTERSECT [ ALL ]
                [ CORRESPONDING [ BY ( column-commalist ) ] ]
                      table-primary

table-term
    ::=   nonjoin-table-term
        | join-table-exp

table-primary
    :=   nonjoin-table-primary
       | join-table-exp

nonjoin-table-primary
    ::=   TABLE table
        | table-constructor
        | select-exp
        | ( nonjoin-table-exp )

table-constructor
    ::=   VALUES row-constructor-commalist

row-constructor
    ::=   scalar-exp | ( scalar-exp-commalist ) | ( table-exp )

select-exp
    ::=   SELECT [ ALL | DISTINCT ] select-item-commalist
             FROM table-ref-commalist
                [ WHERE cond-exp ]
                   [ GROUP BY colunn-ref-commalist ]
                      [ HAVING cond-exp ]

select-item
    ::=   scalar-exp [ [ AS ] column ]
        | [ range-variable . ] *
```

CONDITIONAL EXPRESSIONS

```
cond-exp
    ::=   cond-term
        | cond-exp OR cond-term

cond-term
    ::=   cond-factor
        | cond-term AND cond-factor

cond-factor
    ::=   [ NOT ] cond-test

cond-test
    ::=   cond-primary
             [ IS [ NOT ] ( TRUE | FALSE | UNKNOWN } ]

cond-primary
    ::=   simple-cond | ( cond-exp )

simple-cond
    ::=   comparison-cond
        | between-cond
        | like-cond
        | in-cond
        | match-cond
        | all-or-any-cond
        | exists-cond
        | unique-cond
        | overlaps-cond
        | test-for-null

comparison-cond
    ::=   row-constructor comparison-operator row-constructor

comparison-operator
    ::=   = | < | <= | > | >= | <>

between-cond
    ::=   row-constructor [ NOT ] BETWEEN row-constructor
                                       AND row-constructor

like-cond
    ::=   character-string-exp
          [ NOT ] LIKE character-string-exp
                [ ESCAPE character-string-exp ]

in-cond
    ::=   row-constructor [ NOT ] IN ( table-exp )
        | scalar-exp [ NOT ] IN ( scalar-exp-commalist )

match-cond
    ::=   row-constructor MATCH [ UNIQUE ]
                          [ PARTIAL | FULL ] ( table-exp )

all-or-any-cond
    ::=   row-constructor
              comparison-operator { ALL | ANY | SOME }
                                        ( table-exp )

exists-cond
    ::=   EXISTS ( table-exp )

unique-cond
    ::=   UNIQUE ( table-exp )
```

```
overlaps-cond
    ::=   ( scalar-exp, scalar-exp )
              OVERLAPS ( scalar-exp, scalar-exp )

test-for-null
    ::=   row-constructor IS [ NOT ] NULL
```

CONSTRAINTS

```
domain-constraint-def
    ::=   [ CONSTRAINT constraint ] CHECK ( cond-exp )
                                          [ deferrability ]

base-table-constraint-def
    ::=   [ CONSTRAINT constraint ]
            candidate-key-def [ deferrability ]
        | [ CONSTRAINT constraint ]
            foreign-key-def [ deferrability ]
        | [ CONSTRAINT constraint ]
            check-constraint-def [ deferrability ]

candidate-key-def
    ::=   { PRIMARY KEY | UNIQUE } ( column-commalist )

foreign-key-def
    ::=   FOREIGN KEY ( column-commalist ) references-def

references-def
    ::=   REFERENCES base-table [ ( column-commalist ) ]
             [ MATCH { FULL | PARTIAL } ;
             [ ON DELETE referential-action ]
             [ ON UPDATE referential-action ]

referential-action
    ::=   NO ACTION | CASCADE | SET DEFAULT | SET NULL

check-constraint-def
    ::=   CHECK ( cond-exp )

column-constraint-def
    ::=   [ CONSTRAINT constraint ]
            NOT NULL [ deferrability ]
        | [ CONSTRAINT constraint ]
            { PRIMARY KEY | UNIQUE } [ deferrability ]
        | [ CONSTRAINT constraint ]
            reterences-def [ deferrability ]
        | [ CONSTRAINT constraint ]
            CHECK ( cond-exp ) [ deferrability ]

set-constraints
    ::=   SET CONSTRAINTS { constraint-commalist | ALL }
                                      { DEFERRED | IMMEDIATE }
```

DYNAMIC SQL

```
execute-immediate
    ::=   EXECUTE IMMEDIATE param-or-var

prepare
    ::=   PREPARE prepared FROM param-or-var

prepared
    ::=   prepped-statement-container I param-or-var

deallocate-prepare
    ::=   DEALLOCATE PREPARE prepared

execute
    ::=   EXECUTE prepared [ INTO places ] [ USING arguments ]

places
    ::=   target-commalist | SQL DESCRIPTOR descriptor

arguments
    ::=   target-commalist | SQL DESCRIPTOR descriptor

descriptor
    ::=   lit-param-or-var

allocate-descriptor
    ::=   ALLOCATE DESCRIPTOR descriptor
                  [ WITH MAX lit-param-or-var ]

deallocate-descriptor
    ::=   DEALLOCATE DESCRIPTOR descriptor

describe-input
    ::=   DESCRIBE INPUT prepared
                  USING SQL DESCRIPTOR descriptor

describe-output
    ::=   DESCRIBE [ OUTPUT ] prepared
                  USING SQL DESCRIPTOR descriptor

get-descriptor-1
    ::=   GET DESCRIPTOR descriptor target = COUNT

get-descriptor-2
    ::=   GET DESCRIPTOR descriptor
              VALUE number get-assignment-commalist

get-assignment
    ::=   target = item-descriptor-element

set-descriptor-1
    ::=   SET DESCRIPTOR descriptor COUNT = lit-param-or-var

set-descriptor 2
    ::=   SET DESCRIPTOR descriptor
              VALUE number set-assignment-commalist

set-assiqnment
    ::=   item-descriptor-element = lit-param-or-var

dynamic-cursor-def
    ::=   DECLARE cursor [ INSENSITIVE ] [ SCROLL ] CURSOR
          FOR prepared
```

```
allocate-cursor
    ::=   ALLOCATE param-or-var [ INSENSITIVE ] [ SCROLL ] CURSOR
          FOR prepared

dynamic-open
    ::=   OPEN dynamic-cursor [ USING arguments ]

dynamic-cursor
    ::=   cursor | param-or-var

dynamic-close
    ::=   CLOSE dynamic-cursor

dynamic-fetch
    ::=   FETCH [ [ row-selector ] FROM ] dynamic-cursor
                INTO places

dynamic-positioned-delete
    ::=   DELETE [ FROM table ]
          WHERE CURRENT OF dynamic-cursor

dynamic-positioned-update
    ::=   UPDATE [ table ] SET assignment-commalist
          WHERE CURRENT OF dynamic-cursor
```

SCALAR EXPRESSIONS

```
scalar-exp
    ::=   numeric-exp
        | character-string-exp
        | bit-string-exp
        | datetime-exp
        | interval-exp

numeric-exp
    ::=   numeric-term
        | numeric-exp { + | - } numeric-term

numeric-term
    ::=   numeric-factor
        | numeric-term { * | / } numeric-factor

numeric-factor
    ::=   [ + | - ] numeric-primary

numeric-primary
    ::=   column-ref
        | lit-param-or-var
        | scalar-function-ref
        | aggregate-function-ref
        | ( table-exp )
        | ( numeric-exp )

aggregate-function-ref
    ::=   COUNT(*)
        | { AVG | MAX | MIN | SUM | COUNT }
                ( [ ALL | DISTINCT ] scalar-exp )

character-string-exp
    ::=   character-string-concatenation
        | character-string-primary

character-string-concatenation
    ::=   character-string-exp || character-string-primary

character-string-primary
    ::=   column-ref
        | lit-param-or-var
        | user-function-ref
        | scalar-function-ref
        | aggregate-function-ref
        | ( table-exp )
        | ( character-string-exp )

bit-string-exp
    ::=   bit-string-concatenation
        | bit-string-primary

bit-string-concatenation
    ::=   bit-string-exp || bit-string-primary

bit-string-primary
    ::=   column-ref
        | lit-param-or-var
        | scalar-function-ref
        | aggregate-function-ref
        | ( table-exp )
        | ( bit-string-exp )
```

```
datetime-exp
    ::=   datetime-term
        | interval-exp + datetime-term
        | datetime-exp { + I - } interval-term

datetime-term
    ::=   datetime-primary
                [ AT { LOCAL | TIME ZONE interval-exp } ]

datetime-primary
    ::=   column-ref
        | lit-param-or-var
        | datetime-function-ref
        | scalar-function-ref
        | aggregate-function-ref
        | ( table-exp )
        | ( datetime-exp )

interval-exp
    ::=   interval-term
        | interval-exp { + | - } interval-term
        | ( datetime exp - datetime-term ) start [ TO end ]

interval-term
    ::=   interval-factor
        | interval-term { * | / } numeric-factor
        | numeric-term * interval-factor

interval-factor
    ::=   [ + | - ] interval-primary [ start [ TO end ] ]

interval-primary
    ::=   column-ref
        | lit-param-or-var
        | scalar-function-ref
        | aggregate-function-ref
        | ( table-exp )
        | ( interval-exp )
```

MISCELLANEOUS

```
schema
    ::=   [ catalog . ] identifier

domain
    ::=   [ schema . ] identifier

table
    ::=   base-table | view

base-table
    ::=   [ schema . ] identifier

view
    ::=   [ schema . ] identifier

constraint
    ::=   [ schema . ] identifier

character-set
    ::=   [ schema . ] identifier

collation
    ::=   [ schema . ] identifier

translation
    ::=   [ schema . ] identifier

conversion
    ::=   [ schema . ] identifier

column-ref
    ::=   [ column-qualifier . ] column

column-qualifier
    ::=   table | range-variable

param-or-var
    ::=   parameter [ [ INDICATOR ] parameter ]
        | host-variable [ [ INDICATOR ] host-variable ]

lit-param-or-var
    ::=   literal | param-or-var

target
    ::=   param-or-var

number
    ::=   lit-param-or-var

niladic-function-ref
    ::=   user-function-ref
        | datetime-function-ref

user-fanction-ref
    ::=   USER
        | CURRENT_USER
        | SESSION USER
        | SYSTEM USER

datetime-function-ref
    ::=   CURRENT DATE
        | CURRENT TIME [ ( integer ] ]
        | CURRENT TIMESTAMP [ ( integer ) ]
```