

SQL-Syntax

query-expression ::=

query-term
| query-expression { UNION | EXCEPT } query-term

query-term ::=

query-primary
| query-term INTERSECT query-primary

query-primary ::=

query-specification
| (query-expression)

query-specification ::=

SELECT [ALL | DISTINCT] select-list
FROM table-reference [{ , table-reference } ...]
[WHERE search-condition]
[order-by-clause]
[group-by-clause]
[having-clause]

select-list ::=

*
| select-sublist [{ , select-sublist } ...]

select-sublist ::=

value-expression [[AS] column-name]
| table-name.*
| range-variable.*

table-reference ::=

table-name [range-variable]
| (query-expression) range-variable
| joined-table

value-expression ::=

numeric-value-expression
| string-value-expression
| datetime-value-expression
| interval-value-expression

numeric-value-expression ::=

term
| numeric-value-expression { + | - } term

term ::=

factor
| term { * | / } factor

factor ::=

[+ | -] numeric-primary

numeric-primary ::=

unsigned-numeric-value
| column-name
| set-function-specification
| numeric-value-function
| (numeric-value-expression)

set-function-specification ::=

COUNT (*)
| set-function-type ([ALL | DISTINCT] numeric-value-expression)

```

set-function-type ::=
    AVG | MAX | MIN | SUM | COUNT

joined-table ::=
    table-reference [ join-type ] JOIN table-reference ON search-condition
    | ( joined-table )

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

search-condition ::=
    boolean-term
    | search-condition OR boolean-term

boolean-term ::=
    [ NOT ] boolean-primary
    | boolean-term AND [ NOT ] boolean-primary

boolean-primary ::=
    predicate
    | ( search-condition )

predicate ::=
    comparsion-predicate
    | null-predicate
    | between-predicate
    | like-predicate
    | in-predicate
    | quantified-comparsion-predicate
    | exists-predicate

comparsion-predicate ::=
    row-value-constructor comp-op row-value-constructor

comp-op ::=
    = | < | < | > | <= | >=

row-value-constructor ::=
    value-expression
    | row-subquery

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

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

like-predicate ::=
    character-value-expression [ NOT ] LIKE character-value-expression

in-predicate ::=
    row-value-constructor [ NOT ] IN in-predicate-value

in-predicate-value ::=
    ( value-expression [{ , value-expression }...] )
    | ( table-subquery )

quantified-comparsion-predicate ::=
    row-value-constructor comp-op { ALL | { SOME | ANY } } ( table-subquery )

exists-predicate ::=

```

EXISTS (table-subquery)

order-by-clause ::=

ORDER BY column-name [ASC | DSC] [{ , column-name [ASC | DSC] } ...]

group-by-clause ::=

GROUP BY column-name [{ , column-name } ...]

having-clause ::=

HAVING search-condition

insert-statement ::=

INSERT INTO table-name [(column-name [{ , column-name } ...])] insert-source

insert-source ::=

VALUES (value-expression [{ , value-expression } ...])
 | TABLE table-name
 | query-expression
 | joined-table

delete-statement ::=

DELETE FROM table-name [range-variable] [WHERE search-condition]

update-statement ::=

UPDATE table-name [range-variable]
 SET column-name = row-value-constructor [{ , column-name = row-value-constructor } ...]
 [WHERE search-condition]

table-definition ::=

CREATE TABLE table-name
 (column-definition [{ , column-definition } ...]
 [table-constraint [{ , table-constraint } ...]])

column-definition ::=

column-name data-type [column-constraint [{ , column-constraint } ...]]

column-constraint ::=

| NOT NULL
 | UNIQUE
 | CHECK (check-condition-1)
 | DEFAULT { const | NULL }

table-constraint ::=

| CHECK (check-condition-2)
 | UNIQUE (column-name [{ , column-name } ...])
 | PRIMARY KEY (column-name [{ , column-name } ...])
 | FOREIGN KEY (column-name [{ , column-name } ...]) reference-specification

reference-specification ::=

REFERENCES table-name [(column-name [{ , column-name } ...])]
 [ON DELETE { CASCADE | SET NULL | RESTRICT | NO ACTION }]

alter-table-statement ::=

ALTER TABLE table-name action

action ::=

ADD column-definition
 | ADD table-constraint

rename-table-statement ::=

RENAME TABLE table-name TO new-table-name

view-definition ::=

```
CREATE VIEW table-name [ ( column-name [{ , column-name }...] )  
AS query-expression  
[ WITH CHECK OPTION ]
```

```
grant-statement ::=  
  GRANT privileg [{ , privileg }...]  
  ON table-name  
  TO grantee [{ , grantee }...]  
  [ WITH GRANT OPTION ]
```

```
privileg ::=  
  ALL PRIVILEGES  
  | SELECT  
  | DELETE  
  | INSERT [ ( column-name [{ , column-name }...] ) ]  
  | UPDATE [ ( column-name [{ , column-name }...] ) ]
```

```
grantee ::=  
  PUBLIC  
  | authorization-identifier
```

```
revoke-statement ::=  
  REVOKE privileg [{ , privileg }...]  
  ON table-name  
  FROM grantee [{ , grantee }...]
```