FIREBIRD SQL

The database for the new millennium

QUICK REFERENCE CARD
VERSION 3.0.0

Elementary datatypes

BIGINT	Signed 64 bit integer (± 2E+63)
BLOB [SUB_TYPE t]	<= 64K Bytes. Can hold CLOB's as well
BOOLEAN	FALSE or TRUE
CHAR[(n)]	<= 32.767 bytes. Fixed length char field
DATE	Date (1-jan-100 upto 29-feb-32768)
DECIMAL (p,s)	16, 32, 64 bits. (p,s) from 1 upto 18.
DOUBLE PRECISION	64 bits, 15 digits. (± 2E±308)
FLOAT	32 bits, 7 digits. (± 2E±38)
INTEGER	32 bits. ±2147483647
NUMERIC(p,s)	Alias for DECIMAL
SMALLINT	16 bits, ±32767
TIME	64 bits, 0:00am to 23:59.9999 pm
TIMESTAMP	64 bits, DATE + TIME
VARCHAR[(m)]	<= 32.767 chars. Size m is max length
<datatype>[n]	Array of type <datatype>

CHAR and VARCHAR variants
CHARACTER [VARYING] [(m)]
NCHAR [VARYING][(m)]
NATIONAL CHAR [VARYING][(m)]
NATIONAL CHARACTER [VARYING][(m)]

Constants

CURRENT_CONNECTION	Current database connection
CURRENT_DATE	Date of current system clock
CURRENT_ROLE	Current logged in role
CURRENT_SESSION	Integer representing current session
CURRENT_TIME[(p)]	Time of current system clock
CURRENT_TIMESTAMP[(p)]	Timestamp of current system clock
CURRENT_TRANSACTION	Integer representing glob. transaction
CURRENT_USER	See: USER
DELETING	TRUE if we are in a delete trigger
GDSCODE	Current native error code
INSERTING	TRUE if we are in a insert trigger
ROW_COUNT	Number of rows of last operation
SQLCODE	Error code – CHAR(5)
SQLSTATE	Error code – CHAR(5)
UPDATING	TRUE if we are in an update trigger
USER	Currently logged in user

Literal constants

'normal string'	Normal SQL String
1234	Normal integer number
123.45678	Normal floating point number
0x123AF	Normal hexadecimal number
x'0123BC'	Hexadecimal number string

SQL STATEMENTS

ALTER CHARACTER SET charset_name
SET DEFAULT COLLATION collation;

ALTER { **DATABASE** | **SCHEMA** }
<add_clause>
[ADD DIFFERENCE FILE 'filepath']
[{BEGIN | END} BACKUP]
[DROP DIFFERENCE FILE]
[DROP SHADOW <number>]
[DECRYPT]
[ENCRYPT WITH <plugin> [KEY keyname]]
[SET DEFAULT CHARACTER SET <charset>]
[SET { DROP LINGER | LINGER TO <sec>]};

<add_clause> ADD 'filespec' [<fileinfo>] [<add_clause> …]
<fileinfo> FILE { <length> <starting> }
<length> LENGTH [=] <n> [PAGE[S]]
<starting> STARTING [AT [PAGE]] <m>

ALTER DOMAIN { name | old_name TO new_name }
[<operation>];

<operation> SET DEFAULT { literal | NULL | USER } |
DROP DEFAULT |
ADD [CONSTRAINT] CHECK (<condition>) |
DROP CONSTRAINT |
{DROP | SET} [NOT] NULL |
TYPE datatype ;

<condition> VALUE <operator> <value> |
VALUE [NOT] BETWEEN <value> AND <value> |
VALUE [NOT] LIKE <value> [ESCAPE <value>] |
VALUE [NOT] IN (<value> [, <value> …]) |
VALUE IS [NOT] NULL |
VALUE IS [NOT] { TRUE | FALSE }
VALUE [NOT] CONTAINING <value> |
VALUE [NOT] STARTING [WITH] <value> |
(<condition>) |
NOT <condition> |
<condition> OR <condition> |
<condition> AND <condition>

<operator> { = | < | > | <= | >= | !< | !> | <> | != }

ALTER EXCEPTION name 'message of exception'

ALTER FUNCTION fname [(param1, param2 [,…])]
RETURNS <type>
AS <body>

ALTER EXTERNAL FUNCTION fname
<modification> [<modification>];
<modification> ENTRY_POINT 'new-entry-point' |
MODULE_NAME 'new-module-name'

ALTER GENERATOR name RESTART [WITH <newvalue>];

ALTER INDEX name { ACTIVE | INACTIVE }

ALTER [GLOBAL] **MAPPING** name USING
{PLUGIN name [IN database] |
ANY PLUGIN [IN database | SEVERWIDE] |
MAPPING [IN database] | '*' [IN database]}
FROM { ANY type | <typename>}
TO { USER | ROLE } name

ALTER PROCEDURE name
[(param <datatype> [{= | DEFAULT} *value*]
[, …])]
[RETURNS (param <datatype> [, …])]
AS <procedure_body>

ALTER SEQUENCE name RESTART [WITH <newvalue>];

ALTER TABLE table <operation> [,<operation> …];

<operation> ADD name <col_def> [<col_modifier>] |
ADD <tconstraint> [cons_implem] |
ADD blob COMPUTED BY <expression> |
ALTER [COLUMN] name <alt_col_clause> |
DROP column |
DROP CONSTRAINT constraint

<alt_col_clause> TO new_col_name |
TYPE new_col_datatype |
POSITION new_col_position |
SET DEFAULT value |
RESTART [WITH value] |
{DROP | SET} NOT NULL |
DROP DEFAULT

<col_def> < datatype> [array_dim] |
COMPUTED [BY] (< expr>) |
GENERATED ALWAYS AS (<expr) |
domain

<col_modifier> DEFAULT { literal | NULL | USER} |
NOT NULL |
<col_constraint> [cons_implem]
COLLATE collation

```
< datatype>     Elementary datatype |
                BLOB [(segment length [,subtype])]

<array_dim>     [[x:]y [,[x:]y…]]

<col_constraint> [CONSTRAINT constraint]
                UNIQUE |
                PRIMARY KEY |
                REFERENCES table [(column [, …])]
                [ON DELETE <cons_action]
                [ON UPDATE <cons_action] |
                CHECK ( <search_condition>)

<cons_implem>   USING [ASC[ENDING] | DESC[ENDING]]
                INDEX <index_name>

<cons_action>   NO ACTION | CASCADE | SET NULL
                SET DEFAULT

<tconstraint>   [CONSTRAINT constraint]
                { PRIMARY KEY | UNIQUE } (col [,col …]) |
                FOREIGN KEY (col [, …])
                      REFERENCES table
                [ON DELETE <cons_action>]
                [ON UPDATE <cons_action>] |
                CHECK ( <search_condition>)

<search_condition> <val> <operator> { <val> | ( <select_one>)} |
                <val> [NOT] BETWEEN <val> AND <val> |
                <val> [NOT] LIKE <val> [ESCAPE <val>] |
                <val> [NOT] IN ( <val> [ , …] | <select_list>) |
                <val> IS [NOT] NULL |
                <val> {>= | <=} |
                <val> [NOT] {= | < | >} |
                {ALL | SOME | ANY} ( <select_list>) |
                EXISTS ( <select_expr>) |
                SINGULAR ( <select_expr>) |
                <val> [NOT] CONTAINING <val> |
                <val> [NOT] STARTING [WITH] <val> |
                ( <search_condition>) |
                NOT <search_condition> |
                <search_condition> OR <search_condition> |
                <search_condition> AND <search_condition>

<val>           { col [ <array_dim>] | :variable
                | <constant> | <expr> | <function>
                | udf ([<val> [,<val> …]])
                | NULL | USER | RDB$DB_KEY | ? }
                [COLLATE collation]

<operator>      { = | < | > | <= | >= | !< | !> | <> | != }

<select_one>    SELECT on a single column;
                returns exactly one value.
<select_list>   SELECT on a single column;
                returns zero or more values.
<select_expr>   SELECT on a list of values;
                returns zero or more values.

ALTER TRIGGER name <modifcation> [,<modification> …] ;

<modification>  [ACTIVE | INACTIVE] |
                [{BEFORE | AFTER} <multiple>] |
                [POSITION number] |
                [AS <trigger_body>] ;

<multiple>      <single_action> [ OR <single_action [OR …]]
<single>        {DELETE | INSERT | UPDATE}

ALTER [CURRENT] USER <username> SET PASSWORD '<pwd>'
                [pwd-options][TAGS (tag [,tag [,…]])
                [USING PLUGIN plugin];

<pwd-options>   [PASSWORD 'password']
                [FIRSTNAME 'firstname']
                [MIDDLENAME 'middlename']
                [LASTNAME 'lastname'];
<tag>           {NAME='string' | DROP tagname}

COMMENT ON <object> IS { 'text' | NULL };

<object>        DATABASE
                | <basic-type> objectname
                | COLUMN relationname.fieldname
                | PARAMETER procname.paramname

<basic-type>    CHARACTER SET | COLLATION | DOMAIN |
                EXCEPTION | EXTERNAL FUNCTION | FILTER |
                GENERATOR | INDEX | PROCEDURE | ROLE |
                SEQUENCE | TABLE | TRIGGER | VIEW

COMMIT [WORK] [TRANSACTION name]
                [RELEASE] [RETAIN [SNAPSHOT]]

CONNECT [TO] {ALL | DEFAULT} <config_opts>
                | <db_specs> <config_opts>
                [,<db_specs> <config_opts>...];

< db_specs>     dbhandle | {'filespec' | :variable} AS dbhandle

< config_opts>  [USER {'username' | :variable}
                [PASSWORD {'password' | :variable}]
                [ROLE {'rolename' | :variable}]
                [CACHE int [BUFFERS]]

CREATE { DATABASE | SCHEMA } 'filespec'
                [USER 'username' [PASSWORD 'password']]
                [PAGE_SIZE [=] int]
                [LENGTH [=] <n> [PAGE[S]]]
                [SET NAMES charset_name]
                [DEFAULT CHARACTER SET charset
                      [COLLATION collation]]
                [ <secondary_file>];

<secondary_file> FILE 'filespec' [ <fileinfo>] [<secondary_file>]
<fileinfo>      [LENGTH [=] <n> [PAGE[S]] |
                STARTING [AT [PAGE]] <m> } [<fileinfo>]

CREATE DOMAIN domain [AS] < datatype> [<array_dim>]
                [DEFAULT { literal | NULL | USER}]
                [NOT NULL]
                [CHECK ( <condition>)]
                [CHARSET { charset | NONE }]
                [COLLATE collation];

< datatype>     Elementary datatype
<array_dim>     [[x:]y [,[x:]y ..]]

<condition>     VALUE <operator> value |
                VALUE [NOT] BETWEEN value AND value |
                VALUE [NOT] LIKE value [ESCAPE value] |
                VALUE [NOT] IN ( value [ , value …]) |
                VALUE IS [NOT] NULL |
                VALUE [NOT] CONTAINING value |
                VALUE [NOT] STARTING [WITH] value |
                ( <condition>) |
                NOT <condition> |
                <condition> OR <condition> |
                <condition> AND <condition>

< operator>     {= | < | > | <= | >= | !< | !> | <> | !=}

CREATE [ OR ALTER ] EXCEPTION name 'message text';

CREATE [ OR ALTER] FUNCTION fname [(param1 [,…])]
                RETURNS <type>
                AS <body> [terminator]

CREATE GENERATOR name [START WITH value];

CREATE [UNIQUE] [ASC[ENDING] | [DESC[ENDING]] INDEX
                indexname ON tablename
                { (columname [, ...]) | COMPUTED BY (expr) }

CREATE [OR ALTER] [GLOBAL] MAPPING name USING
                {PLUGIN name [IN database] |
                ANY PLUGIN [IN database | SEVERWIDE] |
                MAPPING [IN database] | '*' [IN database]}
                FROM { ANY type | <typename>}
                TO { USER | ROLE } name

CREATE [OR ALTER] PACKAGE name AS
                BEGIN
                      [<package_decl> …]
                END
```

```
<package_decl>  <function_decl>;
                <proc_decl>;

<function_decl>  FUNCTION name [(param [,…])] RETURNS type
<proc_decl>      PROCEDURE name [(par [,…])] RETURNS type

{CREATE | RECREATE} PACKAGE BODY name AS
                BEGIN
                  [ <package_item> ]
                END

CREATE [OR ALTER] PROCEDURE name
          [(param <datatype> [{= | DEFAULT} value] [, …])]
          [RETURNS (param <datatype> [,…])]
          AS <body> [ terminator]

<body>          [ <declaration_list>]
                <block>

<declaration_list>
                DECLARE [VARIABLE] var <datatype>;
                [DECLARE [VARIABLE] var <datatype>; …]

<block>         BEGIN
                      <compound_statement>
                      [<compound_statement> …]
                END

<compound_statement>
                {<block> | statement;}

<datatype>      Elementary datatype

CREATE ROLE role-name;

CREATE SEQUENCE sequence-name [START WITH value];

CREATE SHADOW <num> [AUTO | MANUAL] [CONDITIONAL]
                'filespec' [LENGTH [=] <n> [PAGE[S]]]
                [<secondary_file>];

<secondary_file> FILE 'filespec' [<fileinfo>] [<secondary_file>]

<fileinfo>      LENGTH [=] int [PAGE[S]] |
                STARTING [AT [PAGE]] int

CREATE TABLE table [EXTERNAL [FILE] 'filespec']
                ( <col_def> [,<col_def> | <tconstraint> …]);

<col_def>       column <datatype>  [<col_modifier>] |
                column COMPUTED [BY] (<expr>) [<col_mod>] |
                column GENERATED ALWAYS AS (expr) [mod] |
                column GENERATED BY DEFAULT AS
                      IDENTITY [(START WITH value)]
                column domain [<col_modifier>]

<col_modifier>  DEFAULT { literal | NULL | USER} |
                NOT NULL |
                <col_constraint> [cons_implem] |
                COLLATE collation

< datatype>     Elementary datatype [<array_dim>]  |
                BLOB [(segmentlength [,subtype])]

<array_dim>     [[x:]y [,[x:]y …]]

<col_constraint> [CONSTRAINT constraint]
                UNIQUE |
                PRIMARY KEY |
                REFERENCES table [(column)]
                      [ON DELETE <cons_action>]
                      [ON UPDATE <cons_action>] |
                CHECK ( <search_condition>)

<cons_implem>   USING [ASC[ENDING] | DESC[ENDING]]
                INDEX <index_name>

<cons_action>   NO ACTION | CASCADE | SET NULL |
                SET DEFAULT

<tconstraint>   [CONSTRAINT constraint]
                {{ PRIMARY KEY | UNIQUE } (col [,col …]) |
                FOREIGN KEY (col [, …]) REFERENCES table
                      [ON DELETE <cons_action>]
                      [ON UPDATE <cons_action>] |
                CHECK ( <search_condition>) }

<search_condition> = <val> <operator> { <val> | ( <select_one>)} |
                <val> [NOT] BETWEEN <val> AND <val> |
                <val> [NOT] LIKE <val> [ESCAPE <val>] |
                <val> [NOT] IN ( <val> [ , …] | <select_list>) |
                <val> IS [NOT] NULL |
                <val> {>= | <=} |
                <val> [NOT] {= | < | >} |
                {ALL | SOME | ANY} ( <select_list>) |
                EXISTS ( <select_expr>) |
                SINGULAR ( <select_expr>) |
                <val> [NOT] CONTAINING <val> |
                <val> [NOT] STARTING [WITH] <val> |
                ( <search_condition>) |
                NOT <search_condition> |
                <search_condition> OR <search_condition>
                <search_condition> AND <search_condition>
<val>           { col [ <array_dim>] | :variable
                | <constant> | <expr> | <function>
                | udf ([<val> [,<val> …]])
                | NULL | USER | RDB$DB_KEY | ? }
                [COLLATE collation]

<operator>      { = | < | > | <= | >= | !< | !> | <> | != }

<select_one>    SELECT on a single column;
                returns exactly one value.
<select_list>   SELECT on a single column;
                returns zero or more values.
<select_expr>   SELECT on a list of values;
                returns zero or more values.

CREATE GLOBAL TEMPORARY TABLE table
                ….
                [ ON COMMIT
                { DELETE | PRESERVE} ROWS ]

CREATE [ OR ALTER ] TRIGGER name [ FOR table ]
                [ACTIVE | INACTIVE]
                {BEFORE | AFTER} <multiple> [ ON table ]
                [POSITION number]
                AS <body>

 <multiple>     <single> [ OR <single [OR …]]

 <single>       { DELETE | INSERT | UPDATE }

<body>           [<variable_declaration_list>]
                 <block>

<variable_declaration_list>
                DECLARE VARIABLE var <datatype>;
                [DECLARE VARIABLE var <datatype>; …]

<block>         BEGIN
                      <compound_statement>
                      [<compound_statement> …]
                END

<compound_statement>
                {<block> | statement;}

CREATE [OR ALTER] TRIGGER name
                [ACTIVE | INACTIVE]
                { ON <event> |
                { BEFORE | AFTER } <ddl_event>}
                [POSITION n]
                AS <body>

<event>         CONNECT | DISCONNECT |
                TRANSACTION START |
                TRANSACTION COMMIT |
                TRANSACTION ROLLBACK

<ddl_event>     ANY DDL STATEMENT |
                <ddl_item> [ OR <ddl_item> …]

<ddl_item>      {CREATE | ALTER | DROP} <object>
```

CREATE [OR ALTER] **USER** <username> PASSWORD '<pwd>'
[pwd-options][TAGS (tag [,tag [,…]])
[USING PLUGIN plugin];

<pwd-options> [PASSWORD 'password']
[FIRSTNAME 'firstname']
[MIDDLENAME 'middlename']
[LASTNAME 'lastname'];

<tag> {NAME='string' | DROP tagname}

CREATE VIEW name [(view_column [, …])]
AS <select> [WITH CHECK OPTION];

DECLARE [VARIABLE] cursor **CURSOR**
FOR (<select-statement>)
[FOR UPDATE OF <col> [,<col>…]];

DECLARE cursor **CURSOR** FOR
{ READ BLOB column FROM table |
INSERT BLOB column INTO table }
[FILTER [FROM subtype] TO subtype]
[MAXIMUM_SEGMENT length] ;

DECLARE EXTERNAL FUNCTION localname
[<type_decl> [, <type_decl> ...]]
RETURNS {<return_type_decl> |
PARAMETER 1-based_pos} [FREE_IT]
ENTRY_POINT 'function_name'
MODULE_NAME 'library_name'

<type_decl> sqltype [BY DESCRIPTOR] | CSTRING(length)

<return_type> sqltype [BY {DESCRIPTOR | VALUE }] |
CSTRING(length)

DECLARE FILTER filtername
INPUT_TYPE <blobtype>
OUTPUT_TYPE <blobtype>
ENTRY_POINT 'function_name' MODULE_NAME
'library_name' ;

<blobtype> number | <mnemonic>
<mnemonic> binary | text | blr | acl | ranges | summary |
format |
transaction_description |
external_file_description

DELETE [TRANSACTION transaction] FROM table
[WHERE <condition> |
WHERE CURRENT OF cursor]
[PLAN planitems]
[ORDER BY value [,…]]
[ROWS <expr1> [TO <expr2>]]
[RETURNING column [,…] [INTO :var [,…]]] ;

<condition> Search condition as specified in SELECT.

DESCRIBE [INPUT | OUTPUT] statement
{ INTO | USING } SQL DESCRIPTOR xsqlda;

DISCONNECT {{ ALL | DEFAULT} | dbhandle [, …] }

DROP DATABASE;
DROP DOMAIN name;
DROP EXCEPTION name;
DROP EXTERNAL FUNCTION name;
DROP FILTER name;
DROP GENERATOR name;
DROP INDEX name;
DROP [GLOBAL] **MAPPING** name;
DROP PACKAGE name;
DROP PACKAGE BODY name;
DROP PROCEDURE name;
DROP SEQUENCE name;
DROP ROLE name;
DROP SHADOW name;
DROP TABLE name;
DROP TRIGGER name;
DROP USER name;
DROP VIEW name;

EVENT INIT request_name [dbhandle]
[(' string' | :variable [, ' string' | : variable …]);

EVENT WAIT request_name;

EXECUTE [TRANSACTION transaction] statement
[USING SQL DESCRIPTOR xsqlda]
[INTO SQL DESCRIPTOR xsqlda];

EXECUTE IMMEDIATE [TRANSACTION transaction]
{:variable | 'string'}
[USING SQL DESCRIPTOR xsqlda];

EXECUTE PROCEDURE { value [,…] | (value [,…])}
[RETURNING VALUES { param [,…] | (param [,…]) }]

FETCH cursor [INTO [:hostvar [[INDICATOR] :indvar] [, …]]

FETCH cursor INTO [:<buffer> [[INDICATOR] :segment_length];

GEN_ID(generator, step)

GRANT < privileges> { <tab_priv> | <proc_priv> | <role_priv> };

<tab_priv> ON [TABLE] { table | view }
TO { <object> | <userlist> | GROUP UNIX_group}

<proc_priv> EXECUTE ON PROCEDURE name
TO { <object> | <userlist>}
[GRANTED { BY | AS } [USER] username]

<role_priv> <role_granted> TO {PUBLIC | <grantee_list>}

<privileges> {ALL [PRIVILEGES] | <privilege_list>}

<privilege_list> SELECT | DELETE | INSERT |
UPDATE [(col [,col …])] |
REFERENCES [(col [, …])] [,<privilege_list>
…]

<object> PROCEDURE procedure_name |
TRIGGER trigger_name |
VIEW view_name |
PUBLIC |
[, <object> …]

<userlist> [USER] username | rolename | Unix_user |
[, <userlist> …] [WITH GRANT OPTION]

<role_granted> rolename [,rolename …]

<grantee_list> [USER] username [,[USER] username …]
[WITH ADMIN OPTION]

INSERT [TRANSACTION name]
INTO {tablename | viewname} [(<columns>)]
{ <value_clause> | select-expression }

<value_clause> VALUES (<values>) | DEFAULT VALUES
[RETURNING <columns> [INTO <variables>]]

<columns> colname [, colname ...]
<values> value [, value ...]
<variables> :varname [, :varname ...]

INSERT CURSOR cursor VALUES (:buffer [INDICATOR]
:buflen);

OPEN [TRANSACTION transaction] cursor;

OPEN [TRANSACTION transaction] cursor
[USING SQL DESCRIPTOR xsqlda]

OPEN [TRANSACTION name] cursor
{INTO | USING} : blob_id;

PREPARE [TRANSACTION transaction] statement
[INTO SQL DESCRIPTOR xsqlda]
FROM {:variable | 'string'};

RECREATE EXCEPTION See: create exception
RECREATE PROCEDURE See: create procedure
RECREATE TABLE See: create table
RECREATE TRIGGER See: create trigger
RECREATE VIEW See: create view

RELEASE SAVEPOINT <savepointname> [ONLY];

REVOKE [{ GRANT | ADMIN} OPTION FOR]
{ <tab_priv> | <proc_priv> | <role_priv> };

<tab_priv> < privileges> ON [TABLE] { table | view }
FROM { <object> | <userlist> | < rolelist> |
GROUP UNIX_group }
[GRANTED { BY | AS} [USER] username]

<proc_priv> EXECUTE ON PROCEDURE procname
FROM { <object> | <userlist> | <rolelist>}

<role_priv> <role_granted> FROM
{ PUBLIC | <grantee_list>}

<privileges> {ALL [PRIVILEGES] | <privilege_list>}

<privilege_list> SELECT | DELETE | INSERT |
UPDATE [(col [,col …])] |
REFERENCES [(col [,…])] [,<privilege_list> …]

<object> CHARACTER SET name |
COLLATION name |
DOMAIN name |
EXCEPTION name |
GENERATOR name |
PROCEDURE name |
SEQUENCE name |
TRIGGER name |
VIEW name |
PUBLIC | [, <object>]}

<userlist> [USER] username [,[USER] username …]
<rolelist> [ROLE] rolename [,[ROLE] rolename …]
<role_granted> rolename [,rolename …]

<grantee_list> [USER] username [,[USER] username …]

ROLLBACK [WORK] [TRANSACTION name]
[RETAIN SNAPSHOT |
TO [SAVEPOINT] <savepointname>] |
RELEASE];

SAVEPOINT <savepointname>;

SELECT [TRANSACTION transaction]
[FIRST (expr)] [SKIP (expression)]
[DISTINCT | ALL]
{* | <value> [,<value> …]}
[INTO :var [,:var …]]
FROM <tableref> [,<tableref> …]
[WHERE <search_condition>]
[GROUP BY col [COLLATE collation] ,…]
[HAVING <search_condition>]
[UNION <select_expr> [{DISTINCT | ALL}]]
[PLAN <plan_expr>]
[ORDER BY <order_list>]
[ROWS <expr1> [TO <expr2>]]
[FOR UPDATE [OF col [,col …]] [WITH LOCK]]
;

<value> { col [<array_dim>] | :variable |
<constant> | <expr> | <function> | <case>
udf ([<value> [,<value> …]]) |
NULL | USER | RDB$DB_KEY | ? }
[COLLATE collation] [AS alias]

<case> CASE <value>
[WHEN <value> THEN <expression>]
[WHEN <condition> THEN <expression]
[ELSE <expression>]
END

<array_dim> **[**[x:]y [,[x:]y …]**]**

<constant> num | 'string' | charsetname 'string'

<function> COUNT (* | [ALL] <val> | DISTINCT <val>) |
SUM ([ALL] <val> | DISTINCT <val>) |
AVG ([ALL] <val> | DISTINCT <val>) |
MAX ([ALL] <val> | DISTINCT <val>) |
MIN ([ALL] <val> | DISTINCT <val>) |
CAST (<val> AS < datatype>) |
GEN_ID (generator, <val>) |
NEXT VALUE FOR <seq> |
UPPER (<val>) |

<tableref> <joined_table> | table | view | (SELECT …) |
Procedure [(<val> [,<val> …])] [alias]

<joined_table> <tableref> <join_type> JOIN <tableref> ON
<search_condition> | (<joined_table>) |
<tableref> NATURAL <join_type> JOIN
<tableref> USING (column [,…])

<join_type> INNER | CROSS |
{LEFT | RIGHT | FULL } OUTER

<search_condition> <val> <operator> {<val> | (<select_one>) } |
<val> [NOT] BETWEEN <val> AND <val> |
<val> [NOT] LIKE <val> [ESCAPE <val>] |
<val> [NOT] SIMULAR TO <pat> [ESCAPE ‘char’]
<val> [NOT] IN (<val> [,<val> …] |<sel_list>) |
<val> IS [NOT] NULL |
<val> IS [NOT] DISTINCT FROM <val>
<val> {>= | <=} |
<val> [NOT] {= | < | >} |
{ALL | SOME | ANY} (<select_list>) |
EXISTS (<select_expr>) |
SINGULAR (<select_expr>) |
<val> [NOT] CONTAINING <val> |
<val> [NOT] STARTING [WITH] <val> |
(<search_condition>) |
NOT <search_condition> |
<search_condition> OR <search_condition> |
<search_condition> AND <search_condition>

<operator> { = | < | > | <= | >= | !< | !> | <> | != }

<plan_expr> [JOIN | [SORT] [MERGE]] ({<plan_item> |
plan_expr>} [,{<plan_item> | <plan_expr>} …])

<plan_item> { table | alias}
{NATURAL | INDEX (<index> [,<index> …]) |
ORDER <index>}

<order_list> { col | int} [COLLATE collation]
[ASC[ENDING] | DESC[ENDING]]
[NULLS { FIRST | LAST}]
[, <order_list> …]

SET { **DATABASE** | SCHEMA} dbhandle =
[GLOBAL | STATIC | EXTERN]
[COMPILETIME]
[FILENAME] ’dbname’
[USER ’name’ PASSWORD ’string’]
[RUNTIME [FILENAME] {’dbname’ | :var}
[USER {’name’ | :var}
PASSWORD {’string’ |:var}]];

SET GENERATOR name TO new_value;

SET NAMES [character_set | :variable];

SET PLANONLY [{ ON | OFF}];

SET SQL DIALECT { 1 | 2 | 3 };

SET ROLE rolename;
SET TRUSTED **ROLE**;

SET STATISTICS INDEX index_name;

```
SET TRANSACTION [NAME transaction]
                [ READ WRITE | READ ONLY ]
                [ WAIT | NO WAIT ]
                [ LOCK TIMEOUT seconds]
                [ NO AUTO UNDO ]
                [ IGNORE LIMBO ]
                [ [ ISOLATION LEVEL ]
                      {SNAPSHOT [ TABLE STABILITY ] |
                      READ COMMITTED |
                      [ [ NO ] RECORD_VERSION ] } ]
                [ RESERVING <reserving> |
                      USING dbhandle [,dbhandle …]];

<reserving>     table [, table …]
                [ FOR [ SHARED | PROTECTED ]
                { READ | WRITE } ] [,<reserving_clause> …]

UPDATE [TRANSACTION transaction] { table | view } [[AS] alias]
                SET col = <value> [,col = <value> …]
                [ WHERE <search_condition> |
                WHERE CURRENT OF cursor];

UPDATE { table | view} [ [AS] alias]
                SET col = <val> [,col = <val> …]
                [ WHERE { <search_condition> |
                  CURRENT OF cursorname }
                [ PLAN plan_items ]
                [ ORDER BY sort_items ]
                [ ROWS <m> [TO <n>] ]
                [RETURNING column [,…] [INTO :var [,…]]]

WITH [ RECURSIVE]
        alias [(column [,...])]
        AS (<select-statement) [, <alias>AS <select> …]
        SELECT …
           FROM alias [,...] <table> [,...]
```

PSQL Procedural SQL (Stored-procedures & Triggers)

```
/* This is a Firebird comment */
-- This is a ANSI / ISO:9075 comment

<block>         BEGIN
                      < compound_statement> |
                      LEAVE [label]|
                      BREAK  |
                      [<compound_statement> …]
                END

<compound_statement>      {< block> | statement;}

variable = <expression>;

CLOSE cursorname;

CONTINUE label;

DECLARE [VARIABLE] variable  <type> ;

<type>          datatype [{'=' | DEFAULT} value] |
                TYPE OF domain_name |
                TYPE OF COLUMN <table or view>.column

DECLARE  name SCROLL CURSOR FOR ( select-expression);

EXCEPTION [name][value];
EXCEPTION name USING (value [,…]);
                -- values reference '@n' strings (1 based!)

EXECUTE PROCEDURE name [ (:param [,:param …] ) ]
          [RETURNING_VALUES :param [,:param …]];

EXECUTE STATEMENT string
                [INTO :var1 [, :var<n>…. [DO <compound>]] ;

EXIT

FETCH cursorname INTO :var [,…] ;

FETCH {NEXT|PRIOR|FIRST|LAST|ABSOLUTE <n>|RELATIVE n}
                FROM cursorname [INTO :var [,…]];

[ label: ]

FOR <select_expr> [AS CURSOR name]
                DO <compound_statement>

IF (<condition>)  THEN <compound_statement>
                [ELSE <compound_statement>]

IN AUTONOMOUS TRANSACTION
             DO <statement >;

INSERT OR UPDATE <table or view> [ column [,…] ]
                VALUES (value [,…] )
                [ MATCHING ( column [,…]) ]
                [ RETURNING (column [,…] [INTO :var [, …]]]

MERGE INTO <table or view> [ [AS] alias ]
          USING <table or view> [ [AS] alias ]
          ON <condition>
          { [ <matched> ] | [ <not matched> ] }

<matched>       WHEN MATCHED THEN
                UPDATE SET <assignment list>

<not matched>   WHEN NOT MATCHED THEN
                INSERT [ ( column [,…] )]
                VALUES (value [,…])

NEW.column      (Triggers only)

OLD.column      (Triggers only)

OPEN cursorname;

POST_EVENT 'event_name' | col;

SELECT          <select_clause>
                <from_clause>
                [<where_clause>]
                [<group_by_clause>]
                [<having_clause>]
                [<union_expression>]
                [<plan_clause>]
                [<ordering_clause>]
                <into_clause>;

<into_clause>   INTO :param [, :param …]

SUSPEND;

WHEN { < error> [, <error> …] | ANY }
                DO < compound_statement>

< error>        { EXCEPTION exception_name |
                SQLCODE number  |
                GDSCODE errcode |
                SQLSTATE state }

WHENEVER { NOT FOUND | SQLERROR | SQLWARNING}
            { GOTO <label> | CONTINUE}

[label:]
WHILE (<condition>) DO <compound_statement>
```

ESQL Embedded SQL (precompiled SQL)

```
BASED ON [dbhandle.] table.column[.segment] variable;

BEGIN DECLARE SECTION;

END DECLARE SECTION;

EXECUTE <statement>

DECLARE TABLE table (<create-table>);

<create-table>  See the 'CREATE TABLE' statement.
```

Internal functions to be used in DML/DDL/Procedures

Numeric functions

ABS(<number>)	Absolute value
ASCII_CHAR(<number>)	Returns ASCII 0 <= n <= 255
ASCII_VAL(<string>)	Returns n
BIN_AND(<number>[,...])	Bitwise AND of numbers
BIN_NOT(<number>)	Bitwise NOT of number
BIN_OR(<number [,...])	Bitwise OR of numbers
BIN_SHL(<number>,<m>)	Bitwise shift left m places
BIN_SHR(<number>,<m>)	Bitwise shift right m places
BIN_XOR(<number>[,...])	Bitwise XOR of numbers
CEIL(number)	Integer ceiling of number
CEILING(number)	Integer ceiling of number
EXP(<number>)	Exponent from 'e'
FLOOR(<number>)	Integer floor of number
LN(<number>)	Natural logarithm
LOG(<number>,<base>)	Logarithm of base
LOG10(<number>)	10 Logarithm of number
MOD(<n>,<m>)	Modulo of n/m
PI()	Returns 3.14... etc
POWER(<number>,<pow>)	Number to the <pow>er.
RAND()	Random number
ROUND(<number>[,<scale>])	
SIGN(<number>)	Returns -1, 0 or 1
SQRT(<number>)	Square root of number
TRUNC(<number>[,<scale>])	Truncate number

String functions

BIT_LENGTH(string)	Returns number of bits
CHAR_LENGTH(string)	Returns number of bytes
CHARACTER_LENGTH(s)	See char_length
HASH(<string>)	Generate hash value
LEFT(<string>,<number>)	Left <n> characters of string
LOWER(string)	Returns string in lowercase
LPAD(<string>,<n>[,<str>])	Left padding of string
OCTET_LENGTH(string)	Returns number of bytes

OVERLAY(str1 PLACING str2 FROM start [FOR length])
POSITION(<string> IN <string>)
POSITION(<string>,<string>[,start])
REPLACE(<searched>,<find>,<replace>)

RIGHT(<string>,<number>)	Rightmost number characters

RPAD(<string>,<number>[,<string>])
SUBSTRING(string FROM start [FOR length]
SUBSTRING(string [NOT] SIMULAR TO <pattern>
ESCAPE <char>)
TRIM([[BOTH|LEADING|TRAILING][string] FROM] string)

UPPER(string)	Returns string in uppercase

Generating functions

GEN_ID(generator,step)	Returns generated unique value
GEN_UUID()	Generate a new 16 octet UUID

Conversional functions

CAST(expression AS type)	Converts column to another type
CHAR_TO_UID(<string>)	Conversion to 16 octets
UUID_TO_CHAR(string)	16 octets to string

Logical functions

COALASCE(val,val,…)	Returns the first non-NULL in list
DECODE(<expression>,<search>,<result>,..,..<default>)	
IIF(condition,true,false)	Returns true/false part condition
NULLIF(val1,val2)	Null if (val1 = val2), val1 otherwise
NTH_VALUE(expression,off)	Nth (offset) value in expression

Select set functions

DENSE_RANK()	Dense ranking of a select
FIRST_VALUE(expression)	First value in expression
LAG(expr [,offset [,default]])	Lagging behind in select
LAST_VALUE(expression)	Last value in expression
LEAD(expr [,offset [,default]])	Leading value in expression
LIST(expression [, …])	
MAXVALUE(<value>[,...])	Max of all values
MINVALUE(<value>[,...])	Min of all values
RANK()	Ranking of a select
ROW_NUMBER()	Row number of a select

Trigonometric functions

ACOS(<number>)	Arc cosine
ACOSH(<number>)	Arc cosine hyperbole
ASIN(<number>)	Arc sine
ASINH(<number>)	Arc sine hyperbole
ATAN(<number>)	Arc tangent
ATAN2(<n>,<m>)	Arc tangent n/m
ATANH(<number>)	Arc tangent hyperbole
COS(<number>)	Cosine of number
COSH(<number>)	Cosine hyperbolic of number
COT(<number>)	Cotangent = 1/ TAN(number)
SIN(<number>)	Sinus
SINH(<number>)	Sinus hyperbolic of arc
TAN(<number>)	Tangent
TANH(<number>)	Tangent hyperbolic of arc

Internal statistical functions

CORR(expr)	Correlation coëfficient
COVAR_POP(expr)	Covariance of population
COVAR_SAMP(expr)	Covariance of sample
STDDEV_POP(expr)	Population standard deviation
STDDEV_SAMP(expr)	Sample Standard deviation
VAR_POP(expr)	Population variance
VAR_SAMP(expr)	Sample variance

Special datetime functions

DATEADD(<number> <part> TO <time>)
DATEADD(<part>, <number>, <time>)
DATEDIFF(<part> FROM <time> TO <time>)
DATEDIFF(<part>, <time>, <time>)
EXTRACT(<part> FROM <time>)

<part>	YEAR | MONTH | WEEK | DAY | HOUR | MINUTE | SECOND
<time>	DATE | TIME | TIMESTAMP

System context functions

RDB$GET_CONTEXT(n,var)	Gets the context variable
RDB$SET_CONTEXT(n,var)	Sets the context variable

N = Namespaces for context

SYSTEM	System context -> see <sysvars>
USER_SESSION	Users session, initially empty
USER_TRANSACTION	Users transaction, initially empty

var = <sysvars>

DB_NAME	Full database path, or alias
NETWORK_PROTOCOL	TCPv6, WNET, XNET or NULL
CLIENT_ADDRESS	Client's IP address
CURRENT_USER	Same as the constant
CURRENT_ROLE	Same as the constant
SESSION_ID	= CURRENT_CONNECTION
TRANSACTION_ID	= CURRENT_TRANSACTION
ISOLATION_LEVEL	SNAPSHOT, CONSISTENCY or 'READ COMMITTED'
ENGINE_VERSION	Firebird version number

SIMULAR pattern rules in functions

<pattern>	'character string' | <regular expr>
<regular expr>	<regular term> | <regular expr> <|> <regular term>
<regular term>	<regular factor> | <regular term> <regular factor>
<regular factor>	<primary> | <primary> * | <primary> + | <primary> ? | <primary> <repeat>
<repeat>	<{> <low> [,<high>] <}>
<low>	integer value
<high>	integer value
<primary>	'char' | '<escape>char' | % | <char set> | <(> <regular expr> <)>
<char set>	_ | <[><enum><]> | <[>^<enum><]>
<enum>	'char'[…] | 'char'-'char' | :<class>:
<class>	ALPHA | UPPER | LOWER | DIGIT | SPACE | WHITESPACE | ALNUM

CHARACTER_SET_NAME	COLLATION_NAME
ASCII	ASCII
BIG_5	BIG_5
CP943C	CP943C
CP943C	CP943C_UNICODE
CYRL	CYRL
CYRL	DB_RUS
CYRL	PDOX_CYRL
EUCJ_0208	EUCJ_0208
GBK	GBK
GBK	GBK_UNICODE
GB_2312	GB_2312
ISO8859_1	DA_DA
ISO8859_1	DE_DE
ISO8859_1	DU_NL
ISO8859_1	EN_UK
ISO8859_1	EN_US
ISO8859_1	ES_ES
ISO8859_1	ES_ES_CI_AI
ISO8859_1	FI_FI
ISO8859_1	FR_CA
ISO8859_1	FR_FR
ISO8859_1	FR_FR_CI_AI
ISO8859_1	ISO8859_1
ISO8859_1	IS_IS
ISO8859_1	IT_IT
ISO8859_1	NO_NO
ISO8859_1	PT_BR
ISO8859_1	PT_PT
ISO8859_1	SV_SV
ISO8859_13	ISO8859_13
ISO8859_13	LT_LT
ISO8859_2	CS_CZ
ISO8859_2	ISO8859_2
ISO8859_2	ISO_HUN
ISO8859_2	ISO_PLK
ISO8859_3	ISO8859_3
ISO8859_4	ISO8859_4
ISO8859_5	ISO8859_5
ISO8859_6	ISO8859_6
ISO8859_7	ISO8859_7
ISO8859_8	ISO8859_8
ISO8859_9	ISO8859_9
KOI8R	KOI8R
KOI8R	KOI8R_RU
KOI8U	KOI8U
KOI8U	KOI8U_UA
KSC_5601	KSC_5601
KSC_5601	KSC_DICTIONARY
NEXT	NEXT
NEXT	NXT_DEU
NEXT	NXT_ESP
NEXT	NXT_FRA
NEXT	NXT_ITA
NEXT	NXT_US
NONE	NONE
OCTETS	OCTETS
SJIS_0208	SJIS_0208
TIS620	TIS620
TIS620	TIS620_UNICODE
UNICODE_FSS	UNICODE_FSS
UTF8	UCS_BASIC
UTF8	UNICODE
UTF8	UNICODE_CI
UTF8	UNICODE_CI_AI
UTF8	UTF8
WIN1250	BS_BA
WIN1250	PXW_CSY
WIN1250	PXW_HUN
WIN1250	PXW_HUNDC
WIN1250	PXW_PLK
WIN1250	PXW_SLOV
WIN1250	WIN1250
WIN1250	WIN_CZ
WIN1250	WIN_CZ_CI_AI
WIN1251	PXW_CYRL
WIN1251	WIN1251
WIN1251	WIN1251_UA
WIN1252	PXW_INTL
WIN1252	PXW_INTL850
WIN1252	PXW_NORDAN4
WIN1252	PXW_SPAN
WIN1252	PXW_SWEDFIN
WIN1252	WIN1252
WIN1252	WIN_PTBR
WIN1253	PXW_GREEK
WIN1253	WIN1253
WIN1254	PXW_TURK
WIN1254	WIN1254
WIN1255	WIN1255
WIN1256	WIN1256
WIN1257	WIN1257
WIN1257	WIN1257_EE
WIN1257	WIN1257_LT
WIN1257	WIN1257_LV
WIN1258	WIN1258

NB: All old MS-DOS character sets and collations are removed from this table for brevity purposes.