APPENDIX B1

Informix SQL Quick Reference

Jonas Mureika and Edriss N. Merchant
University of Southern California Brain Project,
University of Southern California, Los Angeles, California

Introduction

This appendix provides a quick reference for the Informix SQL language. This appendix is intended to be a guide to various SQL commands referenced within this book. For more information on Informix™ products and the SQL language implemented as part of the Informix Dynamic Server, one can visit the Informix website at http://www.informix.com.

SQL Statements

CREATE DATABASE

Syntax: CREATE DATABASE db_name
[with (options)]

Description: The CREATE DATABASE statement creates a new database of the name specified by db_name. The name of the database must be unique on the installation and the name cannot be a keyword.

CREATE FUNCTION

Syntax: CREATE FUNCTION func_name[param_list]
returns type_name
[with (modifier [, modifier . . .])]
as {sql_statement | sql_statement_list} |
external name file_path_spec [(function_name)]
language {C | c} [[not] variant]

Description: The CREATE FUNCTION statement registers a new function in the database. The function body may be written in SQL or C.

CREATE TABLE

Syntax: CREATE TABLE table_name
[with (options)]
[of new type type_name]
({column_description | like table_name}
[{, column_description . . . |, like table name . . .}]
[table_constraints_list])
[under table_name [,table_name . . .]]
[archive]

Description: The CREATE TABLE statement creates a table in the database. The table_name of the table being created must be unique among all table, view, and index names within the user's current schema.

CREATE ROW TYPE

Syntax: CREATE ROW TYPE type_name
([internallength = {integer_constant | variable},]
[input = func_name,]
[output = func_name]
[, modifierb [, modifier . . .]])
[under type_name [, type_name . . .]]

Description: The CREATE ROW TYPE statement creates a new type in the database. The type_name is the name of the type being created. The type_name must be unique within the schema in which it was created.

CREATE INDEX

Syntax: CREATE INDEX index_name_ix ON table_
name ([row_name {ASC | DESC}, . . .]) USING
[index_method];

Description: The CREATE INDEX statement assigns an indexing marker to the data element (row_name) in the table table_name. Indices can be ascending (ASC) or descending (DESC), and only one of each type may be assigned to a given row. The argument index_type describes how the index is to be structured for search/retrieval purposes and may assume the values 'B_TREE,' 'R_TREE.'

DELETE

Syntax: DELETE from {table_name | only (table_name)}
[using suggestion_list]
[where search_condition]

Description: The DELETE statement removes rows from a table.

DROP

Syntax: DROP {database db_name | table table_name | type type_desc | func_name (param_list)}

Description: The DROP statement destroys the specified database and removes the specified table, type, and function from the database.

INSERT

Syntax: INSERT into table_name
[(col_name [, col_name . . .])]
[using suggestion_list]
{values (expression [, expression . . .]) | SELECT_stmt|

Description: The INSERT statement adds rows to a table.

RETURN

Syntax: RETURN expression

Description: The RETURN statement retrieves the results from a function or query. If RETURN is used on a function, it returns a scalar value if that is what the function returns; otherwise, it returns the name of a cursor containing all the returned values.

SELECT

Syntax: SELECT [unique | distinct] SELECT_list
from from_list
[using suggestion_list]
[where search_condition]
[group by expression, [, expression]]
[having search_condition]
[order by order_spec]
[union [all] SELECT_statement]

Description: The SELECT statement retrieves the data specified in the SELECT_list from the tables, views, or versions specified in the from_list.