Select Expressions

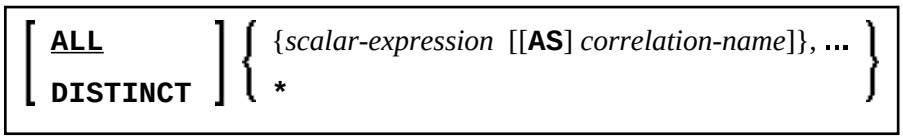
SELECT *selection table-expression*

A *select-expression* specifies a result table. It is used in the following Natural SQL statements: INSERT | SELECT | UPDATE

This chapter covers the following topics:

- Selection
- Table Expression

Selection



The *selection* specifies the items to be selected.

ALL/DISTINCT

Duplicate rows are not automatically eliminated from the result of a *select-expression*. To request this, specify the keyword DISTINCT.

The alternative to DISTINCT is ALL. ALL is assumed if neither is specified.

Scalar Expression

Instead of, or as well as, simple column names, a selection can also include general *scalar-expressions* containing scalar operators and scalar functions which provide computed values (see also the section *Scalar Expressions*).

Example:

```
SELECT NAME, 65 - AGE
  FROM SQL-PERSONNEL
  ...
```

Correlation Name

A *correlation-name* can be assigned to a *scalar-expression* as alias name for a result column.

The *correlation-name* need not be unique. If no *correlation-name* is specified for a result column, the corresponding *column-name* will be used (if the result column is derived from a column name; if not, the result table will have no name). The name of a result column may be used, for example, as column name in the ORDER BY clause of a SELECT statement.

Asterisk Notation - *

All columns of all tables specified in the FROM clause are selected.

Example:

```
SELECT *
  FROM SQL-PERSONNEL, SQL-AUTOMOBILES
  ...
```

Table Expression

from-clause [*where-clause*]

[*group-by-clause*] [*having-clause*]

[*order-by-clause*] [*fetch-first-clause*]

The *table-expression* specifies from where and according to what criteria rows are to be selected.

FROM Clause

FROM *table-reference,…*

This clause specifies from which tables the result set is built.

Table Reference

{ *table-name* [*correlation-clause*]
[**TABLE**] *subquery correlation-clause*
joined-table
TABLE *function-name* (*scalar-expression,...*)) *correlation-clause*
data-change-table-reference [*correlation-clause*]
xmltable-expression correlation-clause }

The tables specified in the FROM clause must contain the column fields used in the selection list.

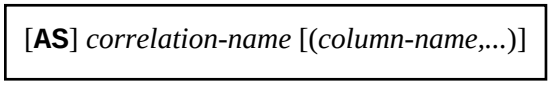
You can either specify a single table or produce an intermediate table resulting from a subquery or a "join" operation (see below).

Since various tables (that is, DDMs) can be addressed in one FROM clause and since a *table-expression* can contain several FROM clauses if *subqueries* are specified, the database ID (DBID) of the first DDM specified in the first FROM clause of the whole expression is used to identify the underlying database involved.

The TABLE *function-name* clause belongs to the SQL Extended Set and requires a *correlation-clause* with a *column-name* list.

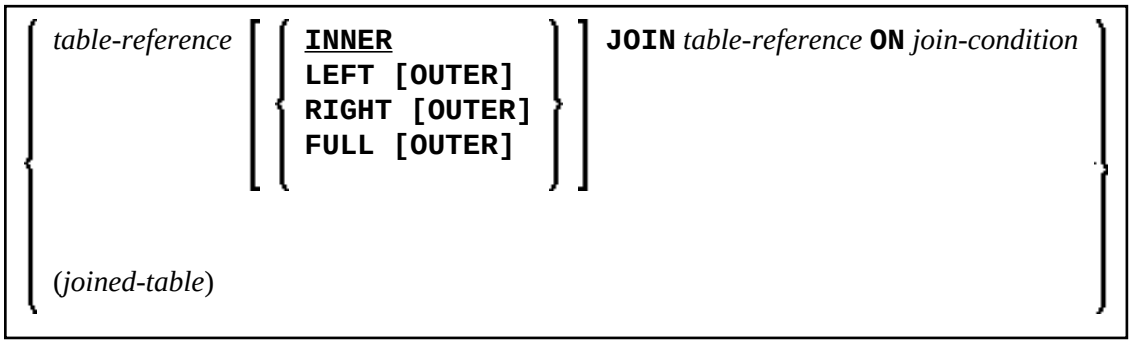
Optionally a *correlation-clause* can be assigned to a *table-name*. For a *subquery*, a *correlation-clause* must be assigned.

Correlation Clause



A *correlation-clause* consists of optional keyword AS and a *correlation-name* and is optionally followed by a plain *column-name* list. The *column-name* list belongs to the SQL Extended Set.

Joined Table



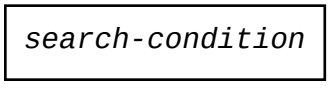
A *joined-table* specifies an intermediate table resulting from a "join" operation.

The "join" can be an INNER, LEFT OUTER, RIGHT OUTER or FULL OUTER JOIN. If you do not specify anything, INNER applies.

Multiple "join" operations can be nested; that is, the tables which create the intermediate result table can themselves be intermediate result tables of a "join" operation or a *subquery*; and the latter, in turn, can also have a *joined-table* or another *subquery* in its FROM clause.

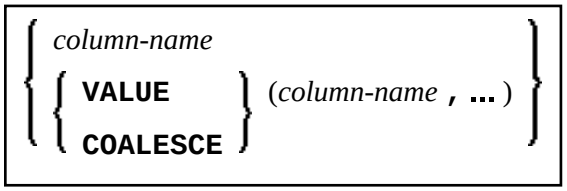
Join Condition

For INNER, LEFT OUTER, and RIGHT OUTER joins:



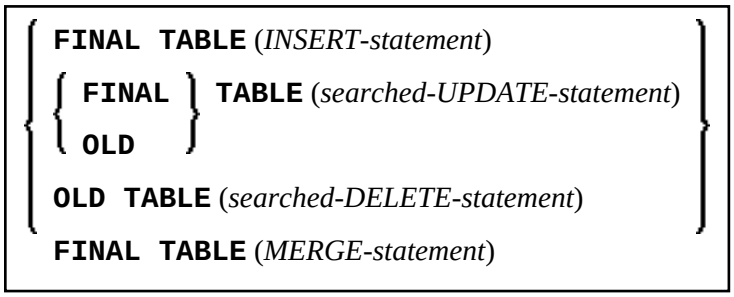
For FULL OUTER joins:

Full Join Expression



Within a *join-expression* only *column-names* and the *scalar-function* VALUE (or its synonym COALESCE) are allowed. See details on *column-name*.

data-change-table-reference



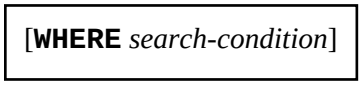
A data-change-table-reference specifies an intermediate result table, which is base on the rows that are changed by the SQL change statement specified in the clause. A *data-change-table-reference* can only be specified as the only table reference in the FROM clause.

FINAL TABLE	Specifies that the rows of the intermediate result table represent the set of rows that are changed by the SQL data change statement as they appear at the completion of the SQL data change statement.
OLD TABLE	Specifies that the rows of the intermediate result table represent the set of rows that are changed by the SQL data change statement as they exist prior to the application of the SQL data change statement.

XMLTABLE Function

The item *xmltable-function* specifies an invocation of the built-in XMLTABLE function.

WHERE Clause



The WHERE clause is used a to specify the selection criteria (*search-condition*) for the rows to be selected.

Example:

```
DEFINE DATA LOCAL
01 NAME   (A20)
01 AGE    (I2)
END-DEFINE
...
SELECT *
  INTO NAME, AGE
  FROM SQL-PERSONNEL
  WHERE AGE = 32
END-SELECT
...
```

See details on *search-condition*.

GROUP BY Clause

[**GROUP BY** *column-reference*,...]

The GROUP BY clause rearranges the table represented by the FROM clause into groups in a way that all rows within each group have the same value for the GROUP BY columns.

Each *column-reference* in the selection list must be either a GROUP BY column or specified within an *aggregate-function*. Aggregate functions are applied to the individual groups (not to the entire table). The result table contains as many rows as groups.

See further details on *column-reference* and *aggregate-function*.

Example:

```
DEFINE DATA LOCAL
1 #AGE     (I2)
1 #NUMBER  (I2)
END-DEFINE
...
SELECT AGE , COUNT(*)
  INTO #AGE, #NUMBER
  FROM SQL-PERSONNEL
  GROUP BY AGE
  ...
```

If the GROUP BY clause is preceded by a WHERE clause, all rows that do not satisfy the WHERE clause are excluded before any grouping is done.

HAVING Clause

[**HAVING** *search-condition*]

If the HAVING clause is specified, the GROUP BY clause should also be specified.

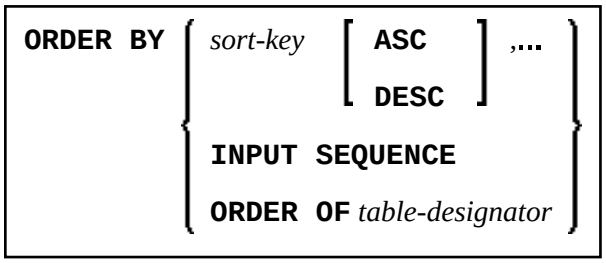
Just as the WHERE clause is used to exclude rows from a result table, the HAVING clause is used to exclude groups and therefore also based on a *search-condition*. Scalar expressions in a HAVING clause must be single-valued per group.

See further details on *scalar-expression* and *search-condition*.

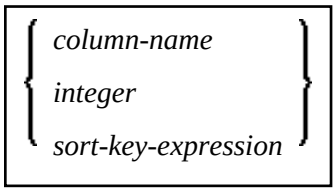
Example:

```
DEFINE DATA LOCAL
1 #NAME     (A20)
1 #AVGAGE   (I2)
1 #NUMBER   (I2)
END-DEFINE
...
SELECT NAME, AVG(AGE), COUNT(*)
  INTO #NAME, #AVGAGE, #NUMBER
  FROM SQL-PERSONNEL
  GROUP BY NAME
  HAVING COUNT(*) > 1
  ...
```

ORDER BY Clause



sort-key



FETCH FIRST Clause