|  |
| --- |
| EPAM Systems, RD Dep., RD Dep. |
| MTN.NIX.07 Oracle SQL Introduction  **Oracle SQL Basics** |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| REVISION HISTORY | | | | | |
| Ver. | Description of Change | Author | Date | Approved | |
| Name | Effective Date |
| 1.0 | Initial Version | Siarhei Kandrashevich | 15-Jan-2014 |  |  |
|  |  |  |  |  |  |

Contents

[1. HAVING clause 3](#_Toc380474707)

[2. Joins 3](#_Toc380474708)

[2.1. Join Conditions 3](#_Toc380474709)

[2.2. Self Joins 4](#_Toc380474710)

[2.3. Inner Joins 4](#_Toc380474711)

[2.4. Outer Joins 4](#_Toc380474712)

[2.5. Antijoins 6](#_Toc380474713)

[2.6. Semijoins 6](#_Toc380474714)

[3. External Sources 7](#_Toc380474715)

# HAVING clause

A HAVING clause restricts the results of a GROUP BY in a SelectExpression. The HAVING clause is applied to each group of the grouped table, much as a WHERE clause is applied to a select list. If there is no GROUP BY clause, the HAVING clause is applied to the entire result as a single group. The SELECT clause cannot refer directly to any column that does not have a GROUP BY clause. It can, however, refer to constants, aggregates, and special registers.

Syntax

HAVING searchCondition

The searchCondition, which is a specialized booleanExpression, can contain only grouping columns (see GROUP BY clause), columns that are part of aggregate expressions, and columns that are part of a subquery. For example, the following query is illegal, because the column SALARY is not a grouping column, it does not appear within an aggregate, and it is not within a subquery:

-- SELECT COUNT(\*)

-- FROM SAMP.STAFF

-- GROUP BY ID

-- HAVING SALARY > 15000

Aggregates in the HAVING clause do not need to appear in the SELECT list. If the HAVING clause contains a subquery, the subquery can refer to the outer query block if and only if it refers to a grouping column.

Example

-- Find the total number of economy seats taken on a flight,

-- grouped by airline,

-- only when the group has at least 2 records.

SELECT SUM(ECONOMY\_SEATS\_TAKEN), AIRLINE\_FULL

FROM FLIGHTAVAILABILITY, AIRLINES

WHERE SUBSTR(FLIGHTAVAILABILITY.FLIGHT\_ID, 1, 2) = AIRLINE

GROUP BY AIRLINE\_FULL

HAVING COUNT(\*) > 1

# Joins

A join is a query that combines rows from two or more tables, views, or materialized views. Oracle Database performs a join whenever multiple tables appear in the FROM clause of the query. The select list of the query can select any columns from any of these tables. If any two of these tables have a column name in common, then you must qualify all references to these columns throughout the query with table names to avoid ambiguity.

## Join Conditions

Most join queries contain at least one join condition, either in the FROM clause or in the WHERE clause. The join condition compares two columns, each from a different table. To execute a join, Oracle Database combines pairs of rows, each containing one row from each table, for which the join condition evaluates to TRUE. The columns in the join conditions need not also appear in the select list.

To execute a join of three or more tables, Oracle first joins two of the tables based on the join conditions comparing their columns and then joins the result to another table based on join conditions containing columns of the joined tables and the new table. Oracle continues this process until all tables are joined into the result. The optimizer determines the order in which Oracle joins tables based on the join conditions, indexes on the tables, and, any available statistics for the tables.

A WHERE clause that contains a join condition can also contain other conditions that refer to columns of only one table. These conditions can further restrict the rows returned by the join query.

Note:

You cannot specify LOB columns in the WHERE clause if the WHERE clause contains the join condition. The use of LOBs in WHERE clauses is also subject to other restrictions. See Oracle Database SecureFiles and Large Objects Developer's Guide for more information.

Equijoins

An equijoin is a join with a join condition containing an equality operator. An equijoin combines rows that have equivalent values for the specified columns. Depending on the internal algorithm the optimizer chooses to execute the join, the total size of the columns in the equijoin condition in a single table may be limited to the size of a data block minus some overhead. The size of a data block is specified by the initialization parameter DB\_BLOCK\_SIZE.

## Self Joins

A self join is a join of a table to itself. This table appears twice in the FROM clause and is followed by table aliases that qualify column names in the join condition. To perform a self join, Oracle Database combines and returns rows of the table that satisfy the join condition.

If two tables in a join query have no join condition, then Oracle Database returns their Cartesian product. Oracle combines each row of one table with each row of the other. A Cartesian product always generates many rows and is rarely useful. For example, the Cartesian product of two tables, each with 100 rows, has 10,000 rows. Always include a join condition unless you specifically need a Cartesian product. If a query joins three or more tables and you do not specify a join condition for a specific pair, then the optimizer may choose a join order that avoids producing an intermediate Cartesian product.

## Inner Joins

An inner join (sometimes called a simple join) is a join of two or more tables that returns only those rows that satisfy the join condition.

## Outer Joins

An outer join extends the result of a simple join. An outer join returns all rows that satisfy the join condition and also returns some or all of those rows from one table for which no rows from the other satisfy the join condition.

To write a query that performs an outer join of tables A and B and returns all rows from A (a left outer join), use the LEFT [OUTER] JOIN syntax in the FROM clause, or apply the outer join operator (+) to all columns of B in the join condition in the WHERE clause. For all rows in A that have no matching rows in B, Oracle Database returns null for any select list expressions containing columns of B.

To write a query that performs an outer join of tables A and B and returns all rows from B (a right outer join), use the RIGHT [OUTER] JOIN syntax in the FROM clause, or apply the outer join operator (+) to all columns of A in the join condition in the WHERE clause. For all rows in B that have no matching rows in A, Oracle returns null for any select list expressions containing columns of A.

To write a query that performs an outer join and returns all rows from A and B, extended with nulls if they do not satisfy the join condition (a full outer join), use the FULL [OUTER] JOIN syntax in the FROM clause.

You cannot compare a column with a subquery in the WHERE clause of any outer join, regardless which form you specify.

You can use outer joins to fill gaps in sparse data. Such a join is called a partitioned outer join and is formed using the query\_partition\_clause of the join\_clause syntax. Sparse data is data that does not have rows for all possible values of a dimension such as time or department. For example, tables of sales data typically do not have rows for products that had no sales on a given date. Filling data gaps is useful in situations where data sparsity complicates analytic computation or where some data might be missed if the sparse data is queried directly.

Oracle Database Data Warehousing Guide for a complete discussion of group outer joins and filling gaps in sparse data

Oracle recommends that you use the FROM clause OUTER JOIN syntax rather than the Oracle join operator. Outer join queries that use the Oracle join operator (+) are subject to the following rules and restrictions, which do not apply to the FROM clause OUTER JOIN syntax:

You cannot specify the (+) operator in a query block that also contains FROM clause join syntax.

The (+) operator can appear only in the WHERE clause or, in the context of left-correlation (when specifying the TABLE clause) in the FROM clause, and can be applied only to a column of a table or view.

If A and B are joined by multiple join conditions, then you must use the (+) operator in all of these conditions. If you do not, then Oracle Database will return only the rows resulting from a simple join, but without a warning or error to advise you that you do not have the results of an outer join.

The (+) operator does not produce an outer join if you specify one table in the outer query and the other table in an inner query.

You cannot use the (+) operator to outer-join a table to itself, although self joins are valid. For example, the following statement is not valid:

-- The following statement is not valid:

SELECT employee\_id, manager\_id

FROM employees

WHERE employees.manager\_id(+) = employees.employee\_id;

However, the following self join is valid:

SELECT e1.employee\_id, e1.manager\_id, e2.employee\_id

FROM employees e1, employees e2

WHERE e1.manager\_id(+) = e2.employee\_id

ORDER BY e1.employee\_id, e1.manager\_id, e2.employee\_id;

The (+) operator can be applied only to a column, not to an arbitrary expression. However, an arbitrary expression can contain one or more columns marked with the (+) operator.

A WHERE condition containing the (+) operator cannot be combined with another condition using the OR logical operator.

A WHERE condition cannot use the IN comparison condition to compare a column marked with the (+) operator with an expression.

If the WHERE clause contains a condition that compares a column from table B with a constant, then the (+) operator must be applied to the column so that Oracle returns the rows from table A for which it has generated nulls for this column. Otherwise Oracle returns only the results of a simple join.

In previous releases of Oracle Database, in a query that performed outer joins of more than two pairs of tables, a single table could be the null-generated table for only one other table. Beginning with Oracle Database 12c, a single table can be the null-generated table for multiple tables. For example, the following statement is allowed in Oracle Database 12c:

SELECT \* FROM A, B, D

WHERE A.c1 = B.c2(+) and D.c3 = B.c4(+);

In this example, B, the null-generated table, is outer-joined to two tables, A and D. Refer to SELECT for the syntax for an outer join.

## Antijoins

An antijoin returns rows from the left side of the predicate for which there are no corresponding rows on the right side of the predicate. It returns rows that fail to match (NOT IN) the subquery on the right side.

## Semijoins

A semijoin returns rows that match an EXISTS subquery without duplicating rows from the left side of the predicate when multiple rows on the right side satisfy the criteria of the subquery.

Semijoin and antijoin transformation cannot be done if the subquery is on an OR branch of the WHERE clause.

# External Sources

1. Oracle Database SQL Language Reference 11g Release 2 (11.2) E26088-02
2. Oracle Database PL/SQL Packages and Types Reference 11g Release 2 (11.2) B28419-03
3. ISO/IEC 9075-1 Part 1 Framework (SQL/Framework)
4. ISO/IEC 9075-2 Part 2 Foundation (SQL/Foundation)
5. ISO/IEC 9075-11 Part 11 Information and Definition Schemas (SQL/Schemata)
6. **Oracle Database SQL Language Reference 12g Release12 (12.1) E17209-15**