

DBMS_FREQUENT_ITEMSET

The DBMS_FREQUENT_ITEMSET package enables frequent itemset counting. The two functions are identical except in the input cursor format difference.

This chapter contains the following topics:

- [Summary of DBMS_FREQUENT_ITEMSET Subprograms](#)

Summary of DBMS_FREQUENT_ITEMSET Subprograms

The DBMS_FREQUENT_ITEMSET package includes the FI_HORIZONTAL function and FI_TRANSACTIONAL function subprograms.

Table 91-1 DBMS_FREQUENT_ITEMSET Package Subprograms

Subprogram	Description
FI_HORIZONTAL Function	Counts all frequent itemsets given a cursor for input data which is in 'HORIZONTAL' row format, support threshold, minimum itemset length, maximum itemset length, items to be included, items to be excluded
FI_TRANSACTIONAL Function	Counts all frequent itemsets given a cursor for input data which is in 'TRANSACTIONAL' row format, support threshold, minimum itemset length, maximum itemset length, items to be included, items to be excluded

FI_HORIZONTAL Function

The purpose of this table function is to count all frequent itemsets given a cursor for input data which is in 'HORIZONTAL' row format, support threshold, minimum itemset length, maximum itemset length, items to be included, items to be excluded. The result will be a table of rows in form of itemset, support, length, total transactions counted.

In 'HORIZONTAL' row format, each row contains all of the item ids for a single transaction. Since all of the items come together, no transaction id is necessary.

The benefit of this table function is that if an application already has data in horizontal format, the database can skip the step of transforming rows that are in transactional format into horizontal format.

Syntax

```
DBMS_FREQUENT_ITEMSET.FI_HORIZONTAL (
    tranx_cursor      IN      SYSREFCURSOR,
    support_threshold IN      NUMBER,
    itemset_length_min IN      NUMBER,
    itemset_length_max IN      NUMBER,
    including_items    IN      SYS_REFCURSOR DEFAULT NULL,
    excluding_items    IN      SYS_REFCURSOR DEFAULT NULL)
RETURN TABLE OF ROW (
    itemset [Nested Table of Item Type DERIVED FROM tranx_cursor],
```

```

support      NUMBER,
length       NUMBER,
total_tranx  NUMBER);

```

Parameters

Table 91-2 FI_HORIZONTAL Function Parameters

Parameter	Description
tranx_cursor	The cursor parameter that the user will supply when calling the function. There is no limits on the number of returning columns. Each column of cursor represents an item. All columns of the cursor must be of the same datatype. The item id must be number or character type (for example, VARCHAR2(n)).
support_threshold	A fraction number of total transaction count. An itemset is termed "frequent" if [the number of transactions it occurs in] divided by [the total number of transactions] exceed the fraction. The parameter must be a NUMBER.
itemset_length_min	The minimum length for interested frequent itemset. The parameter must be a NUMBER between 1 and 20, inclusive.
itemset_length_max	The maximum length for interested frequent itemset. This parameter must be a NUMBER between 1 and 20, inclusive, and must not be less than itemset_length_min.
including_items	A cursor from which a list of items can be fetched. At least one item from the list must appear in frequent itemsets that are returned. The default is NULL.
excluding_items	A cursor from which a list of items can be fetched. No item from the list can appear in frequent itemsets that are returned. The default is NULL.

Return Values

Table 91-3 FI_HORIZONTAL Return Values

Parameter	Description
support	The number of transactions in which a frequent itemset occurs. This will be returned as a NUMBER.
itemset	A collection of items which is computed as frequent itemset. This will be returned as a nested table of item type which is the item column type of the input cursor.
length	Number of items in a frequent itemset. This will be returned as a NUMBER.
total_tranx	The total transaction count. This will be returned as a NUMBER.

Example

Suppose you have a table horiz_table_in.

```

horiz_table_in(iid1 VARCHAR2(30), iid2 VARCHAR2(30), iid3 VARCHAR2(30), iid4
VARCHAR2(30), iid5 VARCHAR2(30));

```

and the data in horiz_table_in looks as follows:

```
('apple', 'banana', NULL, NULL, NULL)
('apple', 'milk', 'banana', NULL, NULL)
('orange', NULL, NULL, NULL, NULL)
```

Suppose you want to find out what combinations of items is frequent with a given support threshold of 30%, requiring itemset containing at least one of ('apple','banana','orange'), but excluding any of ('milk') in any itemset. You use the following query:

```
CREATE TYPE fi_varchar_nt AS TABLE OF VARCHAR2(30);
SELECT CAST(itemset as FI_VARCHAR_NT) itemset, support, length, total_tranx
FROM table(DBMS_FREQUENT_ITEMSET.FI_HORIZONTAL(
    CURSOR(SELECT iid1, iid2, iid3, iid4, iid5
        FROM horiz_table_in),
    0.3,
    2,
    5,
    CURSOR(SELECT * FROM table(FI_VARCHAR_NT
        ('apple','banana','orange'))),
    CURSOR(SELECT * FROM table(FI_VARCHAR_NT('milk'))));
```

FI_TRANSACTIONAL Function

This procedure counts all frequent itemsets given a cursor for input data which is in 'TRANSACTIONAL' row format, support threshold, minimum itemset length, maximum itemset length, items to be included, items to be excluded. The result will be a table of rows in form of itemset, support, length, total number of transactions.

In 'TRANSACTIONAL' row format, each transaction is spread across multiple rows. All the rows of a given transaction have the same transaction id, and each row has a different item id. Combining all of the item ids which share a given transaction id results in a single transaction.

Syntax

```
DBMS_FREQUENT_ITEMSET.FI_TRANSACTIONAL (
    tranx_cursor      IN      SYSREFCURSOR,
    support_threshold IN      NUMBER,
    itemset_length_min IN      NUMBER,
    itemset_length_max IN      NUMBER,
    including_items    IN      SYS_REFCURSOR DEFAULT NULL,
    excluding_items    IN      SYS_REFCURSOR DEFAULT NULL)
RETURN TABLE OF ROW (
    itemset [Nested Table of Item Type DERIVED FROM tranx_cursor],
    support      NUMBER,
    length       NUMBER,
    total_tranx  NUMBER);
```

Parameters

Table 91-4 *FI_TRANSACTIONAL Function Parameters*

Parameter	Description
tranx_cursor	The cursor parameter that the user will supply when calling the function. It should return two columns in its returning row, the first column being the transaction id, the second column being the item id. The item id must be number or character type (for example, VARCHAR2(n)).

Table 91-4 (Cont.) FI_TRANSACTIONAL Function Parameters

Parameter	Description
support_threshold	A fraction number of total transaction count. An itemset is termed "frequent" if [the number of transactions it occurs in] divided by [the total number of transactions] exceed the fraction. The parameter must be a NUMBER.
itemset_length_min	The minimum length for interested frequent itemset. The parameter must be a NUMBER between 1 and 20, inclusive.
itemset_length_max	The maximum length for interested frequent itemset. This parameter must be a NUMBER between 1 and 20, inclusive, and must not be less than itemset_length_min.
including_items	A cursor from which a list of items can be fetched. At least one item from the list must appear in frequent itemsets that will be returned. The default is NULL.
excluding_items	A cursor from which a list of items can be fetched. No item from the list can appear in frequent itemsets that will returned. The default is NULL.

Return Values

Table 91-5 FI_TRANSACTIONAL Return Values

Parameter	Description
support	The number of transactions in which a frequent itemset occurs. This will be returned as a NUMBER.
itemset	A collection of items which is computed as frequent itemset. This will be returned as a nested table of item type which is the item column type of the input cursor.
length	Number of items in a frequent itemset. This will be returned as a NUMBER.
total_tranx	The total transaction count. This will be returned as a NUMBER, and will be the same for all returned rows, similar to a reporting aggregate.

Usage Notes

Applications must predefine a nested table type of the input item type and cast the output itemset into this predefined nested table type before further processing, such as loading into a table.

Examples

Suppose that the input table tranx_table_in looks as follows:

```
(1, 'apple')
(1, 'banana')
(2, 'apple')
(2, 'milk')
(2, 'banana')
(3, 'orange')
```

and the user is trying to find itemsets that satisfy a support-threshold of 60% and have the itemset-length greater than 1 (namely, (apple, banana)).

The output of this function would contain the following output row:

```
itemset=('apple','banana'), support=2, length=2, total_tranx=3
```

You need to create a nested table of item type before you submit a query to perform the frequent itemset counting. In this example, since item is of VARCHAR2(30), you must create a nested table of VARCHAR2(30):

```
CREATE TYPE fi_varchar_nt AS TABLE OF VARCHAR2(30);
SELECT CAST(itemset as FI_VARCHAR_NT) itemset, support, length, total_tranx
FROM table(DBMS_FREQUENT_ITEMSET.FI_TRANSACTIONAL(
    cursor(SELECT tid, iid FROM tranx_table_in),
    0.6,
    2,
    5,
    NULL,
    NULL));
```

Here is another example to illustrate how to include certain items and exclude certain items in the counting.

```
SELECT CAST(itemset as FI_VARCHAR_NT) itemset, support, length, total_tranx
FROM table(DBMS_FREQUENT_ITEMSET.FI_TRANSACTIONAL(
    CURSOR(SELECT tid, iid FROM tranx_table_in),
    0.6,
    2,
    5,
    CURSOR(SELECT * FROM table(FI_VARCHAR_NT
        ('apple','banana','orange'))),
    CURSOR(SELECT * FROM table(FI_VARCHAR_NT('milk')))));
```

Using the including/excluding items parameter, you are able to further optimize the execution by ignoring itemsets that are not expected by application.

You can also use transactional output through collection unnesting:

```
SELECT
    bt.setid, nt.*
FROM
    (SELECT cast(Itemset as FI_VARCHAR_NT) itemset, rownum setid
    FROM table(
        DBMS_FREQUENT_ITEMSET.FI_TRANSACTIONAL(
            CURSOR(SELECT tid, iid FROM tranx_table_in), 0.6, 2, 5,
            NULL, NULL))) bt,
    table(bt.itemset) nt;
```

If you want to use an insert statement to load frequent itemsets into a nested table, it is better to use the NESTED_TABLE_FAST_INSERT hint for performance:

```
CREATE TABLE fq_nt (coll FI_VARCHAR_NT) NESTED TABLE coll STORE AS
    coll_nest;
INSERT /*+ NESTED_TABLE_FAST_INSERT */ INTO fq_nt
SELECT cast(itemset as FI_VARCHAR_NT)
FROM table(DBMS_FREQUENT_ITEMSET.FI_TRANSACTIONAL(
    cursor(SELECT tid, iid FROM tranx_table_in), 0.6, 2, 5,
    NULL, NULL));
```

Note that if you want to use the package inside a PL/SQL cursor, you must cast the return type of the table function:

```

CREATE TYPE fi_res AS OBJECT (
    itemset      FI_VARCHAR_NT,
    support      NUMBER,
    length       NUMBER,
    total_tranx  NUMBER
);
/
CREATE TYPE fi_coll AS TABLE OF fi_res;
/

DECLARE
    cursor freqC is
        SELECT Itemset
        FROM table(
            CAST(DBMS_FREQUENT_ITEMSET.FI_TRANSACTIONAL(
                cursor(SELECT tid, iid FROM tranx_table_in), 0.6, 2, 5,
                NULL, NULL) AS fi_coll));
    coll_nt  FI_VARCHAR_NT;
    num_rows int;
    num_itms int;
BEGIN
    num_rows := 0;
    num_itms := 0;
    OPEN freqC;
    LOOP
        FETCH freqC INTO coll_nt;
        EXIT WHEN freqC%NOTFOUND;
        num_rows := num_rows + 1;
        num_itms := num_itms + coll_nt.count;
    END LOOP;
    CLOSE freqC;
    DBMS_OUTPUT.PUT_LINE('Totally ' || num_rows || ' rows ' || num_itms || '
items were produced.');
```

```

END;
/
```