Задание 1

AVERAGE Command

Computes the arithmetic average of numeric expressions or fields.

```
AVERAGE [ExpressionList]   [Scope] [FOR lExpression1] [WHILElExpression2]
   [TO VarList | TO ARRAY ArrayName]   [NOOPTIMIZE]
```

Parameters

ExpressionList

Specifies the expressions to average. ExpressionList can be a list of fields from the table separated by commas, or numeric expressions involving fields from the table.

Scope

Specifies the record or range of records to include in the average. Only the records that fall within the range of records specified by the scope are averaged. The scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, and REST. The default scope for AVERAGE is ALL records.

Commands that include Scope operate only on the table in the active work area.

FOR lExpression1

Specifies a condition whereby only the records that satisfy the logical condition lExpression are included. This argument allows you to filter out undesired records.

Rushmore Query Optimization optimizes an AVERAGE FOR query if lExpression is an optimizable expression. For best performance, use an optimizable expression in the FOR clause. For information on Rushmore optimizable expressions, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

WHILE lExpression2

Specifies that as long as the logical expression lExpression2 evaluates to true (.T.), records are included in the average.

TO VarList

Specifies the list of variables or array elements to which the results of the average are stored.

TO ARRAY ArrayName

Specifies the one-dimensional array to which the results of the average are stored. The one-dimensional array can be created before the execution of AVERAGE.

If the array you include in AVERAGE doesn't exist, Visual FoxPro automatically creates it. If the array exists and isn't large enough to contain all the results, Visual FoxPro increases the size of the array automatically to accommodate the information.

NOOPTIMIZE

Задание 2

DELETE Command

Marks records for deletion.

```
DELETE [Scope] [FOR lExpression1] [WHILE lExpression2]
   [IN nWorkArea | cTableAlias] [NOOPTIMIZE]
```

Parameters

Scope

Specifies a range of records to mark for deletion. The default scope for DELETE is the current record (NEXT 1). The scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, and REST.

For more information on scope clauses, see Scope Clauses.

FOR lExpression1

Specifies a condition whereby only the records that satisfy the logical condition lExpression1 are marked for deletion.

Rushmore Query Optimization optimizes a query created with DELETE ... FOR if lExpression1 is an optimizable expression and the table is indexed on DELETED( ). For best performance, use an optimizable expression in the FOR clause.

For information on Rushmore optimizable expressions, see SET OPTIMIZE Command, and Using Rushmore Query Optimization to Speed Data Access in Optimizing Applications.

WHILE lExpression2

Specifies a condition whereby records are marked for deletion for as long as lExpression2 evaluates to true (.T.).

IN nWorkArea

Specifies the work area of the table in which records are marked for deletion.

IN cTableAlias

Specifies the alias of the table in which records are marked for deletion.

If you omit nWorkArea and cTableAlias, records are marked for deletion in the table in the currently selected work area.

NOOPTIMIZE

Задание 3

RECALL Command

Unmarks records marked for deletion in the selected table.

```
RECALL [Scope] [FOR lExpression1] [WHILE lExpression2] [NOOPTIMIZE]
   [IN nWorkArea | cTableAlias]
```

Parameters

Scope

Specifies a range of records to recall. The default scope for RECALL is the current record (NEXT 1).

Only the records that fall within the range specified are recalled. The scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, and REST.

For more information on scope clauses, see Scope Clauses.

FOR lExpression1

Specifies that only the records for which lExpression1 evaluates to true (.T.) are recalled. This option allows you to filter out undesired records.

Rushmore Query Optimization optimizes a RECALL FOR if lExpression1 is an optimizable expression. For best performance, use an optimizable expression in the FOR clause.

For more information, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

WHILE lExpression2

Specifies a condition whereby records are recalled for as long as lExpression2 evaluates to true (.T.).

NOOPTIMIZE

Prevents Rushmore optimization of RECALL

For more information, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

IN nWorkArea | cTableAlias

Specifies the workarea or table alias affected by the RECALL command. Use this clause to specify a workarea or a table outside the current work area.

Задание 4

COUNT Command

Counts table records.

```
COUNT   [Scope] [FOR lExpression1] [WHILE lExpression2]   [TO VarName]
   [NOOPTIMIZE]
```

Parameters

Scope

Specifies a range of records to be included in the count. The default scope for COUNT is ALL records.

The scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, and REST. Commands that include Scope operate only on the table in the active work area.

For more information on scope clauses, see Scope Clauses.

FOR lExpression1

Specifies that only the records that satisfy the logical condition lExpression1 are counted. Including FOR lets you conditionally count records, filtering out undesired records.

Rushmore Query Optimization will optimize a COUNT FOR query if lExpression1 is an optimizable expression. For best performance, use an optimizable expression in the FOR clause.

For more information on optimizable expressions, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

WHILE lExpression2

Specifies a condition whereby records are counted for as long as the logical expression lExpression2 evaluates to True (.T.).

TO VarName

Specifies the variable or array to which the record count is stored. If the variable you specify doesn't exist, Visual FoxPro creates it.

NOOPTIMIZE

Задание 5

LOCATE Command

Sequentially searches the table for the first record that matches the specified logical expression.

```
LOCATE [FOR lExpression1]   [Scope]   [WHILE lExpression2]   [NOOPTIMIZE]
```

Parameters

FOR lExpression1

Sequentially searches the current table for the first record that matches the logical expression lExpression1.

Rushmore Query Optimization optimizes a query created with LOCATE FOR if lExpression1 is an optimizable expression. For best performance, use an optimizable expression in the FOR clause.

For more information, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

Scope

Specifies a range of records to locate. Only the records that fall within the range are located. The scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, and REST. Commands that include Scope operate only on the table in the active work area.

The default scope for LOCATE is ALL records.

WHILE lExpression2

Specifies a condition whereby records are searched for as long as the logical expression lExpression2 evaluates to true (.T.).

NOOPTIMIZE

Disables Rushmore Query Optimization of LOCATE.

Задание 6

REPLACE Command (Visual FoxPro)

Updates table records.

```
REPLACE FieldName1 WITH eExpression1 [ADDITIVE]
   [, FieldName2 WITH eExpression2 [ADDITIVE]] ... [Scope]
   [FOR lExpression1] [WHILE lExpression2] [IN nWorkArea | cTableAlias]
   [NOOPTIMIZE]
```

Parameters

FieldName1 WITH eExpression1[, FieldName2 WITH eExpression2...]

Specifies that the data in FieldName1 be replaced with the value of the expression eExpression1; that the data in FieldName2 be replaced with the value of the expression eExpression2; and so on.

When the expression value is longer than the width of a numeric field, REPLACE forces the value to fit by carrying out the following steps:

- First, REPLACE truncates decimal places and rounds the remaining decimal portion of the field.
- If the value still doesn't fit, REPLACE stores the field contents using scientific notation.
- If the value still doesn't fit, REPLACE replaces the field contents with asterisks.

ADDITIVE

Appends to the end of the memo fields replacements to memo fields. ADDITIVE applies to replacements in memo fields only. If you omit ADDITIVE, the memo field is overwritten with the value of the expression.

Scope

Specifies a range of records to replace. The default scope for REPLACE is the current record (NEXT 1).

Only the records that fall within the range are replaced. The scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, and REST. For more information on scope clauses, see Scope Clauses.

FOR lExpression1

Specifies that the designated fields be replaced only in records for which lExpression1 evaluates to true (.T.). Including FOR makes it possible for you to conditionally replace records, filtering out those you don't want replaced.

Rushmore Query Optimization optimizes REPLACE FOR if lExpression1 is an optimizable expression. For best performance, use an optimizable expression in the FOR clause.

For more information, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

WHILE lExpression2

Specifies a condition whereby fields are replaced for as long as the logical expression lExpression2 evaluates to true (.T.).

IN nWorkArea

Specifies the work area of the table in which records are updated.

IN cTableAlias

Specifies the alias of the table in which records are updated.

If you omit nWorkArea and cTableAlias, records are updated in the table in the currently selected work area.

NOOPTIMIZE

Prevents Rushmore optimization.

For more information, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

Задание 7.

SORT Command

Sorts records in the currently selected table and outputs the sorted records to a new table.

```
SORT TO TableName ON FieldName1 [/A | /D] [/C]
   [, FieldName2 [/A | /D] [/C] ...]   [ASCENDING | DESCENDING]
   [Scope] [FOR lExpression1] [WHILE lExpression2]
   [FIELDS FieldNameList   | FIELDS LIKE Skeleton
   | FIELDS EXCEPT Skeleton]   [NOOPTIMIZE]
```

Parameters

TableName

Specifies the name of the new table containing the sorted records. Visual FoxPro assumes a .dbf file name extension for tables. A .dbf extension is automatically assigned if the file name you include doesn't have an extension.

ON FieldName1

Specifies the field in the currently selected table on which the sort is based. The contents and data type of the field determine the order of the records in the new table. By default, the sort is done in ascending order. You can't sort on memo or general fields.

The following example sorts a table on the `cust_id` field. The `customer` table is opened and sorted, creating a new table named `temp`. The records in `temp` are ordered by the `cust_id` field.

```
CLOSE DATABASES
OPEN DATABASE (HOME(2) + 'data\testdata')
USE customer  && Opens Customer table
CLEAR
LIST FIELDS company, cust_id NEXT 3
SORT TO temp ON cust_id
USE temp
LIST FIELDS company, cust_id NEXT 3
WAIT WINDOW 'Now sorted on CUST_ID' NOWAIT
```

You can include additional field names (FieldName2, FieldName3) to further order the new table. The first field FieldName1 is the primary sort field, the second field FieldName2 is the secondary sort field, and so on.

[/A | /D] [/C]

For each field you include in the sort, you can specify an ascending or descending sort order. /A specifies an ascending order for the field. /D specifies a descending order. /A or /D can be included with any type of field.

By default, the field sort order for character fields is case sensitive. If you include the /C option after the name of a character field, case is ignored. You can combine the /C option with the /A or /D option. For example, /AC or /DC.

In the following example, a new table named `clients` is created. The `orders` table is sorted on the `order_date` field in ascending order and the `freight` field in descending order.

```
USE orders
SORT TO clients ON order_date/A,freight/D
```

ASCENDING

Specifies an ascending order for all fields not followed by /D.

DESCENDING

Specifies a descending order for all fields not followed by /A.

If you omit either ASCENDING or DESCENDING, the sort order is ascending by default.

Scope

Specifies a range of records to sort. The scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, and REST.

The default scope for SORT is ALL records.

FOR lExpression1

Specifies that only the records in the current table for which the logical condition lExpression1 evaluates to true (.T.) are included in the sort. Including FOR lets you conditionally sort records, filtering out undesired records.

Rushmore Query Optimization optimizes a SORT ... FOR command if lExpression1 is an optimizable expression. For best performance, use an optimizable expression in the FOR clause.

A discussion of expressions that Rushmore can optimize appears in Optimizing Applications.

WHILE lExpression2

Specifies a condition whereby records from the current table are included in the sort for as long as the logical expression lExpression2 evaluates to true (.T.).

FIELDS FieldNameList

Specifies fields from the original table to include in the new table that SORT creates. If you omit the FIELDS clause, all fields from the original table are included in the new table.

FIELDS LIKE Skeleton

Specifies that fields from the original table that match the field skeleton Skeleton are included in the new table that SORT creates.

FIELDS EXCEPT Skeleton

Specifies that all fields except those that match the field skeleton Skeleton are included in the new table that SORT creates.

The field skeleton Skeleton supports wildcards. For example, to specify that all fields that begin with the letters A and P are included in the new table, use the following:

```
SORT TO mytable ON myfield FIELDS LIKE A*,P*
```

The LIKE clause can be combined with the EXCEPT clause:

```
SORT TO mytable ON myfield FIELDS LIKE A*,P* EXCEPT PARTNO*
```

NOOPTIMIZE

Disables Rushmore optimization of SORT.

For more information, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

Задание 8.

SUM Command

Totals all or specified numeric fields in the currently selected table.

```
SUM [eExpressionList]   [Scope] [FOR lExpression1] [WHILE lExpression2]
   [TO MemVarNameList | TO ARRAY ArrayName]   [NOOPTIMIZE]
```

Parameters

eExpressionList

Specifies one or more fields or field expressions to total. If you omit the field expression list, all numeric fields are totaled.

Scope

Specifies a range of records to include in the total. The scope clauses are: ALL, NEXT nRecords, RECORD nRecordNumber, and REST. For more information on scope clauses, see Scope Clauses.

The default scope for SUM is ALL records.

FOR lExpression1

Specifies that only the records for which the logical condition lExpression1 evaluates to true (.T.) are included in the total. Including FOR makes it possible for you to conditionally total records, filtering out undesired records.

Rushmore Query Optimization optimizes a SUM ... FOR command if lExpression1 is an optimizable expression. For best performance, use an optimizable expression in the FOR clause.

For more information on Rushmore optimization, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

WHILE lExpression2

Specifies a condition whereby records from the current table are included in the total for as long as the logical expression lExpression2 evaluates to true (.T.).

TO MemVarNameList

Stores each total to a variable. If you specify a variable in MemVarNameList that doesn't exist, Visual FoxPro automatically creates it. Separate the variable names in the list with commas.

TO ARRAY ArrayName

Stores totals to a variable array. If the array you specify in SUM doesn't exist, Visual FoxPro automatically creates it. If the array exists and is too small to contain all the totals, the size of the array is automatically increased to accommodate the totals.

NOOPTIMIZE

Disables Rushmore optimization of SUM.

For more information, see SET OPTIMIZE Command and Using Rushmore Query Optimization to Speed Data Access.

Задание 9.

DO WHILE ... ENDDO Command

Executes a set of commands within a conditional loop.

```
DO WHILE lExpression
      Commands
   [LOOP]
   [EXIT]
ENDDO
```

Parameters

lExpression

Specifies a logical expression whose value determines whether the commands between DO WHILE and ENDDO are executed. As long as lExpression evaluates to true (.T.), the set of commands are executed.

Commands

Specifies the set of Visual FoxPro commands to be executed as long as lExpression evaluates to true (.T.).

LOOP

Returns program control directly back to DO WHILE. LOOP can be placed anywhere between DO WHILE and ENDDO.

EXIT

Transfers program control from within the DO WHILE loop to the first command following ENDDO. EXIT can be placed anywhere between DO WHILE and ENDDO.

Задание 10.

FOR ... ENDFOR Command

Executes a set of commands a specified number of times.

```
FOR VarName = nInitialValue TO nFinalValue [STEP nIncrement]
      Commands
   [EXIT]
   [LOOP]
ENDFOR | NEXT
```

Parameters

VarName

Specifies the name of the variable to act as the counter. The counter keeps track of the number of times the Visual FoxPro commands execute inside the **FOR ... ENDFOR** loop. The variable does not need to exist before executing **FOR ... ENDFOR**.

nInitialValueTO nFinalValue

Specifies the initial and final values of the counter. Both nInitialValue and nFinalValue can be array elements.

[STEP nIncrement]

Specifies the amount to increment or decrement the counter value. If nIncrement is negative, the counter value is decremented. If you omit the **STEP** clause, VarName increments by 1.

Commands

Specifies the Visual FoxPro commands to execute. Commands can include any number of commands.

[EXIT]

Transfers control from within the **FOR ... ENDFOR** loop to the command immediately following **ENDFOR**. You can place **EXIT** anywhere between **FOR** and **ENDFOR**. For more information, see EXIT Command.

[LOOP]

Returns control to the **FOR** clause without executing the statements between the **LOOP** and **ENDFOR** keywords. The counter value increments or decrements as if **ENDFOR** was reached. For more information, see LOOP Command.

ENDFOR

Specifies the end of the **FOR ... ENDFOR** loop.

NEXT

Specifies the location to continue program execution after the counter value exceeds nFinalValue.