

UNIT 1: FUNCTIONS

- **A Function is an operation denoted by a function name
Followed by one or more operands enclosed by parenthesis.
The operands of functions are called Arguments.**
- **Classification of function**
 - **Scalar**
 - **Column**

Figure: 1.1 Functions

Notes:

Functions are classified as Scalar functions and Column functions. The argument of a column function is a set of values; an argument of a scalar function is a single value.

TYPES OF FUNCTIONS

COLUMN FUNCTIONS

AVG
COUNT
MAX
MIN
SUM

SCALAR FUNCTIONS

CHAR
COALESCE
DAY
DATE
DAYS
DECIMAL
DIGITS
FLOAT
HEX
HOUR
INTEGER
LENGTH
MICROSECOND
MINUTE
MONTH
NULLIF
SECOND
STRIP
SUBSTRING
TIME
TIMESTAMP
VALUE
VARGRAPHIC
YEAR

Figure: 1.2 Column Function

COLUMN FUNCTIONS

- **The arguments of a column function are a set of values derived from one or more columns.**
- **Keyword DISTINCT**
- **Column name**
- **WHERE clause**
- **The result of a COUNT function may not be a NULL value**

Figure: 1.3 Column Function

Notes:

The keyword DISTINCT is not considered an argument of the function.

ALL expressions in the column function should include a column name and must not include a column function.


A column function can be used in a WHERE clause only if that clause is part of a sub query of a HAVING clause and the column name specified in the expression is a correlated reference to a group.

The result of a COUNT function cannot be null value.

AVG

The AVG function returns the average of a set of numbers.

Syntax:

AVG ( expression)

Argument:

The argument values must be numbers and their sum must be within the range of the data type of the result.

Result:

The data type of the result is the same as the data type of the argument values, except that the result is a large integer if the argument values are small integers, and the result is double precision floating-point if the argument values are single precision floating-point. The result can be null.

Example:

```
EXEC SQL SELECT AVG (SALARY)
      INTO :AVERAGE
      FROM DSN8510.EMP
      WHERE WORKDEPT = 'D11'
END-EXEC.
```

Figure: 1.4 Avg

Notes:

Assuming DEC15 set the DECIMAL (15,2) variable AVERAGE to the average salary in department D11 of the employees in the sample table DSN8510.EMP.

COUNT

The COUNT function returns the number of rows or values in a set of rows or values.

Syntax:

COUNT (DISTINCT expression)

Argument:

The argument values can be any values other than character strings with a maximum length greater than 254 or graphic strings with a maximum length greater than 127.

Result:

The result of the function must be within the range of large integers and cannot be null.

The data type of the result is INTEGER.

Example:

- 1. EXEC SQL SELECT COUNT (*)
 INTO :FEMALE
 FROM DSN8510.EMP
 WHERE SEX = 'F'
EN-EXEC.**
- 2. EXEC SQL SELECT COUNT (DISTINCT WORKDEPT)
 INTO :FEMALE_IN_DEPT
 FROM DSB8510.EMP
 WHERE SEX = 'F'
END-EXEC.**

Figure: 1.5 Avg.

Notes:

Example1: Set the integer host variable FEMALE the number of females represented in the sample table DSN8510.EMP.

Example2: Set the integer host variable FEMALE_IN_DEPT to the number of departments that have at least one female as a member.

MAX

The MAX function returns the maximum value in a set of values.

Syntax:

MAX (ALL
DISTINCT expression)

The argument values can be any values other than character strings with a maximum length greater than 254 or graphic strings with a maximum length greater than 127.

Result:

The data type of the result and its other attributes are the same as the data type and attributes of the argument values. The result can be null.

Example:

```
EXEC SQL SELECT MAX(SALARY) / 12
          INTO :MAX_SALARY
          FROM DSN8510.EMP
END-EXEC.
```

Example2:

```
EXEC SQL SELECT MAX (LASTNAME)
          INTO :LAST_NAME
          FROM DSN8510.EMP
END-EXEC.
```

Figure: 1.6 Max

Notes:

Example1: Set the DECIMAL (8,2) variable MAX_SALARY to the maximum monthly salary of the employees represented in the sample table DSN8510.EMP.

Example2:

Find the surname that comes last in the collating sequence for the employees represented in the sample table DSN8510.EMP. Set the archar variable LAST_NAME to that surname

MIN

The MIN function returns the minimum value in a set of values.

Syntax:

MIN ( expression)

Argument:

The argument values can be any values other than character strings with a maximum length greater than 254 or graphic strings with a maximum length greater than 127.

Result:

The data type of the result and its other attributes of the argument values.

The result can be null.

Example1:

```
EXEC SQL SELECT MIN (SALARY) / 12
          INTO :MIN_SALARY
          FROM DSN8510.EMP
END-EXEC.
```

Example2:

```
EXEC SQL SELECT MIN (LASTNAME)
          INTO :FIRST_NAME
          FROM DSN8510.EMP
END-EXEC.
```

Figure: 1.7 Min

Notes:

Example1: Set the DECIMAL (15,2) variable MIN_SALARY to the minimum monthly salary of the employees represented in the sample table DSN8510.EMP.

Example2: Find the surname that comes first in the collating sequence for the employees represented in the sample table DSN8510.EMP. Set the VARCHAR (15) variable FIRST_NAME to that surname.

SUM

The SUM function returns the sum of a set of numbers.

Syntax:

SUM ( **expression**)

Argument: The argument values must be numbers and their sum must be within the range of the data type of the result.

Result:

The data type of the result is the same as the data type of the argument values, except that the result is a large integer if the argument values are small integers, and the result is double precision floating-point if the argument values are single precision floating-point. The result can be null.

Example:

```
EXEC SQL SELECT SUM (SALARY+COMM+BONUS)  
            INTO :INCOME  
            FROM DSN8210.EMP
```

Figure: 1.8 Sum

Notes:

Set the INTEGER variable INCOME to the total income from all sources (salaries, commissions, and bonuses) of the employees represented in sample table DSN8510.EMP. If DEC31 is not in effect, then because all three columns are DECIMAL (9,2), the resultant sum is DECIMAL (15,2).

SCALAR FUNCTIONS

- **A scalar function can be used whenever an expression can be used.**
- **applied to a single value**
- **The argument of a scalar function can be a function.**

Figure: 1.9 Scalar Function

Notes:

The restrictions on the use of column functions do not apply to scalar functions because a scalar function is applied to a single value rather than set of values. The argument of a scalar function can be a function.

CHAR

The CHAR function returns a string representation of a date time value or a decimal number.

Syntax:

CHAR (expression)

Argument:

The first argument must be a date, time, timestamp, or decimal number. The second argument, if applicable, is the same name of a date time format.

Result:

The result of the function is a fixed-length character string.

Example:

- 1. EXEC SQL SELECT CHAR (HIREDATE, USA)
 INTO :DATESTRING
 FROM DSN8510.EMP
 WHERE EMPNO = '0123'
 END-EXEC.**
- 2.CHAR (STARTING+: HOURS, USA)**
- 3. EXEC SQL SELECT CAHR (AVG(SALARY))
 INTO :AVERAGE
 FROM DSN8510.EMP
 END-EXEC.**

Figure: 1.10 Char

Example1:

HIREDATE is a DATE column in the sample table DSN8510.EMP.

When it represents 15 December 1976. Returns the string value '12/15/1976' in Character-string variable DATESTRING.

Example2:

HOURS is a DECIMAL (6,0) variable with a value of 50000. Interpreted as a time duration, this value is 5 hours. Assume that STARTING is a TIME column in some table. Then, when STARTING represents 17 hours, 30 minutes, and 12 seconds after midnight. Returns the value '10:30 PM'.

Example3:

The above SQL statement sets the CHAR (33) variable AVERAGE to the character string representation of the average employee salary with DEC31, the result of the AVG applied to a decimal number is a decimal number with a precision of 31 digits.

COALESCE

The Coalesce function is a synonym for the VALUE function.

Syntax:

COALESCE (expression -, expression -)



It return the first argument that is not null

COALESCE is used to conform to the SQL standard.

Figure: 1.11 Coalesce

DATE

The DATE function returns a date derived from its argument.

Syntax:

DATE (expression)

Argument:

The argument must be a timestamp, a date, a positive number less than or equal to 3652059, a valid sting representation of a date, or a character string of length Result:

The result of the function is a date. If the argument can be null the result can be null.

Example:

- 1. DATE (RECEIVED)**
- 2. DATE (DATCOL)**
- 3. DATE ('1989-03-02')**

Figure: 1.12 Date

Notes:

Example1: Assume that RECEIVED is a TIMESTAMP column in some table, and that one of its values is equivalent to the timestamp '1988-12-25-17.12.30.000000', then the internal representation of 25 December 1988 is returned.

Example2: Assume that DATCOL is a CHAR (7) column in some table, and that one of its values is the character string '1989061', then the internal representation of 2 March 1989 is returned.

Example3: DB2 recognizes '1989-03-02' as the ISO representation of 2 March 1989, then the internal representation of 2 March 1989 is returned.

DAY

The DAY function returns the day part of its argument.

Syntax:

DAY (expression)

Argument must be a date, timestamp, date duration, or time stamp duration.

Result:

The result of the function is a large integer. If the argument can be null the result can be null.

Example:

- 1. EXEC SQL SELECT DAY (HIREDATE)
 INTO :DAYVAR
 FROM DSN8510.EMP
 WHERE WMPNO = '0123'
 END-EXEC.**
- 2. DAY (DATE1 – DATE2)**

Figure: 1.13 Day

Notes:

Example1. Set the INTEGER variable DAYVAR to the day of the month on which employee 0123 in the sample table DSN8510.EMP was hired.

Example2. Assume that DATE1 and DATE2 are DATE columns in the same table. Assume also that for a given row in this table, DATE1 and DATE2 represent respectively 15 January 2000 AND 31 December 1999, then the above expression returns a value 15.

DAYS

Day's function returns an integer representation of a date.

Syntax:

DAYS (expression)

Argument:

- The argument must be a date, a timestamp, or a valid string representation of a date.

Result:

- The result of a function is a large integer.
- If the argument can be null, the result can also be null.

Example:

1. **EXEC SQL**
 SELECT DAYS ('1990-01-08') – DAYS (HIREDATE) + 1
 INTO: DAYSVAR
 FROM DSN8510.EMP
 WHERE EMPNO = '01230'
END-EXEC.
2. **EXEC SQL**
 SELECT (DAYS (HIREDATE) – DAYS (HIREDATE) / 7 * 7) + 1
 INTO: DAYOFWEEK
 FROM DSN8510.EMP
 WHERE EMPNO = '01230'
END-EXEC.

Notes:

Figure: 1.14 Days

Example 1: Set an INTEGER variable DAYS VAR to the number of days employee 01230 had been with the enterprise. Sample table DSN8510.EMP represents the number of days up to and including 8th Jan 1990.

Example 2: Set the INTEGER variable DAYOFWEEK to the numerical day of the week that employee 01230 was hired where 1 represents Sunday, 2 represent Monday ...etc and 7 represent Saturday. HIREDATE is a column of date type in sample table DSN8510.EMP.

DECIMAL

The DECIMAL function returns a decimal representation of value.

Syntax:

DECIMAL (expression, integer, integer)

Argument:

The first argument must be a character string or a number. The second argument, if specified, must range in value from 1 to 31. The third argument if specified must range in value from 1 to p, where p is the value of the second argument.

Result:

The data type of the result is DECIMAL (p, s) where p and s are the second and third arguments. If the first argument can be null, the results can be null.

Example:

EXEC SQL

SELECT DECIMAL (AVG (SALARY), 8,2)

FROM DSN8510.EMP

END-EXEC.

Figure: 1.15 Decimal

Notes:

Represent the average salary of the employees in DSN8510.EMP as an 8-digit decimal number with two of these digits to the right of the decimal point.

DIGITS

DIGITS function returns a character representation of its argument.

Syntax

DIGITS (expression)

Argument:

- The argument must be an integer or a decimal number.
- The length of a string is
 - 5 for small integer
 - 10 for large integer
 - p for decimal number where p is a precision.

Result:

- The result of the function is a fixed-length character string.
- The result does not include a sign or a decimal point.
- It consists of digits with leading zeroes to fill out the string.

Example 1:

EXEC SQL

```
SELECT DISTINCT SUBSTR (DIGITS (INTCOL), 1,4)
FROM TABLEX
```

END-EXEC.

Example 2:

DIGITS (COLUMNX)

The value '000628' is returned.

Figure: 1.16 Digits

Example 1:

Assume that an INTEGER column called INTOCL containing a 10-digit number is in a table called TABLEX. INTCOL has the data type INTEGER instead of CHAR (10) to save space. List all combinations of the first four digits in column INTCOL.

Example 2:

Assume that COLUMNX has the data type DECIMAL (6,2), and that one of its values is -6.28. The, for this value. The result is a string of length six (the precision of the column) with leading zeros padding the string out to this length. Neither sign nor decimal points appear in the result.

FLOAT

The FLOAT function returns a floating-point representation of the argument.

Syntax

FLOAT (expression)

Argument:

The argument must be a number.

Result:

The result of the function is a double precision floating-point number. If the argument can be null, the result can also be null.

Example:

EXEC SQL

SELECT EMPNO, FLOAT (SALARY) /COMM

FROM DSN8510.EMP

WHERE COMM > 0

END-EXEC.

Figure: 1.17 Float

Notes:

Using the sample table in DSN8510.EMP, find the ratio of salary to commission for employees whose commission is not zero. The columns involved (SALARY and COMM) have DECIMAL data types. To eliminate the possibility of out-of-range results, FLOAT is applied to SALARY so that the division is carried out in floating-point.

HEX

The **HEX** function returns a hexadecimal representation of its argument.

Syntax:

HEX (expression)

Argument:

- The argument can be any value other than a character string with a maximum length greater than 254 or a graphic string with a maximum length greater than 127.

Result:

- The result of the function is a character string.
- If the argument can be null, the result can be null; if the argument is null, the result is the null value.

Example:

EXEC SQL

SELECT HEX (START_RBA) FROM SYSIBM.SYSCOPY
END-EXEC.

Figure: 1.18 Hex

Notes:

Return the hexadecimal representation of START_RBA in the SYSIBM.SYSCOPY catalog table.

HOUR

The HOUR function returns the hour part of its argument.

Syntax:

HOUR (expression)

Argument:

- **The argument must be a**
- **Time**
- **Timestamp**
- **Time duration**
- **Timestamp duration**

Result:

- **The result of the function is a large integer.**
- **If the argument can be null, the result can be null; if the argument is null, the result is the null value.**

Example.

```
EXEC SQL  
SELECT *  
FROM CALSSES  
WHERE HOUR (STARTTM) > 12  
END-EXEC.
```

Figure: 1.19 Hour

Notes:

Assume that a table named CLASSES contains a row for each scheduled class. Assume also that the class starting times are in the TIME column named STARTTM. Using these assumptions, select those rows in CLASSES that represent classes that start after the noon hour.

INTEGER

The INTEGER function returns an integer representation of its argument.

Syntax:

INTEGER (expression)

Argument:

The argument must be a number.

Result:

The result of the function is a large integer. If the argument can be null the result can also be null

Example:

EXEC SQL

SELECT INTEGER (AVG (SALARY)+. 5)

FROM DSN8510.EMP

WHERE WORKDEPT = 'A00'

END-EXEC.

Figure: 1.20 Integer

Notes:

Finds the average salary of the employees in department 'A00', rounding the result to the nearest dollar.

LENGTH

The LENGTH function returns the length of its arguments.

Syntax:

LENGTH (expression)

Argument:

The argument can be any value.

Result:

The result of the function is a large integer. If the argument can be null, the result can also be null.

Example:

1. EXEC SQL

```
SELECT LENGTH (FIRSTNME)
FROM DSN8510.EMP
WHERE EMPNO = '280'
END-EXEC.
```

2.LENGTH (HIREDATE) and LENGTH (HIREDATE, EUR))

Figure: 1.21 Length

Notes:

Example1: Assume that FIRSTNME is a VARCHAR (12) column that contains 'ETHEL' for employee 280 and returns the value 5.

Example2: Assume that HIREDATE is a column of data type DATE. Then, regardless of value the value returned for the first expression is 4 and that for second is 10.

MICROSECONDS

The **MICROSECOND** function returns the microsecond part of its argument.

Syntax:

MICROSECOND (expression)

Argument:

The argument must be a

- **Timestamp**
- **Timestamp duration**

Result:

- The result of the function is a large integer.
- If the argument can be null, the result can be null.

Example:

EXEC SQL

SELECT MICROSECOND (TSTMPCOL)

FROM TABLEX

WHERE INTCOL = 1234

END-EXEC.

Figure: 1.22 Microseconds

Notes:

Assume that table TABLEX contains a **TIMESTAMP** column named TSTMPCOL and a **SMALLINT** column named INTCOL. Select the microseconds part of the TSTMPCOL column of the rows where the INTCOL value is 1234.

MINUTE

The MINUTE function returns the minute part of its argument.

Syntax:

MINUTE (expression)

Argument:

- **The argument must be a**
- **Time**
- **Timestamp**
- **Time duration**
- **Timestamp duration**

Result:

- **The result of the function is a large integer.**
- **If the argument can be null, the result can also be null.**

Example:

EXEC SQL

SELECT * FROM CLASSES

WHERE MINUTE (STARTTM) = 0

END-EXEC.

Figure: 1.23 Minute

Notes:

Assume that a table a table named CLASSES contains one row for each scheduled class. Assume also that the class starting times are in the TIME column named STARTTM. Using these assumptions, select those rows in CLASSES that represent classes that start on the hour.

MONTH

The **MONTH** function returns the month part of its argument.

Syntax:

MONTH (expression)

Argument:

The argument must be a date, timestamp, date duration, or timestamp duration.

Result:

The result of the function is a large integer. If the argument can be null, the result can also be null.

Example:

```
EXEC SQL SELECT *  
FROM DSN8510.EMP  
WHERE MONT (BIRTHDATE) = 5  
END-EXEC.
```

Figure: 1.24 Month

Notes:

Selects all rows in the sample table DSN8510.EMP for employees who were born in May.

NULLIF

The NULLIF function returns null if the two arguments are equal otherwise it returns the value of the first argument.

Syntax:

NULLIF (expression, expression)

Argument:

The two arguments must be compatible. The attributes of the result are the attributes of the first argument.

Result:

The result of using NULLIF (e1, e2) is the same as using the CASE expression

CASE WHEN e1=e2 THEN NULL ELSE e1 END.

Example (:PROFIT + :CASH, :LOSSES)

Figure: 1.25 Nullif

Notes:

Assume that host variables PROFIT, CASH, and LOSSES have decimal data types with the values of 4500.00, 500.00, and 5000.00 respectively.

SECOND

The SECOND function returns the seconds part of the argument.

Syntax:

SECOND (expression)

Argument:

The argument must be a time, timestamp, time duration, or timestamp duration.

Result:

The result of the function is a large integer. If the argument can be null, the result can also be null.

Example:

SECOND (:TIME_DUR)

SECOND (RECEIVED)

Figure: 1.26 Second

Notes:

Assume that the variable TIME_DUR is declared in the PL/1 program as DECIMAL (6,0) and can be interpreted as a time duration. Then when time duration has the value 53045, then the value returned is 45.

Example2: Assume that RECEIVED is a TIMESTAMP column and that one of its values is the internal equivalent of '1988-12-25-17.12.30.000000'. Then the value returned is 30.

STRIP

The STRIP function removes blanks or another specified cahraction from the end, the begining, or at both ends of a string expression.

Syntax:

**STRIP(expression, B (BOTH), strip-character)
L (LEADING)or
T(TRAILING**

Argument:

The first argument must be a string expression.

The second argument indicates whether cahraction are removed from the end or begining of the string.

The third argument is the character to be removed.

Result:

The result of the function is a varying length string with the same maximum length as the lenght attribute of the string. The actual length of the result is the length of the expression minus te number of bytes removed.

Example:

1. Assume the host variable HELLO of type CHAR (9) has a Value of ' HELLO',
STRIP (:HELLO)
Results in 'Hello'
STRIP (:HELLO<TRAILING)
Results in 'Hello'

2 Assume the host variable BALANCE of type CHAR (9) has a value of '000345.50',
STRIP (:BALANCE,L,'0')
Results in '345.50'.

Figure: 1.27 Strip

SUBSTR

The SUBSTR function returns a substring of a string.

Example:

1. SUBSTR (FIRSTNME, 2, 3)

Returns 'AUD'

SUBSTR (FIRSTNME, 2)

Returns 'AUDE'

SUBSTR (FIRSTNME, 2, 6)

Returns 'AUDE' followed by two blanks.

SUBSTR (FIRSTNME, 6)

Returns a string of length zero.

SUBSTR (FIRSTNME, 6, 4)

Returns four blanks.

2. EXEC SQL SELECT * FROM DSN8510.EMP

WHERE SUBSTR (PROJNME, 1, 12) = 'W L PROGRAM'

END-EXEC.

Figure:1.28 Substr

Notes:

Arguments:

String denotes an expression that specifies the string from which the result is derived.

Start denotes an expression that specifies the position of the first character of the result. Length denotes an expression that specifies the length of the result.

Result:

If length is explicitly specified by an integer constant less than 255 the result is a fixed-length string. If length is not specified but the string is a fixed length string, then the result is a fixed length string. In all other cases the result is a varying length string with a maximum length that is same as that of the length of the string.

TIME

The TIME function returns a time derived from its argument.

Syntax:

TIME (expression)

Argument: The argument must be a timestamp, a time or a valid string representation of a time.

Result: The result of the function is a time. If the argument can be null, the result can also be null.

Example:

```
EXEC SQL SELECT *  
FROM CLASSES  
WHERE TIME (STARTTM) = '13:30:00'  
END-EXEC.
```

Figure: 1.29 Time

Notes:

Example: Assume that a table named CLASSES contains one row for each scheduled class. Assume also that the class starting times are in the TIME column named STARTTM. Using these assumptions, select those rows in CLASSES that represent classes that start at 1:30 P.M.

TIMESTAMP

The TIMESTAMP function returns a timestamp derived from its argument.

Syntax:

TIMESTAMP (expression)

Argument: If only one argument is specified, it must be a timestamp with a character string of length 8 or 14.

If both arguments are specified, the first argument must be a date or a valid string representation of time.

Result: If both arguments are specified, the result is a timestamp with the date specified by the first argument and the time specified by the second argument.

The microsecond part of the timestamp is zero.

If only one argument is specified and it is a timestamp, the result is a timestamp.

Example:

TIMESTAMP(DATECOL,TIMECOL)

Returns the value '1988-12-25-17.12.30.000000'

Figure: 1.30 Timestamp

Notes:

Example: Assume that table TABLEX contains a DATE column named DATECOL and a TIME column named TIMECOL. Assume also that for some row in the table, DATECOL represents 25 December 1988 and TIMECOL represents 17 hours, 12 minutes and 30 seconds after midnight.

VALUE

The **VALUE** function returns the first argument that is not null. **COALESCE** can be used as a synonym for **VALUE**.

Syntax:

VALUE (expression)

Argument: The argument must be compatible.

The arguments are evaluated in the order in which they are specified.

Result: The result of the function is the first argument that is not null.

The result can be null only if all arguments can be null.

The result is null only if all arguments are null.

Example:

```
1.      EXEC SQL SELECT * FROM GRADES
WHERE VALUE (SCORE1,0) +
SCORE2 > 100
      END-EXEC.
```

```
2.      EXEC SQL SELECT * FROM DSN8510.EMP
WHERE VALUE (HIREDATE, DATE ('1959-12-31')) < '1960-01-01'
      END-EXEC.
```

Figure: 1.31 Value

Example1:

Assume that SCORE1 and SCORE2 are SMALLINT columns in table GRADES, and that nulls are allowed in SCORE1 but not in SCORE2. Select all the rows in GRADES for which $\text{SCORE1} + \text{SCORE2} > 100$, assume a value of 0 for SCORE1 when SCORE1 is null.

Example2:

Assume that a table named DSN8510.EMP contains a DATE column named HIREDATE, and that nulls are allowed for this column. The query selects all rows in DSN8510.EMP for which the date in HIREDATE is either unknown (null) or earlier than 1 January 1960. In this case the predicate value would be invalid because `VALUE (HIREDATE, DATE('1959-12-31'))`, the strings and date time values are not compatible.

VARGRAPHIC

The VARGRAPHIC function returns a graphic string representation of a character string.

Argument: The arguments must be an EBCDIC-encoded character string with a maximum length not greater than 254.

Result: The result of the function is a varying-length graphic string. If the argument can be null, the result can also be null.

The YEAR function returns the year part of its argument.

Argument: The argument must be a date, timestamp, date duration or Timestamp duration.

Syntax:

VARGRAPHIC (expression)

Figure: 1.32 VARGRAPHIC

Notes:

Example1: Assume that GRPHCOL is a GRAPHIC column in the table TABLEX, and the INSTRING is a character string variable. For various rows in TABLEX, the value of GRPHCOL is being replaced with the value of instring through the use of a positioned UPDATE statement. Before an update can be made, the current value of INSTRING must be converted to a GRAPHIC string. Within the UPDATE statement, this can be done using the VARGRAPHIC function.

Example2: From the table DSN8510.EMP select all rows for employees who are born in 1941.

Example1:

```
EXEC SQL UPDATE TABLEX  
SET GRPHCOL = VARGRAPHIC (:INSTRING)  
WHERE CURRENT OF CRSNAME  
END-EXEC.
```

Example2:

```
EXEC SQL SELECT * FROM DSN8510.EMP  
WHERE YEAR (BIRTHDATE) = 1941  
END-EXEC.
```