Unit 4. Scalar Functions and Arithmetic

What This Unit Is About

Scalar functions can be used to manipulate column or expression values. This unit will discuss the format and syntax of basic scalar functions.

Arithmetic allows us to work with or present calculated values, such as salary + comm.

Date columns can also be used in arithmetic calculations.

What You Should Be Able to Do

After completing this unit, you should be able to

- Use arithmetic in the SELECT and WHERE clauses
- Specify basic scalar functions such as COALESCE/VALUE, DECIMAL, SUBSTR
- Use date and time scalar functions
- Use the CONCAT operator

How You Will Check Your Progress

- Checkpoint questions
- Machine labs
Unit Objectives

After completing this unit, you should be able to:

- Use arithmetic in the SELECT and WHERE clauses
- Use basic scalar functions such as COALESCE/VALUE, DECIMAL, SUBSTR
- Use date and time scalar functions
- Use the CONCAT operator

Notes:
4.1 Scalar Functions and Arithmetic
Selecting Calculated Values

Notes:

- You can use the following arithmetic operators and parentheses() to calculate values:
  
  addition   +
  subtraction -
  multiplication *
  division    /
  
- Precedence of operations:
  
  Expressions within parentheses are evaluated first. When the order of elevation is not specified by parentheses, multiplication and division are applied before addition and subtraction. Operators at the same precedence level are applied from left to right.
  
- In the result table, derived columns, such as $SALARY+COMM$, do not have meaningful column names associated with them.
Naming Result Columns

```
SELECT EMPNO, SALARY, COMM, SALARY + COMM AS SALARY_COMM
FROM EMPLOYEE
WHERE SALARY < 20000
ORDER BY EMPNO
```

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>SALARY</th>
<th>COMM</th>
<th>SALARY_COMM</th>
</tr>
</thead>
<tbody>
<tr>
<td>000210</td>
<td>16270.00</td>
<td>1462.00</td>
<td>17732.00</td>
</tr>
<tr>
<td>000250</td>
<td>19180.00</td>
<td>1534.00</td>
<td>20714.00</td>
</tr>
<tr>
<td>000260</td>
<td>17250.00</td>
<td>1360.00</td>
<td>18610.00</td>
</tr>
<tr>
<td>000290</td>
<td>15940.00</td>
<td>1227.00</td>
<td>17167.00</td>
</tr>
<tr>
<td>000300</td>
<td>17750.00</td>
<td>1420.00</td>
<td>19170.00</td>
</tr>
<tr>
<td>000310</td>
<td>15900.00</td>
<td>1272.00</td>
<td>17172.00</td>
</tr>
<tr>
<td>000320</td>
<td>19950.00</td>
<td>1596.00</td>
<td>21546.00</td>
</tr>
</tbody>
</table>

Figure 4-3. Naming Result Columns (CF124030)

Notes:

Derived columns ordinarily have no column names or have dummy column names specified by the SQL tool used (SPUFI, QMF, Command Center, QMF for Windows). You can use the AS clause to rename the column headings for these derived column results.

Because the SELECT clause is executed after the WHERE clause the new column name may only be used in the ORDER BY clause.
Substitution of NULL Values

The COALESCE scalar function is equivalent to VALUE scalar function.

COALESCE allows multiple parameters:
COALESCE(expression_1, expression_2, expression_n, 'constant').

The COALESCE function takes a variable number of parameters and returns the value of the first parameter that is not null. The result is null if the values of all parameters are null.

In the example, the value "UNKNOWN" is returned if a department does not have a manager, that is, MGRNO is NULL.

Every expression must be of a compatible data type (if the first expression is numeric, every expression must be numeric, and so on).
Arithmetic with NULL Values

If one of the columns used in the expression contains a NULL value, the result of the expression is unknown.

In the left example, when commission is NULL, SALARY + COMM is null. In the right example, for SALARY + COMM, nulls are replaced with zeros.

Figure 4-5. Arithmetic with NULL Values (CF124050)
Calculated Values

```
SELECT EMPNO, SALARY, SALARY * 1.0375
FROM EMPLOYEE
WHERE SALARY < 20000
ORDER BY EMPNO
```

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>SALARY</th>
<th>SALARY * 1.0375</th>
</tr>
</thead>
<tbody>
<tr>
<td>000210</td>
<td>18270.00</td>
<td>18955.125000</td>
</tr>
<tr>
<td>000250</td>
<td>19180.00</td>
<td>19899.250000</td>
</tr>
<tr>
<td>000260</td>
<td>17250.00</td>
<td>17896.875000</td>
</tr>
<tr>
<td>000290</td>
<td>15340.00</td>
<td>15915.250000</td>
</tr>
<tr>
<td>000300</td>
<td>17750.00</td>
<td>18415.625000</td>
</tr>
<tr>
<td>000310</td>
<td>15900.00</td>
<td>16496.250000</td>
</tr>
<tr>
<td>000320</td>
<td>19650.00</td>
<td>20691.250000</td>
</tr>
</tbody>
</table>

Figure 4-6. Calculated Values (CF124060)

**Notes:**

The above example shows multiplication using a constant. You can also use calculations with multiple columns (for example, COL1 * COL2).
Decimal Representation of a Value

SELECT EMPNO, SALARY, DECIMAL (SALARY * 1.0375, 9, 2) 
FROM EMPLOYEE 
WHERE SALARY < 20000 
ORDER BY EMPNO

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>000210</td>
<td>18270.00</td>
</tr>
<tr>
<td>000250</td>
<td>19180.00</td>
</tr>
<tr>
<td>000260</td>
<td>17250.00</td>
</tr>
<tr>
<td>000290</td>
<td>15340.00</td>
</tr>
<tr>
<td>000300</td>
<td>17750.00</td>
</tr>
<tr>
<td>000310</td>
<td>15900.00</td>
</tr>
<tr>
<td>000320</td>
<td>19950.00</td>
</tr>
</tbody>
</table>

Figure 4-7. Decimal Representation of a Value (CF124070)

Notes:

The DECIMAL scalar function returns a decimal representation of a numeric value or a decimal representation of a numeric looking character value. In the example above, the DECIMAL scalar function is used to ensure that the result column has 2 decimal positions.

The result is truncated after the second decimal position. No rounding occurs with the DECIMAL scalar function.
## Decimal Values - Truncation and Rounding

```sql
SELECT EMPNO, SALARY,
     DECIMAL (SALARY * 1.0375 - 0.005, 8, 2)
FROM EMPLOYEE
WHERE SALARY < 20000
ORDER BY EMPNO
```

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>SALARY</th>
<th>SALARY_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>000210</td>
<td>18270.00</td>
<td>18955.13</td>
</tr>
<tr>
<td>000250</td>
<td>19180.00</td>
<td>19899.25</td>
</tr>
<tr>
<td>000260</td>
<td>17250.00</td>
<td>17896.88</td>
</tr>
<tr>
<td>000290</td>
<td>15340.00</td>
<td>15915.25</td>
</tr>
<tr>
<td>000300</td>
<td>17750.00</td>
<td>18415.63</td>
</tr>
<tr>
<td>000310</td>
<td>15900.00</td>
<td>16496.25</td>
</tr>
<tr>
<td>000320</td>
<td>19950.00</td>
<td>20695.13</td>
</tr>
</tbody>
</table>

---

Figure 4-8. Decimal Values - Truncation and Rounding (CF124080)

### Notes:

If you need a rounded result, add 5 to the first decimal position to be truncated and apply the DECIMAL function to this expression.

Some environments support a ROUND scalar function as another method of rounding. See the *SQL Reference* manual for your environment for additional information.
Condition on Calculated Values

```
SELECT EMPNO, COMM, SALARY, (COMM/SALARY) * 100
FROM EMPLOYEE
WHERE (COMM/SALARY) * 100 > 8
ORDER BY EMPNO
```

EMPNO  COMM  SALARY
-------  -----  ------
000140  2274.00  28420.00  8.001400
000210  1462.00  18270.00  8.002100
000240  2301.00  28760.00  8.000600
000330  2030.00  25370.00  8.001500

Figure 4-9. Condition on Calculated Values (CF124090)

**Notes:**

Arithmetic expressions can be used in the SELECT list and in the WHERE clause.

The above SELECT returns rows for employees with a commission higher than 8 percent of their salary.
Date and Time

- DATE, TIME, TIMESTAMP data internally stored as packed decimal, without sign

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Internal Format</th>
<th>Internal Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>DATE</td>
<td>yyyy-mm-dd</td>
<td>4 bytes</td>
</tr>
<tr>
<td>TIME</td>
<td>hh:mm:ss</td>
<td>6 bytes</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>yyyy-mm-dd-hh:mm:ss.nnnnnn</td>
<td>10 bytes</td>
</tr>
</tbody>
</table>

- Program uses an external format, that is,

<table>
<thead>
<tr>
<th>Format</th>
<th>Time Format</th>
<th>Length</th>
<th>Date Format</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISO</td>
<td>hh:mm:ss</td>
<td>6 bytes</td>
<td>yyyy-mm-dd</td>
<td>10 bytes</td>
</tr>
<tr>
<td>USA</td>
<td>hh:mm AM</td>
<td>8 bytes</td>
<td>mm/dd/yyyy</td>
<td>10 bytes</td>
</tr>
<tr>
<td>EUR</td>
<td>hh:mm:ss</td>
<td>8 bytes</td>
<td>dd.mm.yyyy</td>
<td>10 bytes</td>
</tr>
<tr>
<td>JIS</td>
<td>hh:mm:ss</td>
<td>8 bytes</td>
<td>yyyy-mm-dd</td>
<td>10 bytes</td>
</tr>
<tr>
<td>LOCAL</td>
<td>???</td>
<td>???</td>
<td>???</td>
<td>???</td>
</tr>
<tr>
<td>TIMESTAMP DATA: yyyy-mm-dd-hh:mm:ss.nnnnnn</td>
<td>26 bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4-10. Date and Time (CF124100)

Notes:

The visual shows the internal and external formats for DATE/TIME data.

A timestamp is a seven-part value (year, month, day, hour, minute, second, and microsecond) that represents a date and a time including microseconds.
Comparison with Dates

```
SELECT EMPNO, LASTNAME, BIRTHDATE
FROM EMPLOYEE
WHERE BIRTHDATE >= '1955-01-01'
ORDER BY BIRTHDATE
```

<table>
<thead>
<tr>
<th>EMPNO</th>
<th>LASTNAME</th>
<th>BIRTHDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>000160</td>
<td>PIANKA</td>
<td>1955-04-12</td>
</tr>
<tr>
<td>000100</td>
<td>SPENCER</td>
<td>1956-12-18</td>
</tr>
</tbody>
</table>

Figure 4-11. Comparison with Dates (CF124110)

**Notes:**

Columns defined with a DATE or TIME data type can be compared with other DATE or TIME columns or with a valid date or time.

Date or time constants (character string representations of a valid date or time) must be placed in single quotes. All legal formats can be used.

In the shown example, all employees are selected that were born '1955-01-01' or later.
DATE / TIME Arithmetic

• Subtraction only
  time - time → time duration (decimal (6,0))
  date - date → date duration (decimal (8,0))
  timestamp - timestamp → timestamp duration
                     (decimal (20,6))

• Labeled durations: YEARS, MONTHS, DAYS,
  HOURS, MINUTES, SECONDS, MICROSECONDS
  time ± labeled duration → time
  date ± labeled duration → date
  timestamp ± duration → timestamp

Notes:
A date duration represents the period between two dates. The result is presented in format yyyyymmdd.

A time duration represents the period between two times. The result is presented in format hhmmss.

A timestamp duration represents the period between two timestamps. The result is presented in format yyyyymmddhhmmss.nnnnnn (the nnnnnn portion represents microseconds).
Subtraction of Dates

Notes:

In above example, the values of two date columns are subtracted. The result of the subtraction is presented in the format yyyymmdd. Leading zeros are not displayed.

For example, in the first result line, the value 721116 means 72 years, 11 months, and 16 days.

In the WHERE clause, the calculated durations are compared with 650000, that is, 65 years, 00 months, and 00 days (older than 65 years).

CURRENT_DATE, CURRENT_TIME, and CURRENT_TIMESTAMP are special registers provided by DB2 UDB containing the system date, system time, or system timestamp.

On some platforms these registers are know by the names CURRENT DATE, CURRENT TIME, and CURRENT TIMESTAMP (without an underscore in the name).
Date / Time Scalar Functions

- **CHAR** controls external format of date / time data

  ```sql
  SELECT CHAR(TIMECOL, 'USA'), CHAR(TIMECOL, 'ISO'),
  SELECT 03:30 PM, 15:30:00
  ```

- **DAY, MONTH, YEAR, HOUR, MINUTE, SECOND, MICROSECOND, DATE, TIME**

  Extract portions of a date, time, timestamp, or duration

- **DAYS - Converts a date to the number of days since 12/31/0000**

Notes:
The visual shows some scalar functions for date and time values.
Date Scalar Functions

```
SELECT LASTNAME, FIRSTNAME,
       CURRENT_DATE - BIRTHDATE AS AGE,
       YEAR(CURRENT_DATE - BIRTHDATE) AS YEARS,
       MONTH(CURRENT_DATE - BIRTHDATE) AS MONTHS,
       DAY(CURRENT_DATE - BIRTHDATE) AS DAYS
FROM EMPLOYEE
WHERE YEAR(CURRENT_DATE - BIRTHDATE) > 56
ORDER BY AGE DESC, LASTNAME
```

<table>
<thead>
<tr>
<th>LASTNAME</th>
<th>FIRSTNAME</th>
<th>AGE</th>
<th>YEARS</th>
<th>MONTHS</th>
<th>DAYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Geyer</td>
<td>John</td>
<td>721118</td>
<td>72</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>Quintana</td>
<td>Dolores</td>
<td>721118</td>
<td>72</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>Gounot</td>
<td>Jason</td>
<td>720915</td>
<td>72</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>Lucches</td>
<td>Vincenzo</td>
<td>660926</td>
<td>66</td>
<td>9</td>
<td>26</td>
</tr>
<tr>
<td>Setright</td>
<td>Maude</td>
<td>670410</td>
<td>67</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>Mehta</td>
<td>Ramla</td>
<td>660021</td>
<td>66</td>
<td>0</td>
<td>21</td>
</tr>
</tbody>
</table>

Figure 4-15. Date Scalar Functions (CF124150)

Notes:

In the above example, year, month, and day are displayed separately using appropriate scalar functions.

The resulting columns normally do not have column headings. The AS clause is used to name the columns.
DATE Arithmetic

```
SELECT PROJNO,  
    DAYS(PRENDATE) - DAYS(PRSTDATE) AS DAYS  
FROM PROJECT  
WHERE DAYS(PRENDATE) - DAYS(PRSTDATE) <= 300  
ORDER BY DAYS
```

![Query Result](image)

**Notes:**

The example shows the duration of projects in days. To obtain a positive result, the project start date (PRSTDATE) must be subtracted from the project end date (PRENDATE) because the end date is higher.
DATE Arithmetic (Cont)

```
SELECT PROJNO, PRENDATE, PRENDATE + 2 MONTHS + 15 DAYS
FROM PROJECT
WHERE PROJNO = 'AD3100'
ORDER BY PROJNO
```

![SQL query result]

<table>
<thead>
<tr>
<th>PROJNO</th>
<th>PRENDATE</th>
<th>PRENDATE + 2 MONTHS + 15 DAYS</th>
</tr>
</thead>
<tbody>
<tr>
<td>AD3100</td>
<td>1993-02-01</td>
<td>1993-04-16</td>
</tr>
</tbody>
</table>

Figure 4-17. DATE Arithmetic (Cont) (CF124170)

**Notes:**

The example shows the use of labeled durations and adds 2 months and 15 days to a date.
Substring of Strings

If only a part of a character string is to be selected, use the SUBSTR scalar function to extract the part.

Assume that COURSINF is defined CHAR(20) and that both course code and course title are stored in that column, separated by a blank. To only display the course title, you can use the SUBSTR scalar function.

SUBSTR(COURSINF,6,15) returns a substring of length 15 beginning at position 6, that is, the course title in the above example.
**Notes:**

The first two characters of the project number represents the project class. To display only the project class, the SUBSTR function is used.

SUBSTR(PROJNO,1,2) extracts the first two characters. The function can be interpreted to read: Within the PROJNO column, start with the first character and return two characters.
Concatenation of Values

The above example displays last name and first name as a single column, separated by a comma and a space.

CONCAT combines character strings into a single column. The word CONCAT works in all environments.

Double bars (||) are often used in OS/390 to represent concatenation.

Double exclamation points (!!!) may be used in some environments.

Examples:

```
SELECT LASTNAME !! ',' !! FIRSTNME FROM EMPLOYEE

SELECT LASTNAME | | ',' | | FIRSTNME FROM EMPLOYEE
```

Check your environment's SQL documentation to see what additional concatenation symbols are supported.
Checkpoint

Exercise — Unit 4 Checkpoint

T F 1. If you use the DECIMAL scalar function to have two decimal positions instead of six, the result will be rounded.

2. If you subtract two dates, the format of the result will be:
   a. DD.MM.YYYY
   b. YYYY-MM-DD
   c. YYYYMMDD

3. Name several scalar functions.

   _______________________________________________________
   _______________________________________________________
   _______________________________________________________
   _______________________________________________________
   _______________________________________________________
   _______________________________________________________
   _______________________________________________________
   _______________________________________________________
   _______________________________________________________
   _______________________________________________________
Unit Summary

Since completing this unit, you should be able to:

- Use arithmetic in the SELECT and WHERE clauses
- Use basic scalar functions such as COALESCE/VALUE, DECIMAL, SUBSTR
- Use date and time scalar functions
- Use the CONCAT operator

Notes: