Typical consumers of ABAP applications are large corporations with operations spanning the globe. As a result, ABAP programs need to be written with the expectation that they can handle working across time zones and with different currencies. This sample chapter explains how to handle business data representing dates, times, units of measurement, and currencies in ABAP applications.

Brian O’Neill, Jelena Perfiljeva

ABAP: An Introduction
684 pages, 2nd, edition 2020, $69.95

www.sap-press.com/4955
Chapter 12

Working with Dates, Times, Quantities, and Currencies

This chapter explains how to handle business data that represents date and time, quantities in different units of measure, and amounts in different currencies.

Typical consumers of ABAP applications are large corporations with operations spanning the globe. As a result, ABAP programs need to be written with the expectation that they can handle working across time zones and with different currencies.

In addition, when working with any large company, the topics of inventory and quantity can be quite important. An ABAP program may need to work with converting measurements of products or keep track of how many items can fit in a given storage location for that item.

In this chapter, we’ll cover various key items to take into account when calculating and converting dates and times in ABAP programs. We’ll also cover how quantities are handled in ABAP transparent tables and how to convert quantities between different units of measurement. Then we’ll cover how currencies are stored in ABAP transparent tables and how to convert amounts between different currencies. Finally, we’ll update the shopping cart example program to use quantity and currency fields.

12.1 Dates

In this section, we’ll cover the details on how the date type (D) works and some different ways to use dates in ABAP programs. Chapter 3 introduced the date data type (D), showing how to define a date and that it is stored in a YYYYMMDD format where the first four digits represent the year, the second two represent the month, and last two represent the day.
12.1.1 Date Type Basics

Using substring

A date is a character string of eight characters, meaning that you can apply some of the string concepts from the previous chapter, such as using substring, to get specific year, month, and day values, as shown in Listing 12.1.

```plaintext
DATA: my_date TYPE d VALUE '20190102',
    year TYPE i,
    month TYPE i,
    day TYPE i.
year = substring( val = my_date off = 0 len = 4 ).
my_month = substring( val = my_date off = 4 len = 2 ).
my_day = substring( val = my_date off = 6 len = 2 ).
```

Listing 12.1 Using SUBSTRING to Get Year, Month, and Day

Adding/subtracting days

Adding days to a date data type can be done by adding an integer to the date; the date data type will handle updating the day, month, and year values, as shown in Listing 12.2, in which the new date is February 6, 2019.

```plaintext
DATA: my_date TYPE d VALUE '20190102'.
my_date = my_date + 35.
```

Listing 12.2 Adding 35 Days to January 2, 2019

You can also subtract days from a date variable in a similar fashion. In the example shown in Listing 12.3, the result will be January 2, 2019.

```plaintext
DATA: my_date TYPE d VALUE '20190102'.
my_date = my_date - 35.
```

Listing 12.3 Subtracting 35 days from February 6, 2019

Leap year

ABAP systems can easily handle leap years when calculating dates. The example in Listing 12.4 shows the result of adding 70 days to January 1 in the years 2019 and 2020.

```plaintext
DATA: start_date TYPE d,
    my_date TYPE d.
start_date = '20190101'.
my_date = start_date + 70.
WRITE: 'Date in a regular year:', 25 my_date.

start_date = '20200101'.
my_date = start_date + 70.
WRITE: / 'Date in a leap year:', 25 my_date.
```

Listing 12.4 Date Calculation Comparing Regular and Leap Years

Because 2020 is a leap year, the same operation results in a different date (March 11 instead of March 12), as shown in Figure 12.1.

Figure 12.1 Result of Date Calculation in Regular versus Leap Year

Adding a month to a date is not as easy. If you want to add a month to your date, you can call the `RP_CALC_DATE_IN_INTERNAL` function module shown in Listing 12.5, which will change the date from January 1, 2019, to February 1, 2019.

```plaintext
DATA: my_date TYPE d VALUE '20190101'.
CALL FUNCTION 'RP_CALC_DATE_IN_INTERNAL' 
    EXPORTING 
        date = my_date 
        days = 0 
        months = 1 
        years = 0 
    IMPORTING 
        calc_date = my_date.
```

Listing 12.5 Adding Month to Date Using Function Module
Other date operations

Dates also can be compared using the regular comparison operators, such as < or >. The example in Listing 12.6 will always return "True" because you’re adding 1 to the date and then comparing the result. If you change the sign from + to - in this example, it will always return "False".

Listing 12.6 Comparing Dates

```
DATA: one_date TYPE d VALUE '20190801',
     another_date TYPE d.

another_date = one_date + 1.
IF another_date > one_date.
  WRITE: 'True'.
ELSE.
  WRITE: 'False'.
ENDIF.
```

Listing 12.7 Assigning Invalid Date to Integer Variable

```
DATA: my_integer TYPE i,
     my_date TYPE d.

my_date = '20190230'.
my_integer = my_date.
IF my_integer = 0.
  WRITE: / 'Date is invalid'.
ENDIF.
```

12.1.2 Factory Calendars

There may be a situation in which you need to write a program that will act differently based on which days are working days for a given company. Factory calendar days will take into account any public holidays and any relevant working days, allowing you to complete date calculations based on the number of actual working days.

Factory calendars are client-independent and typically are created for each relevant country or region in which a company operates. This allows for a factory calendar to incorporate the local public holiday schedule. To view all of the calendars in an ABAP system, enter Transaction SCAL, select the Factory Calendar radio button, and click the Display button, as pictured in Figure 12.2.

Figure 12.2 Initial Screen for Transaction SCAL
In the next screen, you’ll see a list of the factory calendars defined in the system. These are usually based on country or region and will have **Valid From** and **Valid To** dates. Select the calendar you’re interested in viewing (USA in this example) and click the **Calendar** button, as shown in Figure 12.3.

**Figure 12.3** Selecting Factory Calendar

Next, select a relevant year for the calendar that you want to view and click the **Year** button, as shown in Figure 12.4.

**Figure 12.4** Selecting Calendar Year in Transaction SCAL

Now you’ll see a calendar display for the selected year, showing which days are considered working days and which days are nonworking days, including a list of public holidays for the year (Figure 12.5).

**Figure 12.5** Viewing Working/Nonworking Days and Public Holidays for Factory Calendar

Now that you understand what a factory calendar is and how to set one up, you can use factory dates in your ABAP code with the help of some standard function modules.

Use function module **DATE_CONVERT_TO_FACTORYDATE** to convert a regular date into a factory date. The factory date will be a number representing the number of working days since the factory calendar began. If you convert a date that isn’t a working day, the next working day will be used.

You can then perform some calculations using the factory date and convert it back to a calendar date using the **FACTORYDATE_CONVERT_TO_DATE** function module. An example of this is shown in Listing 12.8, which calculates a date 10 working days in the future.
PARAMETERS: p_date TYPE d OBLIGATORY.
DATA: simple_date TYPE d,
     factory_date TYPE scal-facdate.

simple_date = p_date.
WRITE: / 'Date entered:', p_date.
CALL FUNCTION 'DATE_C ONVERT_TO_FACTORYDATE'
  IMPORTING
    date = simple_date
  EXPORTING
    date = simple_date
    factory_calendar_id = 'US'
    factorydate = factory_date
EXCEPTIONS
    calendar_buffer_not_loadable = 1
    correct_option_invalid = 2
    date_after_range = 3
    date_before_range = 4
    date_invalid = 5
    factory_calendar_not_found = 6
    others = 7
.
ENDIF.
ADD 10 TO factory_date.
CALL FUNCTION 'FACTORYDATE_CONVERT_TO_DATE'
  IMPORTING
    date = simple_date
  EXPORTING
    date = factory_date
    factory_calendar_id = 'US'
EXCEPTIONS
    calendar_buffer_not_loadable = 1
    factorydate_after_range = 2
    factorydate_before_range = 3
    factorydate_invalid = 4
    factory_calendar_id_missing = 5
    factory_calendar_not_found = 6
    others = 7
.
IF sy-subrc <> 0.
MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno
  WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.
WRITE: / 'Result:', simple_date.

Listing 12.8 Using Factory Date Conversions to Complete Date Calculation

When executing this example with August 3, 2019, given as a parameter, the result is not August 13 but August 19, 2019, as shown in Figure 12.6.

Fiscal Year Calendar
Another type of calendar in SAP ERP systems is the fiscal year calendar, which is relevant to financial transactions and reports. The fiscal calendar configuration determines when each fiscal period begins and ends and how many periods are in a fiscal year for a specific company. Some companies may choose to use a fiscal calendar that differs from the physical calendar and has less or more than 12 periods. There are special function modules that help to work with the fiscal calendar dates. An important thing to remember here is that the postings in fiscal period 1 do not necessarily mean the dates from January 1 through January 31 in SAP ERP.

12.1.3 Datum Date Type
The d date type will output the day and month in the order matching the user’s preferences when using WRITE or an ALV grid, but it won’t include slashes or dots to separate the day, month, and year.
Datum formatting

Instead of using the d date type, you can use the datum date type, which will be displayed per the style of the user's date preference any time it's displayed to the user (see Listing 12.9 and the resulting output shown in Figure 12.7).

```plaintext
DATA: my_datum TYPE datum VALUE '20190801'.
WRITE: 'Type DATUM:', my_datum.

DATA: my_date TYPE d VALUE '20190801'.
WRITE: '/Type D:', my_date.
```

Listing 12.9 Demonstrating Date Type Differences

User preferences for the date and time format can be maintained in Transaction SU3, on the Defaults tab. Figure 12.8 shows an example of the date and time settings typical for a user located in the United States.

![Figure 12.8 User's Date and Time Preferences](image)

It's better to use the datum data type in your program instead of the d data type whenever you'll be displaying a date to the user. The datum and d date types can be used interchangeably in your code; the underlying data will be exactly the same.

12.1 Dates

12.1.4 System Date Fields

There are three system date fields: sy-datum, sy-datlo, and sy-fdayw. All these system dates return values related to the current date. sy-datum returns the current system date. The system date is dependent on the system's current time zone. If you're concerned about the date for the logged in user's time zone, you can use sy-datlo. Finally, sy-fdayw will return the factory day of week as an integer, meaning 0 for Sunday and 6 for Saturday.

These system date fields can all be directly assigned to valid variables and can also be used in the variable declaration, as shown in Listing 12.10.

```plaintext
DATA: today type sy-datum,
    today_local type sy-datlo,
    factory_day TYPE sy-fdayw.
```

Listing 12.10 System Date Fields

12.1.5 Date-Limited Records

A date-limited record is a record in a database with valid_from and/or valid_to date fields indicating the dates between which the record is considered valid. There are many reasons that a table may contain date-limited records; for example, a table may contain promotional discounts that are only valid between two dates, or a table containing an employee's position in the company may be date-limited so that it can contain all current and past positions.

Often, you don’t know when a record will no longer be valid, in which case the valid_to date will be set to the latest date possible, which is December 31, 9999. An HR record displaying an employee's current position would have a valid_to date like that; if the employee’s position changes, the valid_to date is updated to show the last day at that position.

Selecting a date-limited record valid today is easy. First, determine if you will be using the system date (sy-datum) or the local date (sy-datlo).
In the following example, we’ll select a record from user table `USR02`, which has optional `Valid From` and `Valid To` fields, as shown in Figure 12.9.

Using dates in SELECT

To select only the records that have a limited validity and are valid for today, you need to select records with a valid from date less than or equal to today and a valid to date greater than or equal to today. For this example, the valid from field is `GLTGV` and the valid to field is `GLTGB`, as shown in Listing 12.11, which will only return the record for user `BGRFC_SUPER`.

```
"New Open SQL
SELECT *
FROM USR02
INTO TABLE @DATA(results)
WHERE GLTGV <= @sy-datum
AND GLTGB >= @sy-datum.

"old Open SQL
DATA: results TYPE STANDARD TABLE OF USR02.
SELECT *
FROM USR02
INTO TABLE results
WHERE GLTGV <= sy-datum
AND GLTGB >= sy-datum.
```

Listing 12.11 Selecting Record Valid for Today

### 12.2 Times

In this section, we’ll cover how the time data type (t) works and some additional functions for calculating time, as well as other time data types. Chapter 3 introduced the time data type (t), and that time is stored with two characters for hours, two characters for minutes, and two characters for seconds, in the format HHMMSS.

If you need time that is more exact than seconds, you can also use a **timestamp**, which will be covered in this section. A timestamp will calculate time to the precision of 100 nanoseconds (ns), which is seven decimal places, and will also include the date. The timestamp format is YYYYMMDDHHMMSSSSSSSS.

#### 12.2.1 Calculating Time

Just as with dates, you can compare time using the regular comparison operators and calculate time using regular math expressions. The example in Listing 12.12 creates two variables of type t and fills in one of them with the current system time using the `sy-zeit` system field. Then the number 5 is added to that variable and placed in the `another_time` variable. At the end, the variable values are compared for equality.

```
DATA: one_time TYPE t,
     another_time TYPE t.

one_time = sy-zeit.
WRITE: / 'Current time:', one_time.
another_time = one_time + 5.
WRITE: / 'Current time + 5:', another_time.
IF one_time = another_time.
  WRITE: / 'Time is the same'.
ELSE.
  WRITE: / 'Time is different'.
ENDIF.
```

Listing 12.12 Calculating and Comparing Time Variables
Figure 12.10 shows an example of executing this program. Note that calculations with time occur in seconds; therefore, adding the number 5 to a time variable adds five seconds as a result.

### 12.2.2 Timestamps

A timestamp is a combination of date and time for the Coordinated Universal Time (UTC) time zone. This can be used to record the exact date and time when an action was completed by a user without having to rely on the local time zone for the user or the server.

There are two types of timestamp data types: `timestamp` and `timestampl`. `timestamp` is a short-form timestamp, which represents only the date and time combined with a format of YYYYMMDDHHMMSS. `timestampl` is a long-form timestamp and will include up to seven decimal places to capture the time with the precision of up to 100 ns using a format of YYMMDDHHMMSS.SSSSSSS.

#### Actual Time Precision

The maximum time precision depends on the operating system of the application server and can be less precise than allowed by the long-form timestamp format.

You can get the current timestamp using the `GET TIME STAMP FIELD` function, as shown in Listing 12.13. This function will work with both `timestamp` and `timestampl` data types. If an inline data declaration is used, `timestamp`, not `timestampl`, will be used.

```plaintext
DATA: my_timestamp TYPE timestamp,
     my_timestampl TYPE timestampl.
GET TIME STAMP FIELD my_timestamp.
```

#### Comparing timestamps

The regular comparison operators can only be used between two timestamps, not between a timestamp and a date or time. The comparisons will work between `timestamp` and `timestampl` type variables, as shown in Listing 12.14. In this example, the `my_timestampl` variable will be found to be bigger and "True" will be displayed.

```plaintext
DATA: my_timestamp TYPE timestamp,
     my_timestampl TYPE timestampl.
my_timestamp = '20190101000000'.
my_timestampl = '20190101000000.0000001'.
IF my_timestampl > my_timestamp.
  WRITE: 'True'.
ENDIF.
```

#### Converting timestamps

You can also convert a timestamp to a local date and time variable using the `CONVERT TIME STAMP` function. When using this function, you must specify a time zone that exists in table `TTZZ`. The result can be stored in explicitly defined date and time variables or using inline declarations, which will be created as date and time variables. You also can store a Boolean value for whether or not daylight saving time is in effect. The example in Listing 12.15 gets the current UTC timestamp, converts it to a Pacific Standard Time (PST) time zone date and time, and captures the daylight savings indicator. The `DATE`, `TIME`, and `DAYLIGHT SAVING TIME` additions are all optional, so you can use only the ones that you need.

```plaintext
GET TIME STAMP FIELD my_timestampl.
GET TIME STAMP FIELD data(my_inline_timestamp).

Listing 12.13 GET TIME STAMP FIELD

The regular comparison operators can only be used between two timestamps, not between a timestamp and a date or time. The comparisons will work between `timestamp` and `timestampl` type variables, as shown in Listing 12.14. In this example, the `my_timestampl` variable will be found to be bigger and "True" will be displayed.

DATA: my_timestamp TYPE timestamp,
     my_timestampl TYPE timestampl.
my_timestamp = '20190101000000'.
my_timestampl = '20190101000000.0000001'.
IF my_timestampl > my_timestamp.
  WRITE: 'True'.
ENDIF.

Listing 12.14 Comparing TIMESTAMP and TIMESTAMPL Data Types

You can also convert a timestamp to a local date and time variable using the `CONVERT TIME STAMP` function. When using this function, you must specify a time zone that exists in table `TTZZ`. The result can be stored in explicitly defined date and time variables or using inline declarations, which will be created as date and time variables. You also can store a Boolean value for whether or not daylight saving time is in effect. The example in Listing 12.15 gets the current UTC timestamp, converts it to a Pacific Standard Time (PST) time zone date and time, and captures the daylight savings indicator. The `DATE`, `TIME`, and `DAYLIGHT SAVING TIME` additions are all optional, so you can use only the ones that you need.

GET TIME STAMP FIELD DATA(my_utc).
CONVERT TIME STAMP my_utc TIME ZONE 'PST' INTO DATE DATA(my_pst_date)
TIME DATA(my_pst_time)
DAYLIGHT SAVING TIME DATA(my_dst).

Listing 12.15 Converting Timestamp from UTC to PST Date and Time
The examples in this section use inline declarations. If inline declaration syntax isn’t yet available in your system, use the explicit declaration with the DATA command, as shown in Listing 12.16. Other examples can be adjusted similarly.

```abap
DATA: my_utc type timestamp,
     my_pst_date type d,
     my_pst_time type t,
     my_dst type abap_bool.
GET TIME STAMP FIELD my_utc.
CONVERT TIME STAMP my_utc TIME ZONE 'PST'
INTO DATE my_pst_date
     TIME my_pst_time
     DAYLIGHT SAVING TIME my_dst.
```

Listing 12.16 Converting Timestamp from UTC to PST Date and Time without Inline Declarations

You can also perform a reverse operation and convert a local date and time into a timestamp using the CONVERT INTO TIMESTAMP function. A DATE and TIME ZONE are both required when using this function, but the TIME and DAYLIGHT SAVING TIME variables are optional. An example converting the results from Listing 12.15 back into a timestamp is shown in Listing 12.17. Remember that the time zone passed is used to convert the local date and time from that time zone into a UTC time zone.

```abap
CONVERT DATE my_pst_date
     TIME my_pst_time
     DAYLIGHT SAVING TIME my_dst
INTO TIME STAMP my_utc TIME ZONE 'PST'.
```

Listing 12.17 Converting PST Date and Time to UTC Timestamp

When outputting timestamps to a string template, there are special formatting options that can be applied to display the timestamp in a more user-friendly format. Table 12.1 provides a list of all the possible formatting options.

<table>
<thead>
<tr>
<th>Timestamp Format</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SPACE</td>
<td>Based on ISO 8601. Format: YYYY-MM-DD HH:MM:SS.ZZZZZZZ. For example, 2019-01-01 12:00:00.0000000.</td>
</tr>
<tr>
<td>ISO</td>
<td>Also based on ISO 8601, but uses a “T” to separate the date and time and always uses a comma as the seconds decimal. Format: YYYY-MM-DDTHH:MM:SS.ZZZZZZZ. For example, 2019-01-01T12:00:00.0000000.</td>
</tr>
<tr>
<td>USER</td>
<td>Based on the user’s preferences, which can be changed by going to System • User Profile • Own Data under the Defaults tab. For example: 01/01/2019 12:00:00 PM.</td>
</tr>
<tr>
<td>ENVIRONMENT</td>
<td>Based on the user’s currently selected language environment. Will display the default for that country, regardless of the user defaults. For example: 01/01/2019 12:00:00 PM.</td>
</tr>
</tbody>
</table>

Table 12.1 Timestamp Formats for String Templates

The syntax to use these different timestamp formats is simple. Within the string template, between the curly braces you put the timestamp variable name followed by TIMESTAMP = and then any of the formats listed in Table 12.1. The example shown in Listing 12.18 will return 2019-01-01T12:00:00.0000000.

```abap
DATA: unformatted TYPE timestamp1
     VALUE '20190101120000.0000000'.
DATA(my_string) = #{ unformatted TIMESTAMP = ISO }.
```

Listing 12.18 Setting Timestamp Format from within String Template

You also can convert the time used in the string template to reflect a given time zone by adding TIMEZONE = followed by the desired time zone, as shown in Listing 12.19, which will return 2019-01-01T04:00:00.0000000.

```abap
my_string = #{ unformatted TIMESTAMP = ISO
    TIMEZONE = "PST" }.
```

Listing 12.19 Setting Timestamp Output and Time Zone for Output in String Template
There are some additional timestamp-related functions that use the CL_ABAP_TSTMP system class. This class should be used if you want to add or subtract seconds from a timestamp, calculate the difference between two timestamps, or convert a long timestamp to a short timestamp or vice versa.

### 12.2.3 SY-UZEIT (System Time versus Local Time)

There are a few system fields related to time. Just as with system dates, there are times based on the system time zone and times based on the user’s time zone. The system time can be found with the `sy-uzeit` system variable, whereas the user’s local time can be found with `sy-timlo`; both will return the result in a variable of type `t`. You can also get the system time zone using `sy-tzone` or the user’s local time zone using `sy-zonlo`, both of which will return a `char` variable with a length of 6.

Just as with system dates, you can declare system variables as data types, as shown in Listing 12.20.

```abap
DATA: sys_time TYPE sy-uzeit,
     sys_timezone TYPE sy-tzone,
     loc_time TYPE sy-timlo,
     loc_timezone TYPE sy-zonlo.
```

Listing 12.20 Declaring and Getting System Time and Local Time Data

### 12.3 Quantities

Quantitative fields are used in SAP ERP systems to track and convert the results of different types of measurements, ranging from lengths (e.g., meters and inches) to masses (e.g., grams and pounds) and units of packaging (e.g., box and pallet). These fields are always associated with a unit of measure (UOM), which allows for converting the quantitative values from one unit to another.

You can see the different possible units of measure that are set up in your ABAP environment by entering Transaction CUNI, shown in Figure 12.11. From there, select a dimension such as **Length** from the dropdown, and click **Units of Measurement**.

![Figure 12.11 Initial Screen of Transaction CUNI](image)

Next, you’ll see a list of possible measurements for that dimension, as shown in Figure 12.12.

![Figure 12.12 List of Length Units of Measurement from Transaction CUNI](image)
12.3.1 Quantity Fields in Data Dictionary

When storing quantities in transparent tables, you need to store the quantity as a QUAN data type and the unit of measure as a UNIT data type. Then, when creating the table using Transaction SE11, you need to relate the two fields in the Currency/Quantity Fields tab.

To demonstrate this, create table ZQUANTITY, shown in Figure 12.13, using Transaction SE11. The table should contain field MANDT with data element MANDT, field ID with data element INTEGER, field QUANTITY with data element BASM (type QUAN), and field UNIT_OF_MEASURE with data element CGSUNIT (type UNIT). If some of the data elements don’t look familiar, don’t worry: they’re SAP-standard data types that should exist in any ABAP system.

Next, click the Currency/Quantity Fields tab, and you’ll see that only the QUANTITY field has an editable row. From here, you can add a reference to the table and field that will describe the quantity’s unit. For this example, enter the current table, “ZQUANTITY”, and the field, “UNIT_OF_MEASURE”, as pictured in Figure 12.14. Assigning a corresponding unit of measure field for any quantity data type is required to activate the table.

Because you store the quantity and unit in the same table, different records can have data stored in different quantities.

It’s possible to store the unit of measure in one table and reference it within a quantity in another table, if this is required by your business rules. In this case, you would enter a different table name in the Reference Table field in Transaction SE11.

12.3.2 Converting Quantities

Within an ABAP program, you can read data stored in a certain unit of measure and then convert the value into a different unit of measure so long as both are part of the same dimension. For example, you can convert one length unit, such as centimeters (cm), into another length unit, such as meters (m).

You can complete the conversion using the UNIT_CONVERSION_SIMPLE function module, which allows you to pass in a quantity with a unit to convert from and a unit to convert to. This function module will also round results if necessary, and you can optionally pass in a + or – to force it to round up or down. Listing 12.21 demonstrates converting a quantity of 5 from meters (M) to centimeters (CM). The unit of measure used can also be seen in the length units of measure list in Figure 12.12.

DATA: my_quantity TYPE i VALUE 5.
CALL FUNCTION 'UNIT_CONVERSION_SIMPLE'
  EXPORTING
    input = my_quantity
    unit_in = 'M'
    unit_out = 'CM'
IMPORTING
    output = my_quantity
EXCEPTIONS
  conversion_not_found = 1
  division_by_zero = 2

Figure 12.13 Table Containing Quantity and Unit of Measure

Figure 12.14 Currency/Quantity Fields Data for Assigning Units to Quantities

ABAP unit conversion
input_invalid = 3
output_invalid = 4
overflow = 5
type_invalid = 6
units_missing = 7
unit_in_not_found = 8
unit_out_not_found = 9
others = 10

IF sy-subrc <> 0.
MESSAGE ID sy-mgid TYPE sy-mgty NUMBER sy-msgno
  WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.
WRITE: my_quantity.

Listing 12.21 ABAP Code for Converting Data between Measurements

12.4 Currencies

Currencies work a lot like quantities; they’re used to capture both a value (amount) and a type (currency code). In today’s world of global business, transactions can be captured in one currency and then transferred to another. This makes it very important to keep track of the currency being referenced in any financial transaction, which is easy with an ABAP system.

You can see a list of currencies defined in your system by entering Transaction OY03. The results will be displayed as shown in Figure 12.15.

The SAP ERP currency codes are listed in the Currency column and corresponding ISO currency codes in the ISO column. Note that even though most SAP currency codes match the ISO standard, there could be sometimes multiple internal SAP codes for one ISO currency code.

When converting amounts across different currencies, the current exchange rate can be found in Transaction OB08. Because exchange rates regularly change, it’s possible to have the exchange rates in your system automatically updated on a daily or weekly basis, which is a topic outside the scope of this book. Figure 12.16 shows that the currency exchange rate between the US dollar and the euro is 0.94 euros to 1 US dollar.

Figure 12.15 List of Currencies from Transaction OY03

12.4.1 Currency Fields in Data Dictionary

Just as quantities use a QUAN data type to keep track of the quantity and a UNIT data type to keep track of the unit of measure, a currency value requires that the amount of currency be stored in a field with a CURR data type, and the currency key needs to be stored with a CUKY data type. You can then relate the fields in the Currency/Quantity Fields tab in the Change Table screen from Transaction SE11.

To demonstrate this, let’s create table ZCURRENCY, shown in Figure 12.17, using Transaction SE11. The table should contain field MANDT with data element MANDT, field ID with data element INTEGER, field PRICE with data element PREIS (type CURR), and field CURRENCY with data element USRCUKY (type CUKY). If some of the data elements don’t look familiar, don’t worry: they’re standard SAP data types that should exist in any ABAP system.

Figure 12.16 Currency Exchange Rates from Transaction OB08

Currency in a transparent table
You won’t be able to activate table ZCURRENCY until you add a reference for the currency field. This works just as you saw with the quantity field; select the Currency/Quantity Fields tab and enter “ZCURRENCY” as the reference table and “CURRENCY” as the reference field, so that the currency and currency key fields are linked as shown in Figure 12.18.

You won’t be able to activate table ZCURRENCY until you add a reference for the currency field. This works just as you saw with the quantity field; select the Currency/Quantity Fields tab and enter “ZCURRENCY” as the reference table and “CURRENCY” as the reference field, so that the currency and currency key fields are linked as shown in Figure 12.18.

Because you have a field for currency in the table, each record in the table can have any currency denomination. And just like with quantity fields, you also can reference the currency key in another table.

#### 12.4.2 Converting Currencies

Within an ABAP program, you may need to convert an amount from one currency to another using the current exchange rates shown in Transaction OB08. You can do so using the CONVERT_TO_LOCAL_CURRENCY function module.

When using CONVERT_TO_LOCAL_CURRENCY, you’re required to pass a date that will be used to determine the exchange rate for converting foreign_amount from foreign_currency to local_currency. The result will be returned as local_amount, as shown in Listing 12.22, and you can also gather additional information from the function module, such as the exchange rate used, or you can specify an exchange rate to be used in the conversion.

```
DATA: local_amount TYPE P DECIMALS 2.
CALL FUNCTION 'CONVERT_TO_LOCAL_CURRENCY' 
  EXPORTING 
    date = sy-datlo 
    foreign_amount = 100 
    foreign_currency = 'EUR' 
    local_currency = 'USD'
  IMPORTING 
    local_amount = local_amount
EXCEPTIONS 
  no_rate_found = 1 
  overflow = 2 
  no_factors_found = 3 
  no_spread_found = 4 
  derived_2_times = 5 
  others = 6.
```

```
IF sy-subrc <> 0.
  MESSAGE ID sy-msgid TYPE sy-msgty NUMBER sy-msgno 
    WITH sy-msgv1 sy-msgv2 sy-msgv3 sy-msgv4.
ENDIF.
```

```
Listing 12.22 Currency Conversion
```

#### Currency Conversion Factor and Decimal Places

Due to historic inflation, some currencies, such as Japanese yen (JPY), operate in much larger amounts than other currencies. Such special currencies can be configured with a conversion factor (stored in table TCURF) and the
factor would be applied to the amounts when stored in the database. This means that the amount of 100 JPY, for example, would be stored as 0.01 in the database.

Some currencies may also use a different number of decimals than two, or no decimals at all. The information about decimal places in currencies is stored in table TCURX.

The good news is that the standard SAP functionality incorporates this configuration into currency field display and conversion. That’s why it’s important to always reference the currency code when dealing with currency-specific amounts in ABAP code.

You should now be familiar with currencies and the aspects required for storing and converting them. In the next section, we’ll update the shopping cart example with currencies and quantities.

12.5 Updating the Shopping Cart Example

Now that you know how to use dates, times, quantities, and currencies, let’s revisit the shopping cart example to take advantage of the currency and quantity features.

First, we need to update the ERD to include currency and quantity values. In Figure 12.19, note the addition of PRICE and CURRENCY in table ZPRODUCT, allowing you to set a price per item for a specified currency. Also, a unit of measurement (UOM) is added to table ZPRODUCT, allowing you to determine the quantity unit that the price is set for. This quantity unit also is referenced by the quantity of the material in a cart (the QUANTITY field in table ZCART).

First, create the data element for the product price field. To begin, enter Transaction SE11, select the Data Type radio button, enter “ZPRODUCT_PRICE” in the corresponding textbox, and click the Create button. Select the Data Element radio button in the Data Type popup and click the Continue button.

In the Change Data Element screen, enter “Product Price for Shopping Cart Example” in the Short Description textbox. Then, with the Data Type tab selected, choose the Elementary Type radio button and the Built-in Type radio button. Enter the Data Type “CURR”, Length “11”, and Decimal Places “2”. We’re using the built-in type option instead of a domain in the new data elements because we don’t need any of the additional settings available within a domain. Your data element should look like the one pictured in Figure 12.20.

You’re now ready to update the transparent tables to match the updated ERD diagram. This section will walk through the steps to accomplish that.

First, create the data element for the product price field. To begin, enter Transaction SE11, select the Data Type radio button, enter “ZPRODUCT_PRICE” in the corresponding textbox, and click the Create button. Select the Data Element radio button in the Data Type popup and click the Continue button.

In the Change Data Element screen, enter “Product Price for Shopping Cart Example” in the Short Description textbox. Then, with the Data Type tab selected, choose the Elementary Type radio button and the Built-in Type radio button. Enter the Data Type “CURR”, Length “11”, and Decimal Places “2”. We’re using the built-in type option instead of a domain in the new data elements because we don’t need any of the additional settings available within a domain. Your data element should look like the one pictured in Figure 12.20.

Next, select the Field Label tab and enter “Price” for all Field Label textboxes. Then press Enter and the Length textboxes will be automatically populated, as shown in Figure 12.21. Click the Activate button to activate the data element and save it as a local object in the $TMP package.
Currency data element

Next, return to Transaction SE11, select the Data Type radio button, enter “ZPRODUCT_CURRENCY” in the corresponding textbox, and click the Create button. Select the Data Element radio button in the Data Type popup and click the continue button.

In the Change Data Element screen, enter “Product Currency for Shopping Cart Example” in the Short Description textbox. Then, with the Data Type tab selected, choose the Elementary Type radio button and the Built-in Type radio button. In the Data Type field, use the input help dropdown and select the CUKY type. The Length field will be automatically set to 5. Your data element should look like the one pictured in Figure 12.22.

Next, select the Field Label tab and enter “Currency” for all Field Label textboxes. Then press Enter and the Length textboxes will be automatically populated, as shown in Figure 12.23. Click the Activate button to activate the data element.

UOM data element

Next, return to Transaction SE11, select the Data Type radio button, enter “ZPRODUCT_UOM” in the corresponding textbox, and click the Create button. Select the Data Element radio button in the Data Type popup and click the Continue button.

In the Change Data Element screen, enter “Product UOM for the Shopping Cart Example” in the Short Description textbox. Then, with the Data Type tab selected, choose the Elementary Type radio button and the Built-in Type radio button. Enter the Data Type “UNIT” and Length “3”. Your data element should look like the one pictured in Figure 12.24.
Next, select the **Field Label** tab and enter “Unit” for the **Short Field Label** textbox and “Unit of Measure” for all other **Field Label** textboxes. Then press Enter and the **Length** textboxes will be automatically populated, as shown in Figure 12.25. Click the **Activate** button to activate the data element.

![Figure 12.25 ZPRODUCT_UOM Field Label Settings](image)

Next, return to Transaction SE11, select the **Data Type** radio button, enter “ZCART_QUANTITY” in the corresponding textbox, and click the **Create** button. Select the **Data Element** radio button in the **Data Type** popup and click the **Continue** button.

In the **Change Data Element** screen, enter “Cart Quantity for the Shopping Cart Example” in the **Short Description** textbox. Then, with the **Data Type** tab selected, choose the **Elementary Type** radio button and the **Built-in Type** radio button. Enter the **Data Type** “QUAN”, **Length** “13”, and **Decimal Places** “3”. Your data element should look like the one pictured in Figure 12.26.

Next, select the **Field Label** tab and enter “Quantity” for all the **Field Label** textboxes. Then press Enter and the **Length** textboxes will be automatically populated, as shown in Figure 12.27. Click the **Activate** button to activate the data element.

![Figure 12.26 ZCART_QUANTITY Data Type Settings](image)

Now that you've created the new data elements, let's update the transparent tables that use those data elements. First, update transparent table ZPRODUCT. To do this, go back to Transaction SE11, select the radio button for **Database Table**, and enter “ZPRODUCT” in the corresponding textbox. Then press the **Change** button.

With the **Fields** tab selected, add **Field** “PRICE” with **Data element** “ZPRODUCT_PRICE”, **Field** “CURRENCY” with **Data element** “ZPRODUCT_CURRENCY”, and **Field** “UOM” with **Data element** “ZPRODUCT_UOM”, as shown in Figure 12.28.
Next, select the Currency/Quantity Fields tab and enter Reference Table “ZPRODUCT” and Reference Field “CURRENCY” for the Field “PRICE”, as shown in Figure 12.29. This will set the CURRENCY field as the currency key for the PRICE field.

Table ZPRODUCT table is now done! Click the Activate button to save and activate the table.

Now, go back to Transaction SE11, select the radio button for Database Table, and enter “ZCART” in the corresponding textbox. Then press the Change button.

With the Fields tab selected add Field “Quantity” with Data Element “ZCART_QUANTITY”, as shown in Figure 12.30.

Now, select the Currency/Quantity Fields tab and enter Reference Table “ZPRODUCT” and Reference Field “UOM” for Field “Quantity”. This will set the UOM field in table ZPRODUCT as the unit for the QUANTITY field in this table, as shown in Figure 12.31.

Table ZCART is now complete! Click the Activate button to save and activate the table.

In the next section, we’ll update the global class to use the new currencies and quantities.

### 12.5.2 Updating the Global Class

Now that the tables have been updated, you need to update the ZCL_SHOPPING_CART global class to use the new information. You’ll need to update the
ADD_PRODUCT method to take an additional parameter for quantity and will need to update the GET_CART method to return the quantity and unit of measure for all the items in the cart.

First, let’s update the ADD_PRODUCT method. To do this, update the method definition to include the IMPORTING parameter ip_quantity as shown in Listing 12.23. Add the new parameter as an optional parameter so that other programs aren’t impacted by the change.

```
add_product IMPORTING ip_product TYPE zproduct-product
   ip_quantity TYPE zcart-quantity OPTIONAL,
```

Listing 12.23 Updated Definition for ADD_PRODUCT Method

Now you can update the ADD_PRODUCT method implementation. The change will be to check if ip_quantity has a value and, if it does, to use that value in the local structure; otherwise, set the local structure quantity to 1. You can also now change the INSERT statement to a MODIFY statement. This will allow the ADD_PRODUCT method to be called to change the quantity and to add new products. This is accomplished using the code in Listing 12.24, which has the specific changes in bold.

```
METHOD add_product.
   DATA: cart TYPE zcart.
      cart-customer = customer.
      cart-product = ip_product.
      IF ip_quantity IS NOT INITIAL.
         cart-quantity = ip_quantity.
      ELSE.
         cart-quantity = 1.
      ENDIF.
      MODIFY zcart FROM ls_cart.
ENDMETHOD.
```

Listing 12.24 Updated ADD_PRODUCT Method Implementation

Next, update the GET_CART method to return the quantity and unit of measure. To do this, first update the t_cart definition to include the quantity and unit of measure variables, as shown in bold in Listing 12.25.

```
PUBLIC SECTION.
   TYPES: BEGIN OF t_cart,
      product TYPE zproduct-product,
      description TYPE zproduct_text-description,
      quantity TYPE zcart-quantity,
      uom TYPE zproduct-uom,
   END OF t_cart,
```

Listing 12.25 Updates to T_CART Definition in ZCL_SHOPPING_CART

Now in the GET_CART method, add the new fields to the SELECT statement, as shown in Listing 12.26.

```
METHOD get_cart.
   "using new Open SQL
   SELECT zcart~product, description, quantity, uom
   FROM zcart
   INNER JOIN zproduct
   ON zcart~product = zproduct~product
   INNER JOIN zproduct_text
   ON zcart~product = zproduct_text~product
   WHERE zproduct_text~language = @sy-language
   AND zcart~customer = @customer
   INTO TABLE @rt_cart.
   "using old Open SQL
   SELECT zcart~product description quantity uom
   INTO TABLE rt_cart.
   FROM zcart
   INNER JOIN zproduct
   ON zcart~product = zproduct~product
   INNER JOIN zproduct_text
   ON zcart~product = zproduct_text~product
```

Listing 12.26 Updating GET_CART Method
WHERE zproduct_text-language = sy-langu
AND zcart~customer = customer.

ENDMETHOD.

Listing 12.26 Updated GET_CART Method in ZCL_SHOPPING_CART

Now, press the Activate button to activate ZCL_SHOPPING_CART. The ZCL_SHOPPING_CART class is now ready to use quantities. In the next section, we’ll update the ABAP programs to use the new price and quantity fields.

12.5.3 Updating the ABAP Programs

You’re now ready to update the maintenance ABAP programs. First, let’s update the ZPRODUCT_MAINT program. In Listing 12.27, we add the additional parameters needed for price, currency, and unit of measurement, all shown in bold.

…
SELECTION-SCREEN BEGIN OF BLOCK product WITH FRAME TITLE text-001.
PARAMETERS: p_prod TYPE zproduct-product,
           p_desc TYPE zproduct_text-description
           LOWER CASE,
           p_price TYPE zproduct-price,
           p_curr TYPE zproduct-currency,
           p_uom TYPE zproduct-uom.
SELECTION-SCREEN END OF BLOCK product.

Listing 12.27 Adding New Parameters to Selection Screen

Press the Activate button to activate the program so that you can add the selection texts for the new parameters.

Next, navigate to the text elements using Goto • Text Elements • Selection Texts in SAP GUI or right-click and choose Open Others • Text Elements in Eclipse. In the Selection Texts tab, select the checkbox for DDIC Reference for all of the new parameters listed, as shown in Figure 12.32. Then click the Activate button to activate the new texts.

You can now test the updated selection screen by running the program. It should look like Figure 12.33.

Press the Activate button to activate ZPRODUCT_MAINT to save the new parameters to the database. To do this, add the bold code in Listing 12.28.

…
product-product = p_prod.
product-price = p_price.
product-currency = p_curr.
product-uom = p_uom.
product_text-product = p_prod.
product_text-description = p_desc.
...

Listing 12.28 Updated Code to Save New Parameters in ZPRODUCT_MAINT

Figure 12.32 Updated Selection Texts in ZPRODUCT_MAINT

Figure 12.33 Updated ZPRODUCT_MAINT Selection Screen
When using inline data declarations, the `SELECT` statement can be quickly updated to display the new fields, as shown in Listing 12.29.

```sql
SELECT zproduct~product, description, price, currency, uom FROM zproduct
INNER JOIN zproduct_text
ON zproduct~product = zproduct_text~product
WHERE language = @sy-langu
INTO TABLE @DATA(products).
```

Listing 12.29 Updating SELECT Statement with Inline Data Declarations

When not using inline data declarations, you need to update the type definition before you can update the `SELECT` statement, as shown in Listing 12.30.

```abap
ELSEIF p_dis = abap_true.
  TYPES: BEGIN OF t_products,
    product TYPE zproduct-product,
    description TYPE zproduct_text-description,
    price TYPE zproduct-price,
    currency TYPE zproduct-currency,
    uom TYPE zproduct-uom,
  END OF t_products.

  DATA: products TYPE STANDARD TABLE OF t_products,
       alv_table TYPE REF TO cl_salv_table.

  SELECT zproduct~product description price currency uom
  INTO TABLE products
  FROM zproduct
  INNER JOIN zproduct_text
  ON zproduct~product = zproduct_text~product
  WHERE language = sy-langu.
```

Listing 12.30 Updating SELECT Statement without Inline Data Declarations

You can now press the `Activate` button to activate the program. The `ZPRODUCT_MAINT` ABAP program is now ready to handle currencies and quantities.

Next, let’s update the `ZCART_MAINT` program to utilize the new currency and quantity options. In Listing 12.31, you can see that we added a new parameter for the quantity and added it as a parameter when calling the `ADD_PRODUCT` method. Notice that we now have to indicate which parameters we’re passing when calling the `ADD_PRODUCT` method.

```abap
ELSEIF p_add = abap_true.
  o_cart->add_product( ip_product = p_prod, ip_quantity = p_qty ).
```

Listing 12.31 Updates for Program `ZCART_MAINT`

Now, press the `Activate` button to activate these changes and make the new parameter available in the selection texts screen.

Navigate to the selection screen texts, as described earlier for the `ZPRODUCT_MAINT` program, and check the `DDIC Reference` checkbox for the new parameter. After activating the changes, the selection screen should look as shown in Figure 12.34.
The *ZCART_MAINT* program is now ready to handle quantities. Add data using both programs to test the new functionality.

### 12.6 Summary

This chapter covered some important steps and tools for working with dates, times, and timestamps. With the information covered here, you can now create ABAP applications that can accurately calculate dates and times across various time zones and consider working days to calculate important dates.

We also covered how to work with quantities and currencies in both the ABAP Dictionary and in ABAP programs and how to read and convert quantities and currencies.

Finally, we updated the shopping cart program from Chapter 10 and Chapter 11 to use quantities and currencies.

In the next chapter, you’ll learn to work with classic dynpro screens.
Contents

Preface ......................................................................................................................... 17

1 Introduction to SAP’s ERP Systems .............................................................................. 23
  1.1 What Are SAP and ABAP? ...................................................................................... 23
  1.2 Current State of SAP’s ERP Systems ................................................................. 24
  1.3 “Old” versus “New” ABAP ................................................................................. 26
  1.4 Navigating SAP ERP Systems ............................................................................. 28
    1.4.1 Overview of SAP GUI .................................................................................... 29
    1.4.2 Overview of SAP Fiori ................................................................................... 33
  1.5 Summary ............................................................................................................... 34

2 Creating Your First Program ......................................................................................... 35
  2.1 “Hello, World!” ..................................................................................................... 35
    2.1.1 Creating a New Program with Eclipse ......................................................... 35
    2.1.2 Creating a New Program in Transaction SE80 ............................................ 40
    2.1.3 Writing a “Hello, World!” Program ............................................................. 45
    2.1.4 Adding Features to Your First Program ....................................................... 46
  2.2 Summary ............................................................................................................... 49

3 ABAP 101 ..................................................................................................................... 51
  3.1 Variables and Constants in ABAP ......................................................................... 51
    3.1.1 The DATA Keyword ..................................................................................... 51
    3.1.2 Numeric Data Types .................................................................................... 53
    3.1.3 Character-like Data Types ......................................................................... 55
# Contents

3.1 Boolean ................................................................. 57
3.2 Structures and Internal Tables ............................... 58
3.3 Inline Data Declarations ...................................... 61
3.4 Constants ............................................................ 62
3.5 System Fields and Predefined Constants .................. 63

3.2 Arithmetic and Basic Math Functions ....................... 65
3.2.1 Arithmetic Operations ....................................... 65
3.2.2 Math Functions ............................................... 68
3.2.3 CLEAR Command ........................................... 70

3.3 Flow Control ......................................................... 72
3.3.1 IF Statements ................................................. 72
3.3.2 CASE Statements ........................................... 76
3.3.3 DO Loops ....................................................... 79
3.3.4 WHILE Loops .................................................. 79

3.4 Selection Screen Programming ............................... 80
3.4.1 PARAMETER .................................................. 81
3.4.2 SELECT-OPTIONS ......................................... 85
3.4.3 SELECTION-SCREEN ....................................... 88
3.4.4 BLOCK .......................................................... 88
3.4.5 Other Selection Screen Elements ....................... 91

3.5 Event Blocks .......................................................... 92
3.5.1 INITIALIZATION ............................................. 93
3.5.2 AT SELECTION-SCREEN ................................. 93
3.5.3 START-OF-SELECTION ................................. 95

3.6 Formatting Code ..................................................... 95
3.7 Comments ............................................................. 97
3.7.1 Common Commenting Mistakes ......................... 98
3.7.2 Using Comments while Programming ................. 100

3.8 Debugging Basics ................................................... 100
3.8.1 Program to Debug ........................................... 101
3.8.2 Breakpoints in Eclipse .................................... 103
3.8.3 Breakpoints in the SAP GUI ....................... 108
3.8.4 Watchpoints in Eclipse ................................... 113
3.8.5 Watchpoints in SAP GUI ............................... 116

3.8.6 Debugging Basics ............................................. 100
3.8.7 Debugging Basics ............................................. 101
3.8.8 Debugging Basics ............................................. 103
3.8.9 Debugging Basics ............................................. 108
3.8.10 Debugging Basics .......................................... 113
3.8.11 Debugging Basics ........................................... 116

3.8.12 Debugging Basics ........................................... 116
3.8.13 Debugging Basics ........................................... 117
3.8.14 Debugging Basics ........................................... 118
3.8.15 Debugging Basics ........................................... 119

3.8.16 Debugging Basics ........................................... 120
3.8.17 Debugging Basics ........................................... 121

4 Data Dictionary Objects ............................................. 121
4.1 What Is a Database? .............................................. 121
4.2 What Is a Data Dictionary? ..................................... 122
4.3 Database Design ................................................. 123
4.4 Navigation to ABAP Dictionary Objects ................. 127
4.5 Table, Data Element, and Domain ......................... 129
4.6 Training Data Model and Example Definition .......... 132
4.7 Creating a New Domain ........................................ 135
4.8 Creating a New Data Element ............................... 139
4.9 Creating and Editing Tables .................................. 144
4.10 Documentation .................................................... 161
4.11 Viewing Data in the Database Tables .................... 162
4.12 Table Maintenance Dialogs ................................. 170

3.9 Tying It All Together ............................................. 117
3.9.1 The Problem .................................................. 118
3.9.2 The Solution .................................................. 119

3.10 Summary ............................................................. 120
3.11 Viewpoints in SAP GUI ........................................ 169

4.11.1 Data Preview in Eclipse ................................. 162
4.11.2 Data Browser in SAP GUI ............................. 164
4.12.1 Generating Maintenance Dialog for a Table ........ 170
4.12.2 Table Maintenance Transaction SM30 ............. 172
5  Accessing the Database  

5.1 SQL Console in Eclipse ........................................................................................................... 185

5.2 Reading Data ............................................................................................................................ 186
  5.2.1 SELECT Statements ............................................................................................................ 187
  5.2.2 INNER JOIN ...................................................................................................................... 188
  5.2.3 LEFT OUTER JOIN ............................................................................................................. 192
  5.2.4 FOR ALL ENTRIES IN ..................................................................................................... 193
  5.2.5 With SELECT-OPTIONS ................................................................................................. 194
  5.2.6 Aggregate Expressions in SELECT Statements .............................................................. 200
  5.2.7 New Open SQL .................................................................................................................. 201
  5.2.8 ABAP CDS Views .............................................................................................................. 203

5.3 Changing Data .......................................................................................................................... 204
  5.3.1 INSERT .............................................................................................................................. 205
  5.3.2 MODIFY/UPDATE ............................................................................................................. 206
  5.3.3 DELETE ............................................................................................................................. 207

5.4 Table Locks .............................................................................................................................. 208
  5.4.1 Viewing Table Locks ........................................................................................................ 209
  5.4.2 Creating Table Lock Objects .......................................................................................... 210
  5.4.3 Setting Table Locks ........................................................................................................ 211

5.5 Performance Topics ................................................................................................................ 212

5.6 Obsolete Database Access Keywords .................................................................................... 213
  5.6.1 SELECT...ENDSELECT ................................................................................................. 214
  5.6.2 Short Form Open SQL ..................................................................................................... 215

5.7 Summary .................................................................................................................................. 216

6  Storing Data in Working Memory ................................................................................................. 229

6.1 Using ABAP Dictionary Data Types ............................................................................................ 229

6.2 Creating Data Types with the TYPE Keyword .............................................................................. 230

6.3 Field Symbols .................................................................................................................................. 231

6.4 Defining Internal Tables ........................................................................................................... 232
  6.4.1 Defining Standard Tables ................................................................................................. 233
  6.4.2 Keys in Internal Tables ...................................................................................................... 234
  6.4.3 Defining Sorted Tables ...................................................................................................... 235
  6.4.4 Defining Hashed Tables ................................................................................................... 236

6.5 Reading Data from Internal Tables ............................................................................................ 237
  6.5.1 READ TABLE .................................................................................................................... 238
  6.5.2 LOOP AT .......................................................................................................................... 239

6.6 Modifying Internal Tables .......................................................................................................... 240
  6.6.1 Inserting/Appending Rows ............................................................................................... 241
  6.6.2 Changing Rows ................................................................................................................ 242
  6.6.3 Deleting Rows .................................................................................................................... 243

6.7 Other Internal Table Operations ............................................................................................... 244
  6.7.1 Copying Table Data ......................................................................................................... 245
  6.7.2 SORT ............................................................................................................................... 246
  6.7.3 DELETE ADJACENT DUPLICATES FROM ........................................................................ 247

6.8 Which Table Should Be Used? .................................................................................................. 248

6.9 Updating ABAP Dictionary Table Type ...................................................................................... 249

6.10 Obsolete Working Memory Syntax .......................................................................................... 250
  6.10.1 WITH HEADER LINE ..................................................................................................... 251
  6.10.2 OCCURS ........................................................................................................................ 252
  6.10.3 Square Brackets .............................................................................................................. 253
  6.10.4 Short Form Table Access ............................................................................................... 254
  6.10.5 BINARY SEARCH .......................................................................................................... 255

6.11 Summary ..................................................................................................................................... 256
7 Making Programs Modular

7.1 Separation of Concerns ......................................................... 269
7.2 Subroutines ........................................................................ 272
7.3 Introduction to Object-Oriented Programming ...................... 274
  7.3.1 What Is an Object? .............................................................. 274
  7.3.2 Modularizing with Object-Oriented Programming .......... 276
7.4 Structuring Classes ............................................................... 277
  7.4.1 Implementation versus Definition ..................................... 278
  7.4.2 Creating Objects ................................................................. 278
  7.4.3 Public and Private Sections ............................................... 280
  7.4.4 Class Methods ................................................................. 282
  7.4.5 Importing, Returning, Exporting, and Changing ............ 286
  7.4.6 Constructors ................................................................. 293
  7.4.7 Recursion ...................................................................... 295
  7.4.8 Inheritance ................................................................. 297
  7.4.9 Composition ................................................................. 302
7.5 Global Classes .................................................................... 308
  7.5.1 How to Create Global Classes in Eclipse ....................... 309
  7.5.2 How to Create Global Classes in Transaction SE80 ....... 310
  7.5.3 Using the Form-Based View in Transaction SE80 .......... 312
7.6 Design Patterns .................................................................. 317
7.7 Function Modules ............................................................... 317
  7.7.1 Creating Function Groups and Modules in Eclipse ........... 318
  7.7.2 Creating Function Groups in Transaction SE80 ............. 322
  7.7.3 Calling Function Modules ............................................... 326
7.8 Summary ........................................................................... 327

8 Error Handling ................................................................... 329

8.1 SY-SUBRC .......................................................................... 329
8.2 Message Classes ................................................................ 331
  8.2.1 Displaying a Message Class ........................................... 331

9 Presenting Data Using the ABAP List Viewer ......................... 361

9.1 What Is ALV? ................................................................. 361
9.2 Report Example Using an SALV Table .................................. 366
9.3 Report Example Using SALV Tree ...................................... 371
9.4 ALV with Integrated Data Access ........................................ 380
9.5 Outdated ALV Frameworks ................................................ 383
  9.5.1 REUSE_ALV_GRID_DISPLAY Function Module ........... 383
  9.5.2 CL_GUI_ALV_GRID Class .............................................. 385
9.6 Summary ........................................................................... 386

10 Creating a Shopping Cart Example ..................................... 387

10.1 The Design ...................................................................... 388
  10.1.1 The Database ................................................................. 389
  10.1.2 The Global Class ............................................................. 390
  10.1.3 The Access Programs ..................................................... 390
10.2 Database Solution ............................................................ 392
  10.2.1 Data Elements ................................................................. 392
  10.2.2 Transparent Tables ......................................................... 397
10.3 Accessing the Database Solution ........................................................... 405
10.4 Creating a Message Class for the Solution ......................................... 415
10.5 Creating Classic Screens for the Solution ........................................... 416
  10.5.1 Product Maintenance Program .................................................... 417
  10.5.2 Shopping Cart Maintenance Program ........................................ 423
10.6 Summary ........................................................................................................... 429

11 Working with Strings and Texts 431
11.1 String Manipulation ..................................................................................... 431
  11.1.1 String Templates .............................................................................. 432
  11.1.2 String Functions .............................................................................. 435
11.2 Text Symbols ................................................................................................... 437
  11.2.1 Creating Text Symbols ................................................................. 437
  11.2.2 Translating Text Symbols ............................................................... 442
11.3 Translating Data in Tables ......................................................................... 445
11.4 Translating Messages .................................................................................. 451
11.5 Obsolete String and Text Commands ....................................................... 453
11.6 Updating the Shopping Cart Example ....................................................... 454
  11.6.1 Updating the Database ................................................................ 454
  11.6.2 Using the Text Table ...................................................................... 460
11.7 Summary ........................................................................................................... 463

12 Working with Dates, Times, Quantities, and Currencies 465
12.1 Dates ................................................................................................................. 465
  12.1.1 Date Type Basics .............................................................................. 466
  12.1.2 Factory Calendars .............................................................................. 469
12.2 Times ................................................................................................................. 477
  12.2.1 Calculating Time .............................................................................. 477
  12.2.2 Timestamps ....................................................................................... 478
  12.2.3 SY-UZEIT (System Time versus Local Time) .................................... 482
12.3 Quantities .......................................................................................................... 482
  12.3.1 Quantity Fields in Data Dictionary .................................................. 484
  12.3.2 Converting Quantities ...................................................................... 485
12.4 Currencies .......................................................................................................... 486
  12.4.1 Currency Fields in Data Dictionary .................................................. 487
  12.4.2 Converting Currencies ...................................................................... 488
12.5 Updating the Shopping Cart Example ......................................................... 490
  12.5.1 Updating the Database ................................................................ 491
  12.5.2 Updating the Global Class ............................................................... 497
  12.5.3 Updating the ABAP Programs .......................................................... 500
12.6 Summary ........................................................................................................... 504

13 User Interface Technologies 505
13.1 Working with Classic Dynpro Screens ....................................................... 505
  13.1.1 Dynpro Screen Events .................................................................... 506
  13.1.2 Creating a Dynpro Screen .............................................................. 508
13.2 Modern UI Technologies .............................................................................. 528
  13.2.1 Web Dynpro ..................................................................................... 529
  13.2.2 SAP Screen Personas ...................................................................... 531
  13.2.3 SAP Gateway, SAPUI5, and SAP Fiori .......................................... 532
13.3 Summary ........................................................................................................... 534
# Contents

## 14  Working with ABAP Professionally  

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>14.1  ABAP System Architecture Basics</td>
<td>535</td>
</tr>
<tr>
<td>14.2  Transport Management</td>
<td>539</td>
</tr>
<tr>
<td>14.3  Authorization Concept</td>
<td>550</td>
</tr>
<tr>
<td>14.3.1  Authorization Object</td>
<td>550</td>
</tr>
<tr>
<td>14.3.2  Performing an Authorization Check in an ABAP Program</td>
<td>560</td>
</tr>
<tr>
<td>14.4  Working with Files</td>
<td>566</td>
</tr>
<tr>
<td>14.4.1  Files on the Presentation Server</td>
<td>567</td>
</tr>
<tr>
<td>14.4.2  Files on the Application Server</td>
<td>577</td>
</tr>
<tr>
<td>14.5  ABAP Development Guidelines</td>
<td>582</td>
</tr>
<tr>
<td>14.6  Maintaining Documentation</td>
<td>586</td>
</tr>
<tr>
<td>14.7  ABAP Unit Tests</td>
<td>593</td>
</tr>
<tr>
<td>14.8  Background Jobs</td>
<td>602</td>
</tr>
<tr>
<td>14.9  Tools of the Trade</td>
<td>611</td>
</tr>
<tr>
<td>14.10 Backward Compatibility and Dealing with Legacy Code</td>
<td>621</td>
</tr>
<tr>
<td>14.11 Summary</td>
<td>623</td>
</tr>
</tbody>
</table>

## Appendices

<table>
<thead>
<tr>
<th>Appendix</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A  Preparing Your Development Environment</td>
<td>627</td>
</tr>
<tr>
<td>B  Interfaces, Forms, Enhancements</td>
<td>645</td>
</tr>
<tr>
<td>C  Differences Between ABAP Versions</td>
<td>661</td>
</tr>
<tr>
<td>D  Other Resources</td>
<td>667</td>
</tr>
<tr>
<td>E  The Authors</td>
<td>671</td>
</tr>
</tbody>
</table>

Index | 673 |
Deletion anomaly .......... 124
Delivery and maintenance 146
DEQUEUE .................. 216, 220, 223
Design patterns .......... 317
DEV environment .......... 536
Development guidelines .... 582
Development standards .... 42
DevOps .................... 535
DIV ............................ 66
DIVIDE ....................... 68
DO ................................. 79
Documentation ............. 161, 586
Domain ................. 121, 130, 135, 394, 400
range ...................... 137
DSAG guidelines ........... 584
Dynamic breakpoints ...... 106
Dynamic program .......... 505
Dynpro ........................ 505
create screen ................. 508
Screen events ............... 506
screen flow diagram .......... 507
Enhancement category ...... 153, 160
Enhancement framework .... 659, 662
Enhancements .............. 654
ENQUEUE .................... 220
Enterprise resource planning ... 23
Entity ......................... 123
Entity relationship diagrams
   (ERDs) ...................... 123
Entry help/check ............. 159
ERD ............................ 123, 389, 455, 490
crow’s foot notation, relationship
   indication .................. 126
normalized .................... 125
Event .......................... 92
AT SELECTION SCREEN .... 93
INITIALIZATION ............ 93
START-OF-SELECTIO ........ 95
Exception .................... 341
SOO SAP internal server error .... 343
CATCH BEFORE UNWIND .... 350
CATCH INTO ................. 349
categories ..................... 350
custom exception classes .... 350
function modules ............. 356
non-class based exceptions .... 355
RAISE EXCEPTION TYPE .... 349
resumable exceptions ......... 349
RESUME ....................... 350
short dump .................. 342
ST22 ......................... 344
TRY ... CATCH ............... 348
unhandled .................... 342
Exclusive lock ............... 215
Execute program ............ 610
Execution stack ............. 105
EXIT ............................. 79
Extended Program Check ...... 611, 612
File .............................. 645
interfaces .................. 645
logical name ................. 582
on application server .......... 577
on presentation server ....... 567
transfer protocol ............ 645
Flight model ................ 169
load data .................... 620
Floor ......................... 69
Flow control ................. 72
Flow logic .................... 516
FORM ... ENDFORM ......... 272
 FORM CHANGING ............. 274
 Format your code .......... 649
 frac ............................... 69
 Fully buffered ................. 225
 Function group ............... 170, 318, 322
 Function modules ............. 318, 324
 CALL FUNCTION ............. 326
 parameters .................... 321
 IF ................................ 72
 AND ............................. 73
 ELSE ............................. 73
 ELSEIF .......................... 75
 OR ................................. 73
IF statement operators ....... 76
INCLUDE ...................... 517
Initial value ................. 51, 71, 147
INITIALIZATION ............ 92
Inline data declaration .... 67, 189, 430, 427
INNER JOIN .................. 192, 414
Input help ..................... 178
INSERT ......................... 210, 250, 263
hashed table .................. 252
INDEX ......................... 250
LINES FROM .................. 251
sorted table .................. 252
Insert anomaly ............... 125
Integrated development environments (IDEs) ........ 627
Internal tables .... 59, 229, 261
   hashed key .................. 237
   hashed table ................. 240, 261
   key ............................. 236
   secondary keys .............. 237
   SORO ........................... 258
   sorted key ..................... 237
   sorted table ................. 239, 261
   standard key ................ 237
   standard table ............... 234, 261
   INTO ............................ 206
   INTO TABLE ................. 206
   ipow ........................... 69, 102
   ITAB_ILLEGAL_SORT_ORDER .... 240
JavaScript ...................... 34
K ................................ 147
Key fields/candidate .......... 157
Keyword ...................... APPEND ............. 251
AS ............................... 194
L ................................ 120
Identifying relationship ..... 126
IDES solution ............... 627
IDocs ........................... 647
Eclipse ........................ 36
ARAP perspective ............. 36
ARAP projects ............... 636
create program ............... 35
create project ............... 39
dynpro ......................... 509
format code .................. 96
open project .................. 38
outline ......................... 38
perspectives ................. 634
problems ..................... 38
project explorer ............ 38
SQL Console ................. 186, 188
versus Transaction SRO .... 38
views ............................ 36
workspaces .................. 633
Electronic Data Interchange (EDI) .... 647
ELSE ............................. 73
END OF ......................... 58
ENDDO .......................... 79
ENDIF ............................ 72
f ................................ 53
Field ............................ 384
   catalog ..................... 384
   label ........................ 141
   Field symbols ............... 233
   ASSIGN ....................... 233
G ................................ 120
Generic area buffered ........ 225
GETWA NOT_ASSIGNED ....... 233
Global class .................. 389
Global variables ............. 272
Hash board .................... 240, 263
High-level documentation ..... 586
HTML5 .......................... 34
JavaScript ........................ 34
JavaScript ...................... 34
K ................................ 147
Key fields/candidate .......... 157
Keyword ...................... APPEND ............. 251
AS ............................... 194
### L
- **Learning path** ........................................... 18
- **LEFT OUTER JOIN** .................................... 197
- **Legacy ABAP** ........................................... 26
- **LENGTH** ................................................... 55
- **Local variables** ...................................... 272
- **Lock objects** ......................................... 218
- **Log data options** ..................................... 151
- **Logical storage parameters** ....................... 150
- **Loop** ..................................................... 79
- **infinite loop** .......................................... 79
- **LOOP AT** .................................................. 247
- **hashed table** ......................................... 249
- **WHERE** ................................................... 248
- **Maintenance dialog** .................................. 170
- **MANDT** ..................................................... 134, 156, 157, 210, 536
- **Math functions** ........................................ 68
- **syntax** ...................................................... 69
- **Message class** ......................................... 331, 359
- **create** ..................................................... 333
- **long text** ............................................... 337
- **MESSAGE** ................................................. 336
- **WITH** ....................................................... 339
- **Method** ..................................................... 432
- **Middleware** .............................................. 645
- **MOD** ......................................................... 66
- **Modularities** .......................................... 127
- **MODIFY** ................................................... 241
- **sorted table** ............................................ 254
- **MODIFY TABLE** ....................................... 253
- **INDEX** ...................................................... 254
- **WHERE** ..................................................... 253
- **Modularization** ....................................... 269
- **Module pool** ............................................ 306
- **MOVE** ....................................................... 65
- **MOVE-CORRESPONDING** .......................... 258
- **Multi-paradigm programming** ................... 27
- **MULTIPLY** ............................................... 68

### N
- **New ABAP** ............................................... 26
- **New Open SQL** ......................................... 205
- **No input history** ..................................... 142
- **Nonidentifying relationship** ....................... 126
- **NULL** ....................................................... 147
- **Numeric data types** .................................. 53

### O
- **Object-oriented programming** ..................... 27, 274
- **attributes** .............................................. 281
- **CHANGING** ............................................... 286, 292
- **class** ....................................................... 275, 277, 278
- **class definition** ....................................... 278
- **class implementation** ............................... 278
- **composition** .......................................... 302
- **CONSTRUCTOR** ......................................... 294
- **CREATE OBJECT** ....................................... 278
- **CREATE PUBLIC** ....................................... 309
- **EXPORTING** ........................................... 286, 291
- **FINAL** ....................................................... 309
- **global classes** ....................................... 308, 309, 390
- **IMPORTING** ............................................ 286, 289
- **INHERITING FROM** ................................... 298
- **INTERFACES** .......................................... 305
- **method** ..................................................... 275, 282, 283
- **object** ...................................................... 274
- **OPTIONAL** ................................................. 290
- **private section** ....................................... 280
- **protected section** .................................... 280, 301
- **public section** ......................................... 280
- **READ-ONLY attributes** ............................ 281
- **REDEFINITION** ......................................... 300
- **RETURNING** ............................................ 286, 287
- **returning parameters** ............................... 293
- **subclass** ................................................... 298
- **superclass** .............................................. 298
- **TYPE REF TO** .......................................... 278
- **One-to-many relationship** ......................... 127
- **One-to-one relationship** ............................ 127
- **Online analytical processing (OLAP)** ............ 25
- **Online transactional processing (OLTTP)** ....... 25
- **OPEN DATASET... FOR OUTPUT** .................. 579

### Open SQL
- **aggregate expressions** .............................. 28, 185
- **infinite aggregate expressions** .................. 28, 185
- **SELECT** ..................................................... 187
- **SELECT *** ................................................... 188
- **SELECT OPTIONS** ...................................... 201
- **SELECT SINGLE** ....................................... 189
- **SELECT SINGLE *** ...................................... 191
- **SELECT ... ENDSELECT** ......................... 226
- **short form** ................................................ 227
- **SUM** ........................................................ 204
- **UP TO I ROWS** .......................................... 226
- **UP TO n ROWS** .......................................... 191
- **UPDATE** ..................................................... 212
- **WHERE** ..................................................... 191
- **WHERE IS NULL** ........................................ 198
- **WHERE IN** ................................................ 201

### Operator
- **Operands** ................................................... 76
- **operator** ..................................................... 76
- **ADD** ......................................................... 68
- **DIVIDE** ..................................................... 68
- **MULTIPLY** ................................................. 68
- **Optimistic lock** ........................................ 216
- **OR** ............................................................ 73
- **Order of operations** ................................... 66
- **Origin of the input help** ............................ 148
- **Output implementation** .............................. 649
- **Output program** ........................................ 649

### P
- **Package** ..................................................... 145, 540
- **PARAMETER** ........................................... 145
- **packed numbers** ....................................... 54
- **Parameter** ................................................. 80, 81
- **_COLLECT** ................................................ 222
- **_WAIT** ....................................................... 222
S
SAIRPORT 134
SALV 364
demo programs 371
table 366
SAP Basis 25
SAP Build 391
SAP Business Suite powered by
SAP HANA 26
SAP Cloud Infrastructure Library 627
SAP ERP output implementation 649
SAP Fiori 33, 532
SAP Fiori launchpad 33
SAP Gateway 532
SAP GUI 29
application toolbar 31
breakpoints 108
developer user menu 33
favorites menu 31
system information 32
toolbar 30
SAP GUI debugger 109
SAP GUI theme 510
SAP GUI versions 31
SAP HANA 25, 33
SAP Interactive Forms by Adobe 649
SAP List Viewer (ALV) 361
SAP List Viewer with Integrated Data Access (ALV with IDA) 365
SAP NetWeaver 25
SAP R/1 24
SAP Report 242
DEMO_CALL_SELECTION_SCREEN 88
SAP S/4 HANA 649
extensibility 659
SAP S/4HANA forms 26
tables 649
SAP Screen Personas 531
SAP Solution Manager 549, 586
SAP Transport Management System (SAP TMS) 538
SAPBIC_DATA_GENERATOR 169
SAPBIC_DATAMODEL 118
SAPscript 649, 652
SAPTEXTEDIT 523
SAPUI5 33, 532
Scope 216
Screen Painter 509
Screen stack 111
Search help 149, 178
SELECT 187, 224, 414
COUNT 203
INTO CORRESPONDING 374
INTO TABLE 188
MAX 204
MIN 204
SUM 204
SELECT (Cont.)
UP TO 1 ROWS 226
UP TO n ROWS 191
SELECT INTO CORRESPONDING 374
SELECT SINGLE 189
SELECT. UP TO n ROWS 191
SELECT. ENDSELECT 226
Selection screen 201, 423
Selection text 80
Selection screen 87, 418
Selection SCREEN 80
BLOCK. WITH FRAME TITLE 423
SELECT-OPTIONS 80, 85, 86, 149, 201
high 203
low 203
option 202
Separation of concerns 269
Sessions 30
SFIGHT 145
appraoach structure 152
primary key 147
Shared lock 215
Short form 267
||| 267
LOOP AT 267
OCCURS 266
READ TABLE 267
WITH HEADER LINE 266
Short form Open SQL 227
Sign 69, 202
Signature 314
Simple Object Access Protocol (SOAP) 648
Single quotes 45
numeric variables 57
Single records buffered 224
Size category 160
Smart Forms 649
SORTED TABLE 238
Space category 151
Spool requests 609
SQL 185
START-OF-SELECTION 93
event 95
Q
QA environment 536
Quantities 482
QUAN 484
UNIT 484
UNIT_CONVERSION_SIMPLE 485
R
Radio buttons 83
RAISE 357
RAISING 351, 352
Ranges 202
RDBMS 122
READ TABLE 241, 263
ASSIGNING 241
INDEX 242
INTO 241
REFERENCE INTO 244
WITH KEY 244
Recursion 295
Relational database management system (RDBMS) 123
Relational operators 76
Remote breakpoint 110
Remote function call (RFC) 646
REPORT 39, 45
REUSE_ALV_GRID_DISPLAY 383
RFC connection 646
RICEF 611
Runtime Analysis 611
Screen stack 111
Screen Painter 509
Scope 216
Screen stack 111
Search help 149, 178
SELECT 187, 224, 414
COUNT 203
INTO CORRESPONDING 374
INTO TABLE 188
MAX 204
MIN 204
SUM 204
SELECT (Cont.)
UP TO 1 ROWS 226
UP TO n ROWS 191
SELECT INTO CORRESPONDING 374
SELECT SINGLE 189
SELECT. UP TO n ROWS 191
SELECT. ENDSELECT 226
Selection screen 201, 423
Selection text 80
Selection screen 87, 418
Selection SCREEN 80
BLOCK. WITH FRAME TITLE 423
SELECT-OPTIONS 80, 85, 86, 149, 201
high 203
low 203
option 202
Separation of concerns 269
Sessions 30
SFIGHT 145
appraoach structure 152
primary key 147
Shared lock 215
Short form 267
||| 267
LOOP AT 267
OCCURS 266
READ TABLE 267
WITH HEADER LINE 266
Short form Open SQL 227
Sign 69, 202
Signature 314
Simple Object Access Protocol (SOAP) 648
Single quotes 45
numeric variables 57
Single records buffered 224
Size category 160
Smart Forms 649
SORTED TABLE 238
Space category 151
Spool requests 609
SQL 185
START-OF-SELECTION 93
event 95
String ............................................. 45, 56, 61
ALIGN ........................................... 433
chaining strings .......................... 433
concat_line_of ........................... 435
CONDENSE ................................ 435
DECIMALS ................................ 435
PAD ............................................. 433
SIGN ........................................... 434
string functions ......................... 435
string literals ............................. 435
string literals ................................ 435
string template .......................... 432
SUB ................................................................ 435
substring ...................................... 436, 466
substring_before ........................ 436
substring_from .......................... 436
substring_to ............................. 436
timestamp formatting ..................... 480
WIDTH ............................................. 433
Structure ..................................... 174, 210, 235
Subpackages ............................... 540
Subroutines ................................ 272
PERFORM ..................................... 272
Switch Framework ....................... 662
xy-batch ...................................... 64
xy-dalio ....................................... 64
xy-datum ..................................... 64
xy-index ...................................... 64
XY ................................................................ 63
System fields ............................... 47, 63
editing ........................................ 64
xy-batch ...................................... 64
xy-dalio ....................................... 64
xy-datum ..................................... 63, 64
xy-index ...................................... 63
xy-subrc ..................................... 64
xy-tabix ....................................... 64
xy-timlo ...................................... 64
xy-zeit ....................................... 64
xy-subrc ................................. 64, 246, 329, 357
xy-tabix ....................................... 64, 246, 249
xy-timlo ...................................... 64
xy-zeit ....................................... 64
Table  .......................................... 514
Table custom transparent .............. 154
normalized .................................. 134
SBOOK ......................................... 134
SCARR ......................................... 134
SCCUSTOM .................................. 134
SFIGHT ......................................... 134
SPFL ............................................. 134
text table ...................................... 445
view single row ............................ 189
Table configuration ...................... 144
Table locks ................................... 214
create deadlock .......................... 214
mode ........................................... 217
parameter ................................... 217
set ............................................... 220
view ............................................ 216
Table type ............................... 174, 189, 232, 263
Tables ....................................... 227
client-dependent .......................... 537
table-dependent .......................... 537
Technical settings ....................... 150, 160, 400, 404
Template  ................................ 436
ABAP Editor ................................ 596
Text editor .................................. 523
Text formats ................................ 567
Text symbols ......................... 419, 424, 437
comparison .................................. 439
translation .................................. 442
Third normal form ...................... 125
table ........................................... 125
Transaction SE11 .......................... 123
Transaction SE63 .......................... 448
transaction editor ......................... 448
Transaction SE80 ......................... 40
transaction editor ......................... 448
Transaction SE80 (Cont.) ................. 40
create screen ............................ 509
edit program ............................. 44
format code ................................ 96
form-based view ......................... 312
IDE settings .............................. 44
pretty print ............................... 96
program attributes ..................... 41
source code view ....................... 312
Transactions ................................ 29
ABAPNCD ................................. 55, 583
ALII ........................................... 577
CMOD ........................................... 655
CUNI ........................................... 483
FILE ........................................... 582
OB08 ........................................... 487
OY03 ........................................... 486
SAT ............................................. 617
SCC ............................................. 538
SE10 .......................................... 546
SE12 .............................. 122, 128, 129, 135, 139, 142, 144, 154, 170, 174, 176, 216, 218, 264, 393, 395, 397, 400, 401, 455, 484, 487, 491
SE16N ........................................ 167
SE38 ........................................... 169
SE39 ........................................... 169
SE40 ........................................... 446, 459, 589
SE71 ........................................... 652
SE80 ........................................... 40, 408
SE9 ............................................. 331, 333
SEG ........................................... 649
SM30 ........................................... 170, 172
SM36 ........................................... 603
SM57 ........................................... 606
SM59 ........................................... 647
SM62 ........................................... 609
SMDAEMON ................................ 665
SMOD ........................................... 655
SOAMANAGER ............................. 649
SP01 ........................................... 609
ST01 ........................................... 563
ST02 ........................................... 344, 350
STMS ........................................... 539
SU01 ........................................... 551
SU21 ........................................... 551
SU3 ............................................. 474
SUIM (User Information System) ........ 564
VOFM ......................................... 675
Translate .................................... 454
Transparent table ......................... 121, 123, 144, 154, 210, 235, 256, 388, 389
currency/quantity fields ............... 484, 487, 488
Transport .................................... 538
view ............................................. 544
Transport management .................. 539
Transport organizer ..................... 546
Transport request ....................... 542
Transport route ......................... 539
trunc ......................................... 69
TYPE ............................................... 51, 57, 74, 97, 231
U ................................................................ 68
UBTRACT .................................. 408
Units of measurement .................. 483
UPDATE ................................... 212
Update anomaly ......................... 124
User interface (UI) ....................... 80, 505
USING ........................................... 273
V ................................................................ 51
VALUE ........................................... 51, 71
Value range .................................. 137
Variable  ...................................... 65
assign value to chain together ........ 52
Variable naming .......................... 52
Variables .................................... 51
W .................................................. 101, 113
Watchpoints ............................... 101, 113
Web Dynpro ABAP ......................... 529
Web Services ............................. 648
WHERE ........................................ 191, 198, 199
WHERE IS NULL ...................... 198
WHERE IN .............................. 201
WHILE loops ............................... 79
First-hand knowledge.

We hope you have enjoyed this reading sample. You may recommend or pass it on to others, but only in its entirety, including all pages. This reading sample and all its parts are protected by copyright law. All usage and exploitation rights are reserved by the author and the publisher.

Brian O’Neill is an ABAP developer with more than 10 years of experience working across different SAP ERP modules and custom applications. His experience includes working in various IT positions, from SAP support to ABAP development, while working in the IT department of SAP customer IGT (International Game Technology). Brian has also worked as an SAP Mobility consultant with IBM and most recently has been working as an independent consultant focusing on SAP mobility projects. Brian has a bachelor’s degree in computer information systems from California State University, Chico, and a master’s degree in information systems from the University of Nevada, Reno.

Jelena Perfiljeva is a senior ABAP developer at IBM Global Finances. She started her SAP career in 2005 and has been involved in SAP rollouts, upgrades, and support for several US-based companies. Jelena is an SAP Mentor alumna and was chosen SAP Community (formerly known as SCN) Member of the Month in April 2013. She was the recipient of both the SAP Community Citizenship and Appreciation awards in 2017. In 2018, she received the SAP Community Hero award. In 2016, Jelena published the book What on Earth is an SAP IDoc? which explains the nature of IDoc interfaces in SAP ERP. Jelena is a speaker at SAP conferences, including SAP TechEd in Las Vegas in 2018 and Mastering SAP Technologies in Melbourne in 2014. She is originally from Riga, Latvia and received a master’s degree in Computer Science and Economics from the Latvian University.

ABAP: An Introduction
684 pages, 2nd, edition 2020, $69.95

www.sap-press.com/4955