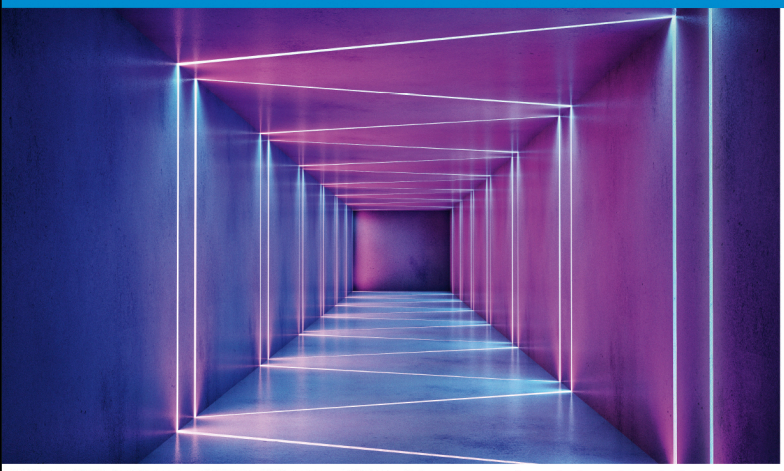
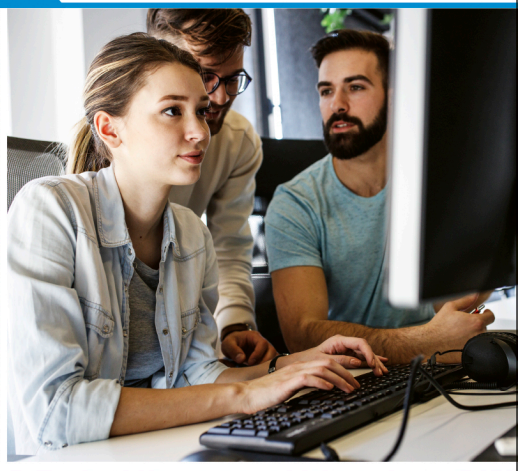
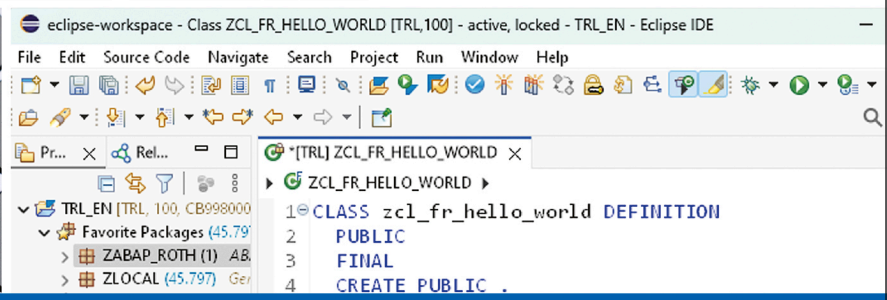
SAP PRESS







Einstieg in ABAP®

Für Ausbildung und Beruf

- Ihre Einführung in die moderne ABAP-Entwicklung
- Alle Grundlagen der Cloud- und On-Premise-Entwicklung
- Praxisnahe Beispiele für die direkte Anwendung

Mit Programmierbeispielen zum Download

Felix Roth

Rheinwerk Publishing

Kapitel 4
ABAP Dictionary

Das ABAP Dictionary ist die zentrale Stelle für alle im SAP-System vorhandenen Datendefinitionen – von einfachen Datenelementen über komplexe Strukturen bis hin zu vollständigen Datenbanktabellen. Damit bildet es die Grundlage für nahezu jede ABAP-Entwicklung.

In Kapitel 2, »Erste Schritte«, habe ich Ihnen gezeigt, wie Sie erste Felder, Konstanten, Strukturen und lokale Typen im ABAP-Quelltext auf Basis von eingebauten Datentypen wie `i` oder `string` definieren. In der Praxis wird jedoch häufig auch auf globale Datentypen, Strukturen und Datenbanktabellen zurückgegriffen, die im ABAP Dictionary definiert sind. In diesem Kapitel möchte ich Ihnen erläutern, was das ABAP Dictionary ist und wie Sie damit arbeiten.

Die Hauptaufgabe des ABAP Dictionary ist die Verwaltung der an das SAP-System angebundenen Datenbank *SAP HANA*, die unter dem System liegt. Damit ist das Dictionary eine Sammlung von Werkzeugen, um auf Tabellen auf dieser Datenbank zuzugreifen, sie zu erzeugen und zu verwalten. Dazu kann mit ABAP und ABAP SQL über das ABAP Dictionary auf die Datenbanktabellen zugegriffen werden, ohne diesen Zugriff (z. B. den Aufbau und Abbau der Verbindung) explizit orchestrieren zu müssen. Listing 4.1 zeigt einen solchen Zugriff auf die Datenbanktabelle `/DMO/FLIGHT` mit einer `SELECT`-Anweisung.

```
SELECT FROM /dmo/flight
  FIELDS *
  INTO TABLE @DATA(lt_ergebnis).
```

Listing 4.1: Selektion auf eine Datenbanktabelle

In diesem Beispiel werden alle Spalten und Zeilen dieser Datenbanktabelle in der internen Tabelle `LT_ERGEBNIS` geladen, mit der nun auf dem Applikationsserver mit ABAP weitergearbeitet werden kann. Die Arbeit mit internen Tabellen erläuterte ich Ihnen in Kapitel 6, »Mit internen Tabellen arbeiten«. Der Zugriff auf Datenbanken mit ABAP SQL wird in Kapitel 7, »Zugriff auf die Datenbank«, erläutert.

In diesem Kapitel konzentrieren wir uns zunächst auf die grundlegenden Objekte des ABAP Dictionary, auf denen u. a. alle Datenbanktabellen basieren. Zentrale Grundlage ist dabei das *zweistufige Domänenprinzip*, das im gleichnamigen Abschnitt 4.1 erläutert wird und zeigt, wie die Datenbanktabelle aus Listing 4.1 aufgebaut ist.

Anschließend lernen Sie die Anlage der wichtigsten Entwicklungsobjekte kennen:

- Domänen (siehe Abschnitt 4.2)
- Datenelemente (siehe Abschnitt 4.3)
- Datenbanktabellen (siehe Abschnitt 4.4)
- Strukturen (siehe Abschnitt 4.5)

Diese Entwicklungsobjekte zu verstehen und anlegen zu können, ist elementar, zum einen, um Quelltext von SAP zu analysieren, zum anderen, um auch größere kundeneigene Entwicklungen umsetzen zu können, die auf eigenen Datenbanktabellen basieren.

Zusammenfassend lässt sich sagen, dass das ABAP Dictionary ein zentraler Bestandteil der Entwicklungsumgebung von SAP ist. Es dient dazu, Datenstrukturen einheitlich zu definieren, zu verwalten und systemweit zur Verfügung zu stellen. Das ABAP Dictionary ist daher nicht nur ein Werkzeug, sondern besteht aus einer Sammlung von mehreren Werkzeugen, die jeweils ein Entwicklungsobjekt bearbeiten.

Das ABAP Dictionary sorgt für:

- zentrale Datenbeschreibung
- Konsistenz im gesamten System
- Wiederverwendbarkeit von Definitionen
- automatische Datenbankanpassung

4.1 Das zweistufige Domänenprinzip

Das ABAP Dictionary basiert auf einem zweistufigen Domänenprinzip, das technische und semantische Domänen vorsieht:

- Die technischen Domänen beschreiben den Wertebereich eines Felds, der durch die Angabe eines eingebauten Datentyps, der Ausgabelänge und eventueller Festwerte festgelegt wird. Im ABAP-Umfeld werden diese technischen Domänen nur *Domänen* genannt.
- Die semantischen Domänen weisen den technischen Domänen durch die Vergabe von Texten einen bestimmten Zusammenhang zu. Im ABAP-Umfeld werden diese semantischen Domänen *Datenelement* genannt.

Wie in Abbildung 4.1 dargestellt, verweist ein Feld einer Struktur oder Tabelle auf ein Datenelement. Dieses Datenelement referenziert wiederum eine Domäne, die die technischen Eigenschaften festlegt.

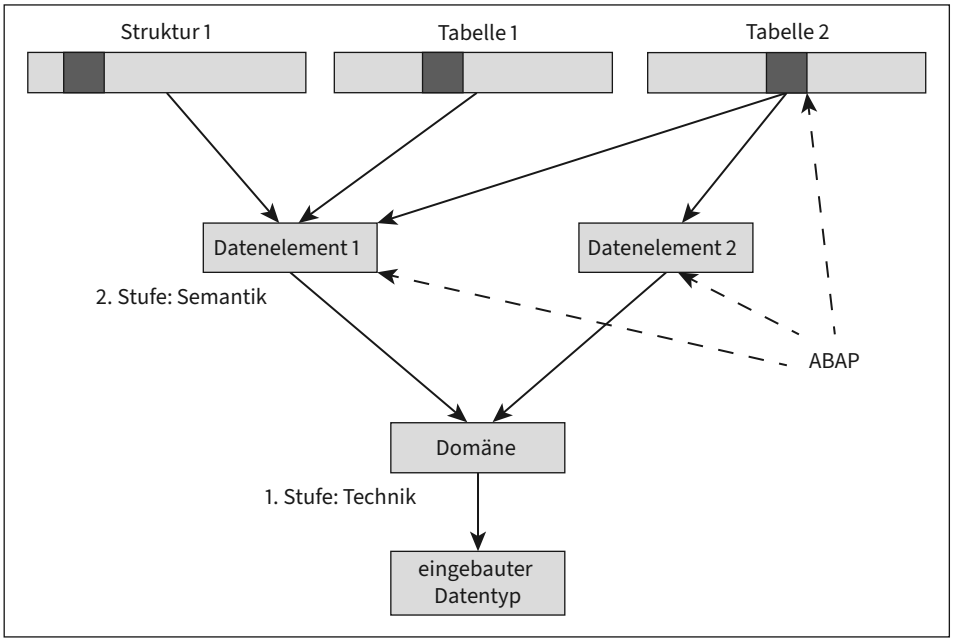


Abbildung 4.1: Das zweistufige Domänenprinzip

Eine Domäne kann demzufolge in mehreren Datenelementen verwendet werden, und ein Datenelement kann in vielen Feldern von Tabellen und Strukturen verwendet werden. Die Domäne fasst also technische Informationen über mehrere Tabellen hinweg zusammen und kann in Form von Datenelementen verschiedene Ausprägungen haben. Darüber hinaus können Sie auf die Datenelemente auch aus ABAP heraus zugreifen und z. B. eine Variable von diesem Typ mit der Anweisung `DATA` und dem Zusatz `TYPE` definieren. Es geht bei den Domänen und Datenelementen also primär um das Thema der Wiederverwendbarkeit, aber auch darum, die eingebauten Datentypen mit Zusatzinformationen anzureichern, wie z. B.:

- Suchhilfen
- verschiedene Texten
- Wertebereiche
- Konvertierungsbausteine

Abbildung 4.2 zeigt beispielhaft die Datenbanktabelle `/DMO/FLIGHT`. Die Tabelle besteht aus neun Feldern. Während für das Feld `client` der eingebaute Datentyp `clnt` verwendet worden ist, basieren die anderen acht Felder jeweils auf einem Datenelement.

```
[TRL] /DMO/FLIGHT
 1 @EndUserText.label : 'Flight Reference Scenario: Flight'
 2 @AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
 3 @AbapCatalog.tableCategory : #TRANSPARENT
 4 @AbapCatalog.deliveryClass : #A
 5 @AbapCatalog.dataMaintenance : #RESTRICTED
 6 define table /dmo/flight {
 7
 8   key client        : abap.clnt not null;
 9   key carrier_id    : /dmo/carrier_id not null;
10   key connection_id : /dmo/connection_id not null;
11   key flight_date   : /dmo/flight_date not null;
12   @Semantics.amount.currencyCode : '/dmo/flight.currency_code'
13   price             : /dmo/flight_price;
14   currency_code     : /dmo/currency_code;
15   plane_type_id     : /dmo/plane_type_id;
16   seats_max         : /dmo/plane_seats_max;
17   seats_occupied    : /dmo/plane_seats_occupied;
18
19 }
```

Abbildung 4.2: Definition der Datenbanktabelle /DMO/FLIGHT

Für das Feld `CARRIER_ID` aus Zeile 9 ist beispielsweise das Datenelement `/DMO/CARRIER_ID` angegeben. Wenn Sie den Cursor auf den Namen des Datenelements (`/DMO/CARRIER_ID`) setzen und die Taste F3 drücken (alternativ Strg + Mausklick oder den Eintrag **Navigate to** über das Kontextmenü wählen), wird Ihnen das Datenelement als eigene Maske wie in Abbildung 4.3 dargestellt.

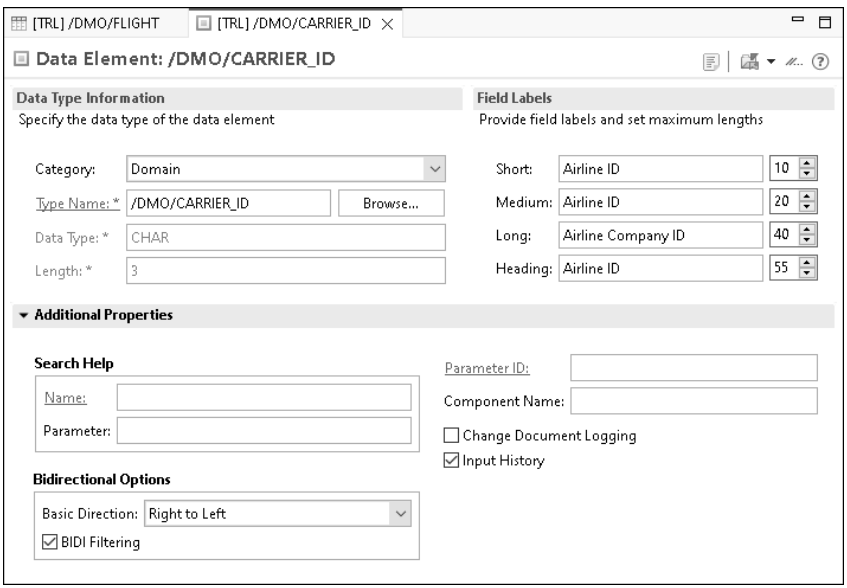


Abbildung 4.3: Definition des Datenelements /DMO/CARRIER_ID

In dieser Ansicht sind alle Eigenschaften des Datenelements dargestellt. Unter anderem erkennen Sie, dass dieses Datenelement auf der gleichnamigen Domäne `/DMO/CARRIER_ID` aufbaut. Über einen Klick auf das Feld **Type Name** gelangen Sie zur Definition der Domäne (siehe Abbildung 4.4).

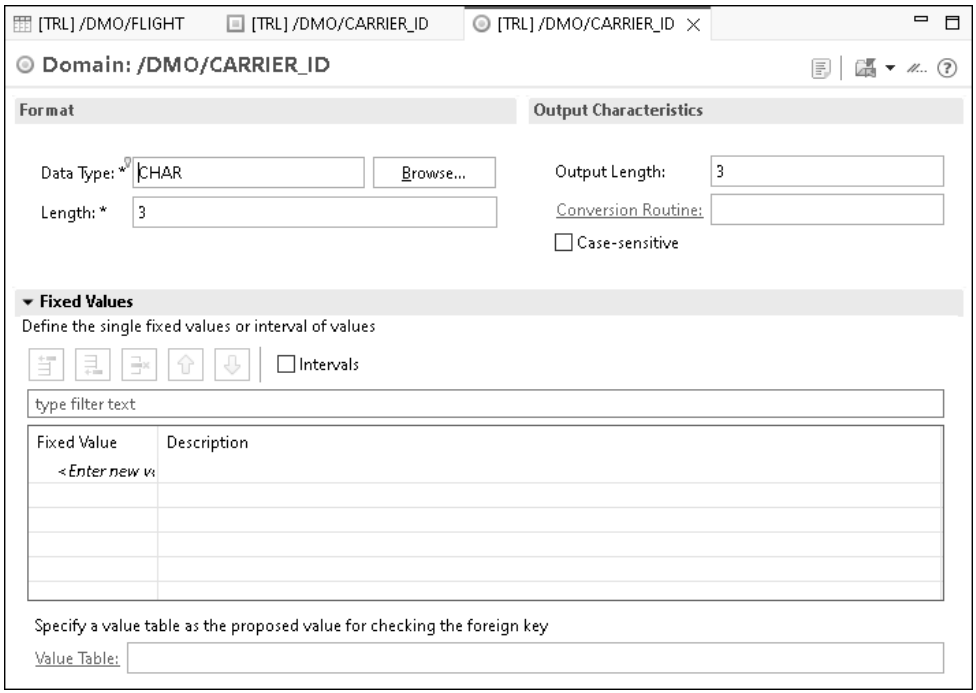


Abbildung 4.4: Definition der Domäne /DMO/CARRIER_ID

In der Definition ist letztlich der eigentliche eingebaute Datentyp `CHAR` (Feld **Data Type**) mit der Länge 3 (Feld **Length**) hinterlegt. Weitere Informationen, wie eine Konvertierungsroutine (Feld **Conversion Routine**), ein Wertebereich (Bereich **Fixed Values**) oder eine Wertetabelle (Feld **Value Table**) sind in diesem Beispiel nicht hinterlegt. Auch die Groß- und Kleinschreibung (Feld **Case-sensitive**) ist für diese Domäne nicht aktiviert, sodass die Inhalte nur in Großschreibung gespeichert werden.

Diese Domäne kann in beliebig vielen anderen Datenelementen wiederverwendet werden. Die Datenelemente können dann wiederum in weiteren Strukturen, Datenbanktabellen oder in Datendefinitionen mit ABAP verwendet werden.

Im nächsten Abschnitt zeige ich Ihnen, wie Sie Datenelemente und Domänen anlegen und diese in Strukturen und Datenbanktabellen verwenden.

4.2 Domänen anlegen

Um eine Domäne anzulegen, gehen Sie im Menü der ABAP Development Tools für Eclipse auf **File • New • ABAP Repository Object** und wählen im sich öffnenden Pop-up-Fenster aus Abbildung 4.5 im Ordner **Dictionary** den Eintrag **Domain** aus. Zusätzlich können Sie im Eingabefeld **type filter text** den Begriff »Domain« eingeben, um die Liste zu filtern. Klicken Sie anschließend auf **Next >**.

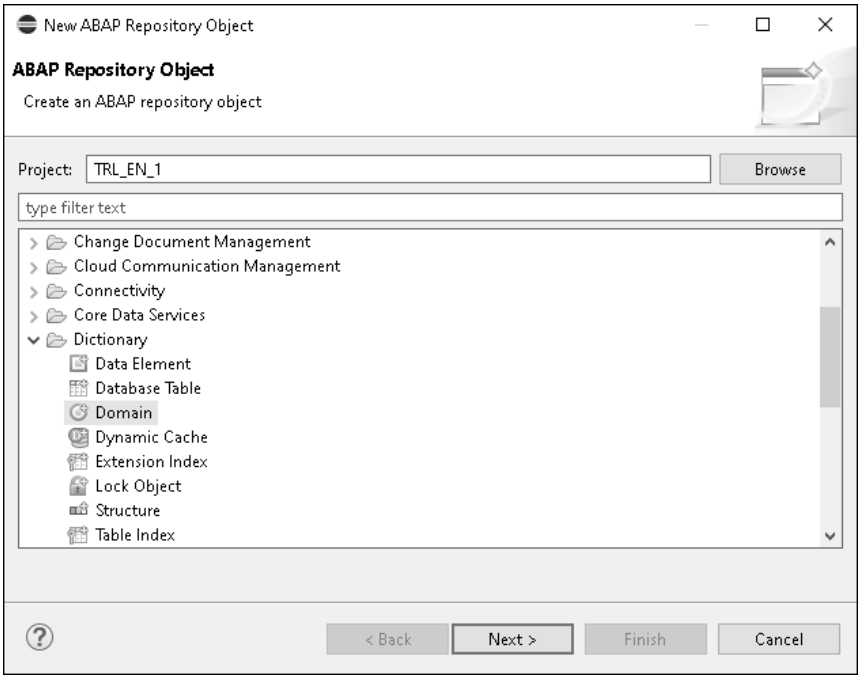


Abbildung 4.5: Dictionary-Entwicklungsobjekt anlegen

Im nächsten Schritt geben Sie im Feld **Package** Ihr Paket aus Abschnitt 2.2.1, »Paket anlegen«, ein (siehe Abbildung 4.6). Vergeben Sie zusätzlich im Feld **Name** einen Namen (hier »ZD_EMP_NAME«), gegebenenfalls mit Ihrem Namenskürzel, und im Feld **Description** eine Beschreibung (hier »Mitarbeitername«). Klicken Sie anschließend auf **Next >**.

Nach dem Bestätigen öffnet sich die Eigenschaftsseite der Domäne aus Abbildung 4.7. Hier können Sie nun in den Feldern **Data Type** und **Length** den Datentyp und die gewünschte Länge eingeben. In unserem Beispiel soll die Domäne vom Typ CHAR sein und die Länge 60 haben. Zusätzlich wird über die Checkbox **Case-sensitive** festgelegt, ob der Inhalt in Groß- und Kleinschreibung sein kann.

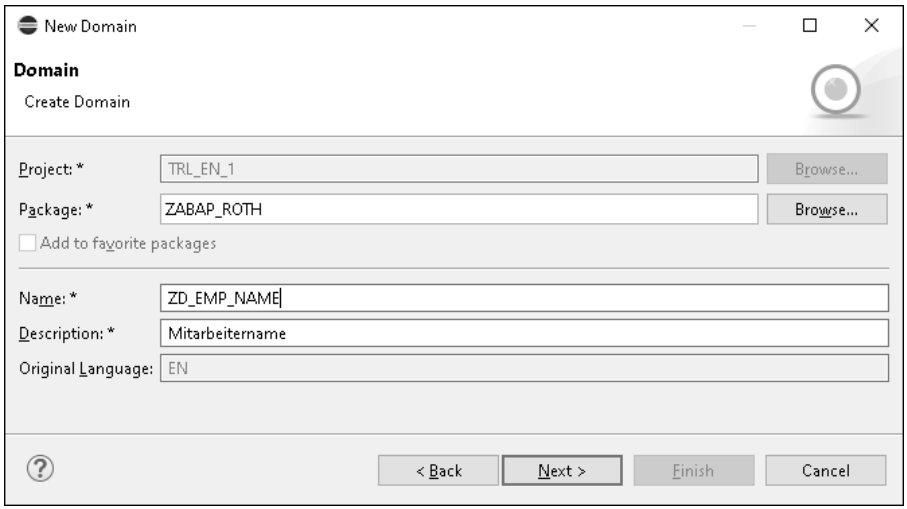


Abbildung 4.6: Namen und Beschreibung einer Domäne angeben

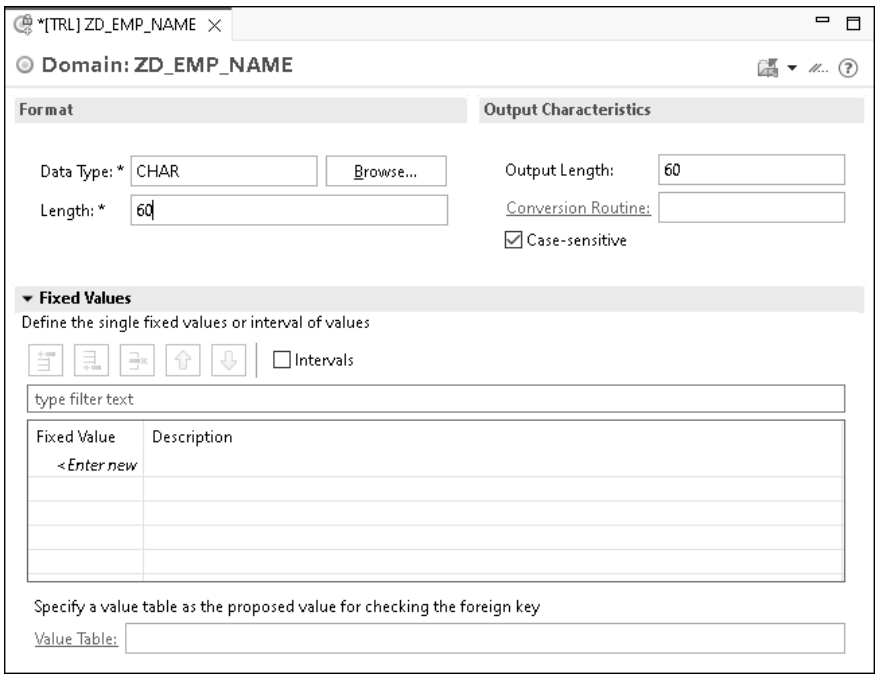


Abbildung 4.7: Eigenschaften einer Domäne

Neben diesen grundlegenden Eigenschaften können Sie im Feld **Conversion Routine** eine Konvertierungsroutine angeben.

Konvertierungsroutinen

Der Zweck einer Konvertierungsroutine im ABAP Dictionary ist es, Daten automatisch zwischen interner und externer Darstellung umzuwandeln. So wird beispielsweise die Materialnummer 123456 intern auf der Datenbanktabelle mit 40 Zeichen gespeichert, wobei die Länge durch führende Nullen erreicht wird:

0000000000000000000000000000000123456

Eine Konvertierungsroutine sorgt dafür, dass bei Eingabe die Nullen ergänzt und bei der Anzeige die Nullen ausgeblendet werden.

Zusätzlich können im Bereich **Fixed Values** Domänenfestwerte oder im Feld **Value Table** eine Werttabelle angegeben werden. Dies ignorieren wir jedoch für den Moment. Aktivieren Sie abschließend die Domäne über das Icon (**Activate**). Die Domäne kann nun in einem Datenelement verwendet werden, das wir im nächsten Abschnitt anlegen werden.

4.3 Datenelemente anlegen

Um ein Datenelement anzulegen, gehen Sie im Menü auf **File • New • ABAP Repository Object** und wählen im sich öffnenden Pop-up-Fenster (siehe Abbildung 4.5) im Ordner **Dictionary** diesmal den Eintrag **Data Element** aus. Zusätzlich können Sie im Eingabefeld darüber bei **type filter text** den Begriff »Data Element« (oder einen Teil davon) eingeben, damit die Ergebnisliste entsprechend gefiltert wird. Klicken Sie anschließend auf **Next >**.

Geben Sie im nächsten Schritt, wie bei den Domänen auch, im Feld **Package** Ihr Paket aus Abschnitt 2.2.1, »Paket anlegen«, ein. Vergeben Sie zusätzlich im Feld **Name** einen Namen (hier »ZE_EMP_NAME«), gegebenenfalls mit Ihrem Namenskürzel, und im Feld **Description** eine Beschreibung (hier »Mitarbeitername«). Klicken Sie anschließend auf **Next >**. Dadurch öffnet sich die Eigenschaftsseite des Datenelements aus Abbildung 4.8.

Wählen Sie im Bereich **Data Type Information** im Drop-down-Menü **Category** den Eintrag **Domain** aus und tragen Sie im Feld **Type Name** den Namen Ihrer Domäne aus Abschnitt 4.2, »Domänen anlegen«, ein. Danach können Sie Ihr Datenelement bereits über einen Klick auf das Icon (**Activate**) aktivieren.

Direkte Verwendung von eingebauten Datentypen

Grundsätzlich hätten Sie den eingebauten Datentyp **CHAR** auch direkt über den Eintrag **Predefined Type** auswählen können. Dann hätten Sie aber beispielsweise nicht die Möglichkeit gehabt, zwischen Groß- und Kleinschreibung zu wählen. Außerdem stünde auch die Option des Wertebereichs nicht zur Verfügung.

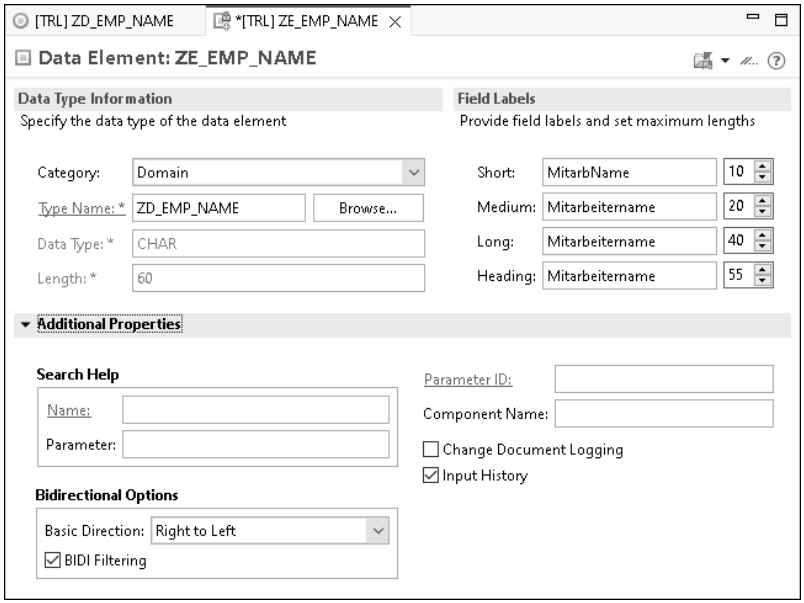


Abbildung 4.8: Eigenschaften eines Datenelements

Bei der Anlage des Datenelements haben Sie darüber hinaus aber noch die Möglichkeit, im Bereich **Field Labels** (hier »Mitarbeitername« in verschiedenen Längen) die Texte zum Datenelement und im Bereich **Additional Properties** beispielsweise eine Suchhilfe oder eine Parameter-ID anzugeben.

Nachdem Sie nun eine Domäne und ein darauf aufbauendes Datenelement angelegt haben, wollen wir im nächsten Schritt eine Datenbanktabelle anlegen, die dieses Datenelement verwendet.

4.4 Datenbanktabellen

Eine der Hauptfunktionen des ABAP Dictionary ist die Verwaltung der zentralen *Datenbanktabellen* des SAP-Systems. Eine Datenbanktabelle beinhaltet eine Menge von persistenten Daten. Das heißt, die Daten existieren nicht nur zur Laufzeit, sondern werden dauerhaft auf einer Festplatte des Datenbankservers vorgehalten. Die Struktur einer Datenbank lässt sich gut mit einem Excel-Sheet vergleichen: Daten sind in *Zeilen* und *Spalten* strukturiert. Die Spaltennamen definieren, welche Art von Daten in jeder Spalte stehen sollen, während die einzelnen Zeilen einen konkreten Datensatz repräsentieren. Über das ABAP Dictionary als Schnittstelle zur an das SAP-System angebundenen Datenbank können Sie sowohl die Strukturen als auch die Daten aller Datenbanktabellen anzeigen lassen. Zusätzlich können Sie auch eigene Datenbanktabellen anlegen, um Daten für Ihre Anwendung dauerhaft zu speichern.

Im gleichnamigen Abschnitt 4.4.1 lernen Sie, wie Sie bestehende Datenbanktabellen anzeigen, während ich in Abschnitt 4.4.2 die Anlage einer Datenbanktabelle zeige. Abschließend erfahren Sie in Abschnitt 4.4.3, wie Sie eine Datenbanktabelle umsetzen können. Dies wird insbesondere dann relevant, wenn sich Eigenschaften wie Datentyp oder Länge einer Domäne ändern und dadurch bereits gespeicherte Daten auf der Datenbanktabelle in »Gefahr« sind.

4.4.1 Datenbanktabellen anzeigen

Um eine Datenbanktabelle anzuzeigen, klicken Sie in der Funktionsleiste der ABAP Development Tools auf das Icon (**Open ABAP Development Object**), oder Sie nutzen die Tastenkombination Strg + ⇧ + A). Geben Sie im sich öffnenden Pop-up-Fenster aus Abbildung 4.9 den Namen der gewünschten Tabelle ein. Wie in Abbildung 4.9 dargestellt, empfiehlt es sich, die Suche mithilfe des Präfixes »type:dtab« gefolgt von einem Leerzeichen einzuschränken. Dadurch wird gezielt nach Datenbanktabellen gesucht, während andere Entwicklungsobjekte ausgeblendet werden.

In Abbildung 4.9 wird beispielsweise die Datenbanktabelle `/DMO/FLIGHT` des Flugdatenmodells geöffnet, da diese auf jedem SAP-System vorhanden ist und wir so die umgangsweise mit Eclipse ADT üben können. Natürlich können Sie auf diese Art und Weise jede in Ihrem SAP-System vorhandene Datenbanktabelle öffnen und so nachvollziehen, wie diese Datenbanktabelle aufgebaut ist und welche Daten sie enthält.

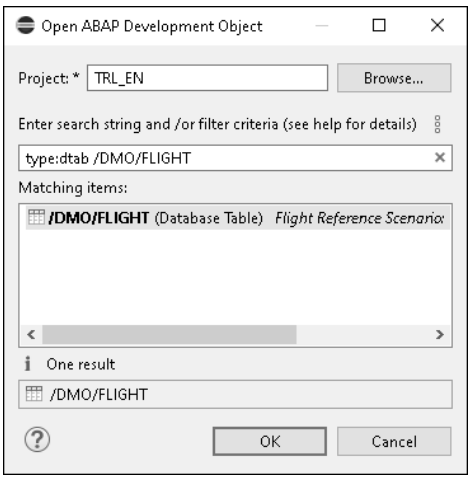


Abbildung 4.9: Nach einer Datenbanktabelle suchen

Öffnen Sie die gewünschte Tabelle mit einem Doppelklick. Dadurch wird Ihnen die Definition der Datenbanktabelle wie in Abbildung 4.10 angezeigt. Wenn Sie nun den Inhalt der Datenbanktabelle sehen möchten, können Sie mit einem Rechtsklick das Kontextmenü öffnen und

über die beiden Einträge **Open with • Data Preview** (ohne Feldauswahl) und **Open with • Data Preview…** (mit Feldauswahl) die Datenvorschau aufrufen.

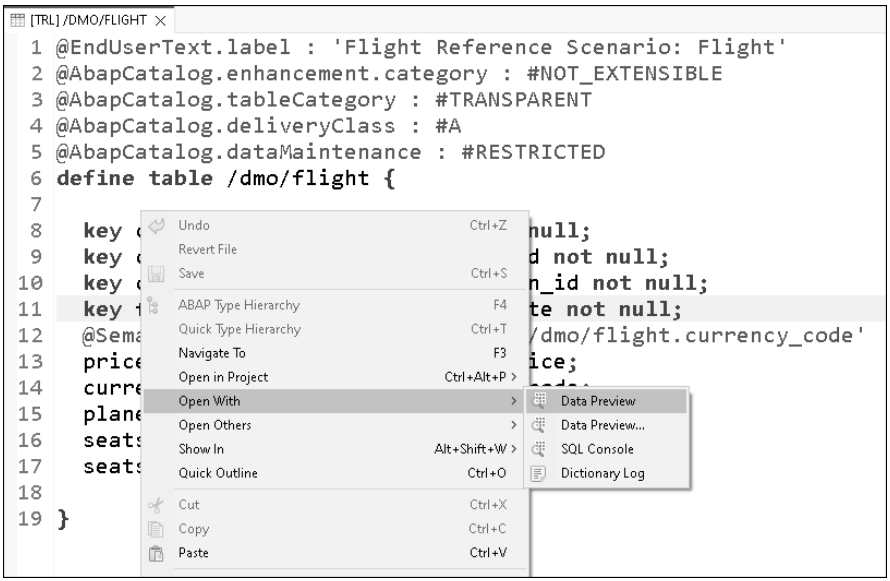
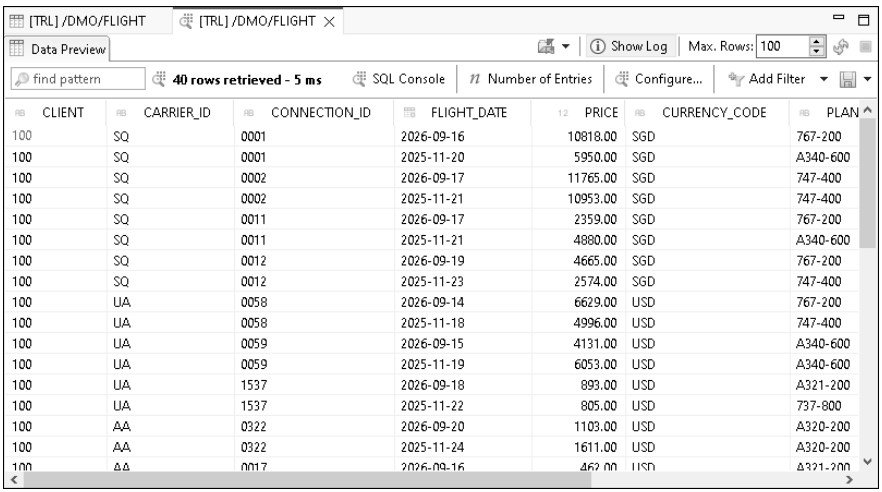


Abbildung 4.10: Datenvorschau für eine Datenbanktabelle aufrufen

In der sich öffnenden Datenvorschau (siehe Abbildung 4.11) sehen Sie nun (falls vorhanden) die ersten 100 Einträge der Datenbanktabelle. Hier haben Sie die Möglichkeit, nach Einträgen zu suchen, Filter hinzuzufügen oder über den Button **SQL Console** gezielt die SQL-Anweisungen zu modifizieren.



CLIENT	CARRIER_ID	CONNECTION_ID	FLIGHT_DATE	PRICE	CURRENCY_CODE	PLAN
100	SQ	0001	2026-09-16	10818.00	SGD	767-200
100	SQ	0001	2025-11-20	5950.00	SGD	A340-600
100	SQ	0002	2026-09-17	11765.00	SGD	747-400
100	SQ	0002	2025-11-21	10953.00	SGD	747-400
100	SQ	0011	2026-09-17	2359.00	SGD	767-200
100	SQ	0011	2025-11-21	4880.00	SGD	A340-600
100	SQ	0012	2026-09-19	4665.00	SGD	767-200
100	SQ	0012	2025-11-23	2574.00	SGD	747-400
100	UA	0058	2026-09-14	6629.00	USD	767-200
100	UA	0058	2025-11-18	4996.00	USD	747-400
100	UA	0059	2026-09-15	4131.00	USD	A340-600
100	UA	0059	2025-11-19	6053.00	USD	A340-600
100	UA	1537	2026-09-18	893.00	USD	A321-200
100	UA	1537	2025-11-22	805.00	USD	737-800
100	AA	0322	2026-09-20	1103.00	USD	A320-200
100	AA	0322	2025-11-24	1611.00	USD	A320-200
100	AA	0017	2026-09-16	462.00	USD	A321-200

Abbildung 4.11: Datenvorschau für eine Datenbanktabelle

4.4.2 Datenbanktabellen anlegen

Nun legen wir eine neue Datenbanktabelle an: Gehen Sie hierzu im Menü auf **File** • **New** • **ABAP Repository Object**, und wählen Sie im sich öffnenden Pop-up-Fenster aus Abbildung 4.5 im Ordner **Dictionary** den Eintrag **Database Table** aus. Zusätzlich können Sie im Eingabefeld darüber bei **type filter text** den Begriff »Database Table« (oder einen Teil davon) eingeben, damit die Ergebnisliste danach gefiltert wird. Bestätigen Sie anschließend mit einem Klick auf **Next >**.

Geben Sie im nächsten Schritt, wie Sie es bereits von den Domänen und den Datenelementen kennen, im Feld **Package** Ihr Paket aus Abschnitt 2.2.1, »Paket anlegen«, ein. Vergeben Sie im Feld **Name** einen Namen (hier »ZFR_EMPLOYEES«), gegebenenfalls mit Namenskürzel, und im Feld **Description** eine Beschreibung (hier »Mitarbeitername«). Klicken Sie anschließend auf **Next >**. Dadurch öffnet sich der Texteditor, in dem der grundlegende Quellcode der Datenbanktabelle dargestellt wird.

Wie in Listing 4.2 dargestellt, werden die Felder der Tabelle innerhalb der geschweiften Klammern `{}` definiert. Jedes Feld erhält einen Namen und wird über einen Doppelpunkt (`:`) einem Datenelement oder einem eingebaute Datentypen zugeordnet.

Wenn Ihre Tabelle mandantenabhängig sein soll (das ist in der Regel der Fall), müssen Sie als erstes Feld das Schlüsselfeld `CLIENT` vom Typ `abap.clnt` hinzufügen. Dieses Feld wird dann automatisch bei jeder ABAP-SQL-Anweisung mit dem aktuellen Mandanten gefüllt bzw. beim lesenden Zugriff entsprechend verarbeitet.

Mandantenabhängigkeit

Mandantenabhängigkeit bedeutet, dass Daten für verschiedene Systeme auf einer Datenbanktabelle liegen können, die sich nur anhand der Mandantennummer voneinander unterscheiden. Für den fiktiven Mandanten 100 könnten 50 Mitarbeitende existieren, für einen anderen Mandanten 10. Je nachdem, auf welchem Mandanten sich die Anwenderinnen und Anwender einloggen, können Sie nur den jeweiligen Datenbereich sehen. Nur wenn Sie wollen, dass beide (oder mehr) Mandanten sich die Daten teilen, können Sie diese Spalte weglassen.

Wählen Sie als Tabellenkategorie hier immer `TRANSPARENT` aus. Die weiteren möglichen Annotationen sind im weiteren Verlauf dieses Abschnitts beschrieben.

Ergänzen Sie Ihren Quellcode mit den Feldern aus Listing 4.2.

```
@EndUserText.label : 'Mitarbeiter'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
@AbapCatalog.tableCategory : #TRANSPARENT
@AbapCatalog.deliveryClass : #C
```

```
@AbapCatalog.dataMaintenance : #RESTRICTED
define table zfr_employees {
  key client : abap.clnt not null;
  key emp_id : abap.numc(5);
  emp_name   : ze_emp_name;
  mgr_id     : abap.numc(5);
}
```

Listing 4.2: Definition einer eigenen Datenbanktabelle

Nach der Definition können Sie Ihre Datenbanktabelle erneut über das Icon  (**Activate**) aktivieren. Diese Tabelle werden wir zu einem späteren Zeitpunkt, in Abschnitt 7.2, »Ändernde ABAP-SQL-Anweisungen«, mit Inhalt befüllen.

Erweiterungskategorie

Sie haben die Möglichkeit, in der Tabellendefinition eine *Erweiterungskategorie* anzugeben. Diese bestimmt, ob und wie die Datenbanktabelle erweitert werden darf, und wird am Kopf mit der Annotation `@AbapCatalog.enhancement.category` angegeben. Die folgenden Optionen stehen Ihnen zur Verfügung:

- `#NOT_EXTENSIBLE`: Tabelle ist nicht erweiterbar.
- `#EXTENSIBLE_CHARACTER`: Die Struktur darf mit zeichenartigen flachen Komponenten (`c`, `n`, `d` oder `t`) erweitert werden.
- `#EXTENSIBLE_CHARACTER_NUMERIC`: Die Tabelle ist erweiterbar und zeichenartig oder numerisch: Die Struktur darf erweitert werden, die Erweiterung darf aber keine tiefen Datentypen enthalten.
- `#EXTENSIBLE_ANY`: Die Struktur darf mit beliebigen Komponenten erweitert werden.

Für eigene Entwicklungen spielt dies meist keine Rolle, da sie üblicherweise nicht an Drittanbieter ausgeliefert werden. Wichtig wird dies vor allem bei den Standard-Datenbanktabellen von SAP, da diese Annotation darüber entscheidet, ob Sie neue Z-Felder über eine sogenannte Append-Struktur hinzufügen können.

Auslieferungsklasse

Ein weiterer Teil der Tabellendefinition ist die *Auslieferungsklasse*. Sie wird am Kopf mit der Annotation `@AbapCatalog.deliveryClass` angegeben und bestimmt das Verhalten der Datenbanktabelle bei der Installation, beim Upgrade, bei einer Mandantenkopie und beim Transport zwischen Kundensystemen. Für Sie sind die Werte `A` für Anwendungstabelle (Stamm- und Bewegungsdaten) als auch `C` für Customer Table (dt. Kundentabelle) am wichtigsten. Eine Kundentabelle bedeutet in diesem Fall, dass die Daten ausschließlich von uns als Kunden gepflegt werden.

Datenpflege

Die Berechtigung zur *Datenpflege* wird am Kopf mit der Annotation `@AbapCatalog.dataMaintenance` angegeben und bestimmt, wie auf die Datenbanktabelle über Pflegedialog oder in der Datenvorschau zugegriffen werden kann.

- `#DISPLAY`: Datenbanktabelle kann nur angezeigt werden.
- `#ALLOWED`: Datenbanktabelle kann in der Datenanzeige angezeigt und bearbeitet werden. Pflegedialoge können erstellt werden.
- `#NOT_ALLOWED`: Datenbanktabelle kann in der Datenanzeige angezeigt und bearbeitet werden. Es können keine Pflegedialoge für die Datenbanktabelle erstellt werden.
- `#RESTRICTED`: Datenbanktabelle kann in der Datenanzeige angezeigt, aber nicht bearbeitet werden.

4.4.3 Datenbanktabelle umsetzen

Eine Datenbanktabelle müssen Sie immer dann umsetzen, wenn die Datenbanktabelle bereits Daten beinhaltet und Sie Änderungen auf Feldebene vornehmen, durch die sich diese Daten ändern würden. »Umsetzen« bedeutet in diesem Kontext, dass die Daten der Datenbanktabelle vom alten Format in das neue Format umgewandelt, also »transformiert«, werden. Das Ziel dieser Umsetzung ist immer, die Daten zu erhalten.

Eine solche Situation ist in Abbildung 4.12 dargestellt. Die in Abschnitt 4.4.2, »Datenbanktabellen anlegen«, angelegte Datenbanktabelle wurde dahingehend geändert, dass das Feld `EMP_NAME` von 60 auf 50 Zeichen verkürzt wurde. Dazu wurde das vorher angelegte Datenelement mit der direkten Angabe des eingebautes Typs `char` ersetzt. Das Aktivieren schlägt daher fehl, und es wird die Fehlermeldung **Structure change at field level (convert table …)** angezeigt.

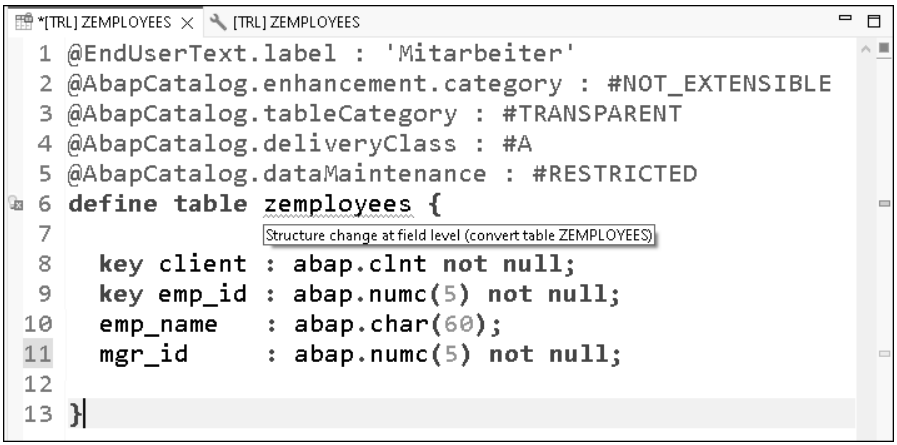


Abbildung 4.12: Strukturänderung einer Tabelle auf Feldebene

Um diesen Fehler zu beheben, rufen Sie die Quick-Fix-Funktion auf, indem Sie auf den Namen der Datenbanktabelle in Zeile 6 klicken und die Tastenkombination Strg + 1 drücken. Dadurch wird Ihnen das Menü aus Abbildung 4.13 angezeigt.

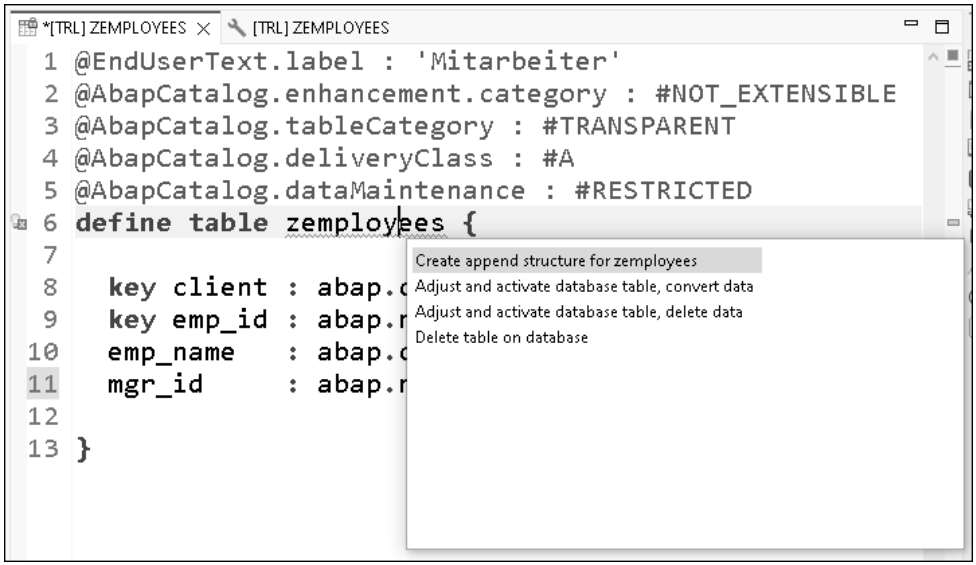


Abbildung 4.13: Tabelle über die Quick-Fix-Funktion umsetzen

Zum Umsetzen der Datenbanktabelle können Sie nun zwischen den beiden folgenden Funktionen wählen:

- **Adjust and activate database table, convert data**: Ändert die Datenbanktabelle auf der Datenbank und konvertiert die Daten in die neue Form.
- **Adjust and activate database table, delete data**: Ändert die Datenbanktabelle auf der Datenbank und löscht alle Daten.

Das Löschen der Daten ist immer dann notwendig, wenn die Daten nicht erhalten werden können. Dies ist beispielsweise der Fall, wenn Sie ein Feld verkürzen. Änderungen sind also mit Vorsicht zu genießen, weil diese direkte Auswirkungen auf gespeicherte Produktivdaten haben können.

Wenn die Verkürzung des Mitarbeiternamens stattdessen auf Domänenebene stattfindet, würde das Aktivieren der Domäne fehlschlagen, da dies indirekt zu Datenverlust in darauf aufbauenden Datenbanktabellen führen würde. Abbildung 4.14 stellt diese Situation dar.

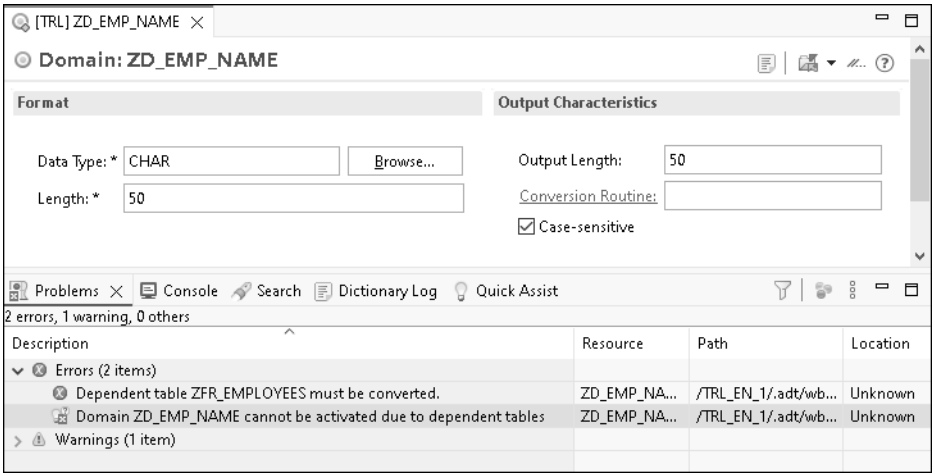


Abbildung 4.14: Fehlermeldung beim Aktivieren der geänderten Domäne

Markieren Sie in diesem Fall die Fehlermeldung in der Ansicht **Problems** und rufen Sie hier ebenfalls die Quick-Fix-Funktion über die Tastenkombination Strg + 1 auf. Dadurch öffnet sich das Pop-up-Fenster aus Abbildung 4.15.

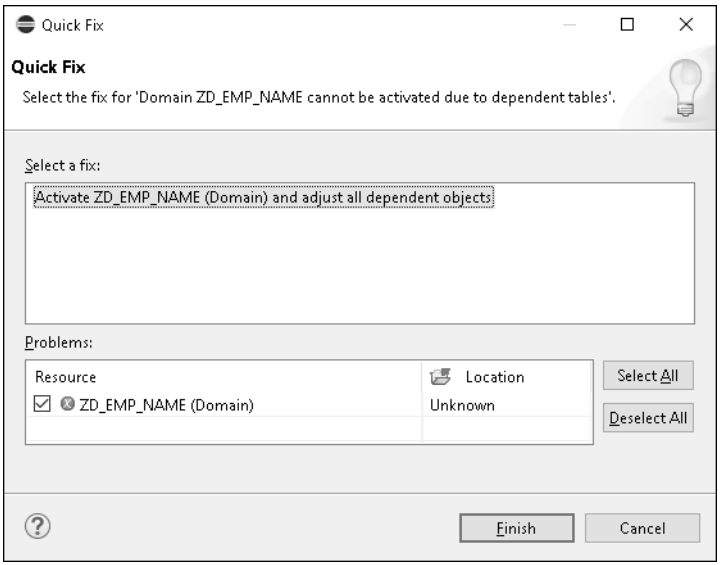


Abbildung 4.15: Quick Fix bei einer indirekten Änderung

Selektieren Sie hier die die Lösung **Activate … (Domain) and adjust all dependent objects** und bestätigen Sie Ihre Auswahl mit **Finish**.

4.5 Strukturen

In Abschnitt 2.7.5, »Strukturen«, hatte ich Ihnen bereits erläutert, was Strukturen sind und wie Sie eine Variable vom Typ einer Struktur definieren können. Hier möchte ich Ihnen zeigen, wie Sie selbst eine solche Feldleiste als Typ im ABAP Dictionary definieren können. Nach der Definition steht ein solcher Strukturtyp global im System zur Verfügung und kann aus jedem ABAP-Quellcode heraus zur Definition einer Struktur referenziert werden.

4.5.1 Struktur anlegen

Um eine neue Struktur anzulegen, gehen Sie im Menü auf **File • New • ABAP Repository Object** und wählen im sich öffnenden Pop-up-Fenster aus Abbildung 4.5 im Ordner **Dictionary** den Eintrag **Structure** aus. Zusätzlich können Sie im Eingabefeld darüber bei **type filter text** den Begriff »Structure« eingeben, damit die Ergebnisliste danach gefiltert wird. Klicken Sie anschließend auf **Next >**.

Folgen Sie auch hier dem bekannten Schema und geben Sie im Feld **Package** Ihr Paket aus Abschnitt 2.2.1, »Paket anlegen«, ein und vergeben Sie zusätzlich im Feld **Name** einen Namen (hier »ZST_FR_EMPLOYEES«), gegebenenfalls mit Namenskürzel, und im Feld **Description** eine Beschreibung (hier »Mitarbeiter«). Klicken Sie anschließend auf **Next >**. Dadurch öffnet sich der Texteditor, in dem der grundlegende Quellcode der Struktur dargestellt wird.

Analog zur Definition von Datenbanktabellen werden die Felder innerhalb der geschweiften Klammern { } angegeben (siehe Listing 4.4). Jedes Feld erhält einen Namen und wird über einen Doppelpunkt (:) einem Datentyp oder Datenelement zugeordnet.

```
@EndUserText.label : 'Mitarbeiter'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
define structure zst_fr_employees {
  emp_id     : abap.numc(5);
  emp_name   : ze_emp_name;
  mgr_id     : abap.numc(5);
}
```

Listing 4.3: Definition einer eigenen Struktur

Die Annotation `@AbapCatalog.enhancement.category` steuert, wie bei den Datenbanktabellen, die Erweiterbarkeit der Struktur. Die möglichen Ausprägungen habe ich bereits in Abschnitt 4.4.2, »Datenbanktabellen anlegen«, beschrieben.

Nach dem Aktivieren können Sie diese Strukturdefinition mit ABAP verwenden, in dem Sie sich beispielsweise eine Struktur damit definieren:

```
DATA: ls_mitarbeiter TYPE zst_employees.
```

Dies wäre natürlich auch mit der in Abschnitt 4.4.2 beschriebenen Datenbanktabelle möglich gewesen. Manchmal möchten Sie jedoch noch zusätzliche Spalten zur Struktur hinzufügen, die in der Datenbank nicht vorhanden sind. Dazu gehören insbesondere, aber nicht ausschließlich, Textfelder, die die technischen Begriffe lesbarer für Anwenderinnen und Anwender machen. So könnte beispielsweise zu einer Spalte mit Materialnummer eine zweite Spalte hinzugefügt werden, die den Text zum Material in der angemeldeten Sprache anzeigt.

4.5.2 Verwendung von Währungsfeldern

Wenn Sie in Ihrer Struktur *Währungsfelder* verwenden, werden Sie beim Aktivieren die folgende Fehlermeldung erhalten: **Annotation with reference to currency code for field PRICE is missing**.

Jedes Währungsfeld muss mit einem Einheitenfeld verknüpft sein. Diese Information wird bei der Anzeige benötigt, um die Währung entsprechend formatieren zu können. Die japanische Währung Yen, die beispielsweise keine Nachkommastellen hat, muss anders dargestellt werden als die europäische Währung Euro.

In Listing 4.4 sehen Sie eine definierte Struktur, in der die beiden Felder zu Betrag und Währung korrekt miteinander verknüpft wurden. Die zusätzliche Annotation `@Semantics.amount.currencyCode`, die sich oberhalb des Betragsfelds befindet, stellt die Beziehung zwischen den Feldern her. Dadurch kann das System Beträge korrekt formatieren und anzeigen.

```
@EndUserText.label : 'Beispielstruktur für Währung'
@AbapCatalog.enhancement.category : #NOT_EXTENSIBLE
define structure zst_curr {
  @Semantics.amount.currencyCode: 'zst_curr.currency_code'
  price             : /dmo/flight_price;
  currency_code     : /dmo/currency_code;
}
```

Listing 4.4: Definition einer Struktur mit einem Währungsfeld

4.6 Checkliste

Die Checkliste in Tabelle 4.1 prüft, ob Sie die Grundlagen des ABAP Dictionary und die Anlage der wichtigsten Dictionary-Objekte verstanden haben, und dient Ihrer Selbstkontrolle.

Ich habe…	Abschnitt
… verstanden, dass das ABAP Dictionary die zentrale Stelle für alle Datendefinitionen im SAP-System ist und für zentrale Datenbeschreibung, Konsistenz, Wiederverwendbarkeit und automatische Datenbankanpassung sorgt.	Einleitung
… verstanden, dass das ABAP Dictionary die Schnittstelle zur SAP-HANA-Datenbank bildet und ABAP-SQL-Zugriffe wie `SELECT` auf Datenbanktabellen ermöglicht.	Einleitung
… das zweistufige Domänenprinzip mit technischen Domänen (Domänen) und semantischen Domänen (Datenelementen) verstanden.	Abschnitt 4.1
… nachvollzogen, wie ein Feld einer Tabelle oder Struktur auf ein Datenelement und dieses wiederum auf eine Domäne verweist.	Abschnitt 4.1
… eine eigene Domäne mit Datentyp, Länge und ggf. weiteren Eigenschaften (z. B. Konvertierungsroutine, Festwerte, Wertetabelle) angelegt und aktiviert.	Abschnitt 4.2
… verstanden, wozu Konvertierungsroutinen dienen und wie sie zwischen interner und externer Darstellung von Daten umwandeln.	Abschnitt 4.2
… ein Datenelement angelegt, einer Domäne zugeordnet und mit Field Labels sowie ggf. zusätzlichen Eigenschaften (z. B. Suchhilfe) versehen.	Abschnitt 4.3
… verstanden, wann es sinnvoll ist, eine Domäne zu verwenden, statt direkt einen eingebauten Datentyp im Datenelement zu hinterlegen.	Abschnitt 4.3
… bestehende Datenbanktabellen über das Icon (**Open ABAP Development Object**) mit dem Filter »type:dtab« gefunden und geöffnet.	Abschnitt 4.4.1
… die Datenvorschau für eine Datenbanktabelle aufgerufen und dort Filter sowie die SQL Console verwendet.	Abschnitt 4.4.1
… eine eigene Datenbanktabelle mit Schlüsselfeldern, weiteren Feldern und dem Mandantenfeld `CLIENT` (`abap.clnt`) angelegt.	Abschnitt 4.4.2
… verstanden, was Mandantenabhängigkeit bedeutet und wann eine Tabelle mandantenabhängig sein sollte.	Abschnitt 4.4.2
… die wichtigsten Annotationen einer Tabellendefinition (Erweiterungskategorie, Tabellenkategorie, Auslieferungsklasse, Datenpflege) kennengelernt und sinnvoll gesetzt.	Abschnitt 4.4.2

Tabelle 4.1: Checkliste zum ABAP Dictionary

Ich habe…	Abschnitt
… gelernt, wann eine Datenbanktabelle umgesetzt werden muss und welche Auswirkungen Strukturänderungen auf vorhandene Daten haben.	Abschnitt 4.4.3
… die Quick-Fix-Funktion (Strg + 1) zum Umsetzen einer Datenbanktabelle bzw. zum Aktivieren abhängiger Objekte bei Domänenänderungen angewendet.	Abschnitt 4.4.3
… den Unterschied zwischen den Optionen **Adjust and activate database table, convert data** und **Adjust and activate database table, delete data** verstanden.	Abschnitt 4.4.3
… eine eigene Struktur als Dictionary-Objekt angelegt und mit ABAP über die Anweisung `DATA … TYPE` referenziert.	Abschnitt 4.5.1
… verstanden, warum es sinnvoll sein kann, eine Struktur statt einer Datenbanktabelle als Typ zu verwenden, etwa um zusätzliche (nicht persistierte) Felder zu ergänzen.	Abschnitt 4.5.1
… gelernt, dass Währungsfelder über die Annotation `@Semantics.amount.currencyCode` mit einem Einheitenfeld verknüpft werden müssen, damit sie korrekt formatiert werden können.	Abschnitt 4.5.2

Tabelle 4.1: Checkliste zum ABAP Dictionary (Forts.)

Kapitel 5

ABAP-Grundbefehle

Dieses Kapitel bietet Ihnen eine kompakte und praxisorientierte Einführung in die wichtigsten ABAP-Grundbefehle. Sie lernen die zentralen Sprachelemente kennen, die Sie in nahezu jedem ABAP-Programm benötigen.

In Kapitel 2, »Erste Schritte«, habe ich Ihnen neben einfachen Syntaxregeln und der Möglichkeit des Kommentierens von Quellcode bereits gezeigt, wie Sie einfache Variablen mit der Anweisung `DATA` definieren, befüllen und wieder initialisieren können. Ziel dieses Kapitels ist es, Ihnen einen Überblick über die ABAP-Grundbefehle zu geben, also die grundlegenden Bestandteile der Syntax, die Sie vielleicht schon aus anderen Programmiersprachen kennen:

- Steueranweisungen (siehe Abschnitt 5.1)
- Arbeiten mit Zeichenketten (siehe Abschnitt 5.2)
- Rechenoperationen (siehe Abschnitt 5.3)
- Typdefinitionen (siehe Abschnitt 5.4)

Dieses Kapitel ist so konzipiert, dass Neulinge schrittweise in die ABAP-Welt eingeführt werden. Es erhebt keinen Anspruch auf Vollständigkeit. Stattdessen konzentriere ich mich bewusst auf eine Auswahl der wichtigsten und am häufigsten verwendeten Anweisungen. Wenn Sie diese sicher beherrschen, können Sie den Großteil der Problemstellungen des ABAP-Alltags bestreiten und sind bestens auf die darauf aufbauenden Funktionen vorbereitet, die in den weiteren Kapiteln des Buchs behandelt werden.

Ich erläutere jede ABAP-Anweisung anhand eines kurzen Beispiels, das sich auf die wesentliche Funktion der Anweisung konzentriert. Sie sollten so ein Verständnis dafür gewinnen, was der Nutzen einer Anweisung ist und wie Sie diese verwenden.

5.1 Steueranweisungen

ABAP stellt Ihnen Steueranweisungen zur Verfügung, um *Verzweigungen* und *Schleifen* in Ihren Programmen zu realisieren. Die wichtigsten Steueranweisungen in ABAP sind:

- bedingte Anweisungen mit `IF` (siehe Abschnitt 5.1.1)
- verwendbare logische Ausdrücke (siehe Abschnitt 5.1.2)

- bedingte Verzweigungen mit `CASE` (siehe Abschnitt 5.1.3)
- nicht bedingte Schleifenverarbeitung mit `DO` (siehe Abschnitt 5.1.5)
- bedingte Schleifenverarbeitung mit `WHILE` (siehe Abschnitt 5.1.6)
- bedingte Durchführung von Schleifen mit `CHECK` (siehe Abschnitt 5.1.7)
- Verlassen des aktuellen Anweisungsblocks mit `EXIT` (siehe Abschnitt 5.1.8)
- Überspringen von Durchläufen mit `CONTINUE` (siehe Abschnitt 5.1.9)

Innerhalb dieser Konstrukte können Sie beliebige weitere Anweisungen verschachteln, beispielsweise Methodenaufrufe, Schleifen oder weitere Bedingungen. Achten Sie dabei stets darauf, dass alle Blöcke korrekt abgeschlossen werden, und ebenso darauf, dass die verwendeten Bedingungen und Schleifen innerhalb eines Ereignisses abgeschlossen werden.

5.1.1 IF-Abfragen

Mit der `IF`-Abfrage steuern Sie den Programmablauf abhängig von einer Bedingung. In der einfachsten Form der Abfrage einer Bedingung führt das System alle Anweisungen `anweisungen1` zwischen den Schlüsselwörtern `IF` und `ENDIF` durch, sobald die Bedingung `bedingung1` der `IF`-Anweisung erfüllt ist. Dabei kann der Anweisungsteil wiederum neue Abfragen oder Schleifen enthalten.

Verwenden Sie `ELSE` in der `IF`-Anweisung, führt das System die Anweisungen `anweisungen2` nach `ELSE` aus, wenn die Bedingung `bedingung1` der `IF`-Anweisung nicht erfüllt ist.

Verwenden Sie `ELSEIF` in der `IF`-Anweisung, wird eine neue Bedingung `bedingung2` abgefragt, wenn die Bedingung `bedingung1` der ersten `IF`-Anweisung nicht erfüllt ist.

Die wichtigsten möglichen logischen Ausdrücke, die Sie hierfür verwenden können, finden Sie in Abschnitt 5.1.2, »Logische Ausdrücke«.

Syntax

In Listing 5.1 sehen Sie die Syntax einer IF-Anweisung.

```
IF bedingung1.
  [anweisungen1]
[ELSEIF bedingung2.
  [anweisungen2]]
...
[ELSE.
  [anweisungen3]]
ENDIF.
```

Listing 5.1: Syntax der IF-Anweisung

Beispiele

Die folgenden Beispiele zeigen typische Einsatzszenarien der `IF`-Anweisung in unterschiedlichen Ausprägungen. Eine einfache Bedingung sieht wie folgt aus:

```
IF sy-subrc <> 0.
  "Variable ist ungleich 0
ENDIF.
```

Eine Bedingung mit einer Alternative sehen Sie in Listing 5.2:

```
IF sy-subrc <> 0.
  "Variable ist ungleich 0
ELSE.
  "Die obere Bedingung wurde nicht erfüllt
ENDIF.
```

Listing 5.2: IF-Anweisung mit einem ELSE-Zweig

Eine komplexe Bedingung mit Alternative könnte hingegen aussehen wie in Listing 5.3.

```
IF sy-subrc <> 0.
  "Variable ist ungleich 0
ELSEIF sy-dbcnt = 1.
  "Variable ist gleich 1
ELSE.
  "Beide oberen Bedingungen wurden nicht erfüllt
ENDIF.
```

Listing 5.3: Komplexe IF-Bedingung mit mehreren Prüfungen

5.1.2 Logische Ausdrücke

Logische Ausdrücke werden in Vergleichen z. B. zur Fallunterscheidung gebraucht. ABAP stellt Ihnen die in Tabelle 5.1 gezeigten logischen Ausdrücke zur Verfügung. Aus historischen Gründen gibt es für einen logischen Ausdruck jeweils mehrere Schreibweisen.

Bedeutung	Mögliche Schreibweisen
gleich	`A = B.` oder `A EQ B.`
größer als	`A > B.` oder `A GT B.`
kleiner als	`A < B.` oder `A LT B.`

Tabelle 5.1: Logische Ausdrücke

Bedeutung	Mögliche Schreibweisen
ungleich	A <> B., A >< B., oder A NE B.
größer gleich	A >= B. oder A GE B.
kleiner gleich	A <= B. oder A LE B.
zwischen	A BETWEEN B AND C BT (für Ranges-Tabellen)
initiale Belegung	A IS INITIAL

Tabelle 5.1: Logische Ausdrücke (Forts.)

Bedingungen können mit Klammern gruppiert werden und über die Anweisungen NOT (nicht), OR (oder) und AND (und) verknüpft werden. Die Auswertung erfolgt immer von links nach rechts. Sobald das Ergebnis gefunden wurde, wird die nächste Anweisung verarbeitet.

5.1.3 CASE-Anweisungen

Die CASE-Anweisung nutzen Sie, um abhängig vom Inhalt eines bestimmten Datenfelds verschiedene Anweisungsblöcke zu verarbeiten. Dabei vergleicht das System den Wert der CASE-Anweisung operand mit dem Wert der WHEN-Anweisungen (operand1, operand2 … operandn). Sobald eine Übereinstimmung gefunden wird, führt das System den zugehörigen Anweisungsteil der WHEN-Anweisung aus und setzt die Verarbeitung nach ENDCASE fort. Wird kein passender Wert gefunden, wird, sofern vorhanden, die optionale Anweisung WHEN OTHERS durchlaufen.

Die Anweisung CASE hat gegenüber der IF-Anweisung einen Performancevorteil, kann aber immer nur den Wert *eines* Felds abfragen.

Syntax

In Listing 5.4 sehen Sie die Syntax der CASE-Anweisung.

```
CASE operand.
  [WHEN operand1 [OR operand2 [OR operandn [...]]].
    [anweisungen1]]
  ...
  [WHEN OTHERS.
    [anweisungen0]]
ENDCASE.
```

Listing 5.4: Syntax der CASE-Anweisung

Beispiel

Listing 5.5 zeigt die Definition einer CASE-Anwendung.

```
CASE sy-subrc.
  WHEN 0.
   "Satz gefunden
  WHEN 4.
   "Satz nicht gefunden
  WHEN OTHERS.
   "Unbekannter Fehler
ENDCASE.
```

Listing 5.5: CASE-Anweisung mit mehreren Prüfungen

5.1.4 Weitere Fallunterscheidungen

Neben den klassischen Anweisungen IF und CASE stellt ABAP mit den Konstruktorausdrücken COND und SWITCH zwei weitere Anweisungen zur Verfügung, mit denen Sie Fallunterscheidungen abbilden können. Ein wesentlicher Vorteil dieser Ausdrücke besteht darin, dass Sie direkt an lesenden Operandenpositionen angegeben werden können und nicht in eine separate Zeile geschrieben werden müssen.

Operandenpositionen

Eine ABAP-Anweisung besteht aus einem Operator (z. B. = oder +) und Operanden (Variablen oder Konstanten). Die Position des Operanden bestimmt, ob dieser geschrieben oder gelesen wird. Am Beispiel vom Operator = wird der Operand davor geschrieben und danach gelesen.

Durch die Verwendung solcher Ausdrücke an lesenden Operandenpositionen wird der Quellcode kürzer, sodass Zuweisungen und gegebenenfalls auch die Adressierung von Speicherbereichen eingespart werden.

Mit der Anweisung COND können Sie eine IF-Abfrage an lesenden Operandenpositionen durchführen. Für einfache Fallunterscheidungen zur Überprüfung des Werts einer Variablen steht Ihnen die Anweisung SWITCH zur Verfügung.

COND: Bedingte Ausdrücke

Während bei einer klassischen IF-Anweisung nur eine Fallunterscheidung stattfindet, erzeugt die Anweisung COND als Ergebnis einen Wert mit dem angegebenen Datentyp datentyp. Das heißt, es wird immer ein Wert zurückgegeben. Ist der Typ vollständig erkennbar, kann auch die Raute # anstelle des Typs angegeben werden.

Inhalt

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

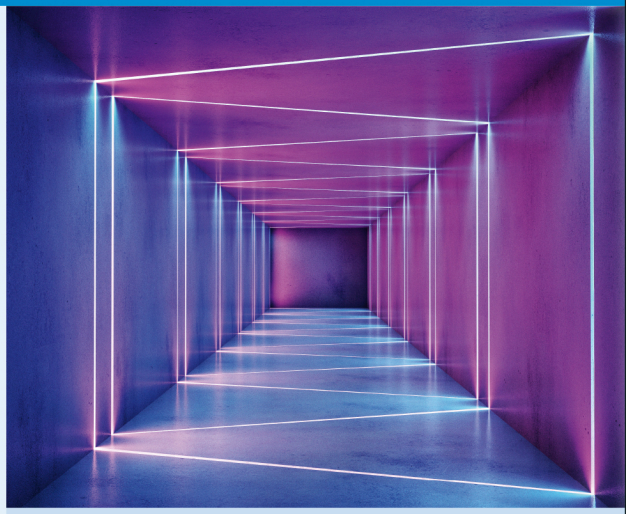
Z

Ihr Sprungbrett in die Welt der SAP-Entwicklung

Lernen Sie ABAP von Grund auf und erzielen Sie schnell erste Erfolge mit eigenem Code. Von der Entwicklungsumgebung über Syntaxgrundlagen bis hin zu Datenbanken und objektorientierter Programmierung führt Sie dieses Buch Schritt für Schritt zum Ziel. Die Beispiele decken klassische und moderne Entwicklungsansätze ab – in On-Premise- ebenso wie in Cloud-Umgebungen. Mit CDS Views und dem ABAP RESTful Application Programming Model steigen Sie tiefer in die moderne ABAP-Entwicklung ein. Der ideale Begleiter – auch zur SAP-Schulung S4D400!

Mit diesem Buch lernen Sie:

- ABAP Development Tools für Eclipse einrichten
- Datentypen deklarieren
- Debugging durchführen
- ABAP Dictionary verwenden
- Steueranweisungen und Tabellen einsetzen
- ABAP SQL für den Datenbankzugriff nutzen
- Objektorientierte Programmierung anwenden
- CDS Views erstellen
- ABAP RESTful Application Programming Model einsetzen
- Auf Geschäftsobjekte mit EML zugreifen



Der Autor
Felix Roth ist selbstständiger ABAP-Trainer, -Entwickler und -Berater (LOOP AT Consulting). Er berät Kunden in Entwicklungsprojekten und beschäftigt sich mit den neuesten SAP-Technologien. Seit 2014 hält er regelmäßig SAP-Schulungen, u. a. beim Rheinwerk Verlag.

Programmierbeispiele stehen zum Download bereit

Rheinwerk Publishing

Gedruckt in Deutschland
Mineralölfreie Druckfarben
Zertifiziertes Papier

€ 39,90 [D] € 41,10 [A]
ISBN 978-3-367-11124-4