

SAP PRESS

ABAP-Entwicklung für SAP HANA

Bearbeitet von
Hermann Gahm, Thorsten Schneider, Christiaan Swanepoel, Eric Westenberger

2., aktualisierte und erweiterte Auflage 2015. Buch. 653 S. Hardcover

ISBN 978 3 8362 3661 4

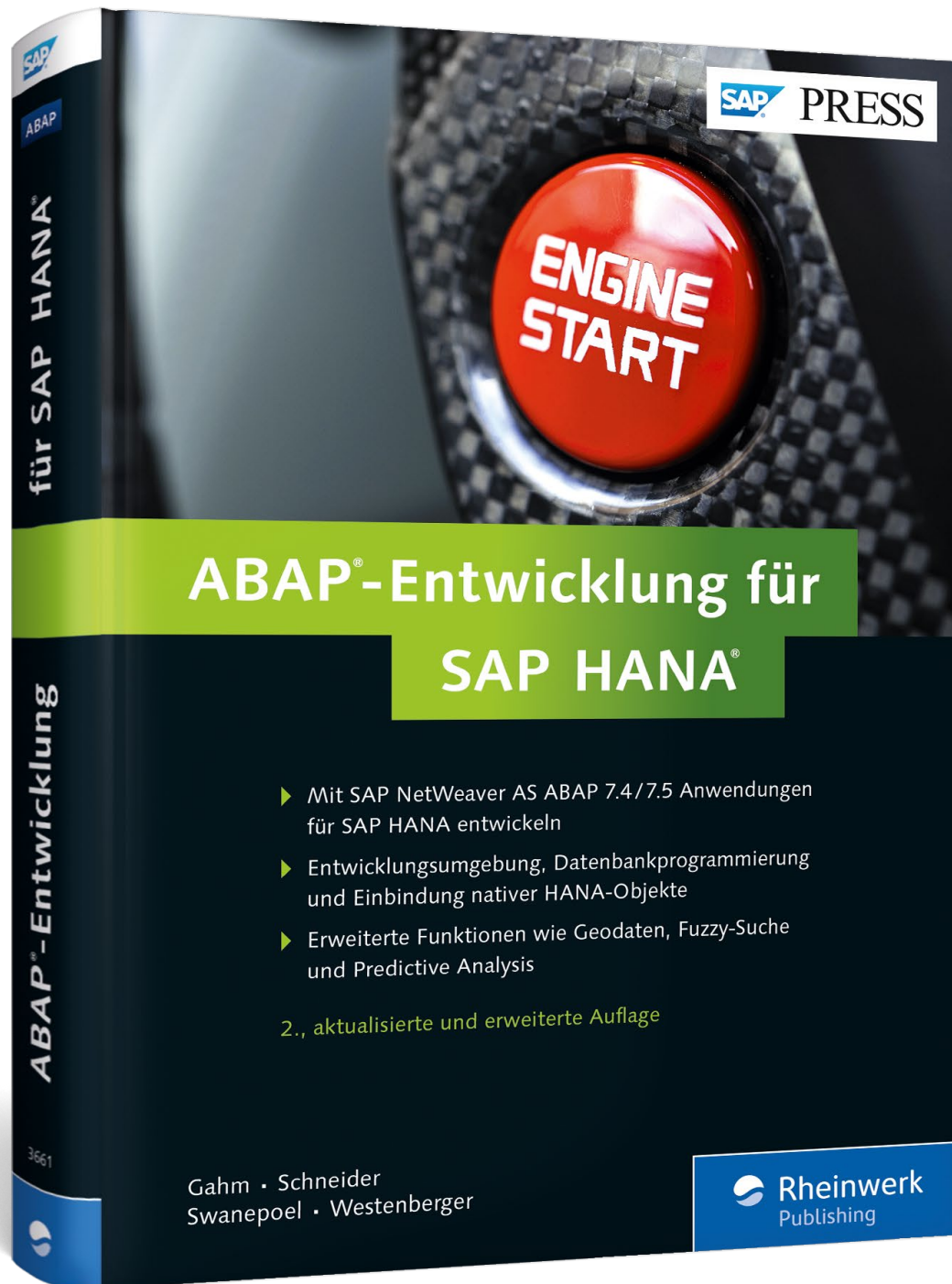
Format (B x L): 16 x 24 cm

[Weitere Fachgebiete > EDV, Informatik > Datenbanken, Informationssicherheit,
Geschäftssoftware > SAP](#)

schnell und portofrei erhältlich bei


DIE FACHBUCHHANDLUNG

Die Online-Fachbuchhandlung beck-shop.de ist spezialisiert auf Fachbücher, insbesondere Recht, Steuern und Wirtschaft. Im Sortiment finden Sie alle Medien (Bücher, Zeitschriften, CDs, eBooks, etc.) aller Verlage. Ergänzt wird das Programm durch Services wie Neuerscheinungsdienst oder Zusammenstellungen von Büchern zu Sonderpreisen. Der Shop führt mehr als 8 Millionen Produkte.



Leseprobe

In dieser Leseprobe lernen Sie, wie Sie native HANA-Objekte aus ABAP heraus aufrufen und ABAP-Programme, die native HANA-Objekte nutzen, konsistent in Ihre Systemlandschaft transportieren können.



Kapitel 5:

»Einbindung nativer SAP-HANA-Entwicklungsobjekte in ABAP«



Inhaltsverzeichnis



Index



Die Autoren



Leseprobe weiterempfehlen

Hermann Gahm, Thorsten Schneider,
Christiaan Swanepoel, Eric Westenberger

ABAP-Entwicklung für SAP HANA

653 Seiten, gebunden, 2. Auflage 2015
69,90 Euro, ISBN 978-3-8362-3661-4



www.sap-press.de/3773

ABAP-Entwickler möchten über das SAP HANA Studio angelegte Views und Datenbankprozeduren in ABAP nutzen. Außerdem sind sie ein leistungsfähiges Transportwesen gewöhnt und erwarten einen konsistenten Transport nativer SAP-HANA-Entwicklungsobjekte über das Change and Transport System.

5 Einbindung nativer SAP-HANA-Entwicklungsobjekte in ABAP

In Kapitel 4, »Native Datenbankentwicklung mit SAP HANA«, haben wir Ihnen gezeigt, wie Sie analytische Modelle (Views) und Datenbankprozeduren über das SAP HANA Studio anlegen können. Nun möchten wir Ihnen erklären, wie Sie diese nativen HANA-Objekte aus ABAP heraus aufrufen können.

Außerdem möchten wir Ihnen erläutern, wie Sie ABAP-Programme, die native HANA-Objekte nutzen, konsistent in Ihrer Systemlandschaft transportieren können.

5.1 Einbindung von analytischen Views

Sie haben in den vorangegangenen Abschnitten gelernt, wie Sie die unterschiedlichen Arten von Views im SAP HANA Studio modellieren können und wie Sie mit dem Data Preview oder in Microsoft Excel auf die Ergebnisse des Views zugreifen können. Wir haben Ihnen in Abschnitt 4.4.4, »Laufzeitobjekte und SQL-Zugriff«, auch bereits gezeigt, wie Sie über SQL die generierten Column Views adressieren können.

In diesem Abschnitt gehen wir auf den Zugriff aus ABAP ein. Wir müssen dabei zwischen ABAP-Release 7.4 und älteren Versionen unterscheiden. Vor ABAP 7.4 ist die einzige Zugriffsmöglichkeit die über natives SQL, was wir Ihnen in Abschnitt 5.1.1, »Zugriff über natives SQL«, kurz vorstellen werden. Ab ABAP 7.4 können Sie die

Zugriff vor und mit
ABAP-Release 7.4

Views aus dem SAP HANA Repository in das ABAP Dictionary importieren und danach auf diese über Open SQL zugreifen. Darauf gehen wir in den Abschnitten 5.1.2, »Externe Views im ABAP Dictionary«, und 5.1.3, »Zugriffsmöglichkeiten auf externe Views«, im Detail ein. Im letzten Abschnitt geben wir Ihnen einige Empfehlungen, Tipps und Tricks für die SAP-HANA-View-Modellierung.

5.1.1 Zugriff über natives SQL

Bei allen vorgestellten View-Typen in SAP HANA entsteht bei der Aktivierung ein Column View im Datenbankkatalog im Schema `_SYS_BIC` mit einem öffentlichen Synonym (*Public Synonym*), z. B. `'test.a4h.book.chapter04::AT_FLIGHT'`.

Über diesen Bezeichner können Sie aus ABAP auf diesen View zugreifen. Listing 5.1 zeigt den Zugriff auf den Attribute View `AT_FLIGHT`, den wir in Abschnitt 4.4.1, »Attribute Views«, angelegt haben über ADBC.

```
" Definition der Resultatstruktur
TYPES: BEGIN OF ty_data,
        carrid  TYPE s_carr_id,
        connid  TYPE s_conn_id,
        fldate  TYPE s_date,
        route   TYPE string,
        END OF ty_data.

CONSTANTS: gc_view TYPE string VALUE
            'test.a4h.book.chapter04::AT_FLIGHT'.
DATA: lt_data TYPE TABLE OF ty_data.

" Zugriff auf Attribute View
DATA(lv_statement) =
  | SELECT carrid, connid, fldate, route |
&& | FROM "{ gc_view }"|
&& | WHERE mandt = '{ sy-mandt }' ORDER BY fldate|.

TRY.
  " SQL-Verbindung und Statement vorbereiten
  DATA(lo_result_set) =
    cl_sql_connection=>get_connection(
      )->create_statement(
        tab_name_for_trace = conv #( gc_view )
      )->execute_query( lv_statement ).
```

```
" Resultat abholen
lo_result_set->set_param_table( REF #( lt_data ) ).
lo_result_set->next_package( ).
lo_result_set->close( ).
CATCH cx_sql_exception INTO DATA(lo_ex).
  " Fehlerbehandlung
&& | WHERE mandt = '{ sy-mandt }' ORDER BY fldate|.
  WRITE: | { lo_ex->get_text( ) } |.
ENDTRY.

LOOP AT lt_data ASSIGNING FIELD-SYMBOL(<1>).
  WRITE: / <1>-carrid , <1>-connid, <1>-fldate,
        <1>-route .
ENDLOOP.
```

Listing 5.1 Zugriff auf einen Attribute View über ADBC

Wie Sie sehen, handelt es sich um einen herkömmlichen nativen SQL-Zugriff. Falls es bei der Ausführung zu einem Fehler kommt, erhalten Sie im Text der SQL-Exception Hinweise auf die Ursache. Neben den SQL-Code-Fehlern, die Sie auch beim Zugriff über die SQL-Konsole sehen, können auch Fehler bei der Abbildung in die ABAP-Zielstruktur auftreten. Wir kommen darauf in den Empfehlungen in Abschnitt 5.1.4 zurück.

5.1.2 Externe Views im ABAP Dictionary

In ABAP 7.4 gibt es im ABAP Dictionary einen neuen View-Typ, einen sogenannten *externen View*, der es erlaubt, im SAP HANA Repository definierte Views in das ABAP Dictionary zu importieren. Der View wird als extern bezeichnet, weil er nicht vollständig im ABAP Dictionary definiert ist, sondern als eine Art *Proxy* (also Stellvertreter) dient, über den ein Zugriff aus ABAP auf den zugehörigen Column View im Schema `_SYS_BIC` möglich ist.

Einen externen View können Sie ausschließlich über die ABAP Development Tools in Eclipse definieren. Dazu legen Sie ein neues Entwicklungsobjekt vom Typ `DICTIONARY VIEW` an. Abbildung 5.1 zeigt den Anlagedialog anhand des Attribute Views `AT_FLIGHT`.

Während der Anlage wird geprüft, ob sich der View in das ABAP Dictionary importieren lässt. Sie müssen dabei beachten, dass nicht alle HANA-Datentypen in ABAP unterstützt werden. Wenn Sie also berechnete Attribute definieren oder in Views auf Tabellen zugreifen,

Externen View in Eclipse anlegen

Prüfung der Importierbarkeit

die nicht über das ABAP Dictionary angelegt wurden, können potenziell solche nicht unterstützten Datentypen auftauchen. In diesem Fall erhalten Sie einen Fehler beim Anlegen des externen Views und können diesen View nicht importieren. Die Liste der unterstützten Datentypen entnehmen Tabelle 3.1 in Abschnitt 3.1.3, »Datentypen«.

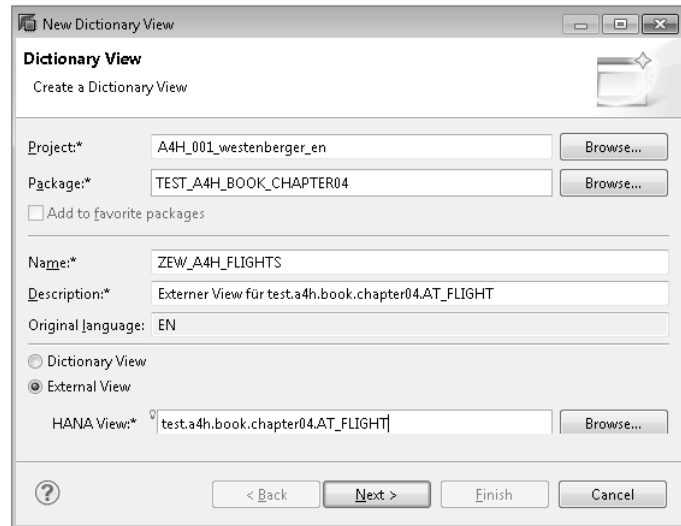


Abbildung 5.1 Anlegen eines externen Views im ABAP Dictionary

View-Struktur und Synchronisierung

Nach erfolgreichem Import des SAP HANA Views in das ABAP Dictionary zeigt der Editor die Struktur des Views mit Abbildung der Datentypen (siehe Abbildung 5.2). Zusätzlich gibt es in dem Dialog einen Button zum Synchronisieren (SYNCHRONIZE) des Views nach einer Änderung der Struktur des zugehörigen Views im SAP HANA Studio. Falls Sie also Attribute in die Ausgabestruktur aufnehmen, Attribute löschen oder Datentypen ändern, müssen Sie den externen View abgleichen, da es ansonsten zu Laufzeitfehlern kommt. Empfehlungen zur Synchronisierung von Entwicklungen in einem Entwicklungsteam geben wir Ihnen in Kapitel 14, »Praxistipps«.

Abbildung von Datentypen

Wie Sie in Abschnitt 3.1.3, »Datentypen«, erfahren haben, ist die Abbildung von SQL-Datentypen auf Dictionary-Typen nicht eindeutig. Vom Datentyp hängt aber die richtige Behandlung von Operationen ab (z. B. die Berechnung von Differenzen bei einem Datum). Aus diesem Grund müssen Sie als Entwickler die Zuordnung des richtigen ABAP-Datentyps manuell vornehmen.

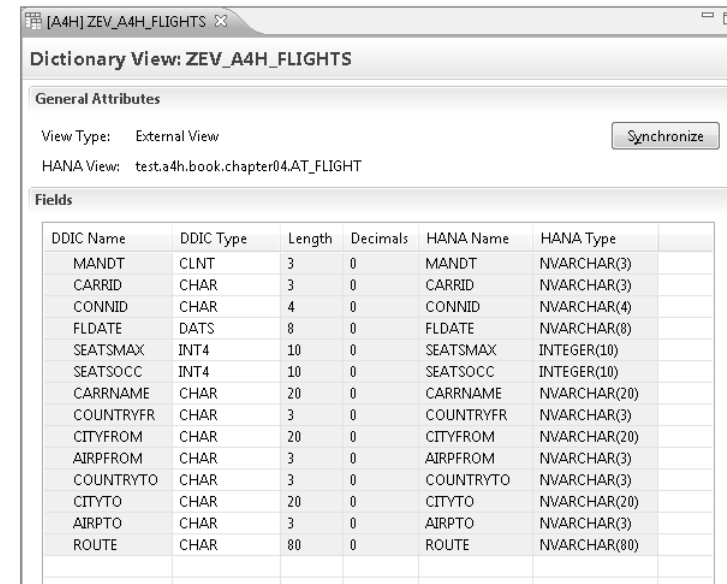


Abbildung 5.2 Externer ABAP Dictionary View, basierend auf einem Attribute View

Tabelle 5.1 zeigt anhand des Beispiel-Views AT_FLIGHT die möglichen Datentypzuordnungen für einige Spalten.

Spalte	SQL-Datentyp	Mögliche Dictionary-Typen
CARRID	NVARCHAR(3)	CHAR(3), NUMC(3), SSTR, CLNT, UNIT, CUKY
FLDATE	NVARCHAR(8)	CHAR(8), NUMC(8), SSTR, DATS
CARRNAME	NVARCHAR(20)	CHAR(20), NUMC(20), SSTR

Tabelle 5.1 Beispiel für mögliche Typzuordnungen

Bei dem externen View in Abbildung 5.2 haben wir daher manuell der Spalte FLDATE den ABAP-Datentyp DATS zugeordnet. Dies mag auf den ersten Blick merkwürdig erscheinen, da diese Information ja in der zugrunde liegenden Dictionary-Tabelle bereits vorliegt, die Spalten von Column Views in SAP HANA haben jedoch keine für das ABAP Dictionary ersichtliche Referenz auf Spalten existierender Tabellen. Zum Beispiel könnte die Spalte FLDATE auch ein berechnetes Attribut sein.

Die Definition eines externen Views, der auf einem Analytic View oder Calculation View basiert, funktioniert wie auf der Basis eines

Voraussetzungen

Attribute Views. Das ABAP Dictionary verfügt bei einem externen View aktuell über kein Wissen darüber, um welchen speziellen HANA-View-Typ es sich handelt. Voraussetzung für die Verwendung externer Views ist lediglich, dass der View im SAP HANA Repository definiert ist. Column Views, die lediglich im Datenbankkatalog vorhanden sind (z. B. durch eine Generierung), können Sie nicht in das ABAP Dictionary importieren.

Auf den Transport von externen Views (und anderen HANA-spezifischen Entwicklungen) gehen wir in Abschnitt 5.3, »Transport nativer Entwicklungsobjekte«, ein.

5.1.3 Zugriffsmöglichkeiten auf externe Views

Vorteile Der wesentliche Vorteil externer Views besteht darin, dass ein Zugriff auf SAP HANA Views auch über Open SQL möglich ist. Sie können daher insbesondere von folgenden Qualitäten in Open SQL profitieren:

- ▶ Syntaxprüfung durch den ABAP Compiler und Vorschlagswerte bei der Entwicklung (*Code Completion*)
- ▶ automatische Mandantenbehandlung
- ▶ Iterieren über eine Ergebnismenge innerhalb einer SELECT-Schleife
- ▶ Verwendung des Ausdrucks `INTO CORRESPONDING FIELDS` zur passenden Selektion in eine Zielstruktur unabhängig von der Reihenfolge in der Projektionsliste
- ▶ Verwendung von `IN` bei der `WHERE`-Bedingung zur Übertragung von Selektionsoptionen

Zugriff per Open SQL Listing 5.2 setzt den Zugriff auf den externen View aus Abbildung 5.2 um und entspricht funktional der ADBC-Zugriffsvariante aus Listing 5.1. Wie Sie sehen, hat sich der für den Zugriff notwendige ABAP-Code deutlich reduziert und entspricht einem Zugriff auf einen Standard-Dictionary-View.

```
REPORT ZR_A4H_CHAPTER4_VIEW_OPEN.
```

```
DATA: wa TYPE zev_a4h_flights.
" Daten von externem View lesen
SELECT carrid connid fldate route
      FROM zev_a4h_flights
```

```
      INTO CORRESPONDING FIELDS OF wa.
WRITE: / wa-carrid, wa-connid, wa-fldate, wa-route.
ENDSELECT.
```

Listing 5.2 Zugriff auf externen View über Open SQL

Mögliche Laufzeitfehler beim Zugriff auf externe Views

Auch bei dem Open-SQL-basierten Zugriff auf einen externen View wird letztlich eine SQL-Abfrage auf den zugehörigen Column View in SAP HANA ausgeführt. Dabei gelten die gleichen Regeln und Restriktionen wie bei einem Zugriff mit nativem SQL.

Wie wir in Abschnitt 4.4.4, »Laufzeitobjekte und SQL-Zugriff«, erklärt haben, müssen Sie etwa beim Zugriff auf Analytic Views über SQL einige Einschränkungen beachten. Eine nicht unterstützte Anfrage über Open SQL führt zu einem Laufzeitfehler. Dies erfordert von Ihnen als ABAP-Entwickler eine gewisse Vorsicht, da beim normalen Open-SQL-basierten Zugriff auf ABAP-Tabellen solche Fehler eher selten vorkommen. Auf die Werkzeuge zur Fehleranalyse und die möglichen Laufzeitfehler bei einem SQL-Zugriff kommen wir in Abschnitt 7.2, »Fehleranalyse«, zurück.

Neben dem Open-SQL-basierten Zugriff können Sie externe Views auch über natives SQL ansprechen. Diese auf den ersten Blick etwas merkwürdig erscheinende Variante ist dennoch sinnvoll, falls Sie etwa über eine SQL-Abfrage auf einen SAP HANA View zugreifen wollen, die über Open SQL nicht möglich ist. Ein Beispiel ist eine *Fuzzy-Suche* in einem Attribute View (siehe Abschnitt 10.4, »Einsatz der Textsuche in ABAP«). Im Vergleich zu einem Zugriff über natives SQL auf den generierten Column View im Schema `_SYS_BIC` hat der externe View den Vorteil, dass es dabei im ABAP Dictionary bereits eine geeignete Zielstruktur für eine Selektion über ADBC gibt.

[!]

Nativer Zugriff
über ADBC

5.1.4 Empfehlungen

Zum Abschluss dieses Abschnitts wollen wir Ihnen einige Empfehlungen für die Verwendung von SAP HANA Views geben. Wir beschränken uns dabei auf funktionale Empfehlungen. Für Werkzeuge und Empfehlungen zur Performanceanalyse verweisen wir auf Kapitel 7, »Laufzeit- und Fehleranalyse auf SAP HANA«, und Kapitel 14, »Praxistipps«, wo wir auch Designaspekte wie Namenskonventionen aufgreifen werden.

Falls der Funktionsumfang von Standard-ABAP-Dictionary-Views für Sie ausreicht und Sie diese in der Vergangenheit verwendet haben,

Verwendung der
View-Typen

gibt es keine Notwendigkeit, Ihre Anwendung auf einen nativen SAP HANA View umzubauen. Im nächsten Kapitel werden Sie über CDS-Views eine Möglichkeit kennenlernen, wie Sie auch direkt in ABAP komplexe Sichten mit berechneten Feldern definieren können.

Für spezielle analytische Szenarien bieten allerdings die modellierten SAP HANA Views einen einfachen Zugang. Welcher der drei vorgestellten View-Typen in SAP HANA für Ihr Szenario der richtige ist, können Sie über den folgenden Fragenkatalog bestimmen:

- ▶ Geht es um eine Sicht auf Stammdaten, die eventuell durch berechnete Attribute erweitert werden? Hier sollte der Attribute View der Startpunkt sein.
- ▶ Handelt es sich um eine Datenanalyse von Bewegungsdaten, basierend auf einem Sternschema? In diesem Fall sollte der Analytic View der erste Anlaufpunkt sein, wobei Sie die Dimensionen als Attribute Views realisieren.
- ▶ Müssen Sie Ergebnisse aus verschiedenen Tabellen und HANA-Views zusammenbringen oder adaptieren? In diesem Fall bietet sich der modellierte Calculation View an. Falls die modellierte Variante für einen Teil Ihres Szenarios nicht ausreicht, können Sie für diesen Teil auf eine SQLScript-basierte Implementierung ausweichen.

Mandanten- behandlung

Bei der Modellierung von Views sollten Sie auf die richtige Behandlung des Mandanten achten. Wir empfehlen Ihnen auch, stets das Mandantenfeld als erstes Feld des Views aufzunehmen und sicherzustellen, dass Sie den Mandanten in die Join-Modellierung aufgenommen haben. In den meisten Fällen sollte der Konfigurationswert `SESSION CLIENT` für Views, die auf ABAP-Tabellen aus dem gleichen System basieren, die richtige Einstellung sein. Falls Tabellen durch eine Replikation aus einem anderen System entstanden sind, kann auch ein fester Wert für den Mandanten sinnvoll sein. Nur in wenigen Fällen macht hingegen ein Zugriff über Mandantengrenzen hinweg Sinn.

Schema- behandlung

Sie sollten für Analytic und Calculation Views stets das richtige Standardschema auswählen. In diesem Schema wird insbesondere das für Konvertierungen relevante Customizing gesucht, wenn keine spezielle Einstellung am Attribut vorgenommen wurde. Noch wichtiger ist die Einstellung des Standardschemas bei der implementierten Variante von Calculation Views.

Sie sollten externe Views nur für solche SAP HANA Views definieren, über die Sie aus ABAP zugreifen wollen, da Sie diese bei Änderungen manuell synchronisieren müssen. Ebenso ist es nicht empfehlenswert, mehrere externe Views für einen SAP HANA View zu definieren.

Definition von externen Views

Falls Sie bei der Aktivierung von SAP HANA Views Fehlermeldungen erhalten, finden Sie in der Regel im Fehlertext Hinweise auf die Ursache. Die richtige Interpretation der Fehlermeldung erfordert jedoch in manchen Fällen ein wenig Erfahrung. Aus diesem Grund ist eine gewisse Heuristik bei der Fehleranalyse sinnvoll. Zunächst sollten Sie sicherstellen, dass Sie bei Attribute Views mindestens ein Feld als Schlüsselfeld ausgezeichnet sowie bei Analytic Views mindestens eine Kennzahl definiert haben. Falls es berechnete Attribute in Ihrem View gibt, sollten Sie prüfen, ob Sie eventuell bei dem zugehörigen Ausdruck einen Fehler gemacht haben.

Fehleranalyse

Falls Sie bei der Fehleranalyse einmal nicht weiterkommen, können Sie das zugehörige Attribut (z. B. in einer Kopie des Views) probe-weise einmal entfernen. Falls Sie beim Aufruf des Data Previews eine Fehlermeldung oder unerwartete Daten bekommen, deutet das in vielen Fällen auf einen Fehler bei der Join-Modellierung hin. Im Fall von Währungskonvertierungen kann es durch einen fehlenden Mandantenkontext zu Fehlern kommen.

Beim Zugriff auf einen SAP HANA View aus ABAP über natives SQL sollten Sie den Namen des Views mitgeben (über den Parameter `tab_name_for_trace` wie in Listing 5.1 oder über die Methode `SET_TABLE_NAME_FOR_TRACE`), der in Support-Szenarien eine einfachere Fehleranalyse erlaubt.

5.2 Einbettung von nativen Prozeduren in ABAP

In Kapitel 4, »Native Datenbankentwicklung mit SAP HANA«, haben Sie gelernt, was SQLScript ist und wie Sie es zur Implementierung von Datenbankprozeduren nutzen können. Nun möchten wir Ihnen erklären, wie Sie Datenbankprozeduren aus ABAP aufrufen. Dabei unterscheiden wir zwei Möglichkeiten:

- ▶ Zugriff über *natives SQL* und *ABAP Database Connectivity (ADBC)*, siehe auch Kapitel 3, »Datenbankprogrammierung mit dem SAP NetWeaver AS ABAP«)
- ▶ Nutzung von sogenannten *Database Procedure Proxies*

Voraussetzungen Der Aufruf von Datenbankprozeduren in SAP HANA über ADBC ist ab dem ABAP-Release 7.0 und SAP-Kernel 7.20 möglich. Die Database Procedure Proxies stehen ab dem Release 7.4 zur Verfügung. Sie setzen voraus, dass SAP HANA als Primärdatenbank genutzt wird. Außerdem unterstützen Database Procedure Proxies ausschließlich das – eigentlich veraltete – XML-Dateiformat (*.procedure*).

5.2.1 Zugriff über natives SQL

Wie in Abschnitt 4.3, »Datenbankprozeduren«, beschrieben, erzeugt das System bei der Aktivierung einer Datenbankprozedur verschiedene Laufzeitobjekte im Datenbankkatalog (z. B. im Schema `_SYS_BIC`) sowie ein öffentliches Synonym. Darüber können Sie über natives SQL aus ABAP auf die Datenbankprozedur zugreifen.

Nachteile von nativem SQL Die Verwendung von nativem SQL zum Aufruf einer Datenbankprozedur ist allerdings verhältnismäßig umständlich und fehleranfällig. Tabellarische Eingabe- und Ausgabeparameter können Sie, wie Sie im weiteren Verlauf dieses Abschnitts sehen werden, nur über temporäre Tabellen mit der Datenbankprozedur austauschen. Außerdem erkennt der SAP NetWeaver AS ABAP Syntaxfehler in nativen SQL-Anweisungen erst zur Laufzeit. Für mehr Details verweisen wir Sie auf die Erläuterungen in Kapitel 3, »Datenbankprogrammierung mit dem SAP NetWeaver AS ABAP«.

Beispiele Wir beschreiben Ihnen die Verwendung von nativem SQL zum Zugriff auf Datenbankprozeduren anhand mehrerer Beispiele im Detail. Zunächst betrachten wir dazu eine Datenbankprozedur, die auf Basis der Kurzbezeichnung den Namen einer Fluggesellschaft ermittelt. Für die anschließend folgenden Beispiele greifen wir auf bereits bekannte Datenbankprozeduren aus Kapitel 4, »Native Datenbankentwicklung mit SAP HANA«, zurück.

Beispiel 1: Aufruf einer Datenbankprozedur

Für den Aufruf einer Datenbankprozedur über ADBC stellt die Klasse `CL_SQL_STATEMENT` die Methode `EXECUTE_PROCEDURE` zur Verfügung.

Diese können Sie nutzen, solange eine Datenbankprozedur keine tabellarischen Eingabe-/Ausgabeparameter hat.

Das Programm `ZR_A4H_CHAPTER5_CARRNAME_ADBC` zeigt die Verwendung der Methode `EXECUTE_PROCEDURE` beispielhaft (siehe Listing 5.3). Es ruft die Datenbankprozedur `DETERMINE_CARRNAME` auf. Diese hat folgende Eingabe- und Ausgabeparameter:

Beispiel für Aufruf

- ▶ `IV_MANDT` (Mandant)
- ▶ `IV_CARRID` (Kurzbezeichnung einer Fluggesellschaft)
- ▶ `EV_CARRNAME` (Name einer Fluggesellschaft)

PARAMETERS: `p_carrid` TYPE `s_carr_id`.

DATA: `lo_sql_statement` TYPE REF TO `cl_sql_statement`,
`lv_carrname` TYPE `s_carrname`.

```
TRY.
  " create statement
  lo_sql_statement =
    cl_sql_connection=>get_connection(
    )->create_statement( ).

  " bind parameters
  lo_sql_statement->set_param( data_ref =
    REF #( sy-mandt )
    inout = cl_sql_statement=>c_param_in ).

  lo_sql_statement->set_param( data_ref =
    REF #( p_carrid )
    inout = cl_sql_statement=>c_param_in ).

  lo_sql_statement->set_param( data_ref =
    REF #( lv_carrname )
    inout = cl_sql_statement=>c_param_out ).

  " call procedure
  lo_sql_statement->execute_procedure(
  'test.a4h.book.chapter04::DETERMINE_CARRNAME' ).

CATCH cx_sql_exception INTO DATA(lo_ex).
  " error handling
  WRITE: | { lo_ex->get_text( ) } |.
ENDTRY.
```

WRITE: / `lv_carrname`.

Listing 5.3 Aufruf einer Datenbankprozedur über natives SQL

Erläuterung des Programms Das Programm erzeugt zunächst eine Instanz der Klasse `CL_SQL_STATEMENT`. Danach belegt es die Eingabe- und Ausgabeparameter der Datenbankprozedur durch Aufruf der Methode `SET_PARAM` mit den Aktualparametern. Anschließend ruft es die Methode `EXECUTE_PROCEDURE` auf.

Beispiel 2: Tabellarische Ausgabeparameter

Alternativ können Sie eine Datenbankprozedur auch über die Methode `EXECUTE_QUERY` (zusammen mit dem Zusatz `WITH OVERVIEW`) ausführen. Dies funktioniert auch bei Datenbankprozeduren mit tabellarischen Eingabe- und Ausgabeparametern.

Beispiel für Ausgabeparameter Das Programm `ZR_A4H_CHAPTER5_TOP_ADDBC` in Listing 5.4 zeigt die Verwendung der Methode `EXECUTE_QUERY` beispielhaft, indem es die Datenbankprozedur `DETERMINE_TOP_CONNECTIONS` aufruft. Diese ermittelt die wichtigsten Verbindungen einer Fluggesellschaft und hat folgende Eingabe- und Ausgabeparameter:

- ▶ `IV_MANDT` (Mandant)
- ▶ `IV_CARRID` (Kurzbezeichnung einer Fluggesellschaft)
- ▶ `IV_ALGORITHM` (steuert, wie die wichtigsten Verbindungen ermittelt werden sollen)
- ▶ `ET_CONNECTIONS` (Tabellenparameter; enthält die Kurzbezeichnung `CARRID` der Fluggesellschaft sowie den Verbindungscode `CONNID`)

PARAMETERS: `p_carrid` TYPE `s_carr_id`.

" Definition der Resultatstruktur

```
TYPES: BEGIN OF ty_connections,
        carrid TYPE s_carr_id,
        connid TYPE s_conn_id,
      END OF ty_connections.
```

```
DATA: lt_connections TYPE TABLE OF ty_connections,
      lv_statement TYPE string,
      lo_result_set TYPE REF TO cl_sql_result_set,
      lo_connections TYPE REF TO data.
```

TRY.

```
" lokale temporäre Tabelle löschen
lv_statement = | DROP TABLE #ET_CONNECTIONS |.
cl_sql_connection=>get_connection(
)->create_statement( )->execute_ddl( lv_statement ).
```

```
CATCH cx_sql_exception.
" unter Umständen existiert die lokale temporäre
" Tabelle nicht, diesen Fehler ignorieren wir
ENDTRY.
```

TRY.

```
" lokale temporäre Tabelle anlegen
lv_statement = | CREATE LOCAL TEMPORARY ROW|
&& | TABLE #ET_CONNECTIONS LIKE "_SYS_BIC".|
&& | "test.a4h.book.chapter04::GlobalTypes.t|
&& | t_connections" |.
cl_sql_connection=>get_connection(
)->create_statement( )->execute_ddl( lv_statement ).
```

" Datenbankprozedur aufrufen

```
lv_statement = | CALL "test.a4h.bo|
&& | ok.chapter04::DETERMINE_TOP_CONNECTIONS|
&& | "( '{ sy-mandt }', '{ p_carrid }', 'P'|
&& | , #ET_CONNECTIONS ) WITH OVERVIEW |.
lo_result_set = cl_sql_connection=>get_connection(
)->create_statement( )->execute_query(
lv_statement ).
lo_result_set->close( ).
```

" lokale temporäre Tabelle auslesen

```
lv_statement = | SELECT * FROM #ET_CONNECTIONS |.
lo_result_set = cl_sql_connection=>get_connection(
)->create_statement( )->execute_query(
lv_statement ).
```

" Resultat auslesen

```
GET REFERENCE OF lt_connections INTO
lo_connections.
lo_result_set->set_param_table( lo_connections ).
lo_result_set->next_package( ).
lo_result_set->close( ).
CATCH cx_sql_exception INTO DATA(lo_ex).
```

" Fehlerbehandlung

```
WRITE: | { lo_ex->get_text( ) } |.
ENDTRY.
```

LOOP AT `lt_connections` ASSIGNING

```
FIELD-SYMBOL(<ls_connections>).
WRITE: / <ls_connections>-carrid ,
       <ls_connections>-connid.
```

ENDLOOP.

Listing 5.4 Behandlung von tabellarischen Ausgabeparametern

Temporäre Tabellen Wir möchten Ihnen anhand des Programms vor allem die Übergabe tabellarischer Eingabe- und Ausgabeparameter an eine Datenbankprozedur erläutern: Das Programm ZR_A4H_CHAPTER5_TOP_ADBC verwendet zur Übergabe des Tabellenparameters ET_CONNECTIONS die *temporäre Tabelle* #ET_CONNECTIONS.

» Temporäre Tabellen

Viele Datenbanken, so auch die HANA-Datenbank, bieten Ihnen die Möglichkeit, Zwischen- und Endergebnisse von Kalkulationen *vorübergehend* in sogenannten *temporären Tabellen* zu speichern. Temporäre Tabellen haben, verglichen mit herkömmlichen Tabellen, verschiedene Vorteile für diesen Anwendungsfall:

- ▶ Tabellendefinition und Tabelleninhalt werden von der Datenbank automatisch gelöscht, wenn sie nicht mehr benötigt werden.
- ▶ Die Datenbank isoliert die Daten paralleler Sitzungen (*Sessions*) automatisch voneinander. Sperren auf temporären Tabellen sind weder notwendig noch möglich.
- ▶ Für temporäre Tabellen schreibt die Datenbank kein Transaktionslog.
- ▶ Die Verwendung temporärer Tabellen ist in der Regel performanter als die Verwendung herkömmlicher Tabellen.

SAP HANA unterstützt globale und lokale temporäre Tabellen:

- ▶ Die Tabellendefinition *globaler* temporärer Tabellen ist sitzungsübergreifend nutzbar. Der Tabelleninhalt ist nur für die aktuelle Session sichtbar, er wird zum Sitzungsende von der Datenbank automatisch gelöscht.
- ▶ Bei *lokalen temporären Tabellen* sind sowohl Tabellendefinition als auch Tabelleninhalt nur für die aktuelle Session sichtbar, d. h., beide werden am Ende der Sitzung von der Datenbank automatisch gelöscht.

Nutzung über den AS ABAP Hinsichtlich der Nutzung temporärer Tabellen für die Datenübergabe zwischen dem SAP NetWeaver AS ABAP und einer Datenbankprozedur sollten Sie folgende Dinge beachten:

- ▶ Wenn Sie mit globalen temporären Tabellen arbeiten, können Sie diese (da sie sitzungsübergreifend nutzbar sind) einmalig anlegen, müssen allerdings organisatorisch sicherstellen, dass der Tabellenname nicht für verschiedene Anwendungsfälle (die eine unterschiedliche Tabellenstruktur voraussetzen) verwendet wird.
- ▶ Die Anlage globaler temporärer Tabellen kann bereits zur *Design-time* erfolgen. Dann müssen Sie dafür sorgen, dass die Tabellen nach einem Transport auch in Test- und Produktivsystemen zur Verfügung stehen.

- ▶ Falls Sie sich zur Anlage globaler temporärer Tabellen zur *Laufzeit (Runtime)* entscheiden, müssen Sie vor jedem Aufruf einer Datenbankprozedur sicherstellen, dass die Tabellenstruktur zur Schnittstelle der aufgerufenen Datenbankprozedur passt (denn diese könnte sich ja zwischenzeitlich geändert haben).
- ▶ Lokale temporäre Tabellen müssen Sie zumindest einmalig pro Session anlegen (beachten Sie dazu auch die nachfolgenden Erläuterungen zum Verhältnis von ABAP-Workprozess und Datenbankverbindung). Daher können Sie lokale temporäre Tabellen nur zur Run Time eines ABAP-Programms anlegen.
- ▶ Da jeder ABAP-Workprozess über eine stehende Verbindung mit der Datenbank verbunden ist, stellen mehrere nacheinander durch den gleichen Workprozess ausgeführte ABAP-Programme aus Datenbanksicht eine Session dar. Nach Beendigung eines ABAP-Programms werden daher weder Definition noch Inhalt lokaler (und globaler) temporärer Tabellen automatisch gelöscht.
- ▶ Sowohl für globale als auch für lokale temporäre Tabellen sollten Sie den Inhalt (der aktuellen Sitzung) vor Aufruf der Datenbankprozedur löschen.

Das Programm ZR_A4H_CHAPTER5_TOP_ADBC in Listing 5.4 arbeitet mit einer lokalen temporären Tabelle. Es löscht zunächst über `DROP TABLE #ET_CONNECTIONS` die eventuell bereits vorhandene lokale temporäre Tabelle #ET_CONNECTIONS. Anschließend legt es über die Anweisung `CREATE LOCAL TEMPORARY ROW TABLE` eine (neue) lokale temporäre Tabelle mit dem Namen #ET_CONNECTIONS an. Dabei bezieht sich das Programm auf den Tabellentyp, der beim Aktivieren der Datenbankprozedur vom System automatisch für den Ausgabeparameter ET_CONNECTIONS angelegt wurde. Durch dieses Vorgehen stellt das Programm vor dem Aufruf der Datenbankprozedur sicher, dass die temporäre Tabelle leer ist und zur aktuellen Struktur des Ausgabeparameters ET_CONNECTIONS passt.

Jetzt ruft das Programm die Datenbankprozedur über die Methode `EXECUTE_QUERY` auf. Dabei übergibt es SY-MANDT, P_CARRID und 'P' an die Eingabeparameter und die temporäre Tabelle #ET_CONNECTIONS an den Ausgabeparameter der Datenbankprozedur.

Nach dem Aufruf der Datenbankprozedur liest das Programm den Inhalt der temporären Tabelle #ET_CONNECTIONS aus. Der Inhalt ent-

Erläuterung des Programms

spricht den wichtigsten Verbindungen der übergebenen Fluggesellschaft.

Beispiel 3: Tabellarische Eingabeparameter

Wenn eine Datenbankprozedur über tabellarische Eingabeparameter verfügt, können Sie analog zu tabellarischen Ausgabeparametern vorgehen. Das Programm ZR_A4H_CHAPTER5_KPIS_ADBC in Listing 5.5 zeigt, wie Sie die Datenbankprozedur GET_KPIS_FOR_CONNECTIONS für eine Menge von Flugverbindungen aufrufen können. Die Datenbankprozedur ermittelt pro übergebener Verbindung einige Kennzahlen.

Beispiel für Eingabeparameter

Sie hat folgende Eingabe- und Ausgabeparameter:

- ▶ IV_MANDT (Mandant)
- ▶ IT_CONNECTIONS (Tabellenparameter; enthält die Kurzbezeichnung CARRID der Fluggesellschaft sowie den Verbindungscode CONNID)
- ▶ ET_KPIS (Tabellenparameter; enthält die Kennzahlen der Verbindungen)

```
...
LOOP AT lt_connections INTO ls_connections.
  lv_statement = | INSERT INTO #IT_CONNECTIONS VALUES
    ( '{ ls_connections-carrid }', '{ ls_connections-connid }' )|.
  cl_sql_connection=>get_connection(
    )->create_statement(
    )->execute_update( lv_statement ).
ENDLOOP.
" Datenbankprozedur aufrufen
lv_statement = | CALL "test.a4h.bo|
&& |ok.chapter04::GET_KPIS_FOR_CONNECTIONS|
&& |( '{ sy-mandt }', #IT_CONNECTIONS, #ET_KPIS )
  WITH OVERVIEW |.
lo_result_set = cl_sql_connection=>get_connection(
  )->create_statement( )->execute_query( lv_statement ).
lo_result_set->close( ).
...
```

Listing 5.5 Behandlung von tabellarischen Eingabeparametern

Erläuterung des Programms

Vor dem Aufruf der Datenbankprozedur füllt das Programm die lokale temporäre Tabelle #IT_CONNECTIONS mit den gewünschten Flugverbindungen. Der Aufruf der Datenbankprozedur erfolgt über EXECUTE_QUERY.

5.2.2 Definition von Database Procedure Proxies

Ab dem ABAP-Release 7.4 können Sie einen sogenannten *Database Procedure Proxy* definieren, um aus ABAP nativ auf Datenbankprozeduren zuzugreifen, die im SAP HANA Repository der Primärdatenbank definiert worden sind. Beachten Sie, dass nur das XML-Dateiformat (*procedure*) unterstützt wird (siehe Abschnitt 4.3, »Datenbankprozeduren«).

Bei einem Database Procedure Proxy handelt es sich (wie der Name bereits vermuten lässt) um ein *Proxy-Objekt*. Dieses repräsentiert eine Datenbankprozedur im ABAP Dictionary.

Mehrere Proxy-Objekte für eine Datenbankprozedur

Technisch ist es möglich, für eine Datenbankprozedur mehrere Database Procedure Proxies anzulegen. Wir empfehlen dies jedoch *nicht*. Legen Sie zu einer Datenbankprozedur immer maximal ein Proxy-Objekt im ABAP Dictionary an.

[!]

Für jeden Database Procedure Proxy legt das System automatisch auch ein Interface an. Über dieses Interface können Sie Parameternamen und Datentypen, die beim Aufruf der Datenbankprozedur mit ABAP verwendet werden, beeinflussen:

Interface des Proxy-Objekts

- ▶ Die Namen von Eingabe- und Ausgabeparametern können Sie in SAP HANA ändern, sobald diese 30 Zeichen überschreiten. In diesem Fall verkürzt das System die Parameternamen zunächst. Sie können die verkürzten Namen bei Bedarf überschreiben.
- ▶ Komponentennamen von Tabellenparametern können Sie immer überschreiben.
- ▶ Jedem Parameter können Sie den zu verwendenden Datentyp zuordnen. Dies ist wichtig, da die Abbildung von SQL-Datentypen auf ABAP-Datentypen und Dictionary-Datentypen nicht eindeutig ist. Daher kann das System bei der Anlage eines Proxy-Objekts nicht (immer) den richtigen ABAP-Datentyp bzw. Dictionary-Typ ableiten.

Wir werden Ihnen nun erläutern, wie Sie für die Datenbankprozedur DETERMINE_TOP_CONNECTIONS_XML ein Proxy-Objekt anlegen. Dazu müssen Sie sich in den ABAP Development Tools in Eclipse befinden. Dort wählen Sie den Menüpunkt FILE • NEW • OTHER... aus. Anschließend selektieren Sie den Eintrag DATABASE PROCEDURE PROXY und kli-

Anlage eines Database Procedure Proxys

cken auf NEXT. Abbildung 5.3 zeigt das Fenster, das daraufhin erscheint.

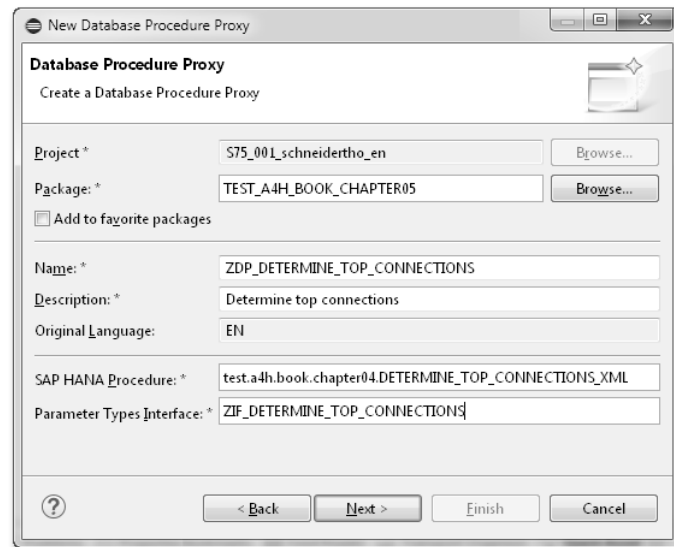


Abbildung 5.3 Anlegen eines Database Procedure Proxys

Parameter für die Anlage

In diesem Fenster erfassen Sie für den Database Procedure Proxy folgende Daten:

- ▶ **NAME:** Über den Namen des Database Procedure Proxys können Sie die Datenbankprozedur (später) nativ in ABAP aufrufen.
- ▶ **DESCRIPTION:** Bei der Beschreibung handelt es sich um einen erläuternden Text.
- ▶ **SAP HANA PROCEDURE:** Name der (bereits existierenden) Datenbankprozedur im SAP HANA Repository
- ▶ **PARAMETER TYPES INTERFACE:** Name des Interface, das beim Anlegen des Proxy-Objekts automatisch angelegt wird (siehe Listing 5.6)

Nach nochmaligem Klick auf NEXT und anschließendem Klicken des Buttons FINISH legt das System den Database Procedure Proxy und das entsprechende Interface an.

Im PROJECT EXPLORER finden Sie den Database Procedure Proxy im entsprechenden Paket unterhalb des Knotens DICTIONARY • DB PROCEDURE PROXIES. Das Interface liegt (ebenso wie andere Interfaces) im entsprechenden Paket unterhalb des Knotens SOURCE LIBRARY.

Abbildung 5.4 zeigt den Database Procedure Proxy für die Datenbankprozedur DETERMINE_TOP_CONNECTIONS_XML. Wenn Sie Parameternamen oder Datentypen anpassen möchten, können Sie dies in den Spalten ABAP NAME, ABAP TYPE und DDIC TYPE OVERRIDE tun. Zum Beispiel können Sie die Spalte CONNID des tabellarischen Ausgabeparameters ET_CONNECTIONS auf das Datenelement S_CONN_ID (und damit auf den ABAP-Datentyp N length 4) abbilden.

Anpassung des Interface

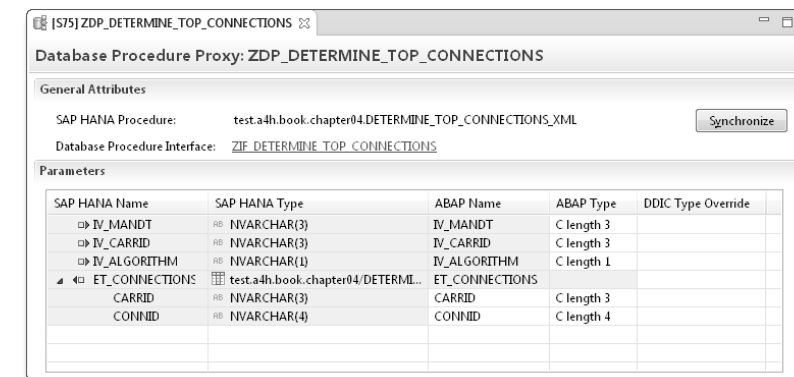


Abbildung 5.4 Database Procedure Proxy und Interface

Listing 5.6 zeigt das automatisch angelegte Interface nach Anpassung der Datentypen.

```
interface ZIF_DETERMINE_TOP_CONNECTIONS public.
types: iv_mandt type mandt.
types: iv_carrid type s_carr_id.
types: iv_algorithm type c length 1.
types: begin of et_connections,
       carrid type s_carr_id,
       connid type s_conn_id,
       end of et_connections.
endinterface.
```

Listing 5.6 Interface des Proxy-Objekts

5.2.3 Aufruf von Database Procedure Proxies

Nach der Aktivierung des Database Procedure Proxys können Sie das Proxy-Objekt zum Aufruf der Datenbankprozedur verwenden. Das Programm ZR_A4H_CHAPTER5_TOP_PROXY in Listing 5.7 zeigt die Verwendung beispielhaft.

```

PARAMETERS: p_carrid TYPE s_carr_id.

DATA: lt_connections TYPE TABLE OF
      zif_determine_top_connections=>et_connections.

TRY.
  CALL DATABASE PROCEDURE
    zdp_determine_top_connections
    EXPORTING
      iv_mandt = sy-mandt
      iv_carrid = p_carrid
      iv_algorithm = 'P'
    IMPORTING
      et_connections = lt_connections.

  CATCH cx_sy_db_procedure_sql_error
    cx_sy_db_procedure_call INTO DATA(lo_ex).
    " Fehlerbehandlung
      iv_algorithm = 'P'
    WRITE: | {lo_ex->get_text( ) } |.
ENDTRY.

LOOP AT lt_connections ASSIGNING
  FIELD-SYMBOL(<ls_connections>).
  WRITE: / <ls_connections>-carrid ,
        <ls_connections>-connid.
ENDLOOP.

```

Listing 5.7 Aufruf eines Database Procedure Proxys

Erläuterung des Programms

Das Programm verwendet die Anweisung `CALL DATABASE PROCEDURE`, um über den Proxy `ZDP_DETERMINE_TOP_CONNECTIONS` die Datenbankprozedur `DETERMINE_TOP_CONNECTIONS_XML` aufzurufen. Bei der Definition der internen Tabelle `LT_CONNECTIONS` bezieht sich das Programm auf das Interface `ZIF_DETERMINE_TOP_CONNECTIONS`. Beim Aufruf der Datenbankprozedur eventuell auftretende Probleme (Ausnahmen vom Typ `CX_SY_DB_PROCEDURE_SQL_ERROR` sowie `CX_SY_DB_PROCEDURE_CALL`) fängt das Programm ab.

5.2.4 Anpassung von Database Procedure Proxies

Wenn Sie eine Datenbankprozedur – genauer gesagt die Schnittstelle einer Datenbankprozedur – im SAP HANA Studio ändern, nachdem Sie einen Database Procedure Proxy angelegt haben, müssen Sie das Proxy-Objekt mit dem SAP HANA Repository synchronisieren. Dazu

steht Ihnen der Button `SYNCHRONIZE` (siehe Abbildung 5.4) zur Verfügung.

Während des Synchronisationsvorgangs können Sie entscheiden, ob Sie am Proxy-Objekt vorgenommene Anpassungen (Komponentennamen oder Datentypen) beibehalten oder überschreiben wollen.

In Kapitel 6, »Erweiterte Datenbankprogrammierung mit ABAP 7.4«, lernen Sie ABAP-Datenbankprozeduren (*ABAP Managed Database Procedures*) kennen. Diese haben – bei der Verwendung im Rahmen von ABAP – im Vergleich zur Verwendung von Prozeduren, die Sie über das SAP HANA Studio angelegt haben, mehrere Vorteile. Daher empfehlen wir grundsätzlich die Nutzung von ABAP-Datenbankprozeduren, wenn Sie SQLScript innerhalb von ABAP nutzen möchten.

5.3 Transport nativer Entwicklungsobjekte

In diesem Abschnitt möchten wir Ihnen erklären, wie Sie ABAP-Programme, die native HANA-Objekte nutzen, konsistent in Ihrer Systemlandschaft transportieren können. Wir beschäftigen uns dazu mit dem sogenannten *HANA-Transportcontainer*. Auf das *erweiterte Change and Transport System (CTS+)*, das Ihnen ebenfalls Möglichkeiten bietet, gehen wir nicht ein.

Wir nehmen für unsere Ausführungen an, dass Sie sich mit der Entwicklungsorganisation und dem Transport im SAP NetWeaver AS ABAP bereits auskennen.

5.3.1 Exkurs: Entwicklungsorganisation und Transport in SAP HANA

Damit Sie die Funktionsweise des HANA-Transportcontainers (besser) verstehen, geben wir Ihnen in diesem Abschnitt einige Hintergrundinformationen zur Entwicklungsorganisation und zum Transport in SAP HANA.

Entwicklungsorganisation

Die Entwicklungsorganisation in SAP HANA ähnelt in vielerlei Hinsicht der im SAP NetWeaver AS ABAP. Sie unterscheidet sich aber auch in einigen wesentlichen Aspekten von ihr. Wie in Kapitel 2,

»Einführung in die Entwicklungsumgebung«, beschrieben, ist das SAP HANA Repository die zentrale Ablage von Entwicklungsobjekten der HANA-Datenbank.

Namensraum für Kunden Innerhalb des Repositorys liefert SAP Content unterhalb des Wurzelpakets `sap` aus. Unterhalb dieses Pakets dürfen daher keine Kundenentwicklungen angelegt werden, da diese andernfalls versehentlich überschrieben werden könnten. Bauen Sie stattdessen eine parallele Pakethierarchie für Kundenentwicklungen auf. Verwenden Sie als Wurzelpaket z. B. Ihren Domänennamen.

Lokale Entwicklungen Einen Sonderfall stellt das Paket `system-local` dar. Es ähnelt von der Idee her den lokalen Paketen des SAP NetWeaver AS ABAP. Verwenden Sie es für Entwicklungsobjekte, die nicht transportiert werden sollen.

Transport

Delivery Units Ein Transport erfolgt in SAP HANA in der Regel auf Basis einer *Delivery Unit* (übersetzt heißt das so viel wie *Liefereinheit*). Eine Delivery Unit fasst Pakete zusammen, die gemeinsam transportiert bzw. ausgeliefert werden sollen. Sie entspricht konzeptionell weitestgehend einer Softwarekomponente im Sinn des AS ABAP. Während Sie im AS ABAP allerdings in der Regel mit der Softwarekomponente `HOME` arbeiten, müssen Sie in SAP HANA immer eigene Delivery Units für Kundenentwicklungen anlegen. Voraussetzung dafür ist, dass Sie bzw. ein Administrator vorher den Systemparameter `content_vendor` in der Datei `indexserver.ini` über die `ADMINISTRATION CONSOLE` des SAP HANA Studios gepflegt haben.

Zuweisung Delivery Unit Betrachten wir die Zuweisung einer Delivery Unit und den anschließenden Transport anhand eines Attribute Views `AT_CUSTOMER`. Bei der Anlage des Attribute Views `AT_CUSTOMER` ordnen Sie diesem ein Paket zu. In den Eigenschaften des Pakets können Sie eine Delivery Unit pflegen. Verwenden Sie dazu den Kontextmenüeintrag `EDIT` des Pakets. Alle im System vorhandenen Delivery Units sehen Sie in der Ansicht `QUICK VIEW` über den Menüeintrag `DELIVERY UNITS`. Dort können Sie auch neue Delivery Units anlegen. Abbildung 5.5 veranschaulicht den Zusammenhang zwischen Entwicklungsobjekt, Paket und Delivery Unit am Beispiel des Attribute Views `AT_CUSTOMER` (die Delivery Unit `ZA4H_BOOK_CHAPTER05` ist *nicht* Teil der mit diesem Buch ausgelieferten Beispiele).

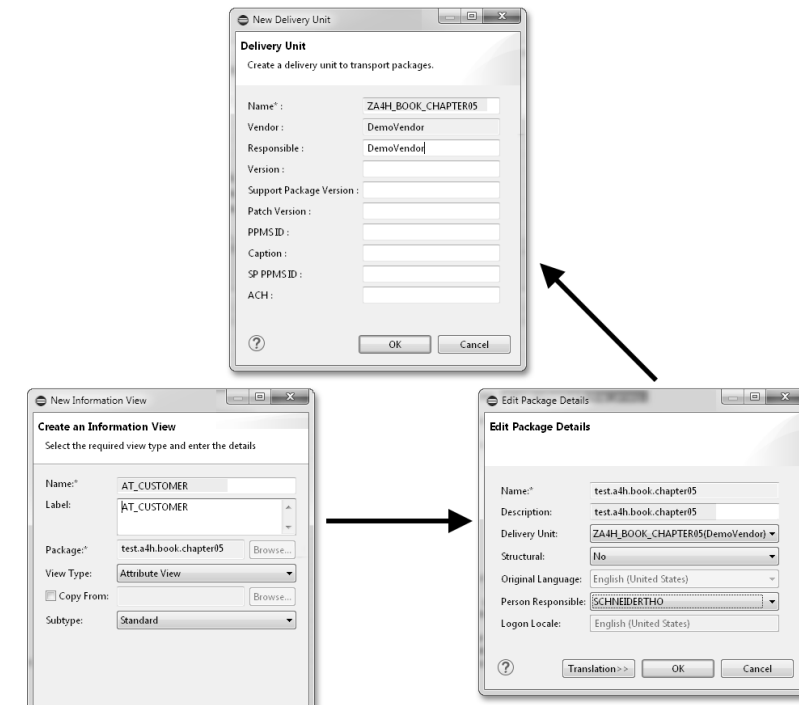


Abbildung 5.5 Entwicklungsobjekt, Paket und Delivery Unit

Im SAP HANA Studio können Sie Entwicklungsobjekte auf zwei Arten transportieren, d. h. exportieren und im Zielsystem importieren: **Import und Export**

- ▶ Export/Import einer Delivery Unit (optional gekoppelt mit CTS+)
- ▶ Export/Import einzelner Objekte (der sogenannte *Developer Mode*)

Für einen konsistenten Transport von HANA-Content (der nicht eng mit einer ABAP-Entwicklung gekoppelt ist) in einer produktiven Systemlandschaft empfehlen wir grundsätzlich den Export/Import auf Basis von Delivery Units und des CTS+.

Schema-Mapping

Eine Besonderheit beim Transport von HANA-Content ist das sogenannte *Schema-Mapping*. Ein Schema-Mapping ist notwendig, wenn die Datenbankschemata im Quellsystem und Zielsystem eines Transports voneinander abweichen. Es handelt sich um eine Abbildung eines Entwicklungsschemas (*Authoring Schema*) auf ein physikalisches Schema (*Physical Schema*).

Sie pflegen ein Schema-Mapping in der Ansicht QUICK VIEW über den Menüeintrag SCHEMA MAPPING. Bevor wir genauer darauf eingehen, wann und wie das System es auswertet, möchten wir die Notwendigkeit für das Schema-Mapping zunächst anhand des Attribute Views AT_CUSTOMER erläutern. Betrachten Sie hierzu Abbildung 5.6.

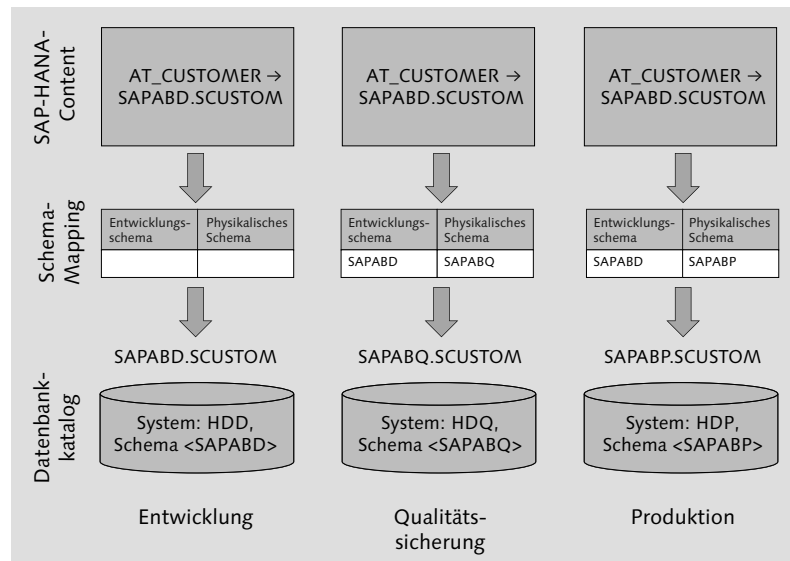


Abbildung 5.6 Prinzip des Schema-Mappings

Beispiel zum Schema-Mapping

Der Attribute View AT_CUSTOMER liest Kundendaten aus der Datenbanktabelle SCUSTOM. Diese Tabelle gehört zum Flugdatenmodell des AS ABAP und liegt im Entwicklungssystem im Datenbankschema SAPABD (weil die Systemkennung des ABAP-Systems ABD ist). Folglich verweist der Attribute View auf SAPABD.SCUSTOM.

Im Qualitätssicherungs- und Produktivsystem gibt es die Tabelle SAPABD.SCUSTOM nicht. Aufgrund der abweichenden Systemkennungen liegt die Datenbanktabelle im Qualitätssicherungssystem im Schema SAPABQ und im Produktivsystem im Schema SAPABP.

Das Schema-Mapping ermöglicht Ihnen, das Schema SAPABD im Qualitätssicherungssystem auf SAPABQ und im Produktivsystem auf SAPABP abzubilden.

Schema-Mapping-Pflege

Bei der Pflege eines Schema-Mappings müssen Sie einige Aspekte beachten:

- ▶ Das Schema-Mapping steuert letztlich, in welchem Datenbankschema ein Entwicklungsobjekt des SAP HANA Repositorys ein Objekt des Datenbankkatalogs sucht.
- ▶ Wenn kein Schema-Mapping gepflegt ist, sind Entwicklungsschema und physikalisches Schema identisch.
- ▶ Sie können mehrere Entwicklungsschemata auf das gleiche physikalische Schema abbilden.
- ▶ Sie können einem Entwicklungsschema *nicht* mehrere physikalische Schemata zuordnen.
- ▶ Der HANA-Content speichert Verweise zu Datenbankobjekten mit dem Entwicklungsschema. Wenn dieses (aufgrund einer Mehrfachzuordnung) nicht eindeutig ermittelt werden kann, speichert das System den Verweis mit dem physikalischen Schema.

Schema-Mapping bei Installation von SAP NetWeaver AS ABAP 7.4

[«]

Wenn Sie SAP NetWeaver AS ABAP 7.4 auf einer HANA-Datenbank installieren, erzeugt das Installationsprogramm das ABAP-Schema SAP<SID>. Außerdem legt das Installationsprogramm auch (mindestens) ein Schema-Mapping an, nämlich vom Entwicklungsschema ABAP auf das physikalische Schema SAP<SID>.

Sollten Sie sich für weiterführende Informationen zur Entwicklungsorganisation und zum Transport in SAP HANA interessieren, konsultieren Sie bitte die Dokumentation der HANA-Datenbank.

5.3.2 Nutzung des SAP-HANA-Transportcontainers

Nun betrachten wir den Transport von ABAP-Programmen, die native HANA-Objekte nutzen, über den HANA-Transportcontainer. Hierzu verwenden wir den Report ZR_A4H_CHAPTER5_LIST_CUSTOMER. Dieser greift über den externen View ZEV_A4H_CUSTOMER des ABAP Dictionarys auf den Attribute View AT_CUSTOMER des SAP HANA Repositorys zu. Den Quelltext des Reports finden Sie in Listing 5.8.

```
REPORT zr_a4h_chapter5_list_customer.
```

```
DATA: lt_customer TYPE STANDARD TABLE OF
      zpv_a4h_customer,
      ls_customer TYPE zpv_a4h_customer.
```

```
IF cl_db_sys=>dbsys_type = 'HDB'.
```

```

SELECT * FROM zev_a4h_customer
  INTO TABLE lt_customer.
ELSE.
  SELECT * FROM zpv_a4h_customer
  INTO TABLE lt_customer.
ENDIF.
LOOP AT lt_customer INTO ls_customer.
  WRITE: / ls_customer-id, ls_customer-name.
ENDLOOP.
    
```

Listing 5.8 Zu transportierender Beispielreport

Probleme beim Transport

Sowohl der Report ZR_A4H_CHAPTER5_LIST_CUSTOMER als auch der externe View ZEV_A4H_CUSTOMER können durch die Änderungsaufzeichnung und das Transportwesen des SAP NetWeaver AS ABAP problemlos transportiert werden (das geschieht im Prinzip »automatisch«). Der dem externen View zugrunde liegende Attribute View AT_CUSTOMER unterliegt hingegen nicht der Änderungsaufzeichnung und dem Transportwesen des Applikationsservers. Damit fehlt er (ohne dass wir entsprechende Maßnahmen ergreifen) nach einem Transport im Zielsystem. Daher kommt es im Zielsystem beim Aufruf des Reports zu einem Laufzeitfehler. Abhilfe kann der HANA-Transportcontainer schaffen.

Grundlegende Funktionsweise

Der HANA-Transportcontainer steht im Release SAP NetWeaver 7.31 ab dem Support Package 5 sowie ab dem Release 7.4 zur Verfügung. Er kann genutzt werden, wenn SAP HANA die Primärdatenbank ist.

Der HANA-Transportcontainer erlaubt Ihnen, über das SAP HANA Studio angelegte Entwicklungsobjekte des SAP HANA Repositorys mit den Mechanismen des Change and Transport Systems des ABAP-Applikationsservers (und insbesondere ohne die Notwendigkeit für einen Java-Stack, wie er für CTS+ benötigt würde) zu transportieren.

Technisch betrachtet handelt es sich beim HANA-Transportcontainer um ein logisches Transportobjekt, das als Proxy-Objekt für genau eine Delivery Unit agiert. Die Funktionsweise des HANA-Transportcontainers veranschaulicht Abbildung 5.7.

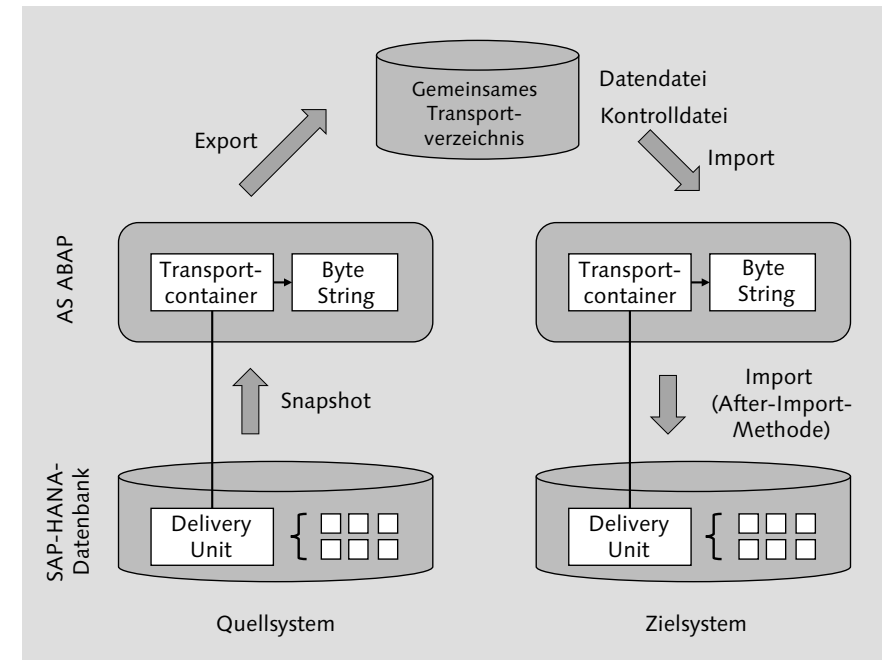


Abbildung 5.7 Funktionsweise des HANA-Transportcontainers

Sie können einen HANA-Transportcontainer über die ABAP Development Tools (und nur dort) anlegen. Dazu folgen Sie in der ABAP-Perspektive z. B. dem Menüpfad FILE • NEW • OTHER... • ABAP • SAP HANA TRANSPORT CONTAINER. Anschließend geben Sie den Namen der Delivery Unit ein, für die Sie den Transportcontainer anlegen möchten. Das System leitet daraus automatisch den Namen des Transportcontainers ab (siehe Abbildung 5.8; der HANA-Transportcontainer ZA4H_BOOK_CHAPTER05 ist *nicht* Teil der mit diesem Buch ausgelieferten Beispiele).

Anlage des Transportcontainers

Falls Sie in ABAP einen Präfixnamensraum verwenden möchten, müssen Sie vor der Anlage des Transportcontainers dem Namen des content_vendor (siehe Abschnitt 5.3.1, »Exkurs: Entwicklungsorganisation und Transport in SAP HANA«) den gewünschten Präfixnamen zuordnen. Dazu können Sie die Datenbanktabelle SNHI_VENDOR_MAPP über die Tabellensicht-Pflege füllen.

Nutzung eines Präfixnamensraums

Wenn die Transporteigenschaften des verwendeten Pakets – im Beispiel TEST_A4H_BOOK_CHAPTER05 – entsprechend gepflegt sind, zeichnet das System die Anlage des Transportcontainers in einem transportierbaren Änderungsauftrag auf.

Änderungsaufzeichnung

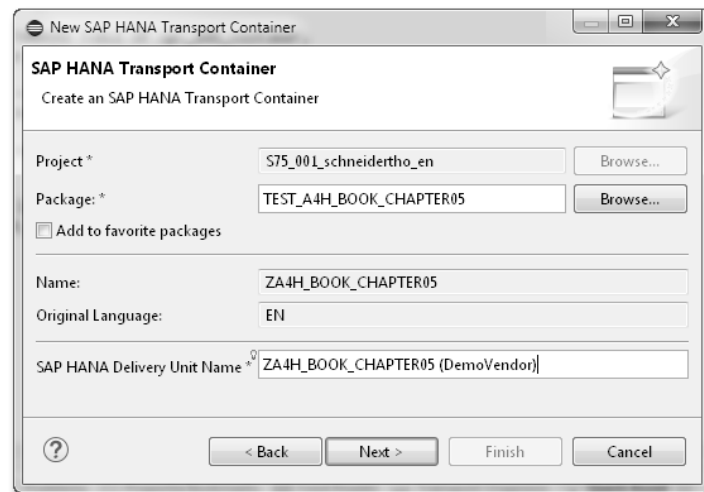


Abbildung 5.8 Anlage eines Transportcontainers

Synchronisation Beim Anlegen eines Transportcontainers synchronisiert das System den Inhalt dieses Containers einmalig automatisch mit dem Inhalt der Delivery Unit. Das bedeutet, dass alle Objekte der Delivery Unit als gepackte Datei auf den ABAP-Applikationsserver geladen und dort als *Byte String* in einer Datenbanktabelle (nämlich der Tabelle SNHI_DU_PROXY) abgelegt werden. Genau genommen liegt der Inhalt der Delivery Unit anschließend zweimal in der HANA-Datenbank:

- ▶ im SAP HANA Repository
- ▶ über die Datenbanktabelle SNHI_DU_PROXY

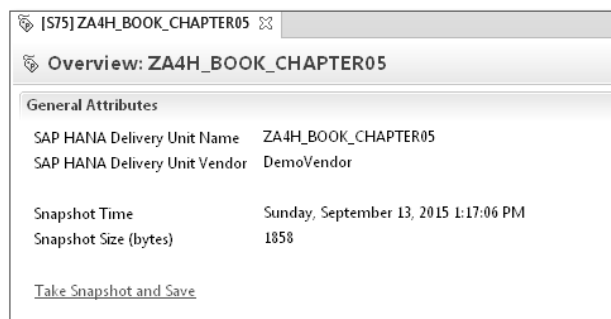


Abbildung 5.9 Synchronisation und Inhalt eines Transportcontainers

Wenn Sie den Transportcontainer nach der Anlage – z. B. weil Sie Änderungen am Attribute View AT_CUSTOMER vorgenommen haben –

mit der Delivery Unit synchronisieren möchten, müssen Sie dies manuell tun. Verwenden Sie dazu den Link TAKE SNAPSHOT AND SAVE. Den aktuellen Inhalt des Transportcontainers können Sie sich über die Registerkarte CONTENTS anschauen (beides ist in Abbildung 5.9 dargestellt).

Der Transport vom Entwicklungs- ins Qualitätssicherungs- und Produktivsystem erfolgt mit den Mechanismen des CTS:

Export und Import

- ▶ Beim Export (genauer beim *Export Release Preprocessing*) schreibt das System den Inhalt des Transportcontainers in die Datendatei im gemeinsamen Transportverzeichnis der am Transport beteiligten Systeme.
- ▶ Beim Import (genauer gesagt in einer *After-Import-Methode*) liest das System den Inhalt des Transportcontainers aus der Datendatei und importiert die Delivery Unit in die HANA-Datenbank des Zielsystems. Eine Aktivierung des Contents findet dabei nur statt, wenn Sie dies für die Softwarekomponente des Transportcontainers in der Tabelle SNHI_DUP_PREWORK aktiviert haben (und zwar im Zielsystem).

Sie können die beiden Schritte anhand des Transportprotokolls jederzeit nachvollziehen.

Gemischte Systemlandschaften

Einen Sonderfall bei der ABAP-Entwicklung auf SAP HANA stellen gemischte Systemlandschaften dar. Stellen Sie sich dazu vor, dass Sie als ABAP-Entwickler ein Programm für SAP HANA optimieren und dabei von spezifischen Möglichkeiten der HANA-Datenbank Gebrauch machen möchten. Gleichzeitig soll dieses Programm aber auch auf traditionellen Datenbanken lauffähig sein, z. B. weil Ihr Arbeitgeber nur in Teilbereichen des Unternehmens SAP HANA als Datenbank nutzt. Eine Systemlandschaft könnte in diesem Fall (vereinfacht) wie in Abbildung 5.10 aussehen.

Durch eine Fallunterscheidung können Sie – um beim Beispiel des Programms ZR_A4H_CHAPTER5_LIST_CUSTOMER zu bleiben – einmal den Projektions-View ZPV_A4H_CUSTOMER und einmal den externen View ZEV_A4H_CUSTOMER aufrufen (siehe Listing 5.8). Dadurch stellen Sie sicher, dass *zur Laufzeit* keine Fehler auftreten.

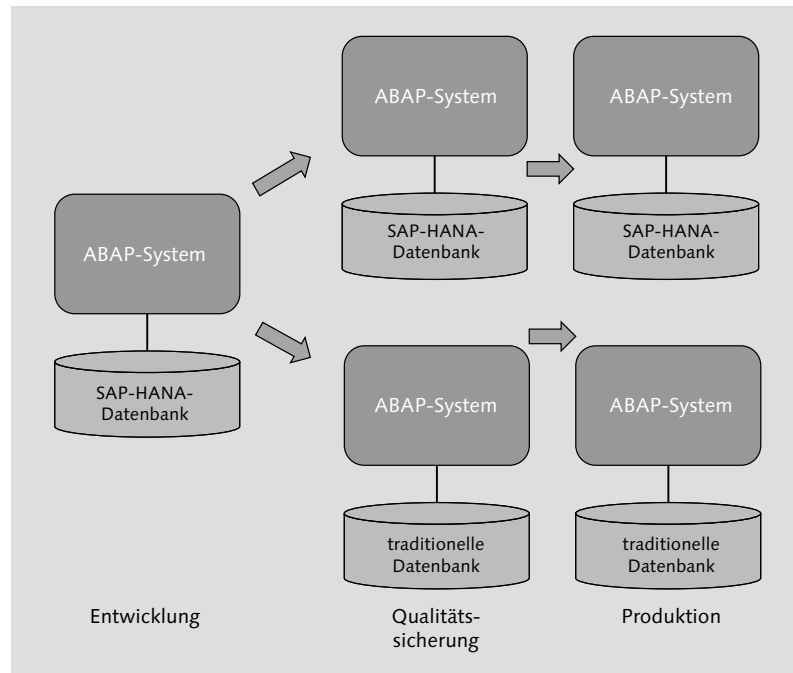


Abbildung 5.10 Gemischte Systemlandschaft

Systeme ohne
HANA-Datenbank

Die Implementierung des Transportcontainers sorgt dafür, dass *beim Transport* keine Fehler auftreten und der HANA-Content nur dann importiert wird, wenn es sich beim Zielsystem des Imports um ein HANA-basiertes System handelt.

Empfehlungen zur Verwendung des Transportcontainers

Einschränkungen

Bei der Verwendung des Transportcontainers sollten Sie einige Einschränkungen beachten:

- ▶ Bei der Verwendung des Transportcontainers transportieren Sie immer die komplette Delivery Unit. Es besteht keine Möglichkeit, nur den Inhalt einer Delivery Unit zu transportieren, der in einem bestimmten Zeitintervall geändert wurde.
- ▶ Anders als bei Entwicklungsobjekten, die im SAP NetWeaver AS ABAP verwaltet werden, zeichnet das System Änderungen am Inhalt einer Delivery Unit nicht automatisch auf, und die Objekte einer Delivery Unit werden nicht exklusiv für einen Transportauftrag gesperrt. Es liegt folglich in Ihrer Verantwortung, den Transportcontainer manuell mit der Delivery Unit zu synchronisieren.

- ▶ Beim Export der Entwicklungsobjekte aus dem Quellsystem berücksichtigt der Transport nur die aktiven Objekte.
- ▶ Das Transportsystem erkennt keine Abhängigkeiten zwischen mehreren gleichzeitig transportierten Transportcontainern.

Im Rahmen der Einschränkungen erlaubt Ihnen der Transportcontainer, Anwendungen, die zum Teil aus ABAP-Objekten und zum Teil aus HANA-Content bestehen, konsistent zu transportieren. Wir empfehlen Ihnen seine Verwendung, wenn die zu Beginn des Abschnitts 5.3.2, »Nutzung des SAP-HANA-Transportcontainers«, beschriebenen Voraussetzungen erfüllt sind.

Wenn Sie von den in Kapitel 6, »Erweiterte Datenbankprogrammierung mit ABAP 7.4«, beschriebenen Möglichkeiten Gebrauch machen, brauchen Sie den HANA-Transportcontainer nicht.

Inhalt

Geleitwort	15
Vorwort	17
Einleitung	19

TEIL I Grundlagen

1	SAP HANA im Überblick	31
1.1	Softwarekomponenten von SAP HANA	32
1.1.1	SAP HANA Database	32
1.1.2	SAP HANA Studio	34
1.1.3	SAP HANA Client	35
1.1.4	SAP HANA XS	37
1.1.5	Zusatzoptionen	38
1.2	Grundlagen der In-Memory-Technologie	41
1.2.1	Hardwareinnovationen	42
1.2.2	Softwareinnovationen	47
1.3	Architektur der In-Memory-Datenbank	58
1.4	Anwendungsfälle und Deployment-Optionen für SAP HANA	61
1.4.1	Anwendungsfälle	61
1.4.2	Deployment-Optionen	64
1.5	Auswirkungen von SAP HANA auf die Anwendungsentwicklung	66
1.5.1	Neue technische Möglichkeiten	66
1.5.2	Code Pushdown	67
1.5.3	Datenbank als Whitebox	69
1.5.4	Qualifizierung des Entwicklers	72
2	Einführung in die Entwicklungsumgebung	75
2.1	Eclipse im Überblick	75
2.2	Die Eclipse-Strategie von SAP	78
2.2.1	Entflechtung von Eclipse und SAP-Software	79
2.2.2	Zentrale Update-Seite	79
2.3	Installation der Entwicklungsumgebung	81

- 2.3.1 Installation der Eclipse IDE for Java Developers 81
- 2.3.2 Installation des SAP HANA Studios 82
- 2.3.3 Installation der ABAP Development Tools for SAP NetWeaver 83
- 2.4 Erste Schritte im Entwicklungssystem 84
 - 2.4.1 Grundlagen von Eclipse 84
 - 2.4.2 ABAP Development Tools for SAP NetWeaver 87
 - 2.4.3 SAP HANA Studio 97

3 Datenbankprogrammierung mit dem SAP NetWeaver AS ABAP 115

- 3.1 Architektur des SAP NetWeaver AS ABAP 117
 - 3.1.1 Datenbankschnittstelle und Datenbanktreiber 118
 - 3.1.2 Nutzung der Datenbank durch den SAP NetWeaver AS ABAP 121
 - 3.1.3 Datentypen 123
- 3.2 ABAP-Datenbankzugriff 129
 - 3.2.1 ABAP Dictionary 130
 - 3.2.2 Open SQL 135
 - 3.2.3 Datenbank-Views im ABAP Dictionary 145
 - 3.2.4 Datenbankzugriff über natives SQL 146
 - 3.2.5 Sekundäre Datenbankverbindungen 153
- 3.3 Datenbankzugriffe mit dem SQL-Trace analysieren 157
 - 3.3.1 Anweisungstransformationen 157
 - 3.3.2 Sekundärverbindungen 164
 - 3.3.3 Natives SQL 165
 - 3.3.4 Puffer 165

TEIL II Einführung in die ABAP-Programmierung mit SAP HANA

4 Native Datenbankentwicklung mit SAP HANA 169

- 4.1 Grundlagen der nativen Datenbankentwicklung 169
 - 4.1.1 Objekte im HANA-Datenbankkatalog 170

- 4.1.2 SQL-Standard und HANA-spezifische Erweiterungen 172
- 4.2 SQLScript 179
 - 4.2.1 Grundlagen von SQLScript 179
 - 4.2.2 SQLScript-Programmierung 186
- 4.3 Datenbankprozeduren 198
- 4.4 Analytische Modelle 203
 - 4.4.1 Attribute Views 204
 - 4.4.2 Analytic Views 219
 - 4.4.3 Calculation Views 228
 - 4.4.4 Laufzeitobjekte und SQL-Zugriff 235
 - 4.4.5 Zugriff auf Column Views über Microsoft Excel 238

5 Einbindung nativer SAP-HANA-Entwicklungsobjekte in ABAP 241

- 5.1 Einbindung von analytischen Views 241
 - 5.1.1 Zugriff über natives SQL 242
 - 5.1.2 Externe Views im ABAP Dictionary 243
 - 5.1.3 Zugriffsmöglichkeiten auf externe Views 246
 - 5.1.4 Empfehlungen 247
- 5.2 Einbettung von nativen Prozeduren in ABAP 249
 - 5.2.1 Zugriff über natives SQL 250
 - 5.2.2 Definition von Database Procedure Proxies 257
 - 5.2.3 Aufruf von Database Procedure Proxies 259
 - 5.2.4 Anpassung von Database Procedure Proxies 260
- 5.3 Transport nativer Entwicklungsobjekte 261
 - 5.3.1 Exkurs: Entwicklungsorganisation und Transport in SAP HANA 261
 - 5.3.2 Nutzung des SAP-HANA-Transportcontainers 265

6 Erweiterte Datenbankprogrammierung mit ABAP 7.4 273

- 6.1 Einführung in Core Data Services 274
- 6.2 ABAP Core Data Services 276
 - 6.2.1 CDS-Views 277

- 6.2.2 Code Pushdown 298
- 6.2.3 View-Erweiterungen 305
- 6.2.4 Annotationen 307
- 6.2.5 CDS-Views in ABAP und in ALV with IDA
verwenden 311
- 6.2.6 Tipps für die Anwendung von ABAP-
CDS-Views 313
- 6.3 SAP HANA Core Data Services 315
- 6.4 Open-SQL-Erweiterungen 317
- 6.5 ABAP-Datenbankprozeduren 320
 - 6.5.1 AMDPs anlegen 322
 - 6.5.2 Fehleranalyse 325
 - 6.5.3 Erweiterungen 327
 - 6.5.4 Praxistipps 331

7 Laufzeit- und Fehleranalyse auf SAP HANA 333

- 7.1 Übersicht der verfügbaren Werkzeuge 334
- 7.2 Fehleranalyse 336
 - 7.2.1 Unit Tests 337
 - 7.2.2 Dump-Analyse 339
 - 7.2.3 Debugging und Tracing in SQLScript 342
- 7.3 ABAP-Code-Analyse 343
 - 7.3.1 Prüfungen und Prüfvarianten 343
 - 7.3.2 Prüfungen in der Entwicklungs-
infrastruktur 348
 - 7.3.3 Globale Prüfläufe im System 349
- 7.4 Laufzeitstatistiken und Traces 351
 - 7.4.1 Laufzeitstatistik 352
 - 7.4.2 ABAP-Trace und ABAP Profiler 356
 - 7.4.3 SQL-Trace 364
 - 7.4.4 Single Transaction Analysis 368
 - 7.4.5 Explain Plan 369
 - 7.4.6 SAP HANA Plan Visualizer 370
- 7.5 Systemweite Analysen 376
 - 7.5.1 DBA-Cockpit 376
 - 7.5.2 SQL Monitor 381
 - 7.5.3 Laufzeitprüfungs-Monitor 390
- 7.6 SQL-Performanceoptimierung 393

8 Beispielszenario: Optimierung einer bestehenden Anwendung 399

- 8.1 Vorgehen bei der Optimierung 400
 - 8.1.1 Migration auf SAP HANA 400
 - 8.1.2 Systemoptimierung 402
 - 8.1.3 Anwendungsoptimierung 403
- 8.2 Szenario und Anforderungen 406
 - 8.2.1 Ausgangssituation 406
 - 8.2.2 Technische Umsetzung 408
 - 8.2.3 Aktuelle Probleme 410
- 8.3 Umsetzung der Anforderungen 411
 - 8.3.1 Eingrenzung des Problems mit der
Laufzeitstatistik 411
 - 8.3.2 Detailanalyse des ABAP-Programms mit
Transaktion SAT 413
 - 8.3.3 Detailanalyse der Datenbankzugriffe 415
 - 8.3.4 Ergebnis der Analyse 417
 - 8.3.5 Optimierung mit Open SQL 417
 - 8.3.6 Analyse der ersten Optimierung 420
 - 8.3.7 Ergebnis der Analyse 421
 - 8.3.8 Optimierung mit einer
Datenbankprozedur 422
 - 8.3.9 Analyse nach der zweiten Optimierung 424
 - 8.3.10 Ergebnis der Analyse 427

TEIL III Fortgeschrittene Techniken für die ABAP-Programmierung mit SAP HANA

9 Integration analytischer Funktionalität 431

- 9.1 Was ist analytische Funktionalität? 432
- 9.2 Das SAP-BusinessObjects-Portfolio 435
- 9.3 Exkurs: SAP Business Warehouse 439
 - 9.3.1 SAP HANA vs. SAP Business Warehouse 439
 - 9.3.2 Terminologie im Überblick 440
 - 9.3.3 InfoProvider bei Nutzung von
SAP HANA 442
- 9.4 Mögliche Architekturen im Überblick 448

9.4.1 Direkter Zugriff auf analytische Funktio-
nalität in SAP HANA 448

9.4.2 Zugriff über SAP NetWeaver AS ABAP 450

**10 Textsuche und Analyse von unstrukturierten
Daten 455**

10.1 Grundlagen der Textsuche in SAP HANA 457

10.1.1 Technische Architektur 458

10.1.2 Fehlertolerante Suche 459

10.1.3 SAP-Komponenten und Produkte für die
Suche 461

10.2 Textdatentypen und Full-Text-Indizes in
SAP HANA 462

10.3 Verwendung der Textsuche über SQL 467

10.3.1 Fuzzy-Suche 469

10.3.2 Synonyme und Stoppwörter 473

10.3.3 Suche über Datumsfelder und
Adressdaten 476

10.4 Einsatz der Textsuche in ABAP 479

10.4.1 Direkter SQL-Zugriff aus ABAP 479

10.4.2 Einbettung in Wertehilfen 480

10.4.3 ABAP-Quelltextsuche 488

10.5 Textanalyse 489

10.6 Ressourcenverbrauch und Laufzeitaspekte der
Textsuche 492

11 Entscheidungstabellen in SAP HANA 495

11.1 Grundlagen von Entscheidungstabellen 496

11.2 Anlegen von Entscheidungstabellen im
SAP HANA Studio 500

11.3 Entscheidungstabellen basierend auf
SAP HANA Views 506

11.4 Laufzeitobjekte und SQL-Zugriff für
Entscheidungstabellen 508

11.5 Zugriff auf Entscheidungstabellen aus ABAP 509

12 Funktionsbibliotheken in SAP HANA 513

12.1 Grundlagen der Application Function Library 516

12.2 Business Function Library 518

12.3 Predictive Analysis Library 522

12.3.1 Generierung der K-Means-Funktion über
die SQL-Konsole 524

12.3.2 Verwendung des Application Function
Modelers 527

13 Verarbeitung von Geoinformationen 535

13.1 Grundlagen von Geoinformationssystemen 536

13.2 Geodatentypen und Geofunktionen in
SAP HANA 538

13.2.1 Datentypen 539

13.2.2 Tabellen anlegen und Daten lesen 540

13.2.3 Operationen auf geografischen
Strukturen 542

13.2.4 Integration von externem
Kartenmaterial 544

13.3 Geoinformationen in ABAP-Anwendungen
einbinden 546

14 Praxistipps 551

14.1 Allgemeine Empfehlungen 552

14.1.1 Empfehlungen zu Column und
Row Store 552

14.1.2 HANA-spezifische Implementierungen 553

14.1.3 Checkliste für datenbankspezifische
Implementierungen 556

14.1.4 Empfehlungen zur Migration 558

14.1.5 Entwicklung in Landschaften 560

14.1.6 Schreibende Zugriffe in SQLScript oder
nativem SQL 562

14.2 Konventionen 564

14.2.1 Namenskonventionen 564

14.2.2 Kapselung von HANA-Paketen 566

14.3 Qualitätsaspekte 566

14.3.1 Testen von Views und Prozeduren 567

14.3.2 Robuste Programmierung 568

14.3.3 Sicherheitsaspekte 569

14.4 Performanceempfehlungen für Open SQL 570

14.4.1 Regel 1: Ergebnismengen klein halten 571

14.4.2	Regel 2: Übertragene Datenmengen klein halten	575
14.4.3	Regel 3: Anzahl der Anfragen reduzieren	581
14.4.4	Regel 4: Suchaufwand minimieren	587
14.4.5	Regel 5: Datenbank entlasten	591
14.4.6	Zusammenfassung der Regeln	596
14.5	Performanceempfehlungen für native Implementierungen in SAP HANA	597
14.5.1	Empfehlungen für natives SQL	597
14.5.2	Empfehlungen für SAP HANA Views	599
14.5.3	Empfehlungen für SQLScript	602
14.6	Zusammenfassung der Empfehlungen	604

Anhang 607

A	Flugdatenmodell	609
B	Erweiterungen der ABAP-Programmiersprache (ab SAP NetWeaver 7.4)	617
C	Lese- und Schreibzugriffe im Column Store	623
D	SAP Business Application Accelerator powered by SAP HANA	633
E	Installation der Beispiele	637
F	Die Autoren	639
	Index	641

Index

@ (für Annotationen) 307
\$parameters 296
\$projection 289

A

ABAP

ABAP Unit 337
ABAP-Anwendung, Transport 561
ABAP-Code-Analyse 335, 400
ABAP-Laufzeitumgebung 118
ABAP-Projekt 88
ABAP-Proxy 565
ABAP-Trace 356, 420
Quelltextsuche 488
Schema 121
Typsystem 125
ABAP 7.4 617
ABAP CDS 275
Compiler 296
Funktionen 298
Modellierung 313
ABAP Connectivity and Integration
 Development Tools 84
ABAP Core Development Tools 82, 83
ABAP Database Connectivity → ADBC
ABAP Development Tools for SAP Net-
 Weaver 34, 243
ABAP-Ressourcen-URL 90, 92
Benutzereinstellungen 90
Berechtigungen 88
Code-Analyse 348
Debugger 95
favorisierte Pakete 89
Komponenten 83
Perspektive 87
Programm anlegen 92
Programm ausführen 95
Project Explorer 89
Projekt 88
SAP-GUI-Integration 90
Systembibliothek 89
Templates 93

ABAP Dictionary 90, 116, 124, 130,
 464
 Typsystem 126
 Wertehilfe 480
ABAP Managed Database Procedure
 147, 172, 274, 547
ABAP Managed Database Procedure
 Framework 321
ABAP Memory 594
ABAP Profiler 335, 358
ABAP Test Cockpit 93, 335, 347, 350
ABAP-Managed-Datenbankprozedur
 321
ABAP-Programm
 Analyse 413
 Ausführungszeit 413
ABAP-Puffer, benutzerübergreifen-
 der 591
ABAP-Tabellenpuffer → Tabellenpuf-
 fer
Accelerator 62
ACID-Prinzip 32
ADBC 147, 250, 465, 479, 558, 599
Administration Console 97
AFL 514, 517
 Software Development Kit 515
After-Import-Methode 269
Aggregatfunktion 139, 299, 301
Aggregation 298, 312
Aktivierungsprotokoll 314
Alias 298
Alternativimplementierung 554
ALV → SAP List Viewer
AMDP 479, 520
 BAdI 327
 Datenbankprozedur aufrufen 325
 Datenbankverbindung 325
 erweitern 327
 Fehleranalyse 325
 Tipps 331
AMDP Framework → ABAP Managed
 Database Procedure Framework
AMDP-Klasse 322
AMDP-Methode 321, 323
Analysis, Edition for Microsoft Office
 437

Analytic Engine 442
 Bericht-Bericht-Schnittstelle 452
 Formel 452
 Hierarchieverarbeitung 452
 Analytic Privilege 107
 Analytic View 107, 204, 219, 237,
 599, 614
 anlegen 221
 Analytical Search 426
 analytische Funktionalität 432
 analytische Query → BW Query
 analytischer Index 443
 Änderungsauftrag 267
 Annotation 275, 307
 Post-Annotation 307
 Scope 307
 Sichtbarkeitsbereich 307
 Anweisungstransformation 157
 Anwendung, Optimierung 399, 403
 Anwendungslogik 68
 Append View 305
 Appliance 31
 Application Function Library 102,
 514, 517
 Application Function Modeler 518,
 524, 527
 Applikationsschicht 68
 Array-Interface 599
 Assoziation 275, 286, 289
 Definition 288
 Filter 294
 lokale 289
 Name 288, 292
 Präfix 292
 sichtbare 289
 Verwendung 291
 Vorteile 291
 Attribut 205
 berechnetes 213, 224
 virtuelles 213
 Attribute View 107, 204, 205, 599
 Attributvektor 50, 623
 Aufrufhierarchie 362, 416
 Ausführungsplan → Explain Plan
 Ausgabeparameter 565
 Ausnahme, CX_SY_SQL_UNSUPPORTED_FEATURE 296

B

BAdI 556
 B-Baum-Index 627
 Bedingung 299
 Berechtigung
 analytische 98, 570
 Paketberechtigung 98
 Prüfung 569
 SAP HANA Studio 98
 Systemberechtigung 98
 Berichtswesen 432
 Bewegungsdaten 220, 612
 BEx Query Designer 447
 BFL 515, 518
 Blocking Factor 162
 Breakpoint 326
 dynamischer 95
 externer 96
 statischer 95
 BRFPplus 496
 Business Function 488
 Business Function Library 515, 518
 Business Rule Mining 514
 BW Query 442, 447

C

Calculation Engine 186, 197
 Calculation View 107, 180, 204, 228,
 443, 509, 599
 Data Category 230
 modellierter 228
 SQLScript 228
 Callstack 362, 416
 CASE-Anweisung 298
 Cashflow 518
 CDS → Core Data Service
 CDS-Entität 278, 311
 CDS-Objekt 278, 315
 CDS-View 276, 470
 Aktivierungsprotokoll 314
 anlegen 278
 erweitern 305
 mit Parameter 295
 Name 311
 Outer-Joins 483
 Verwendung mit Parametern 296

CE-Funktion 422, 600
 CE_AGGREGATION 196
 CE_COLUMN_TABLE 196
 CE_CONVERSION 197
 CE_PROJECTION 196
 CE_VERTICAL_UNION 197, 520
 TRACE 342
 Zugriffsfunktion 197
 CLIENT SPECIFIED 286
 Client-Server 21
 Cloud Deployment 65
 Cluster-Codierung 53
 Cluster-Tabelle 134, 559
 COALESCE-Funktion 283
 Code Completion 93, 105, 246
 Code Inspector 335, 343, 401, 415
 Code Pattern 72
 Code Pushdown 67, 594
 Code Template 279
 Code-to-Data-Paradigma 67, 69, 411
 Codierung
 Cluster-Codierung 53
 Dictionary-Codierung 50
 indirekte 53
 Laufängencodierung 53
 Präfix-Codierung 53
 Sparse-Codierung 53
 Column Store 48, 103, 133, 173, 174,
 623
 Datentypen 462
 Empfehlung 552
 INSERT ONLY 627
 invertierter Index 630
 Lesezugriff 624
 Merge 628
 Schreibzugriff 626
 zusammengesetzter Index 631
 Column View 102, 171, 236, 242,
 245
 Commit, impliziter 144
 CONTAINS 179, 467
 Content 35, 106
 Core Data Service 37, 146, 274
 ABAP 275
 Element 282
 HANA 275, 315
 CPU-Cache 44, 45
 CPU-Kern 42
 CPU-Zeit 404, 426
 Cursor 142, 194, 603

D

Data Aging 41
 Data Control Language (DCL) 129
 Data Definition Language (DDL) 129,
 187, 277, 540
 Data Dictionary → ABAP Dictionary
 Data Manipulation Language (DML)
 129
 Data Modeler 610
 Data Quality Library 515
 Data Scientist 513
 Data Warehouse 63, 434
 DATA() 618
 Database Procedure Proxy 250, 273,
 561, 565
 Synchronisation 260
 Database Shared Library → DBSL
 DataSource 441, 447
 Data-to-Code-Paradigma 67
 Datenanalyse 432, 435
 Datenart 133
 Datenbank
 relationale 32, 47, 175
 Typsystem 126
 Datenbankindex 588
 Datenbank-Interface 559, 562
 Datenbankkatalog 101, 170, 242
 Datenbankobjekt 102
 Datenbankoptimierer 56, 182, 369
 Datenbankprogrammierung, Werk-
 zeuge 157
 Datenbankprozedur 59, 102, 107,
 180, 565
 ABAP-verwaltete 274
 Arten 187
 Ausführung 186
 Kompilierung 186
 Kontrollstruktur 193
 testen 567
 Datenbankschema 101, 109, 170
 Datenbankschicht 68
 Datenbankschnittstelle → DBI
 Datenbanktabelle 102
 Datenbanktrigger 103
 Datenbankunabhängigkeit 277
 Datenbankverbindung
 sekundäre 153, 164
 Standard 164
 Datenbank-View 131, 145

Datendeklaration 617
 Datenelement 296
 Datenexploration 435
 Dateninkonsistenz 564, 567
 Datenlayout 47
 Datenmodell
 Erweiterung 276
 Syntax 276
 virtuelles 449
 Datenquelle, Zieldatenquelle 288
 Datenreferenz 620
 Datenreplikation 39
 SAP Landscape Transformation Replication Server 40
 SAP Replication Server 40
 Datensicht → View
 Datentyp 123, 132, 174, 257, 311
 benutzerdefinierter 275
 Geo-Spatial 539
 Integer 50
 Konvertierung 505
 SHORTTEXT 174, 462
 ST_GEOMETRY 174
 TEXT 174, 462
 Datenvorschau 104, 112
 DBA-Cockpit 154, 376
 DBI 118
 DBSL 36, 120
 DCL → Data Control Language (DCL)
 DDL Editor 280
 DDL → Data Definition Language (DDL)
 DDL Source 277
 anlegen 278
 DDL-Statement 57
 Debugging 342
 Decision Table → Entscheidungstabelle
 deklarative Programmierung 184, 603
 Delivery Unit 106, 566
 Delta Load 39
 Delta Merge 494
 Delta Store 54, 494, 626
 Delta-Komprimierung 52
 Designtime 254
 Design-Time-Objekt 110
 Diagnostics Agent 34
 Dictionary-Codierung 50
 Dictionary-Vektor 50, 623

Dimension 220
 Direct Extractor Connection 40
 DISTINCT 299
 Document Analysis Toolkit 458
 Domäne 132
 DRAM 45
 Drill-down 215
 Dump 339, 569
 Duplikat 298
 Dynamic Random Access Memory 45
 dynamischer Datenbank-View 496

E

Easy Query 453
 Echtzeit 19, 434
 near Realtime 633
 Echtzeitqualität 19
 Eclipse 34
 ABAP-Entwicklungsumgebung 34
 Ansicht 86
 Arbeitsbereich 87
 Editor 86
 Erweiterungspunkt 76
 Fenster 84
 Foundation 34, 75, 77
 Framework 75
 Menüleiste 86
 Perspektive 85
 Plattform 34, 75, 76
 Plug-in 76
 Projekt 77
 Release Train 77
 Repository 80
 Sammelrelease 77
 SAP Release Train for Eclipse 79
 SDK 76
 Symbolleiste 86
 Update Site 80
 Workbench 84
 Eingabeparameter 227, 565
 Einheitenkonvertierung 226
 Element Info 306
 elementare Suchhilfe 481
 Embedded Reporting 440
 Embedded Search 461
 Engine 59, 601
 Enqueue-Server 117
 Enqueue-Service 563

Enqueue-Workprozess 123
 Enterprise Information Management 530
 Enterprise Search 462
 Entitätsname 281
 Entity-Relationship-Modell 203
 Entkopplung 555, 567
 Entscheidungsregel 495, 506
 Entscheidungstabelle 107, 495, 497, 500
 Aktionen 497
 anlegen 500
 Bedingungen 497
 Transport 511
 Entwicklungslandschaft, gemischte 561
 Entwicklungsobjekt 92, 102, 106
 ABAP 565
 Ablage 108
 aktivieren 110
 Namenskonvention 564
 SAP HANA 564
 testen 112
 validieren 109
 Entwicklungsschema 263
 Entwicklungsumgebung
 ABAP Development Tools 83, 87
 Installation 81
 SAP HANA Studio 82
 Entwurfsmuster 339
 Enumerationswert 307
 Equi-Join 211
 ESRI Shapefile 541, 545
 ETL-Prozesse 39
 Existenzcheck 346
 Expensive SQL Statement Trace 336
 Explain Plan 336, 369
 Aufruf 369
 Ausgabe 370
 Export Release Preprocessing 269
 Extended Storage 41
 Extension Index 464
 externer View → View, externer

F

Factory Pattern 555
 Faktentabelle 219, 220
 Fehleranalyse 333, 335, 336

Fehlerbehandlung 568
 Feld
 berechnetes 224
 Referenz 305
 Feldliste 577
 Feldsymbol 618
 Festplatte 44
 Filter 206, 298, 312
 Filterbedingung 294
 Filterwert 212
 Fiskaljahr 217
 Flash-Speicher 45
 Fluchtsymbol 296
 Flugdatenmodell 117, 609, 610
 Flussdiagramm 527
 FOR ALL ENTRIES 137, 162, 163, 345, 585
 Foreign Key → Fremdschlüssel
 Fremdschlüssel 286, 290
 Fremdschlüsselbeziehung 131, 204, 290
 Full-Text-Index 463, 467, 490, 493
 Funktion 171
 benutzerdefinierte 102
 numerische 299
 Funktionsbibliothek → Application Function Library
 Fuzzy Score 473
 Fuzzy-Search-Index 461, 493
 Fuzzy-Suche 456, 457, 459, 473
 Schwellenwert 459
 stopwordListId 475
 stopwordTable 474
 textsearch 475

G

Geo-Coding 546
 Geodatentyp 538
 konvertieren 542
 Geo-Index 546
 Geoinformation 535
 in ABAP 546
 Geoinformationssystem 535, 536
 GeoJSON 541
 Geo-Spatial Engine 536, 539
 Darstellungsformate 541
 Konstrukturen 540
 ST_Area 544

Geo-Spatial Engine (Forts.)
ST_Contains 544
ST_Distance 542
ST_WithinDistance 543
Tabelle anlegen 540
 Geschäftsjahr 217
 Geschäftslogik 185
 Geschäftsprozess 495
 Geschäftsregel-Managementsystem
 495
 gewichteter Durchschnitt 518, 520
 GIS → Geoinformationssystem
 goldene Regeln für Datenbankpro-
 grammierung 571
 Größenkategorie 133
 GROUP BY 298
 GROUPING SETS 178
 Gruppieren 312
 GUID 614

H

Hadoop 40
 HANA CDS 275, 315
Objekt 315
Syntax 315
 HANA XS 37
 HANA-Datenbank 32, 58
 HANA-Entwicklungsobjekt 273
 HANA-Softwarekomponente 32
 HANA-Transportcontainer 266, 561
 Hardwareinnovationen 42
 Hardwaretrends 42
 Hash-Partitionierung 57
 Hashwert 57
 Hauptspeicher 42, 45
 HAVING-Klausel 299, 302, 572, 588
 Help-View 483
 Hierarchie 206, 215, 221
Level 215
Vater-Kind 215
 High Performance Analytical Appli-
 cation → SAP HANA
 Hint 144, 558
 Hitliste 361
 Hostvariable 296
 hybride Anwendung 67
 Hypernym 476
 Hyponym 476

I

IDA → Integrated Data Access
 Identical Select 366, 367
 imperative Programmierung 184, 603
 Index 102, 171
Exklusionsliste 133
Full-Text-Index 463
Inklusionsliste 133
invertierter 590, 629, 630
Primärindex 588
zusammengesetzter 590, 630, 631
 Indexserver 59
 indirekte Codierung 53
 InfoObject 441, 443, 444
 InfoProvider 441, 442
transient 446
transienter 442
virtueller 444
 InfoSet, klassisches 446
 Infrastructure-as-a-Service 65
 Initial Load 39
 Inkonsistenz-Prüfung 303
 Inline-Deklaration 318, 617
 IN-Liste 162
 In-Memory-Datenbank 58
 In-Memory-Technologie 41
 Insight to Action 432
 Integer 50
 Integrated Data Access 312
 Integrität 290
 International Organization for Standar-
 dization (ISO) 539
 interne Tabelle 594

J

Java Runtime Environment (JRE) 82
 JDBC 35
 Job Log 110
 Join 105, 197, 281, 283, 318, 421
aus Assoziation 288
Equi-Join 211
Full Outer Join 177
Inner Join 136, 177, 283
Join-Typ 177
komplexer 602
Left Outer Join 136, 177
Outer Join 177

Join (Forts.)
Referential Join 206, 210
Right Outer Join 177, 294
Self Join 234
Text Join 206
Typ 293
 Join Engine 205, 603
 Join View → Attribute View

K

Kalkulationslogik 68
 Kapselung 566
 Kardinalität 289, 294
 Kartenmaterial 544
 Katalogobjekt 315
 Kennzahl 220
berechnete 224
ingeschränkte 221
 Kerberos 60
 Kernel 118
 key 302
 Klasse
CL_ABAP_DBFEATURES 296
CL_SALV_GUI_TABLE 312
 K-Means 523, 527
 Komprimierung
Delta-Komprimierung 52
Verfahren 50, 52
 Konstruktor-Ausdruck 619
 Kontrollstruktur 193
 Konvertierungs-Exit 557

L

L (Programmiersprache) 185, 187
 Large Object 462, 494
 Lastverteilung 56
 Latenz 19
 Lauflängencodierung 53
 Laufzeit 255
 Laufzeitanalyse 333, 335, 420
 Laufzeitfehler 339
 Laufzeitobjekt 236
 Laufzeitprüfungs-Monitor 336, 390
 Laufzeitstatistik 351, 352, 404, 412
Auswertung 352
Selektion 352
 Lifecycle Management 33

Liste 312
 Literal 295, 297, 304, 307
 L-Knoten 185
 Logical Unit of Work 122
 lokale temporäre Tabelle 342
 LOOP-Schleife 586
 LUW-Konzept 122

M

Main Store 54, 626
 Mainframe 21
 Mandantenabhängigkeit 284
 Mandantenbehandlung 120, 248,
 284, 561
Attribute View 209
 MANDT 285
 Manifest 76
 Massenoperation 599
 Materialisierung 203
 MDX 35, 59, 178
 Measure → Kennzahl
 Mengenoperation 582
 Mengenoperator 298
 Merge 494, 628
 Message-Server 117
 Metadaten 275
 Methode DB_CAPABILITIES 312
 Modeler-Perspektive 98
 MODIFY 161
 Modularisierung 567
 Modularisierungseinheit 346
 Monitoring View 493
 Multidimensional Expressions → MDX
 Multitenant Database Container 59

N

Namensraum 565
 Nameserver 60
 natives SQL 479, 597
ABAP-Tabellen 562
 NEW-Operator 619
 node.js 36
 NULL-Wert 283
 NUMA-Architektur 45
 numerische Funktion 299

O

Objektinstanz, anlegen 619
 ODBC 35
 ODBO 35
 ODS-Objekte 441
 OLAP 67, 220, 434
 OLAP Engine 603
 OLTP 67, 434
 on the fly 435
 ON-Bedingung 281, 294
 On-Premise Deployment 64
 Open Geospatial Consortium 539
 Open SQL 69, 115, 129, 179, 183, 417
 Array-Operation 143
 CDS-View verwenden 296
 Cursor 142
 dynamisches 141
 Erweiterung 317
 Existenzprüfung 140
 Hints 144
 Paketgröße 142
 striktter Modus 318
 Transaktionskontrolle 144
 Unterabfrage 140
 Operational Data Provisioning 446
 Operator 301
 Optimierer → Datenbankoptimierer
 Orchestrierungslogik 68
 OR-Verbindung 162
 Outer Join 128

P

Paging 176, 312
 Paket 106, 107, 566
 SAP HANA 564
 system-local 262
 PAL → Predictive Analysis Library
 Parallelisierung 56, 193, 603
 Parameter 295
 Namenskonventionen 296
 Parameter-Marker 598
 Partition Pruning 56
 Partitionierung 54
 Art 57
 explizites Partition Handling 56
 Hash-Partitionierung 57

Partitionierung (Forts.)
 horizontale 55
 Range-Partitionierung 57
 Round-Robin-Partitionierung 56, 57
 vertikale 55
 PBO-Modul 510
 Performance 570
 Phrase-Index 493
 Phrase-Index-Ratio 493
 physikalisches Schema 263
 Planungs-Engine 59
 PlanViz 336, 370, 405, 426
 Aufzeichnung 370
 Auswertung 371
 Operator List 372
 Tables Used 372
 Timeline 371
 PMML 522
 Pool-Tabelle 134, 559
 Post-Annotation 307
 Präfix 292
 Präfix-Codierung 53
 Präfixnamensraum 267
 Pragma, ##DB_FEATURE_MODE 296
 Präprozessor-Server 60
 Präsentationsschicht 68
 Predictive Analysis 513
 Predictive Analysis Library 38, 515, 522
 Predictive Model Markup Language (PMML) 522
 Prepared-Anweisung 150, 570, 597
 Prepare-Phase 598
 Pretty Printer 92, 93
 Primärdatenbank 250, 444
 Procedure 171
 Projektion 197
 Projektions-View 131
 Proxy-Objekt 257
 Public Synonym 111
 Puffer-Trace 165
 Pufferzugriff 418
 Punktoperator 292

Q

QL → Query Language (QL)
 Query Language 275

R

R (Programmiersprache) 187, 533
 RAM 42
 Range 137
 Range-Partitionierung 57
 räumliches Bezugssystem → Spatial Reference System
 Read Only Procedure 187
 Read Write Procedure 187
 Realtime → Echtzeit
 Redirected Database Access (RDA) 153, 633
 Referential Join 206, 210
 Regel 498
 relationale Funktion 197
 Reporting 432
 Repository → SAP HANA Repository
 Result View 508, 509
 RIGHT OUTER JOIN 294
 robuste Programmierung 568
 Rollback 144
 Rollenverwaltung, SAP HANA Studio 99
 Round Trip 161
 Round-Robin-Partitionierung 57
 Row Store 48, 133, 173
 Runtime-Objekt → Laufzeitobjekt

S

SAML 60
 Sammelsuchhilfe 481
 SAP ASE (Datenbank) 41
 SAP Business Application Accelerator 633
 SAP Business Explorer (BEx) 438, 442
 SAP Business Warehouse 434
 SAP BusinessObjects 442
 SAP BusinessObjects Business Intelligence-Plattform 238
 SAP BusinessObjects Dashboards 436
 SAP BusinessObjects Design Studio 436
 SAP BusinessObjects Web Intelligence 436
 SAP BusinessObjects Werkzeuge 435
 SAP Crystal Reports 435
 SAP Gateway 453

SAP HANA 31
 Advanced Data Processing 39
 Anwendungsfälle 61
 Application Lifecycle Manager 33
 Client 32
 Datenbank 32
 Dynamic Tiering 41
 Engines 38
 Enterprise Information Management 40
 Entwicklungsorganisation 261
 Lifecycle Manager 33
 Migration 400, 558
 Optionen 38
 Predictive Option 38
 Real-Time Replication 40
 Smart Data Streaming 41
 Spatial Option 38
 SQL Command Network Protocol 36
 Studio 32
 Transportcontainer 265
 zertifizierte Hardware 42, 43
 SAP HANA Analytics Foundation 449
 SAP HANA Client 35
 HTTP 37
 JDBC 35
 ODBC 35
 ODBO 35
 SQLDBC-Bibliothek 35
 SAP HANA Cloud Platform 65
 SAP HANA Cockpit 35
 SAP HANA Development 98
 SAP HANA Enterprise Cloud 65
 SAP HANA Extended Application Services 32
 SAP HANA Live 62, 449
 SAP HANA MDX Provider 238
 SAP HANA Plan Visualizer → PlanViz
 SAP HANA Repository 98, 106, 262, 564, 566
 SAP HANA Repository View 243, 246
 SAP HANA Studio 32, 34, 97
 Arbeitsbereich 99
 Benutzereinstellungen 100
 Berechtigungen 98
 Datenbankkatalog 101
 hdbinst 82
 hdbsetup 82
 Perspektive 97
 SQL-Anweisung 426

SAP HANA Studio (Forts.)
SQL-Konsole 105, 187
Systemverbindung 99
Templates 105
View-Modellierung 204, 206

SAP HANA UI for Information Access 462

SAP HANA View 70, 242, 561
Auswahl 248
Performance 599
testen 567
Typ 599

SAP HANA Web Workbench 38

SAP HANA XS → SAP HANA Extended Application Services

SAP Host Agent 33

SAP InfiniteInsight 515

SAP IQ (Datenbank) 41

SAP Landscape Transformation Replication Server 40

SAP List Viewer 312, 510, 511
with Integrated Data Access 312

SAP List Viewer mit Integrated Data Access 302

SAP Lumira 438

SAP Management Console 33

SAP Memory 594

SAP NetWeaver AS ABAP 36
 7.4 296
 7.5 289, 296

SAP Operational Process Intelligence 499

SAP Predictive Analysis 515

SAP Replication Server 40

SAP S/4HANA 63

SAP Solution Manager 34

Scale-out 43, 117

Schattenspalte 463

Schema → Datenbankschema

Schemabehandlung 248, 561

Schema-Mapping 263, 561

Schleife 603

Schlüsselfeld 206, 302

Scope
Element 307
Extend View 307
View 307

Scope-Operator 296

Scorecard 514

Script Server 517

Scrollen 312

SDK → Software Development Kit

Searched Case 298

Segmentierung 523

Sekundärdatenbank 153, 450

Sekundärindex 588

Sekundärverbindung 164

SELECT*-Anweisung 418

SELECT-Anweisung 571
geschachtelte 418, 584

SELECT-Klausel 298

Selektivität 461

Sentiment-Analyse 456, 458, 490

Sequenz 102, 171

Serverkomponente 58

Session-Kontext 209

Session-Variable 297

SFLIGHT 117, 609, 610

Shared Buffer 594

Shared Memory 594

Shared Objects 594

Side-by-Side-Szenario 22

Sidepanel 453

Simple Case 298

Single Transaction Analysis 336, 368

skalarer Parameter 189

Slice-and-dice 220

Smart Data Access 40, 174

Software Development Kit 515

Softwareinnovation 47

Sortieren 312

Sortierverhalten 344, 560

spaltenorientierte Datenablage 48, 103

Sparse-Codierung 53

Spatial Reference Identifier 537

Spatial Reference System 537

Sperre 563

Sperrobjekt 123, 132

SQL 129
 1999 (Standard) 172
ADBC 147
dynamisches 195, 603
Erweiterungen 275
Injection 195
natives 70, 146, 250
Open 69
SQL-Cache 336, 380, 598
SQL-Injection 570
SQL-Konsole 105, 187

SQL (Forts.)
SQL-Performanceoptimierungswerkzeug 401
SQL-Profil 415
SQL-Prozessor 59

SQL Injection 141

SQL Monitor 336, 381, 401, 415
aktivieren 381
Auswertung 382
Einstiegspunkt 384

SQL Performance Tuning Worklist 393

SQL View 175, 277, 278, 293
Definition 285

SQL-92 (Standard) 172

SQL-Analyse, systemweite 376

SQL-Anweisung
Analyse 403, 416
BINARY SEARCH 560
CREATE FULLTEXT INDEX 463
EXEC 569
EXEC SQL 558
FOR ALL ENTRIES 585, 586
INTO CORRESPONDING FIELDS OF 577
ORDER BY 559
*SELECT ** 418
SELECT COUNT()* 579
UP TO n ROWS 575
UPDATE 580
UPDATE ... SET 580

SQL-Ausdruck, in Open SQL 317

SQLDBC-Bibliothek 35

SQL-Dialekt 170

SQLScript 70, 422
ABAP-Tabellen 562
Ausgabeparameter 252
benutzerdefinierte Funktion 188
BREAK 193
CONTINUE 193
CREATE PROCEDURE 189
CREATE TYPE 190
Debugger 325
dynamisches 569
Eingabeparameter 256
EXEC 195
EXECUTE IMMEDIATE 195
explizite Typisierung 191
Fallunterscheidung 184, 193
implizite Typisierung 191

SQLScript (Forts.)
Optimierungen 195
Qualitäten 179
Regeln 602
Schleife 184, 193
skalarer Parameter 189
Tabellenparameter 189
Tabellentyp 189
Tabellenvariable 181
Typisierung 192
UDF 188
User-defined Function 188
Wiederverwendung 181
WITH RESULT VIEW 190

SQL-Trace 159, 335, 364, 405, 415, 421
aufzeichnen 159, 364
auswerten 365

SRS → Spatial Reference System

SRTCM, Prüfung 391

ST_GEOMETRY 174, 539

Stack-Trace 366

Stammdaten 220

Standarddatenbankverbindung 121

Standard-View 175

Statistiksatz 335

Statistikserver 60

Sternschema 203, 219, 220

Stoppwort 461, 473

Stored Procedure → Datenbankprozedur

String 52

Structured Query Language → SQL

Struktur 311, 620

Strukturkomponente 311

Subselect 298

Suche
exakte 468
Freestyle 457
Fuzzy 456, 457
linguistische 457, 460, 468
Synonymsuche 457

Suchfacette 458

Suchhilfe 131, 455, 480
elementare 481
Suchhilfe-Exit 483

Summentabelle 203

Synonym 102, 111, 171

Synonymliste 460, 475

Syntaxprüfung 93

Syntaxwarnung, unterdrücken 296
 Systemfeld 555
 Systemlandschaft, gemischte 269
 Systemoptimierung 402
 Systemschema 101, 121

T

Tabelle 130, 170
 interne 620, 621
 replizierte 633
 Tabellendefinition 103
 Tabelleninhalt 103
 Tabellenparameter 189
 Tabellenpuffer 120, 133, 346, 562,
 587, 593
 Tabellenstatistik 412
 Tabellentyp 189
 Tabellenvariable 181
 temporäre Tabelle 254
 Temporäre Tabelle 173
 Term Mapping 475
 Test 567
 Text Join 206
 Textanalyse 456, 458, 489
 Text-Mining 39
 Textsuche 458, 462
 Thread 379
 Token 462, 489
 Trace 351
 Tracing 342
 Transaktion
 ATC 335
 DBACOCKPIT 154, 336, 376
 RSA1 444
 RSDD_HM_PUBLISH 443
 RSRT 448
 SAT 335, 357, 413, 420
 SCI 335, 343
 SE11 89
 SE80 486
 SFW5 488
 SQLM 336, 381
 SRTCM 390
 ST04 376
 ST05 159, 164, 165, 335, 387, 421
 ST12 336, 368
 ST22 339
 STAD 335, 352, 404, 411, 420

Transaktion (Forts.)
 STATS 356
 SWLT 336, 393, 415
 transaktionales System 434
 Transport
 Änderungsauftrag 267
 Developer Mode 263
 Empfehlungen 270
 gemischte Systemlandschaft 269
 logisches Transportobjekt 266
 Protokoll 269
 Synchronisation 268
 Transportcontainer 266
 Transportcontainer 265
 TREX 461
 Trigger 171

U

UNION 298
 UNION ALL 299, 303
 Unit Test 337, 339, 405, 568
 Unterabfrage 140, 585
 Unterprogramm 144
 UPSERT 161

V

Validierung 109
 VALUE-Operator 620
 Variable
 deklarieren 617
 globale 297
 Gültigkeitsbereich 619
 Session 297
 Verbuchung 117
 Verbuchungsbaustein 144
 Versionshistorie 113
 Vertreterobjekt 273
 Verwendungsnachweis 95
 View 103, 131, 170, 203, 277
 Analytic View 204, 219
 Attribute View 204
 Calculation View 204, 228
 Datenbank-View 145
 Dictionary View 243
 externer 249, 273, 561, 565
 SQL View 175
 View-on-View-Muster 313

Vorschlagsliste 457
 Vorwärtsnavigation 94

W

Wahrheitswert 307
 Währungsumrechnung
 Customizing 176
 parametrisieren 227
 Web Dynpro ABAP 486
 Eclipse 486
 Kontext 486
 Werkzeuge zur Performanceanalyse
 Laufzeitprüfungs-Monitor (SRTCM)
 390
 Worthilfe → Suchhilfe
 WHERE 294, 298
 WHERE-Klausel 571, 587
 White List 570
 Wildcard 455, 468
 WITH PARAMETERS 295
 Workprozess 117, 255

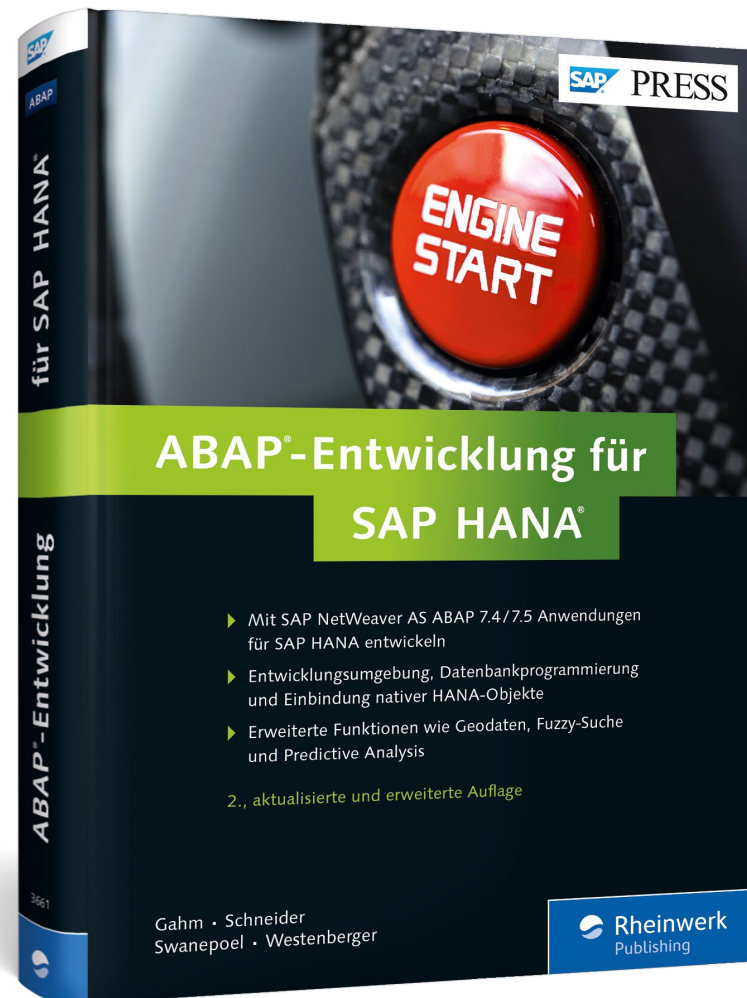
Wort-Dictionary 492
 Wrapper-Funktion, AFL 518

X

XS Engine 37, 98, 102, 570
 XS Server 60

Z

zeilenorientierte Datenablage 48
 Zeitdaten, generieren 217
 Zeitzone 556
 Zieldatenquelle 288, 289
 Kardinalität 289
 Zielkardinalität 294
 Zugriffsfunktion 197
 Zugriffszeit 44
 CPU-Cache 45
 Festplatte 44
 Flash-Speicher 45
 Hauptspeicher 45



Hermann Gahm, Thorsten Schneider,
Christiaan Swanepoel, Eric Westenberger

ABAP-Entwicklung für SAP HANA

653 Seiten, gebunden, 2. Auflage 2015
69,90 Euro, ISBN 978-3-8362-3661-4

 www.sap-press.de/3773

Hermann Gahm ist Principal Consultant im Performance CoE der SAP Global IT Application Services.

Thorsten Schneider ist Produktmanager im Bereich Product & Innovation HANA Platform bei der SAP AG.

Christiaan Swanepoel arbeitet seit 2003 für die SAP AG. Aktuell ist er Product Owner im Bereich der ABAP-Entwicklungstools für Core Data Services (CDS) in Eclipse.

Dr. Eric Westenberger arbeitet seit 2005 für die SAP AG und ist aktuell als Produktmanager für SAP HANA und SAP NetWeaver tätig.

Wir hoffen sehr, dass Ihnen diese Leseprobe gefallen hat. Sie dürfen sie gerne empfehlen und weitergeben, allerdings nur vollständig mit allen Seiten. Bitte beachten Sie, dass der Funktionsumfang dieser Leseprobe sowie ihre Darstellung von der E-Book-Fassung des vorgestellten Buches abweichen können. Diese Leseprobe ist in all ihren Teilen urheberrechtlich geschützt. Alle Nutzungs- und Verwertungsrechte liegen beim Autor und beim Verlag.

Teilen Sie Ihre Leseerfahrung mit uns!

