Fachhochschule Köln

Abteilung Gummersbach

Diplomarbeit

Electronic Publishing mit SAP R/3

Produktklassifizierung und automatisierte Produktdatenaufbereitung für die medienunabhängige Produktkatalogerstellung mit SAP - R/3

Autor:

Adnan Duman

Matrikel-Nr.: 0 10 22 91 98

Wirtschaftsinformatik

Email: pss-software@t-online.de

Betreuung:

Prof. Dr. Heide Faeskorn - Woyke

1.

2. Inhaltsverzeichnis

3.

4. Vorwort

In dieser Diplomarbeit wird die Lösung eines Standardproblems aus dem betrieblichen Umfeld behandelt. Die Aufgabenstellung ist die automatisierte Bereitstellung von Daten, die für die Produktion eines medienunabhängigen Produktkataloges notwendig sind. Ein zentraler Datenbestand soll als Quelle für Printkataloge, CD-ROM- und Internet-Präsentationen dienen.

Der Inhalt dieser Diplomarbeit war ein konkretes Projekt der Firma August Rüggeberg GmbH & Co aus Marienheide bei Gummersbach. Zu Beginn dieser Arbeit waren bereits einige Anbieter von Electronic - Publishing - Systemen bei dem Unternehmen vorstellig geworden, deren Konzepte als Orientierung dienen konnten, jedoch nicht den erhofften Durchbruch brachten. Es handelte sich in allen Fällen um proprietäre Systeme, die eine eigene Datenbank nutzten. Die daraus folgende doppelte Datenhaltung und –pflege war von dem Unternehmen nicht gewünscht. Die Firma Rüggeberg hatte zu dieser Zeit eine ca. einjährige Einführungsphase von SAP R/3 erfolgreich abgeschlossen, so daß die Integration von R/3 in die Lösung natürlich einen hohen Stellenwert hatte, weil alle operativen Daten bereits in R/3 abgelegt wurden. Eine echte Online - SAP R/3 - Anbindung konnte zu diesem Zeitpunkt keiner der Anbieter präsentieren.

In diesem Umfeld sollte die Diplomarbeit sich mit den Realisierungsmöglichkeiten des Projektes befassen. Während der konkreten Definitionsphase des Diplomthemas fiel auf, daß kein Anforderungsprofil existierte, an dem die gebotenen Lösungen konkret gemessen werden konnten und für das ein Lösungskonzept entwickelt werden sollte. Es bot sich an, die Erstellung des Anforderungsprofils mit in die Arbeit aufzunehmen.

Um ein konkretes Anforderungsprofil an das Endprodukt erstellen zu können war erst einmal eine Bestandsaufnahme nötig, an dessen Ergebnis sich die realisierbaren Umsetzungsmöglichkeiten orientieren sollten. Diese Bestandsaufnahme betraf sowohl die technischen Mittel

(insbesondere von R/3), als auch die Definition der zu verarbeitenden Informationen und Daten, die benötigt wurden und die evtl. bereits im R/3 System vorhanden waren.

Da das Unternehmen noch keine langjährige Erfahrung mit dem SAP R/3 System hatte, sollte diese Diplomarbeit ursprünglich nur die grundsätzlichen Möglichkeiten des R/3-Systems darstellen. Das hohe persönliche Interesse des Autors führte dazu, daß auch ein detailliertes Konzept für die konkrete Umsetzung und erste Programmroutinen erstellt werden konnten.

Zu Beginn der Diplomarbeit drückte der Rüggeberg - hausinterne Begriff

Multimediadatenbank aus, daß eine zentrale Datenbank Daten unterschiedlichster Art (Bilder, Texte, Klänge, evtl. sogar Animationen) liefern sollte. Bestehende operative Daten sollten automatisch genutzt werden und zusätzliche Daten sollten transparent und redundanzfrei hinzugepflegt werden können. Es existierte jedoch noch keine Vorstellung darüber, wie das Projekt organisatorisch und technisch realisiert werden könnte.

An diesem Punkt wandte sich die Firma Rüggeberg an die Fachhochschule Köln, Abtl. Gummersbach und bot das Projekt als Diplomarbeit an. Da der Aufwand für eine einzige Diplomarbeit zu umfassend war, beschloß der Autor gemeinsam mit seinem Kommilitonen Michael Cuti, das Projekt zu teilen. Der Autor übernahm den konzeptionellen Teil, der sich mit der Problemanalyse, R/3 – Integration und R/3 – Implementation befaßte und Herr Cuti baute in seiner Diplomarbeit auf dieser Arbeit auf, indem er die Internet / CD-ROM - Präsentation übernahm.

Da der Autor vorher noch keinerlei praktische Erfahrung im Umgang mit dem R/3-System sammeln konnte, war eine intensive ca. 8-wöchige Einarbeitung in SAP R/3 vonnöten.

Für die freundliche und umfassende Unterstützung der Firma Rüggeberg GmbH & Co aus Marienheide, insbesondere der EDV- und der Marketingabteilung, möchte ich an dieser Stelle meinen persönlichen Dank aussprechen. Ohne das Entgegenkommen und das Vertrauen gegenüber meiner (unserer) Person(en) wäre diese Arbeit in der vorliegenden Fassung nicht möglich gewesen. Erwähnenswert sind dabei die umfassenden Zugriffsberechtigungen auf das R/3-System, die Bereitstellung eines eigenen R/3-Arbeitsplatzes und die gute persönliche Zusammenarbeit. Auch der Betreuerin und Prüferin Frau Prof. Dr. Faeskorn - Woyke von der Fachhochschule Köln, Abt. Gummersbach gebührt der Dank des Autors für die flexible und pragmatische Zusammenarbeit.

Die Arbeit bezieht sich auf R/3 in der Version 3.1h. Die rasche Entwicklung auf dem Markt der Standard - Software machte sich auch während des Zeitraums dieser Diplomarbeit bemerkbar. SAP kündigte die Version 4.0 von R/3 an. Der Autor informierte sich über die Neuerungen und stellte fest, daß diese keine grundsätzliche Relevanz für die hier vorgestellten Zusammenhänge haben, so daß die Ergebnisse auch auf die Version 4.0 übertragen werden können.

Einführung

Die fortschreitende Integration übergreifender Prozesse in Unternehmen ist im Informationszeitalter eng mit der Verfügbarkeit entsprechender Software verbunden. Die betriebswirtschaftliche Standardsoftware SAP R/3 der Walldorfer Firma SAP AG hat sich seit ihrer Veröffentlichung als ein weltweiter Standard etabliert. Dieser Standard kann als Basis für Standardlösungen im betrieblichen Umfeld herangezogen werden.

Diese Arbeit befaßt sich in erster Linie mit der Nutzung der Synergien, die durch das integrative Konzept von R/3 möglich werden. Dazu ist die überwiegende Nutzung der in R/3 bereits vorhandenen Standard - Funktionalitäten Voraussetzung. Ziel ist die automatisierte Bereitstellung von Daten, die für die Produktion eines medienunabhängigen Produktkataloges notwendig sind. SAP R/3 soll als zentraler Datenbestand für Printkataloge, CD-ROM und Internet-Präsentationen dienen.

In **Kapitel 1** wird das System SAP R/3 allgemein vorgestellt.

Es werden die Konzepte und Möglichkeiten dargelegt, wie R/3 Informationen speichert und sie systemweit verfügbar macht. Bei der Beschreibung der R/3 – Funktionalitäten hat der Autor das vordergründige Ziel, ein Verständnis für das System R/3 zu vermitteln, d.h. das Zusammenspiel verschiedener R/3 Komponenten zu beschreiben. Es werden (fast) alle von R/3, insbesondere der *ABAP/4 – Workbench*, anwendungsübergreifend zur Verfügung

gestellten Funktionalitäten, Werkzeuge und Verfahren zur Abbildung kundenindividueller Informationen vorgestellt. Dabei wird auch erläutert, was sich hinter den Masken verbirgt und wie eigene Erweiterungen und Anpassungen realisiert werden können.

In **Kapitel 2** wird die Aufgabenstellung *Produktkatalog* am Beispiel des Produktkataloges der Firma Rüggeberg GmbH & Co analysiert.

Dabei werden die konzeptionellen Probleme der Aufgabenstellung dargestellt und die Informationsstrukturen mit ihren logischen Beziehungen untereinander ausgearbeitet.

In **Kapitel 3** findet die Symbiose zwischen den Ergebnissen aus Kapitel 1 und Kapitel 2 statt. Die Informationsstrukturen aus Kapitel 2 werden, unter Beachtung der vorher definierten Anforderungen, mit den technischen Möglichkeiten von SAP R/3 in Einklang gebracht.

Für jedes, von R/3 angebotene Verfahren, werden die Vor- und Nachteile zur Abbildung einer bestimmten Informationsart erläutert. Anschließend wird die jeweils geeignetste Möglichkeit ausgewählt.

In **Kapitel 4** wird ein einsatzfähiges Konzept ausgearbeitet und dargestellt.

Aufbauend auf den Ergebnissen aus Kapitel 3 werden schrittweise alle R/3 – Transaktionen und Bildfolgen, die für die Abbildung und Pflege von Produktkatalogdaten notwendig sind, dargestellt und erläutert.

In **Kapitel 5** schließlich wird die Konzeption einer externen Softwarelösung für die Datenpflege vorgestellt. Es wird der Bedarf nach einer externen Softwarelösung begründet und die technische Anbindung bzw. Integration mit SAP R/3 beschrieben.

Kapitel 1 – Das System SAP R/3

1. Allgemeines

SAP R/3 ist eine Standardsoftware, die betriebswirtschaftliche Abläufe, die in Unternehmen der verschiedensten Branchen und Größen vorkommen, abbilden kann. Ermöglicht wird dies durch umfangreiche Parametrisierungen, die es erlauben Unternehmensprozesse und -strukturen möglichst realitätsgetreu und flexibel in R/3 nachzubilden.

Das R/3-Konzept hat zum Ziel, daß jede Information, die in einem Unternehmen verarbeitet wird, nur einmal erfaßt wird (idealerweise an ihrem Entstehungsort) und in jedem Prozeß des Unternehmens, in dem sie benötigt wird, verfügbar ist.

1. R/3 – Grundlagen

- Mandanten

Alle eingegebenen Daten (Stammdaten etc.) werden von R/3 mandantenbezogen abgelegt. Ein Mandant repräsentiert ein ganzes Unternehmen. Es können mehrere Mandanten innerhalb eines R/3-Systems angelegt werden, von denen nur einer produktiv eingesetzt wird, während ein anderer z.B. für die Schulung von Mitarbeitern genutzt wird.

Auch evtl. Anpassungen und Neuerungen werden üblicherweise nicht sofort im Produktivmandanten durchgeführt, sondern zuerst in einem Testmandanten überprüft, um dann über die R/3-Transportfunktion in den Produktivmandanten übertragen zu werden.

- Relationale Datenbank

  R/3 baut immer auf einer relationalen Datenbank auf, so daß die elementaren Konzepte relationaler Datenbanksysteme wenigstens teilweise auf R/3 übertragen werden können. Alle system- und anwendungsspezifischen Informationen werden in Tabellen einer relationalen Datenbank gespeichert. SAP bietet keine eigene Datenbank an, sondern es stehen verschiedene leistungsfähige Datenbanken von Fremdanbietern wie Oracle, Microsoft, Software AG etc. zur Verfügung.

- Bild

Ein R/3-*Bild* ist eine Bildschirmmaske.

-
- Anwendungssicht

  Es können *Anwendungssichten* auf bestimmte Daten angelegt werden. Für jede Anwendungssicht werden bestimmte Bildschirmfelder ein- bzw. ausgeblendet. Für das Material existieren z.B. Anwendungssichten für den Einkauf und den Verkauf, wo jeweils nur die für den Anwender relevanten Informationen angezeigt werden.

- Änderungsdienst

  Der Änderungsdienst führt Datenpflegetätigkeiten zu einem bestimmten Stichtag durch. Es können z.B. die Preise von Materialien bereits für die Zukunft gepflegt werden. Außerdem können Änderungen an Objekten (Materialdaten etc.) protokolliert werden.

- Status

  Viele Datenpflegetätigkeiten in R/3 können mit einem *Status* versehen werden. Dieser Status gibt Auskunft über den aktuellen Stand einer Datenpflege. Es könnten z.B. Materialien in den Materialstamm aufgenommen werden, die jedoch noch unvollständige Datenfelder haben. Diesen Materialien kann ein Status *Nicht freigegeben* vergeben werden. Erst wenn alle relevanten Daten gepflegt und der Status auf *freigegeben* geändert wird steht das Material im System zur Verfügung.

- Customizing

  Als Customizing wird die kundenspezifische Parametrisierung des R/3 – Systems bezeichnet. Darunter fallen grundsätzlich alle Tätigkeiten, die im Unternehmens-IMG durchgeführt werden.

- Unternehmens – IMG

  Das Unternehmens - IMG ist das zentrale Werkzeug für das Customizing. Es bildet in einer hierarchischen Struktur die Customizing - Funktionen ab und unterstützt ein strukturiertes Vorgehen bei der Anpassung von R/3.

1.

2. Das Klassensystem von SAP R/3

1. Einführung

Das R/3 - Klassensystem hat die Aufgabe, beliebige Objekte mit Hilfe von Merkmalen zu beschreiben und ähnliche Objekte nach frei zu definierenden Kriterien zu gruppieren.

Dieser Vorgang wird Klassifizierung genannt. Objekte können sowohl reale Objekte wie Materialien, Kunden, Lieferanten oder logische Objekte wie z.B. ein Antrag sein.

Ziel der Klassifizierung ist neben der spezifischen Beschreibung von Objekten durch die Merkmale eine gezielte Suche nach Objekten mit ganz bestimmten Merkmalausprägungen. So kann z.B. bei der Zusammenstellung eines Kundenauftrags im SD Modul (Vertrieb) gezielt nach bestimmten Materialien gesucht werden, indem nur Materialien mit bestimmten Merkmalausprägungen angezeigt werden.

Das R/3 - Klassensystem arbeitet anwendungsübergreifend, d.h. alle R/3 - Module können auf die Funktionalität des Klassensystems zugreifen.

Das Klassensystem ist ein mächtiges Werkzeug zur Anpassung von R/3 an unternehmensspezifische Anforderungen. Die Merkmale und ihre Ausprägungen können frei definiert werden. So kann z.B. ein Merkmal *Körpergröße* angelegt und dem Objekt *Kunde* zugeordnet werden, wenn dies für ein bestimmtes Unternehmen relevant ist.

In den folgenden Abschnitten werden die Komponenten des Klassensystems ausführlich dargestellt. Dabei werden auch die R/3 - spezifischen Begriffe eingeführt.

2. Merkmale

Ein Merkmal ist eine Objekteigenschaft. Diese Eigenschaft kann jede Art von Information beinhalten, die logisch einem bestimmten Objekt zuzuordnen ist.

Beispiele für Merkmale:

- Technische Merkmale: Länge, Breite, Höhe, Gewicht, Farbe, Material, Form, etc.
- Verwaltungsinformationen: Verantwortlicher Bearbeiter, Aktenzeichen, Eingangscode, etc.

Jedes Merkmal wird grundsätzlich unabhängig von seinem Verwendungszweck (d.h. seiner Zuordnung zu einem bestimmten Objekttyp) zentral definiert und gepflegt. Bei der Definition eines Merkmals werden u.a. der Merkmalname und die möglichen Merkmalwerte festgelegt. Die Zuordnung von einem Merkmal zu einem Objekttyp geschieht erst später bei der Klassifizierung.

Beispiel:

Merkmal	Merkmalwert
Farbe	Rot
	Gelb
	Grün
Bearbeiter	Schmitz
	Müller
	Meier
Form	Rund
	Eckig

Tabelle 1: Merkmale mit Merkmalwerten

Für jedes Merkmal können verschiedene Einstellungen vorgenommen werden, die das Verhalten des Merkmals beeinflussen:

Werteprüfung

Die verfügbaren Merkmalwerte können, für jedes Merkmal getrennt, einzeln als Liste angegeben werden (siehe Tabelle 1). Alternativ kann eine Tabelle im System angegeben werden, die sog. Prüftabelle. Die Prüftabelle ist eine eigens für diesen Zweck anzulegende Tabelle, die die verfügbaren Merkmalwerte beinhaltet. Damit könnten z.B. mehrere Merkmale die gleichen Merkmalwerte verwenden, indem sie auf dieselbe Prüftabelle verweisen. Für maximale Flexibilität sorgt die Möglichkeit der Verwendung von eigens entwickelten ABAP/4 (die R/3 – eigene Programmiersprache) Funktionsbausteinen zur Werteprüfung. Es kann angegeben werden, ob ausschließlich die verfügbaren Werte erlaubt sind, oder ob der Anwender auch andere (nicht angegebene) Werte für das Merkmal eintragen kann.

Wertehierarchie

Bei einer hohen Zahl von verfügbaren Merkmalwerten kann der Anwender mit Such- und Filterfunktionen die angezeigten Werte einschränken und so einen bestimmten Wert schneller finden. Die verfügbaren Merkmalwerte können aber auch in einer Wertehierarchie strukturiert abgelegt werden. Der Anwender kann bei der Bewertung aus einer Baumstruktur gezielter (als in einer Liste) einen bestimmten Wert suchen. Die Wertehierarchie ist eine gute Alternative zur simplen listenartigen Auflistung von Merkmalwerten.

Beziehungswissen

Für jedes Merkmal kann definiert werden, ob und wie es in Abhängigkeit (in Beziehung) zu anderen Merkmalen steht. Es können z.B. die Merkmale *Form* und *Farbe* in einer logischen Beziehung zueinander stehen. Es kann definiert werden, ob das Merkmal *Farbe* überhaupt bewertet werden soll (darf), wenn das Merkmal Form einen bestimmten Wert hat. Wenn es nicht bewertet werden soll, erscheint es gar nicht auf dem Bild zur Merkmalbewertung. Alternativ kann definiert werden, welche Merkmalwerte für das Merkmal *Farbe* erlaubt sind, wenn das Merkmal *Form* einen bestimmten Wert hat; d.h. die verfügbaren Merkmalwerte für ein Merkmal A können dynamisch in Abhängigkeit vom Merkmalwert eines Merkmals B eingeschränkt werden. Diese Abhängigkeiten zwischen Merkmalen werden *Beziehungswissen* genannt.

Da ein Merkmal mehreren Objekttypen zugeordnet werden kann (z.B. Merkmal *Bearbeiter* für Objekttypen *Kunde* und *Material*), ist die Möglichkeit gegeben, bestimmtes Beziehungswissen auf einen bestimmten Objekttypen einzuschränken (globales / lokales Beziehungswissen), d.h. eine

bestimmte Wertekombination der Merkmale A und B kann für das Objekt *Kunde* gültig sein, während es für das Objekt *Material* unzulässig ist.

Bei der Verwendung von Prüftabellen wird immer der Inhalt der Prüftabelle als verfügbare Wertemenge herangezogen. Evtl. vorhandenes Beziehungswissen wird nicht berücksichtigt.

Eingabe erforderlich

Wenn diese Funktion aktiviert ist muß ein Merkmal bei der Klassifizierung bewertet werden

Bewertungen

Ein Merkmal kann nur einen eindeutigen Wert oder aber mehrere Werte gleichzeitig haben, d.h. das Merkmal *Farbe* kann entweder nur den Wert ROT oder die Werte ROT und GELB gleichzeitig haben.

Die Merkmalbezeichnungen und Merkmalwerte (bei textlichen Merkmalwerten) können mehrsprachig eingetragen werden, d.h. das R/3-System erlaubt es, die Bezeichnung und die Wertevorgaben für das Merkmal *Farbe* (Werte: ROT, GELB, GRÜN) für jede Landessprache getrennt zu pflegen.

Merkmalgruppe

Durch die zentrale Ablage der Merkmale entsteht im Laufe des Aufbaus eines umfangreicheren Klassensystems eine stetig länger werdende Liste von Merkmalen. Zur besseren Verwaltung dieser Merkmale können *Merkmalgruppen* gebildet werden, die ähnliche oder logisch zusammengehörende Merkmale aufnehmen. Diese *Merkmalgruppen* haben keinen Einfluß auf das Verhalten des Klassensystems, sondern dienen lediglich der übersichtlicheren Merkmalpflege.

Das Zusammenspiel von Merkmalen, Merkmalwerten und Beziehungswissen eignet sich ideal für die Abbildung von Variantenkonfigurationen. Es können z.B. komplexe Produkte, die in vielen verschiedenen Ausführungen erhältlich sind, im Klassensystem abgebildet werden. So kann z.B. ein Auto, das in verschiedenen Ausführungen erhältlich ist, im Klassensystem abgebildet werden. Bei der Bestellung eines Autos mit ganz bestimmten Ausprägungen können sich gegenseitig ausschließende Kombinationen von Ausprägungen verhindert werden

(typisches Beispiel: Cabrio mit Schiebedach).

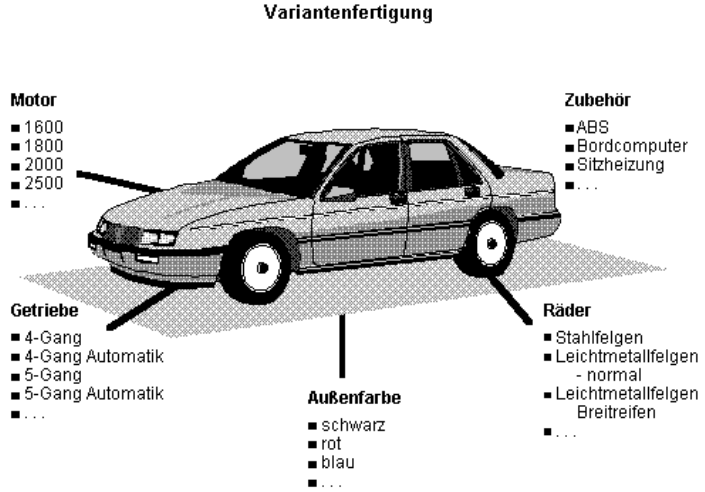


Bild: Ein konfigurierbares Produkt => Variantenprodukt

Objektmerkmal

Der Sonderfall eines Merkmals ist das Objektmerkmal. Ein Objektmerkmal ist ein Merkmal, welches bereits standardmäßig im Datenstamm eines Objektes vorhanden ist und zusätzlich zur Klassifizierung im Klassensystem verwendet wird (z.B. zur Objektsuche). Es herrscht also der umgekehrte Zusammenhang: Während ein *herkömmliches* Merkmal den Datenstamm eines Objektes erweitert, wird bei einem Objektmerkmal ein bestehendes Datenstammfeld in das Klassensystem übernommen.

1. **Klassen**

   Vereinfacht dargestellt ist eine Klasse eine Gruppierung von Merkmalen. Im Gegensatz zur bereits erwähnten *Merkmalgruppe*, die lediglich der Verwaltung von Merkmalen dient, ist eine Klasse funktionaler Bestandteil des Klassensystems. Durch die Zuordnung einer bestimmten Objektinstanz zu einer Klasse werden die Merkmale dieser Klasse der Objektinstanz zugeordnet.

   Um z.B. dem Objekt *Kunde* das Merkmal *Bearbeiter* zuzuordnen, muß zuerst das Merkmal *Bearbeiter* angelegt werden. Anschließend wird eine Klasse Kunden angelegt, der das Merkmal *Bearbeiter* zugeordnet wird. Um einer bestimmten Objektinstanz, d.h. einem bestimmten Kunden (Meier) das Merkmal *Bearbeiter* zuzuordnen und zu bewerten muß diese Objektinstanz, (Kunde Meier), klassifiziert werden. Die Klassifizierung findet dabei in zwei Schritten statt:

   Schritt 1:

   Im ersten Schritt wird der Kunde Meier der Klasse *Kunden* zugeordnet. Damit erweitert sich der Datenstamm des Kunden Meier um die Merkmale der Klasse *Kunden*; in diesem Fall um das Merkmal *Bearbeiter*.

   Schritt 2:

   Im zweiten Schritt werden die Merkmale bewertet, d.h. das Merkmal *Bearbeiter* wird mit dem Namen eines bestimmten Bearbeiters versehen. Es ist naheliegend, daß dieser Name als Liste bei den verfügbaren Merkmalwerten für das Merkmal *Bearbeiter* abgelegt ist.

   Die beschriebene Vorgehensweise zeigt folgenden Zusammenhang auf:

   Es wird nicht eine Klasse fest einem Objekttyp (Kunde) zugeordnet, der dann um die entsprechenden Merkmale (der Klasse) ergänzt wird, sondern es werden (objekt)neutrale

   Klassen angelegt, denen jede Instanz eines Objekttypen (Meier, Müller, ...) einzeln zugeordnet wird. Diese Vorgehensweise hat zur Folge, daß nicht grundsätzlich alle Instanzen eines Objekttyps (alle Kunden) die gleichen Merkmale haben müssen. Die Merkmale einer Objektinstanz sind abhängig von der individuellen Klassifizierung einer Objektinstanz.

   Dieser auf den ersten Blick unnötig erscheinende höhere Pflegeaufwand ist durch die Flexibilität des Klassensystems begründet. Es können z.B. zwei Kundenklassen angelegt werden, die verschiedene Merkmale beinhalten und so einen bestimmten Kundentyp abbilden:

   Privatkunden: Merkmale => Bearbeiter, Kaufverhalten

   Geschäftskunden: Merkmale => Bearbeiter, Rechtsform

Je nach Kundentyp wird ein Kunde in die Klasse *Privatkunde* oder die Klasse *Geschäftskunde* eingeordnet (klassifiziert). Dementsprechend werden ihm völlig unterschiedliche Merkmale zugeordnet. Dabei können natürlich auch einzelne Merkmale in mehreren Klassen vorkommen. Diese Mehrfachverwendung von Merkmalen macht auch den Vorteil der klassenunabhängigen Merkmalpflege deutlich. Eine Änderung der Merkmaleigenschaften (z.B. verfügbare Merkmalwerte) wird sofort in allen Klassen gültig, in denen das Merkmal verwendet wird.

Die Zuordnung eines Merkmals zu einer Klasse schränkt die weitere Verwendbarkeit des Merkmals in keiner Weise ein. Das Merkmal kann parallel dazu in einer anderen Klasse für einen anderen Objekttyp verwendet werden. Bei einer mehrfachen Verwendung eines Merkmals sollte jedoch immer beachtet werden, daß die Änderung einer Merkmaleigenschaft sich grundsätzlich auf alle Klassen, in denen das Merkmal verwendet wird, auswirkt. Klassenbezogene Merkmaleigenschaften sind zwar R/3 – technisch möglich, sollten jedoch vermieden werden, da sie die Übersichtlichkeit verringern.

1. ***Klassengruppe***

Zur besseren Verwaltung der Klassen können, analog zu den Merkmalgruppen, *Klassengruppen* gebildet werden, die ähnliche oder logisch zusammengehörende Klassen aufnehmen. Auch die Klassengruppen haben keinen Einfluß auf das Verhalten des Klassensystems, sondern dienen lediglich der übersichtlicheren Klassenpflege.

2.
3. **Mehrfachklassifizierung**

Ein Objekt kann auch in mehreren Klassen klassifiziert werden. Das Objekt bekommt dann die Summe aller Merkmale der zugeordneten Klassen zugeordnet. Doppelte Merkmale werden dabei ignoriert.

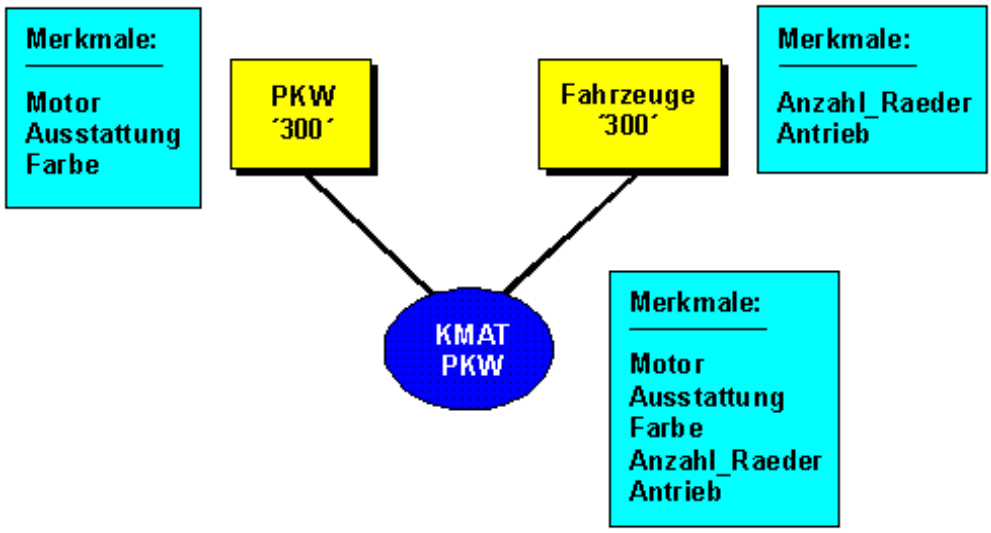


Bild: Mehrfachklassifizierung

4. **Klassenhierarchien**

Besonders bei der Materialklassifizierung können sehr umfangreiche und vielfältige Klassendefinitionen entstehen. Um eine übersichtliche Struktur und redundanzfreie Pflege im Klassensystem zu ermöglichen, können Klassenhierarchien (Baumstruktur) und sogar Klassennetze (Netzstruktur) aufgebaut werden.

Beispielsweise kann in eine Klasse *Kunden* mit dem Merkmal *Bearbeiter* definiert werden. Als Unterklassen der Klasse *Kunden* werden die Klassen *Privatkunden* und *Geschäftskunden* mit ihren entsprechenden Merkmalen *Kaufverhalten* und *Rechtsform* definiert.

Alle drei Klassen sind für die Klassifizierung geeignet:

- die Klassifizierung eines Objektes in der Klasse *Kunden* weist dem Kundenobjekt nur das Merkmal *Bearbeiter* zu.
- Die Klassifizierung eines Objektes in der Klasse *Privatkunden* weist dem Kundenobjekt sowohl das Merkmal *Bearbeiter*, als auch das Merkmal *Kaufverhalten* zu.
- Die Klassifizierung eines Objektes in der Klasse *Geschäftskunden* weist dem Kundenobjekt sowohl das Merkmal *Bearbeiter*, als auch das Merkmal Rechtsform zu.

Wenn nun ein Merkmal hinzugefügt werden soll, das für *Privatkunden* und *Geschäftskunden* relevant ist, muß dieses Merkmal nicht mehr in jede Klasse getrennt aufgenommen werden, sondern es genügt die Aufnahme in die Klasse *Kunden*. Dieses Konzept entspricht der Vererbung von Merkmalen zwischen Klassen. Es können beliebig tiefe baumartige Klassenstrukturen aufgebaut werden, um z.B. die Produkthierarchie eines Unternehmens abzubilden.

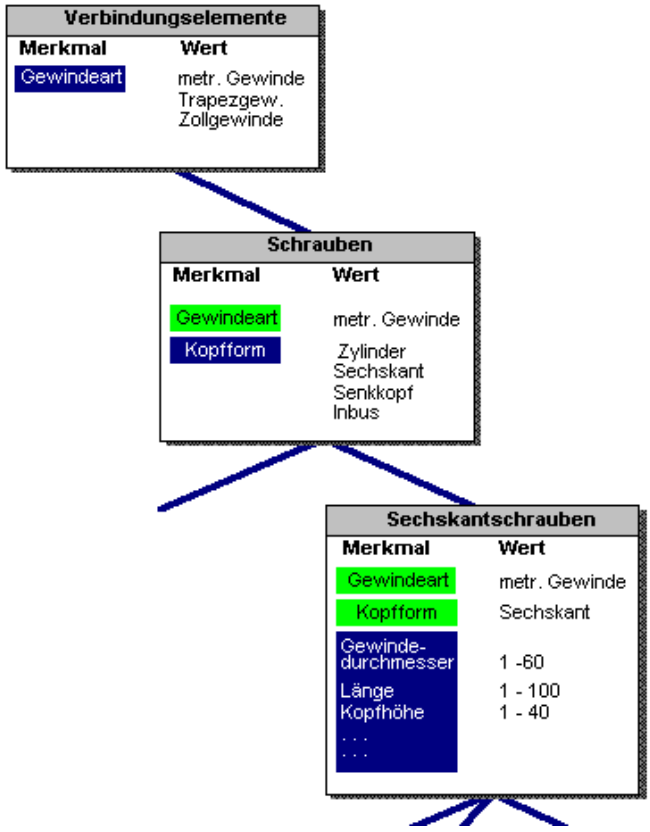
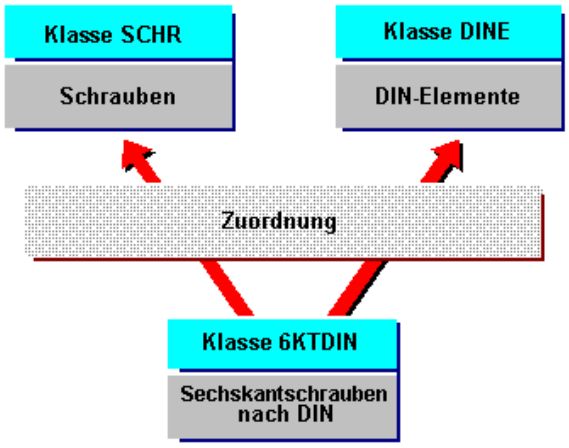


Bild: Merkmalvererbung

Durch die Möglichkeit der Zuordnung von mehreren übergeordneten Klassen zu einer Klasse können sogar Klassennetze aufgebaut werden, die sehr komplexe Abhängigkeiten abbilden



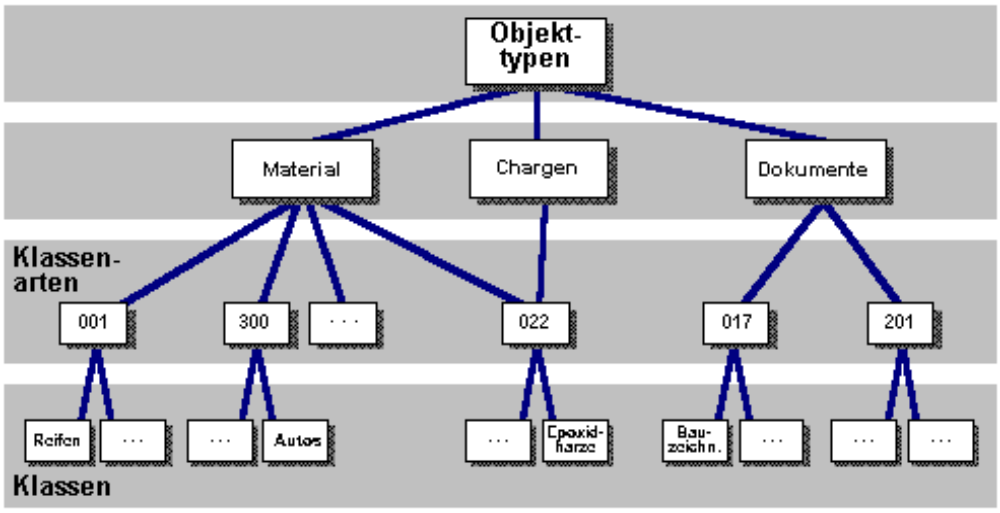
können.

Bild: Klasse mit zwei übergeordneten Klassen => Aufbau eines Hierarchienetz

1. **Klassenart**

Bisher wurde eine Klasse als objekttypunabhängige Gruppierung von Merkmalen betrachtet. Objekttypunabhängig meint, daß einer Klasse nicht zwingend nur ein bestimmter Objekttyp

(Kunde) zugewiesen werden darf. Die Merkmale, die eine Klasse beinhaltet, würden grundsätzlich jeder Objektinstanz, die in dieser Klasse klassifiziert wird, zugewiesen; d.h. es wäre theoretisch möglich, der Klasse *Kunden* ein Material zuzuordnen.



Im R/3 Klassensystem is dies nicht ohne weiteres möglich. Jede Klasse gehört zwingend einer bestimmten *Klassenart* an. Eine Klassenart definiert unter anderem, welche Objekttypen in einer Klasse klassifiziert werden können.

Bild: Die Hierarchie des R/3 Klassensystems

Eine Klassenart wird durch eine dreistellige numerische Kennung bezeichnet (=> 001, 010, 025, ...). Es wäre sehr wohl möglich in einer Klasse *Kunden* Material und Kunden zu klassifizieren, wenn diese Option in der Klassenart der Klasse *Kunden* (R/3 => 011 Debitoren) erlaubt würde. Neben dieser Option werden in einer Klassenart folgende Eigenschaften definiert, die für alle Klassen gelten, die einer bestimmten Klassenart zugeordnet sind:

- welche Objekte einer Klasse zugeordnet werden dürfen und ob einer Klasse Objekte unterschiedlichen Typs zugeordnet werden dürfen.

- welche Bilder in der Klassenpflege bearbeitet werden dürfen. Ein Bild ist eine SAP - Bildschirmmaske (Dynpro), die den Zugriff auf bestimmte Datenfelder und Funktionen

ermöglicht.

- ob Objekte in einer Klassenart in mehreren Klassen klassifiziert werden dürfen. Wenn ein Objekt z.B. in einer Klasse *Privatkunde* klassifiziert ist, darf es vielleicht nicht auch parallel in einer Klasse *Geschäftskunde* klassifiziert werden. In anderen Fällen mag eine Mehrfachklassifizierung durchaus sinnvoll sein (z.B. Materialklassifizierung).

- welche Klassenstati, Anwendungssichten und Textarten in der Klassenpflege zur Verfügung stehen.

- welche Klassifizierungsstati vergeben werden können. Ein Klassifizierungsstatus gibt den aktuellen Stand der Merkmalbewertung an.

- ob die Klassifizierung mit dem R/3 Änderungsdienst bearbeitet werden kann.

- welche Funktionen ausgewählt werden können, um das Ergebnis einer Objektsuche weiterzuverarbeiten.

- mit welchen Filterfunktionen das Suchergebnis weiter eingeschränkt werden kann.

Klassenarten werden im Customizing im Unternehmens IMG gepflegt. Der IMG-Pfad ist:

[Anwendungsübergreifende Komponenten] [Klassensystem] [Klassen] [Objekttypen] [Klassen] [Einstellungen für die Klassendefinition]

1.
2. **Prinzipielle Vorgehensweise im Klassensystem**

Das Konzept und die daraus resultierende Flexibilität des R/3-Klassensystems haben zur Folge, daß beim Aufbau eines Klassensystems planmäßig und in der richtigen Reihenfolge vorgegangen werden muß. Ein einfaches *drauflos - Klassifizieren* wird zwangsläufig eine sehr unübersichtliche Klassenstruktur erzeugen, die mehr schadet als nützt. Der Autor hält, nach einigen *Selbstversuchen*, folgende Vorgehensweise für empfehlenswert:

- Definition der abzubildenden Merkmale mit Merkmalgruppen, Merkmalbezeichnung, Merkmalwerte und Beziehungswissen. Es werden anhand der zu klassifizierenden Objekte (Produkte) Kriterien für die Merkmal-, Klassen- und Hierarchiebildung definiert. Dieser Vorgang erfordert hohe fachliche Kenntnis der zu klassifizierenden Objekte und ist die Voraussetzung für eine sinnvolle Struktur der Klassenhierarchie. Dieser Schritt findet in der Regel in enger Zusammenarbeit mit Fachleuten aus den betroffenen Fachabteilungen statt. Diese Planung sollte noch nicht im R/3 – System durchgeführt werden. Es existieren verschiedene Software - Tools, die das Designen solcher Beziehungen unterstützen.

- Untersuchung der bereits im R/3-System vorhandenen Merkmalgruppen, Merkmale, Merkmalwerte und Beziehungswissen und Abgleich zwischen vorhandenen und anzulegenden Merkmalen. Es können evtl. bereits vorhandene Merkmale verwendet werden. Auch die Verwendungsmöglichkeit von Objektmerkmalen sollte berücksichtigt werden.

- Anlegen der neuen Merkmale und Merkmalgruppen in R/3. Die genauen Schritte werden später in dieser Arbeit näher erläutert.

- Definition einer Klassenhierarchie mit Klassen, Merkmalen und Vererbungen.

- Untersuchung der bereits im System vorhandenen Klassen, Klassenhierarchien und Merkmalvererbungen und Abgleich zwischen vorhandenen und anzulegenden Klassen und Klassenhierarchien. Es können evtl. bereits vorhandene Strukturen verwendet werden. Die

hinzukommenden Klassen und Klassenhierarchien können auch Änderungen an bestehenden Klassen zur Folge haben, wenn logische Abhängigkeiten bestehen.

- Anlegen der Klassen, Zuordnung der Merkmale zu den Klassen und Hierarchiebildung.
- Klassifizierung der Objekte

1.

2. Das R/3 Dokumentenverwaltungssystem (DVS)

1. Einführung

Das SAP R/3-System bietet ein anwendungsübergreifendes Dokumentenverwaltungssystem (DVS) an. Die Hauptaufgabe ist die zentrale Speicherung und Verwaltung von Informationen in Dokumentenform, die jederzeit systemübergreifend von den verschiedenen R/3 Modulen und Anwendungen angefordert und bearbeitet werden können.

Ein Dokument besteht im R/3-DVS aus einer Dokumentenbeschreibung und dem eigentlichen Dokument z.B. einer Text- oder Fotodatei. Im SAP - Sprachgebrauch wird die eigentliche Dokumentendatei als das *Original* bezeichnet. Als *Dokument* wird die Verbindung einer Dokumentenbeschreibung mit einem Original bezeichnet. Die inhaltliche Form des Originals unterliegt keinerlei Restriktionen. Die Datei wird als Black-Box lediglich verwaltet, d.h. es finden vom DVS bzw. dem R/3-System keinerlei Interpretationen des Dateiinhaltes statt.

Die Darstellung und Bearbeitung des Originals kann über geeignete externe Software-Anwendungen (SAP: W orkstation- A pplikation = WS) wie z.B. WinWord, Excel, AutoCAD, etc. durchgeführt werden. Der automatisierte Aufruf der jeweils zu verwendenden Applikation findet durch entsprechende Parametrisierung statt. Dazu müssen während des Customizing im IMG alle zu verwendenden Workstation-Applikationen registriert werden. Die Verknüpfung zwischen einer Originaldatei und einer Applikation geschieht dabei über die 3-stelligen Dateiendungen (z.B. *DOC* => WINWORD).

Je Dokument können max. 2 Originale abgelegt werden, was die Zusammenfassung logisch zusammengehörender Originale erleichtert. So kann für eine CAD-Zeichnung mit der passenden Beschreibung, die z.B. in Form einer Textdatei vorliegt, ein einziges Dokument angelegt werden. Mit dem Abruf der CAD-Zeichnung steht damit auch gleich die Beschreibung zur Verfügung.

Innerhalb des R/3-Systems erscheint ein Dokument immer nur in Form der Dokumentenbeschreibung, die für jedes Dokument bei seiner Aufnahme in das DVS angelegt werden muß. Jede Originaldatei, die im System abgelegt wird, wird damit in ein standardisiertes *Paket* abgelegt, das von allen Modulen einheitlich behandelt werden kann.

Dokumente können mit einzelnen R/3 – Objekten *verknüpft* werden. Es kann z.B. ein bestimmtes Material mit einem Dokument *verknüpft* werden, das technische Zeichnungen zu diesem Material beinhaltet.

2. Konzepte und Funktionalitäten

Die Darstellung der gesamten Leistungsfähigkeit und Komplexität der R/3 – Dokumentenverwaltung würde den Rahmen dieser Arbeit verlassen. Im folgenden Abschnitt werden die elementaren Funktionalitäten der R/3 – DVS umrissen, um eine Vorstellung von seiner Verwendbarkeit zu bekommen. Detailliertere Informationen können in der R/3 – Dokumentation unter *[R/3 – Bibliothek | Anwendungsübergreifende Funktionen | Dokumentenverwaltung]* eingesehen werden.

Beschreibung/Langtext

Es können allgemeine Informationen und Hinweise zum Inhalt des Dokumentes eingetragen werden. Diese Informationen können später auch der Suche nach einem Dokument dienen.

Dokumentenart

Die Dokumentenart legt fest, wie das Dokument in folgenden Bereichen vom R/3-System zu behandeln ist:

- Bildaufbau
  Vergabe der Dokumentnummer
- Festlegung der Nummernkreise
- Versionsnumerierung
- Statusverwaltung
- Vergabe eines Revisionsstandes,
- Ablage im Archiv
- Zusatzdaten

Die verschiedenen Dokumentarten werden während des Customizings von R/3 im Unternehmens-IMG definiert.

Berechtigungsgruppe

Die Zuweisung einer Berechtigungsgruppe an ein Dokument schränkt die Verfügbarkeit eines Dokumentes auf die Benutzer ein, deren Benutzerstammsatz die passende Berechtigungsgruppe enthält. Diese Funktion stellt sicher, daß nur autorisierte Benutzer den Zugriff auf schützenswerte Dokumente haben.

Klassifizierung

Da jedes Dokument ein Objekt im Sinne von R/3 ist, kann das Klassensystem uneingeschränkt für die Dokumentenklassifizierung verwendet werden. Damit stehen auch die entsprechenden Such- und Filterfunktionen des Klassensystems zur Dokumentenfindung zur Verfügung.

Zusatzfelder

Über das Klassensystem kann jedes Dokument um individuelle Merkmale ergänzt werden. Die Funktion *Zusatzfelder* stellt eine besondere, auf das DVS ausgelegte, Integration des Klassensystems dar. Bei der üblichen Klassifizierung muß der Anwender für jedes Dokument, das er klassifizieren möchte, einzeln die allgemeine Transaktion zur Objektklassifizierung aufrufen , eine Klasse wählen und die entsprechenden Merkmale bewerten.

Diese Schritte können für die Dokumentenverwaltung teilweise automatisiert werden:

Durch die einmalige Zuordnung einer Klasse A an eine Dokumentart B kann definiert werden, daß alle Dokumente dieser Dokumentart B Objekte dieser Klasse A sind. Damit findet der erste Schritt der Klassifizierung, nämlich die Auswahl einer Klasse, bereits mit Auswahl der Dokumentart statt. Der zweite Schritt, die Bewertung der Merkmale dieser Klasse, findet ebenfalls innerhalb der Pflegemaske für die Grunddaten des Dokumentes statt. Die Merkmale der Klasse der gewählten Dokumentart erscheinen als zusätzliche Eingabefelder mit auf der Maske der Grunddaten des Dokumentes und können mit den bekannten Mitteln der Klassifizierung bewertet werden. Ziel dieser Integration ist die Erleichterung der Pflege der Klassen - Merkmale für den Anwender, da er nicht für jedes Dokument die allgemeine Transaktion zur Objektklassifizierung aufrufen muß. Außerdem erscheint dem Anwender die Verwendung des Klassensystems völlig transparent, d.h. der Schulungsaufwand für die Benutzer der Dokumentenpflege ist geringer.

Aus dem Bezug einer Klasse zu einer Dokumentart resultiert auch, daß alle Dokumente einer Dokumentart die gleichen Zusatzfelder besitzen. Durch Verwendung von unterschiedlichen Klassen für die verschiedenen Dokumentarten können die Dokumente unterschiedliche Zusatzfelder besitzen. Die Zuordnung einer Klasse zu einer Dokumentart findet im Customizing im Unternehmens-IMG unter [Anwendungsübergreifende Komponenten] [Dokumentenverwaltung] [Steuerungsdaten] [Dokumentarten definieren] statt.

Status

ist der aktuelle Bearbeitungstatus eines Dokumentes. Es können eigene Stati definiert und komplexe Statusnetze aufgebaut werden, die eine vorgegebene Bearbeitungsreihenfolge erzwingen, indem ein Status nur gesetzt werden kann, wenn entsprechende vorherige Stati gesetzt waren. Jede Statusänderung kann Nachrichten auslösen, Workflow-Prozesse anstoßen oder ABAP/4 - Programmroutinen aufrufen, so daß eine systemübergreifende Abstimmung und Integration der verschiedenen Bearbeitungsschritte eines Dokumentes möglich ist.

Dokumenthierarchie

Über eine Dokumenthierarchie können die logischen Beziehungen zwischen verschiedenen Dokumenten abgebildet werden. Dabei kann jedem Dokument ein übergeordnetes anderes Dokument zugewiesen werden, so daß eine baumartige Hierarchiestruktur entsteht. Dieser Hierarchiebaum kann visuell dargestellt werden um eine übersichtliche Verwaltung komplexer Dokumentenbeziehungen zu ermöglichen.

Eine DOKUMENTSTÜCKLISTE ist ebenfalls eine Möglichkeit, Beziehungen zwischen Dokumenten abzubilden. Ihre Funktion entspricht im wesentlichen der Funktion von *Material-stücklisten* , wo mehrere Einzelteile zu Baugruppen zusammengefaßt werden, die wiederum Teil anderer Stücklisten sein können.

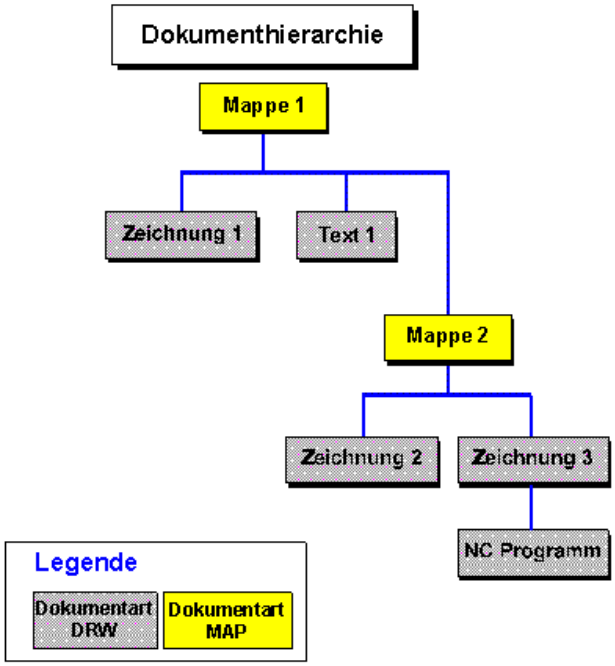


Bild: Dokumenthierarchie

1.

2. Werbemittel

Das Planen und Erstellen von *Werbemittel*n wie Katalogen, Prospekten, CD-ROM's und Internet-Präsentationen (=> *Werbemittel*) erfordert in der Regel eine aufwendige Verwaltung verschiedenster Informationen. Dies sind u.a. *Werbemittel* bezogene Materialstammdaten, Strukturinformationen, Multimediainformationen (Bilder, Töne etc.) und Textinformationen.

Das Modul *Werbemittel* ist das von R/3 angebotene Verfahren für die Bereitstellung der Daten zur Erstellung von Katalogen, CD-ROM's und Internet-Präsentationen.

Die in R/3 integrierte *Werbemittel* planung unterstützt die zentrale Erfassung und Pflege von *Werbemittel*relevanten Daten. Dabei werden die eigentlichen Produkt- und Katalogdaten nicht direkt in der *Werbemittel* planung eingegeben, sondern lediglich *Werbemittel* bezogen neu strukturiert und zusammengefaßt. Ziel der *Werbemittel* planung ist nicht die tatsächliche Erstellung und Produktion der *Werbemittel*, sondern lediglich die Bereitstellung der für diese Produktion relevanten Daten. Diese Daten können über eigens für diesen Zweck integrierte Schnittstellen (BAPI's) in eine professionelle Erstellungssoftware exportiert werden.

1. Aufbau baumartiger Hierarchien

Es kann eine Hierarchiestruktur aufgebaut werden, deren Interpretation von der später verwendeten Erstellungssoftware beeinflußt werden kann. Die anwendungsbezogene Interpretation der *Werbemittel* wird an späterer Stelle bei der Ausarbeitung des Konzeptes detaillierter erläutert. Für einen Produktkatalog kann z.B. die Produkthierarchie abgebildet werden, wie sie für die Kunden in einem Katalog dargestellt werden soll. Diese Hierarchiestruktur kann genau der evtl. vorhandenen Produkthierarchiestruktur im Klassensystem entsprechen; sie kann aber auch völlig von dieser abweichen. Die Hierarchiestruktur der Werbemittelplanung wird unabhängig vom Klassensystem und der R/3-Produkthierarchie zur Preisfindung verwendet und gepflegt (wird u.a. im Modul SD verwendet). Sie repräsentiert die Strukturierung der Produkte für die

Werbemittel - Präsentation.

Der jeweils untersten Hierarchieebene der *Werbemittel* planung werden die entsprechenden Produkte einzeln zugeordnet. Dabei können die Produkte auch mehrfach an unterschiedliche Knoten zugeordnet werden. Im SAP-Sprachgebrauch wird die abgebildete Hierarchiestruktur als LAYOUT des *Werbemittel* s bezeichnet. Es können mehrere voneinander unabhängige *Werbemittel* angelegt werden, die jeweils zielgruppenorientierte Hierarchien beinhalten.

2. **Vollständige Integration**

Alle Daten, die für die Produktion eines *Werbemittel*s notwendig sind, werden in R/3 abgelegt und gepflegt. Dies verringert die Gefahr von Datenredundanz und erhöht die Aktualität der verwendeten Daten, da auch operative Daten direkt genutzt werden können (Preise etc.). Jeder Hierarchieebene und jedem Produkt können beliebig viele Dokumente aus der SAP Dokumentenverwaltung zugeordnet werden. Außerdem kann zu jeder Hierarchieebene und jedem Produkt je ein Kurz- und ein Langtext eingetragen werden. Die Art der Verwendung dieser Informationen ist analog zur Interpretation der Hierarchiestruktur anwendungsbezogen.

3. **Beliebige Auflagenhöhe**

Ein einmal angelegtes *Werbemittel* kann beliebig oft genutzt werden, so daß sich der Aufwand für eine Neuauflage eines Kataloges etc. auf ein Minimum reduziert.

4. **Landessprachen- und währungsabhängige Varianten**

Es können von jedem *Werbemittel* verschiedene Varianten angelegt werden. Varianten unterscheiden sich durch ihre Landeswährungen (Preise) und Landessprachen (Produktinformationen etc.) voneinander. Es kann z.B. von einem *Werbemittel*, das die deutsche Sprache und Währung verwendet, sofort eine Variante angelegt werden, die die Preise in US-Dollar angibt. Für eine englische Sprachversion müssen natürlich die Produktinformationen und sonstigen Texte bereits mehrsprachig gepflegt worden sein. Ein beliebiges Umschalten und Kombinieren zwischen verschiedenen Landessprachen und Landeswährungen ist durch Anlegen einer neuen Variante jederzeit möglich. Produkthierarchie und Produktzuordnungen gelten für alle Varianten.

5. **Automatische Preisermittlung**

Alle in einem *Werbemittel* angegebenen Preise werden aus dem operativen System ermittelt. Dazu werden in jedem neu angelegten *Werbemittel* alle Informationen zur korrekten Ermittlung der Preise angegeben. Dies sind das Belegschema, das Kundenschema, ein Referenzkunde, eine Vertriebsorganisation, ein Vertriebsweg und eine Produktsparte. Außerdem ist die Angabe eines Stichtages erforderlich, an dem die Preise Gültigkeit haben. Damit stehen alle Möglichkeiten der flexiblen

Preisfestlegung zur Verfügung und die Konsistenz der Preise ist gewährleistet, da die Preisfindung der Komponente *Werbemittel* die gleichen Basisinformationen verwendet wie die anderen R/3 – Module (z.B. SD).

3. Die ABAP/4 - Workbench

In diesem Kapitel werden allgemein die programmiertechnischen Möglichkeiten und Verfahren von SAP R/3 dargestellt. Dabei steht nicht die exakte Beschreibung der Programmiersprache ABAP/4 im Vordergrund. Vielmehr geht es um die Darstellung und Bewertung der Werkzeuge und daraus resultierend um deren Nutzwert für den hier zu bearbeitenden Anwendungsfall Produktkatalog. Es soll ein Eindruck über die erreichbaren Ergebnisse und den Aufwand bzw. Weg für diese Ergebnisse gegeben werden.

Die R/3 Online – Dokumentation beinhaltet eine gut lesbare und vollständige ABAP/4 – Sprachreferenz. Daher hat der Autor hier auf die Aufzählung einzelner ABAP/4 – Befehle verzichtet. In der Praxis trat eher das Problem auf, eine bereits gefundene Information später wiederzufinden. Die Masse und Struktur (Windows – Help => Hyperlinks) der R/3 Online – Dokumentation macht es sehr schwer, sich gezielt durch ein Thema durchzuarbeiten, da ständig Verzweigungen zu anderen Themen bestehen. Der Autor verweist in diesem Kapitel daher auf die Stellen der R/3 – Online – Dokumentation, die die hier angesprochenen Themen weiter vertiefen.

1.
2. **Aufruf**

Die ABAP/4 - Workbench ist die Entwicklungsumgebung von R/3. In ihr sind die Entwicklungs- und Administrationswerkzeuge zur Entwicklung und Pflege von ABAP/4 - Programmen integriert. Die ABAP/4 - Workbench kann im Hauptmenü unter WERKZEUGE / ABAP/4 - WORKBENCH aufgerufen werden.

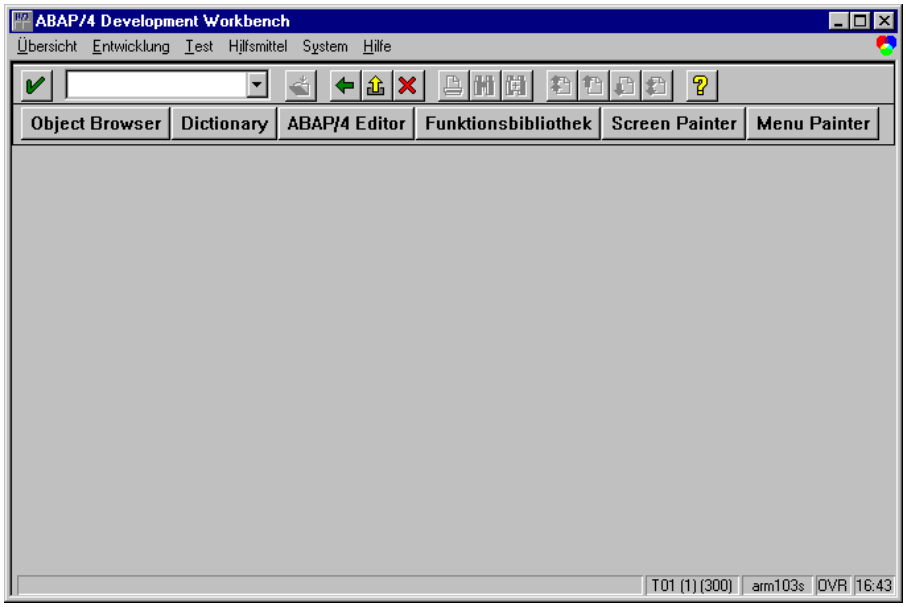


Bild: ABAP/4 Workbench

Dokumentation:

R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench.

3. ABAP/4

ABAP/4 ist die Programmiersprache von SAP R/3. ABAP/4 ist eine Interpreter-Programmiersprache, die an die speziellen Bedürfnisse von betriebswirtschaftlicher Software, insbesondere SAP R/3, angepaßt wurde. Dabei haben sowohl die SAP-typischen Anwendungsoberflächen, als auch die Orientierung an relationalen Datenbanken ihre Spuren in ABAP/4 hinterlassen. Die Programmstruktur und –syntax von ABAP/4 sind stark an COBOL angelehnt, was die Verständlichkeit von fremden Quelltexten erleichtert, da keine kryptischen Abkürzungen und sogenannte *Tricks* erlaubt sind (wie z.B. in C / C++). Der Datenzugriff erfolgt in einer SQL-ähnlichen Abfragesprache, die um anwendungsbezogene Funktionalitäten ergänzt wurde. Bis auf einen C-basierten Systemkern wurde R/3 selbst komplett in

ABAP/4 entwickelt. Dies ermöglicht eine weitgehend betriebssystemunabhängige Entwicklung innerhalb der R/3 - Umgebung. Als sehr überraschend empfand der Autor die Tatsache, daß alle Quellcodes der R/3 – Standard – Dialoge frei einsehbar sind, als ob sie eigene Programme wären. Die dadurch gegebene Möglichkeit, sich die Implementierung interessanter Funktionalitäten anzuschauen, sollte in der Praxis nicht unterschätzt werden. Man stelle sich so etwas bei MS-Windows vor. Eine Änderung dieser Original – Programme ist natürlich unzulässig, jedoch spricht nichts dagegen, sich eine Kopie anzulegen und diese zu verändern.

Dokumentation:

R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench | ABAP/4 - Benutzerhandbuch.

Die Dokumentation beinhaltet eine vollständige ABAP/4 – Sprachreferenz mit vielen kleinen Code – Beispielen. Der Autor hat festgestellt, daß die Vollständigkeit der ABAP/4 – Dokumentation die Nutzung weiterer Informationsquellen (ABAP/4 Bücher etc.) überflüssig macht. Diese waren oft eine Abschrift der Original – Dokumentationen. Eine Ausnahme war das Buch ABAP/4 vom Addison – Wesley - Verlag. Inhaltlich hat es sich zwar auch nicht besonders von der Originaldokumentation unterschieden, aber die Informationen waren gut aufbereitet und schnell zu finden. Als Nachschlagewerk ist es zu empfehlen.

4. Data - Dictionary

Das R/3 Data - Dictionary ist eine zentrale Datenbank, die analog zum Data - Dictionary einer relationalen Datenbank Metainformationen enthält. Metainformationen sind Beschreibungen der Informationsstruktur von (Daten-) Objekten. Die Beschreibung der Struktur eines Materialstammsatzes mit Feldnamen, Feldlängen und Datentypen ist z.B. eine Meta - Information, die im R/3 Data - Dictionary abgelegt ist. Diese Metainformationen werden in ABAP/4 - Programmen verwendet, um z.B. auf die Datensätze einer Tabelle zuzugreifen. Der Zugriff auf die Daten einer Tabelle und die eigentliche Tabellenpflege erfolgt grundsätzlich indirekt über das R/3 – Data - Dictionary und nicht direkt über die zugrundeliegende relationale Datenbank
(z.B. via ODBC). Dies garantiert datenbankunabhängige Programme und bewahrt die Konsistenz der Datenstrukturen (Ausnahmen bestätigen jedoch

auch hier die Regel).

Dokumentation:

R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench | ABAP/4 - Dictionary.

Transparente Tabellen

Eine *transparente Tabelle* ist im SAP-Sprachgebrauch eine *normale* physikalische Datenbanktabelle.

Pool - Tabellen und *Cluster - Tabellen* wurden von SAP eingeführt, um die Ressourcen der relationalen Datenbank, auf der R/3 aufbaut, nicht unnötig stark zu beanspruchen. Sie werden überwiegend für interne Zwecke verwendet. Für den *normalen* ABAP/4 – Entwickler sind sie nicht relevant. Sie werden in dieser Arbeit erwähnt, da die Begriffe recht häufig in der R/3 – Online - Dokumentation (Thema: Data - Dictionary) auftauchen. Mit der steigenden Leistungsfähigkeit der verfügbaren relationalen Datenbanken wird ihr Einsatz sicher stetig seltener werden.

5.
6. **Pool - Tabellen**

Ein Tabellenpool (Pool) dient zur Zusammenfassung mehrerer logischer Tabellen im ABAP/4 Data - Dictionary. Ein Tabellenpool ist eine physikalische Tabelle auf der Datenbank, in der alle Sätze der zugeordneten Pooltabellen abgelegt werden. Die Sätze der einem Pool zugeordneten Pooltabellen werden also in einer gemeinsamen Tabelle auf der Datenbank abgelegt. Die Definition eines Pools besteht aus mindestens zwei Schlüsselfeldern und einem langen Argumentfeld (VARDATA). Das erste Schlüsselfeld enthält den Tabellennamen der Pooltabelle, das zweite Schlüsselfeld (VARKEY) die Primärschlüsselfelder dieser Tabelle. Das lange Argumentfeld (VARDATA) enthält die übrigen Daten eines Datensatzes, d.h. die Daten in Feldern, die nicht zum Schlüssel gehören. Vor diesem langen Argumentfeld steht ein Feld, in das von der Datenbankschnittstelle die tatsächliche Länge des in das VARDATA - Feld geschriebenen Satzes eingetragen wird. Der Tabellenname und das Feld VARKEY bilden gemeinsam den Schlüssel des Pools.

Ein Tabellen - Pool besitzt also folgenden Aufbau:

TABNAME CHAR(10) Tabellenname

VARKEY CHAR(n) maximale Schlüssellänge n =< 110

DATALN INT2(5) Länge des zurückgelieferten VARDATA - Satzes

VARDATA RAW(m) maximale Länge des Datenteils abhängig vom DBMS

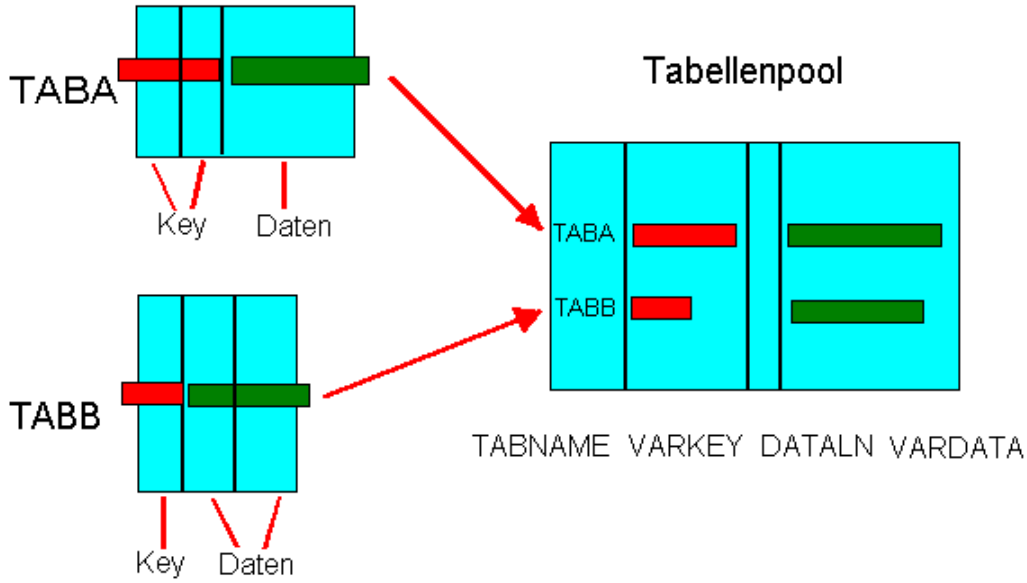


Bild: Speicherung von Pool - Tabellen

7.

8. **Cluster - Tabellen**

Ein Tabellencluster dient, wie die Pool - Tabellen, zur Zusammenfassung mehrerer logischer Tabellen im ABAP/4 Data - Dictionary. Bei diesem Tabellentyp werden mehrere logische Zeilen verschiedener Clustertabellen in einem physischen Satz zusammengefaßt. Die Sätze der einem Cluster zugeordneten Clustertabellen werden also in einer gemeinsamen Tabelle auf der Datenbank abgelegt. Ein Cluster enthält einen transparenten Clusterschlüssel, der als Anfangsstück in den Schlüsseln aller in den Cluster eingehenden logischen Cluster - Tabellen enthalten sein muß. Weiterhin enthält ein Cluster ein langes Feld (VARDATA), das die Daten der Cluster - Tabellen zu diesem Schlüssel enthält. Falls die Daten nicht in ein Feld passen, werden Fortsetzungssätze angelegt.

Aufbau eines Clusters:

CLKEY1 CHAR(*) Erstes Schlüsselfeld

CLKEY2. CHAR(*) Zweites Schlüsselfeld

....

....

CLKEYN CHAR (*) n-tes Schlüsselfeld

PAGENO INT2(5) Nummer der Folgeseite

TIMESTMP CHAR(14) Zeitstempel

PAGELG INT2(5) Länge des zurückgelieferten VARDATA - Satzes

VARDATA RAW(*) maximale Länge des Datenteils, abhängig vom

unterliegenden Datenbanksystem

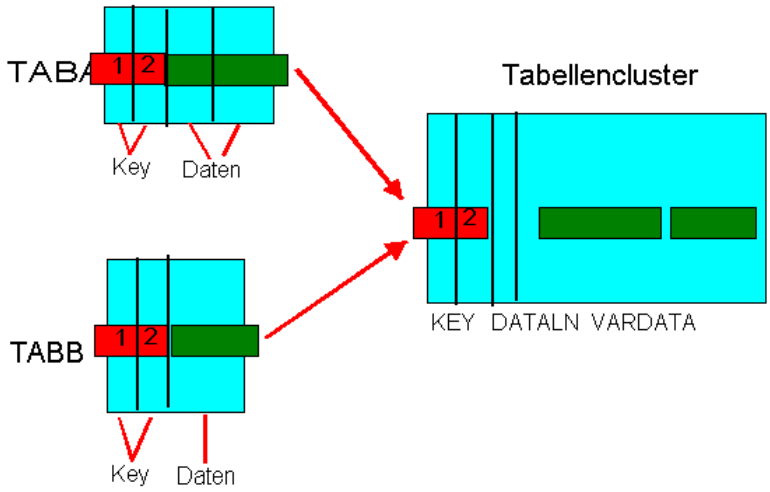


Bild: Speicherung von Cluster - Tabellen

Dokumentation *Pool-/Cluster-Tabellen*:

R/3 - Bibliothek | Basis | ABAP/4 - Development - Workbench | ABAP/4 Dictionary

Stichwort: Pool => Grundkonzepte

9.
10. Landessprachen - Unabhängigkeit

Um eine landessprachenunabhängige Programmierung zu erreichen, sollten (dürfen ?) selbstgeschriebene ABAP/4 - Programme keine konstanten Meldungstexte, die der Benutzer zu sehen bekommt, enthalten. R/3 bietet hier die NACHRICHTENKLASSEN an. In einer Nachrichtenklasse werden alle Meldungstexte 3-stelligen Kürzeln zugeordnet. Diese Kürzel werden in einem Programm als Ausgabeinformationen (wie eine Variable) verwendet. Das System setzt in der Meldung automatisch den Meldungstext in der Landessprache ein, der beim R/3 - Login vom Benutzer gewählt wurde. Natürlich müssen die verfügbaren Landessprachen in der Nachrichtenklasse entsprechend gepflegt werden.

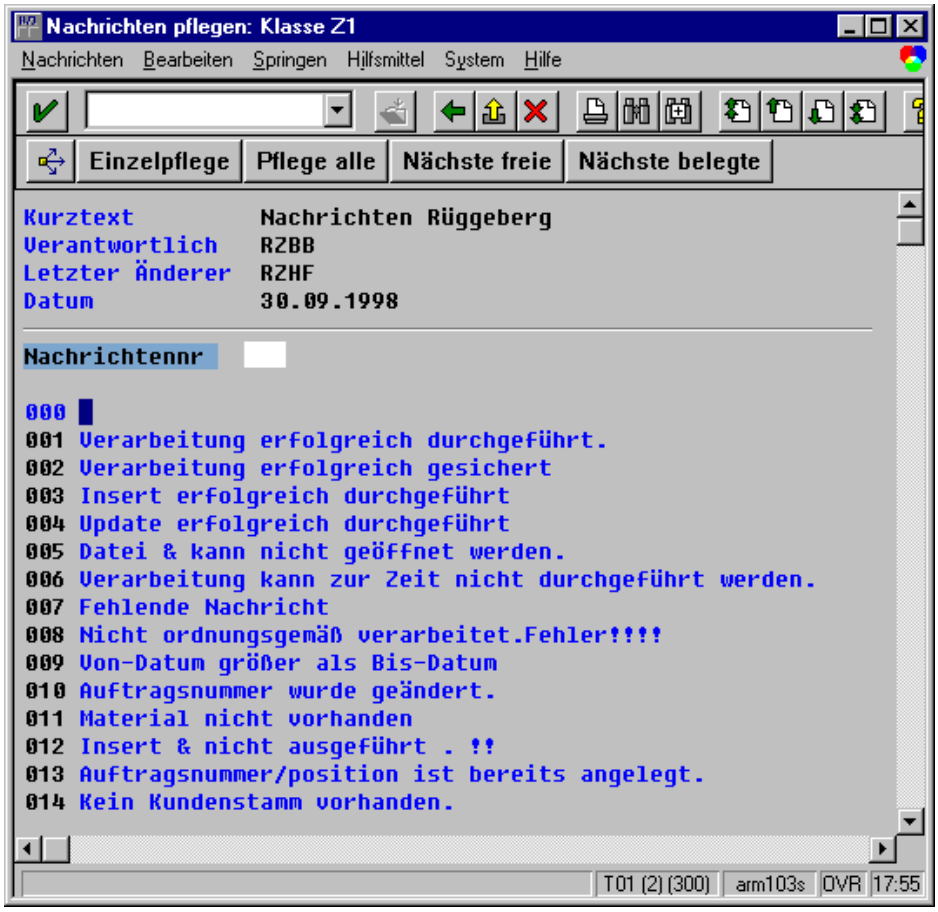


Bild: Nachrichtenklassen

11.
12. **Grundobjekte der ABAP/4 - Workbench**

- **Domänen**
  Eine Domäne beschreibt die technischen Eigenschaften eines Tabellenfeldes, wie Datentyp, Wertebereich, Feldlänge etc. Bei Anlegen von Tabellen kann auf Domänen verwiesen werden, um dessen Eigenschaften auf bestimmte Tabellenfelder zu übertragen.

- **Datenelemente**
  Ein Datenelement beschreibt die semantischen Eigenschaften eines Tabellenfeldes, wie Ausgabeformat, Bezeichnungstexte etc.

  Domänen und Datenelemente werden unabhängig voneinander und unabhängig von bestimmten Tabellen gepflegt. Dies ermöglicht ihre mehrfache Verwendung, was den Pflegeaufwand und die Redundanz verringert.

- **Strukturen**
  Strukturen entsprechen in ihrem Aufbau Tabellen. Der einzige Unterschied ist, daß aus ihnen keine Datenbanktabellen generiert werden, d.h. in Strukturen können keine Daten gespeichert werden. Strukturen sind damit reine Metainformationen, die in ABAP/4 – Programmen Variablen zugewiesen werden können (wie Datentypen).

- **Tabellen**
  Tabellen sind das zentrale Element in R/3 und ABAP/4. In ihnen werden alle Daten des Systems gespeichert. Auch die Parametrisierung des Systems geschieht durch die entsprechende Anpassung von Tabellenwerten.

- **Fremdschlüssel**
  Fremdschlüssel bilden Beziehungen zwischen den Tabellenfeldern verschiedener Tabellen

ab.

1.
2. **Tabelle anlegen**

Am Beispiel *TABELLE ANLEGEN* wird der Umgang mit dem Data - Dictionary demonstriert.

1. In der ABAP/4 - Workbench wird das Data - Dictionary mit ENTWICKLUNG / DICTIONARY aufgerufen. Nach dem Eintragen des Tabellennamens, wird mit ANLEGEN eine neue Tabelle angelegt.

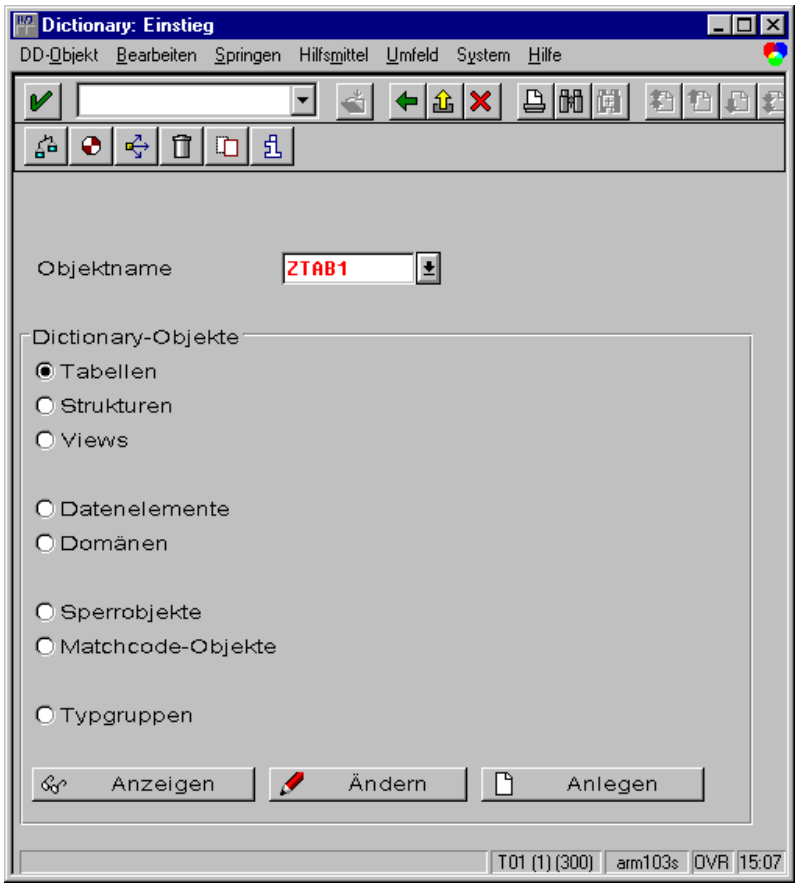


Bild: R/3 - Data - Dictionary - Einstiegsbild

2.
3. Im Fenster *TABELLE / STRUKTUR: FELDER ÄNDERN* können alle relevanten Tabelleneigenschaften eingetragen werden. Mit TABELLE / SICHERN OHNE PRÜFEN werden die Eingaben gespeichert. Mit TABELLE / AKTIVIEREN werden die eingetragenen Tabelleneigenschaften geprüft und anschließend erst wird die Tabelle physikalisch in der Datenbank generiert.

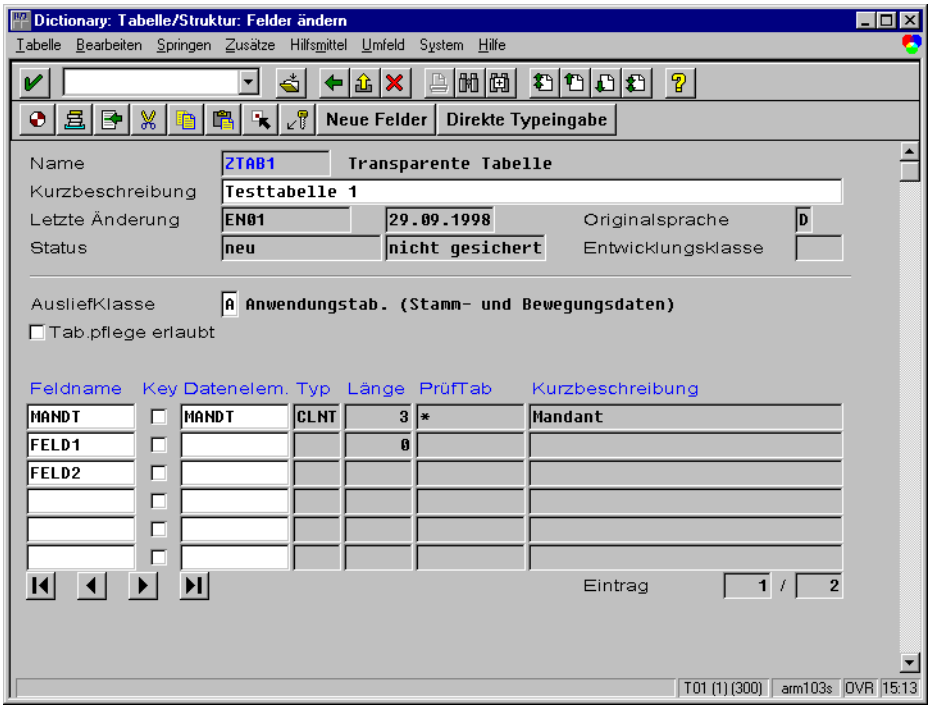


Bild: R/3 - Tabellenpflege

1. Aggregierte Objekte

Aggregierte Objekte fassen mehrere Tabellen zu einem logischen Objekt zusammen. Dies erleichtert die einheitliche Behandlung mehrerer logisch zusammengehörender Tabellen.

- **Views**
  Eine View ist eine logische Sicht auf eine oder mehrere Tabellen, ähnlich den Views einer relationalen Datenbank.

- **Matchcodes**
  Matchcodes unterstützen den Anwender bei der Suche nach bestimmten Datensätzen während der Eingabe in Eingabefelder. Wenn z.B. eine Kundennummer eingetragen werden soll, kann über einen Matchcode in einer Liste nach einem bestimmten Kundennamen gesucht werden. Die Bestätigung eines bestimmten Kunden fügt automatisch die gesuchte Kundennummer in das Eingabefeld ein. Ein Matchcode besteht aus einer oder mehreren Tabellen und Views, die bei Bedarf durchsucht werden können. Funktionell entsprechen Matchcodes ungefähr den Comboboxen der Windows-Oberfläche, wo auch anhand einer vorgegebenen mehrspaltigen Liste ein bestimmter Wert aus einer Liste ausgewählt werden kann und nur ein Spaltenwert in der Eingabezeile angezeigt wird. Ein evtl. verfügbarer Matchcode wird dem Benutzer durch einen kleinen Pfeil rechts neben der Eingabezeile angezeigt (siehe Bildschirmfoto in Kapitel 2 *Selektionsbild* für die *Materialtabelle*). Mit einem Mausklick erscheint ein modales Fenster, in dem ein Eintrag ausgewählt werden kann. Beim Schließen des Fensters wird der Rückgabewert (Kundennummer) des Matchcodes automatisch in eine Eingabezeile übertragen. Ab der Version 4 können auch mehrere Eingabezeilen gleichzeitig beim Schließen des Matchcode - Fensters mit Daten gefüllt werden.

- **Sperrobjekte**
  Wenn ein Benutzer Daten in R/3 bearbeitet, ändert er zwangsläufig Tabelleninhalte. Die betroffenen Tabellen bzw. Datensätze müssen für den Bearbeitungszeitraum vor der parallelen Bearbeitung durch andere Benutzer gesperrt werden. Ein Sperrobjekt faßt mehrere Tabellen zu einem logischen Objekt zusammen. Die Aktivierung eines Sperrobjektes sperrt automatisch alle Tabellen des Sperrobjektes. Sperrobjekte unterliegen

der Kontrolle des ABAP/4 - Programms und sind dadurch besser steuerbar als die automatisch gesetzten Datenbanksperren eines relationalen Datenbanksystems.

1.

2. Transaktionen

Der Begriff Transaktion ist in R/3 gleich mehrfach belegt.

- **Datenbanktransaktion (LUW, logical unit of work).**

  Mehrere Datenbankoperationen (Tabellenänderungen) werden zu einer Datenbanktransaktionen zusammengefaßt. Entweder eine Datenbanktransaktion wird korrekt abgeschlossen (commit) oder alle Änderungen innerhalb der Datenbanktransaktion werden komplett rückgängig gemacht (rollback). Eine LUW ist die niedrigste Form der Transaktionssteuerung in R/3. Sie entspricht der Transaktionssteuerung der relationalen Datenbank, auf der R/3 aufbaut. Jeder Bildwechsel innerhalb eines Dynpro - Programms löst ein LUW – Commit aus.

- **Änderungstransaktion (SAP-LUW)**

  Mehrere Teilaufgaben einer logisch betriebswirtschaftlichen Aufgabe werden zu einer SAP - LUW zusammengefaßt. Analog zu Datenbanktransaktionen (LUW) gilt auch hier das Alles - oder Nichts - Prinzip. Eine Buchung wird entweder vollständig durchgeführt oder vollständig verworfen. Eine SAP - LUW kann mehrere Datenbank - LUW's enthalten.

- **ABAP/4 Transaktion (SAP-Transaktion).**

  Mehrere SAP-LUW's werden zu einer SAP-Transaktion zusammengefaßt. Eine ABAP/4 - Transaktion kann mit einem Transaktionscode versehen und vom Anwender direkt aufgerufen werden. Dem Anwender erscheint eine ABAP/4 Transaktion als Programm, das eine bestimmte Aufgabe erfüllt, z.B. die Pflege der Materialstammdaten ermöglicht.

  SAP-Transaktionen können wiederum in folgende Gruppen unterteilt werden:

- Dialogtransaktionen
  Dialogtransaktionen bestehen aus einer oder mehreren Bildschirmmasken und ermöglichen eine interaktive Benutzerführung mit dem Anwender. Dialogtransaktionen werden für die Datenpflege und -bearbeitung verwendet. Die meisten SAP-Transaktionen sind als Dialogtransaktionen realisiert.

- Reporttransaktionen
  Reporttransaktionen erzeugen parametergesteuerte Auswertungen, die in Listenform ausgegeben werden. Die Ausgabe einer Liste mit allen Kunden eines Postleitzahlengebietes ist z.B. ein Report mit einem Parameter *Postleitzahl*. Ursprünglich waren Reports nur ausgabeorientiert; inzwischen wurden sie jedoch um interaktive Funktionalitäten erweitert, so daß auch in einem Report Benutzereingaben verarbeitet werden können (Mausklick etc.).

- Parametertransaktionen
  Parametertransaktionen sind Dialogtransaktionen, die von einem anderen ABAP/4 - Programm aufgerufen werden können und dabei bestimmte Eingaben automatisch als Parameter erhalten. Dies ermöglicht die Nutzung von Dialogtransaktionen innerhalb anderer Dialogtransaktionen. Es können Dialog - Programmbausteine erstellt werden, die wiederverwendet werden können.

- Variantentransaktionen
  Variantentransaktionen sind Dialogtransaktionen, die auf einer bestehenden Dialogtransaktionen aufbauen, aber andere Vorgabewerte für die Eingabezeilen haben oder bestimmte Eingabefelder ausblenden. Variantentransaktionen stellen also eine Variante einer Dialogtransaktion dar.

1.
2. **Funktionsbausteine**

   Ein Funktionsbaustein ist ein ABAP/4 - Programmbaustein, der über genau definierte Schnittstellen verfügt und eine definierte Aufgabe löst. Da R/3 in ABAP/4 entwickelt wurde, existieren Zehntausende von Funktionsbausteinen im R/3 - System, die die unterschiedlichsten Aufgaben lösen. Viele dieser Funktionsbausteine werden nur intern von den Standard-Dialogtransaktionen genutzt. Es existieren jedoch auch sehr viele Funktionsbausteine, die allgemeine Funktionalitäten anbieten, die das Leben eines ABAP/4 - Entwicklers sehr vereinfachen können. Im Prinzip können die Funktionsbausteine als eine Art R/3 - API angesehen werden, deren Kenntnis und Nutzung die eigentliche Aufgabe eines ABAP/4 - Entwicklers ist. Während die Programmiersprache ABAP/4 für einen erfahrenen Entwickler in wenigen Wochen schon Routine sein dürfte, ist die Einarbeitung in die R/3 - Funktionsbausteine ein wesentlich aufwendigerer und entsprechend längerfristiger Prozeß. In der Praxis ist die Kenntnis der für eine bestimmte Aufgabenstellung relevanten Funktionsbausteine das eigentliche Know-how eines ABAP/4 - Entwicklers. Der Autor machte bei der Erkundung des R/3 - Systems die Erfahrung, daß für fast jeden erdenklichen Fall bereits Funktionsbausteine im System existieren. Die hohe Zahl an vorhandenen Funktionsbausteinen ist entgegen des ersten Eindrucks kein praktisches Hemmnis. Durch gezielte Stichwortsuche können die verfügbaren Funktionsbausteine rasch auf eine überschaubare Anzahl verringert werden.

   Das eigentliche Problem ist, daß die Kenntnis der Existenz eines bestimmten Funktionsbausteins nicht auch den sofortigen Einsatz desselben bedeuten muß. Viele Funktionsbausteine, die auch für eigene Einsatzzwecke interessant sind, wurden von der SAP AG für ihre internen Zwecke entwickelt. Entsprechend ist die Dokumentation dieser Funktionsbausteine nicht besonders üppig ausgefallen. Neben der Kurzbeschreibung der Funktion eines Funktionsbausteins (in der Regel ein Satz) kann der ABAP/4 - Entwickler nur versuchen über die Namensgebung der Funktionsbaustein - Parameter und den Kontext des Funktionsbausteins die Verwendbarkeit eines Funktionsbausteins zu erfahren. Letztendlich bleibt auch in R/3 oft nur die try and error Methode, um das genaue Verhalten eines Funktionsbausteins zu ermitteln. SAP bietet auf seiner Homepage ein kostenloses Tool zum Download an, das den isolierten Aufruf von Funktionsbausteinen innerhalb einer Testumgebung ermöglicht, um das Verhalten derselben zu ermitteln.

   Auch bei ordentlich dokumentierten Funktionsbausteinen (die ebenfalls in R/3 vorkommen) ist für den routinierten Einsatz von Funktionsbausteinen eine intensivere Beschäftigung mit dem R/3 - System Voraussetzung, da immer wieder Standarddomänen und -strukturen aus

   dem Data - Dictionary als Parametertypen auftauchen, deren Kenntnis für den praktischen Einsatz eines Funktionsbausteins unerläßlich ist.

   Für den Autor interessant war die Tatsache, daß die Quellcodes der SAP – Funktionsbausteine sehr viele Variablenbezeichner und Kommentare in

deutscher Sprache beinhalten. In der SAP – Dokumentation wird zwar darauf hingewiesen, daß bei der Entwicklung eigener Funktionsbausteine die Schnittstellenparameter englische (oder besser US - amerikanische) Bezeichner bekommen sollten, jedoch wird in der Implementation der SAP Funktionsbausteine immer wieder *auf deutsch* programmiert. Dem Autor erschienen die Quellen dadurch lesbarer, auch wenn es sehr ungewohnt war. Seit dem weltweiten Erfolg von R/3 wird dies bei den Weiterentwicklungen von R/3 sicher stark nachlassen.

Über ENTWICKLUNG / FUNKTIONSBIBLIOTHEK erscheint das Fenster zur Pflege der Funktionsbausteine. Mit ANLEGEN kann ein neuer Funktionsbaustein angelegt werden.

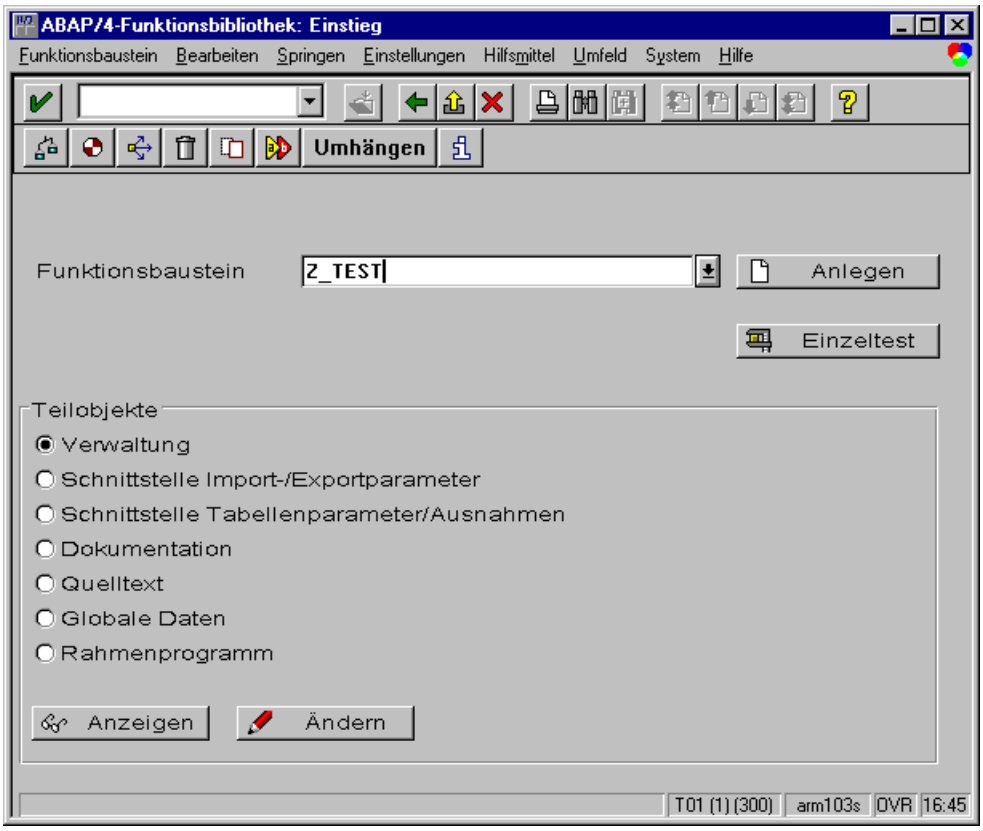


Bild: Funktionsbaustein anlegen

Nach Eingabe der Funktionsbibliothek erscheinen folgende Bildschirmmasken:

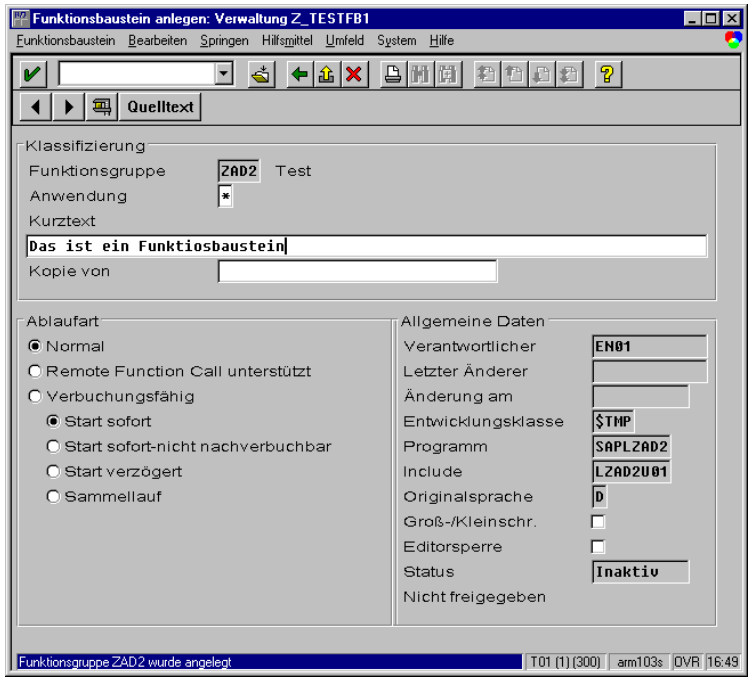


Bild: Basisdaten eines Funktionsbausteins

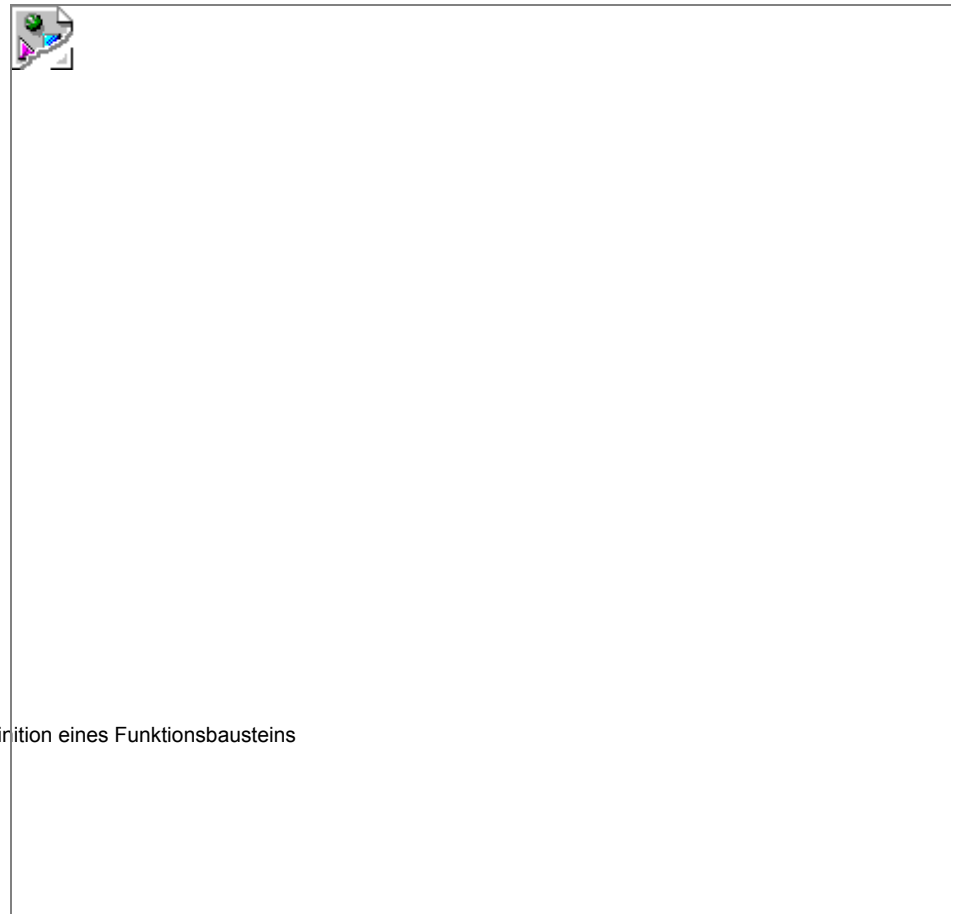


Bild: Schnittstellendefinition eines Funktionsbausteins

Bild: Implementation eines Funktionsbausteins

3.

4. Funktionsbibliotheken

Eine Funktionsbibliothek faßt logisch zusammengehörende Funktionsbausteine zusammen. Dabei dient eine Funktionsbibliothek nicht nur als Hilfsmittel zur besseren und einfacheren Verwaltung der Funktionsbausteine im R/3 - System, sondern ist auch ein ABAP/4 - Programmierobjekt. Eine Funktionsbibliothek kann eigene Variablen enthalten, die nur für die Funktionsbausteine in dieser Funktionsbibliothek sichtbar sind. Es kann also (über globale Variablen) ein Datenaustausch zwischen Funktionsbausteinen einer Funktionsbibliothek stattfinden, ohne Funktionsparameter zu verwenden. Diese in der Programmierpraxis sehr erleichternde Möglichkeit kann sich jedoch bei unachtsamer Verwendung negativ für die Verständlichkeit von Funktionsbausteinen auswirken, da das Verhalten eines Funktionsbaustein nicht mehr nur von seinen Parametern abhängt, sondern auch von globalen Variablen beeinflußt werden kann, deren Behandlung von beliebigen anderen Funktionsbausteinen der Funktionsbibliothek möglich ist.

Beim Anlegen eines Funktionsbausteins muß immer eine Funktionsbibliothek angegeben werden (4-stellige Bezeichnung). Eine Funktionsbibliothek wird beim Anlegen eines Funktionsbausteins automatisch mit angelegt, wenn sie noch nicht existiert.

Ein sehr praktisches Beispiel für eine allgemein verwendbare Funktionsbibliothek, die dem ABAP/4 - Entwickler sehr viel Programmierarbeit ersparen kann, ist die Funktionsbibliothek SEUT. Sie stellt Funktionsbausteine zur Behandlung und Darstellung von Hierarchiebäumen in DYNPRO's zur Verfügung.

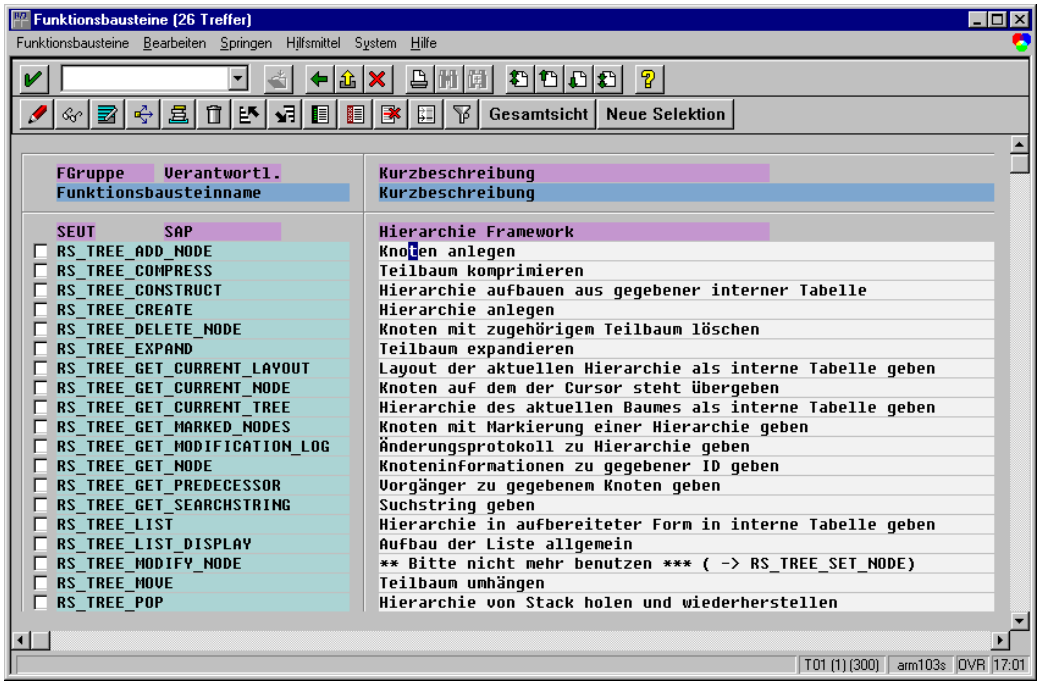


Bild: Funktionsbibliothek SEUT

5.

6. Dynpros

1. Was ist ein Dynpro ?

Der Begriff DYNPRO ist ein historisches Kunstgebilde der SAP und kommt von DYNamisches PROgramm. Als dynamisches Programm wurde in den Anfängen der SAP AG ein Programm bezeichnet, das dynamisch auf Benutzereingaben reagieren konnte und entsprechende Folgebilder auf dem Bildschirm zeigte. Dynamische Programme sind also Programme, die interaktiv mit dem Benutzer kommunizieren.

In der heutigen Version von R/3, wo allgemein Windows - Oberflächen Standard sind, ist ein Dynpro technisch gesehen kein eigenständiges Programm, sondern eine Komponente eines Programms, nämlich ein Bildschirmfenster (bzw. Bildschirmmaske oder WINDOW) mit der zugehörigen Programmlogik. Ein R/3 - Programm (ABAP/4) kann keine, eines oder beliebig viele Dynpros (Bildschirmmasken) haben. Ein ABAP/4 - Programm wiederum wird von einer Transaktion aufgerufen, der das aufzurufende Programm mit dem Start - Dynpro einmal zugewiesen wird. Eine Transaktion kann direkt vom Anwender durch Eingabe einer 4-stelligen Transaktionskennung (z.B. ZAR5) aufgerufen werden.

Mit ÜBERSICHT / OBJECT BROWSER gelangt man in das zentrale Tools zur Verwaltung und Erstellung von ABAP/4 - Programmen. Es können direkt einzelne Objekte wie Programme oder Tabellen durch Eingabe ihres Namens angezeigt und bearbeitet werden. Sollte ein eingegebenes Objekt (z.B. ein Programm) nicht existieren, wird es automatisch angelegt. Mit einem Doppelklick auf *Objektarten Programm* kann ein neues Dynpro angelegt werden.

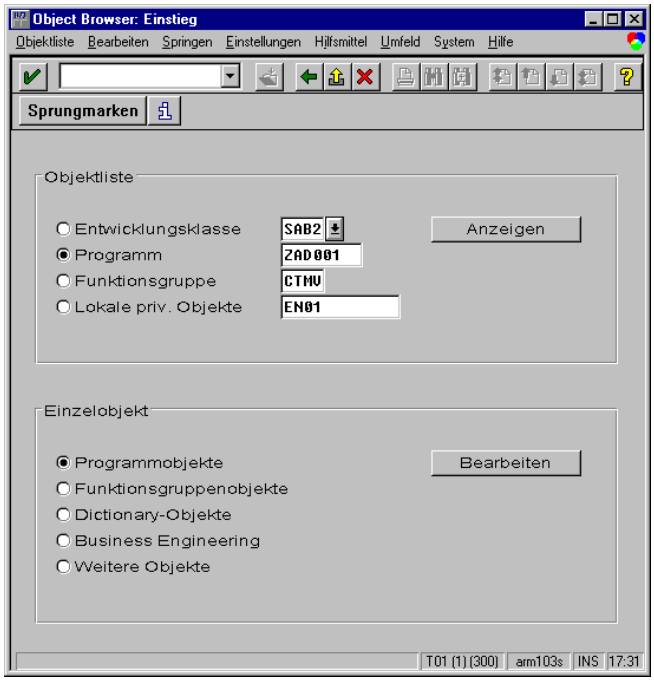


Bild: R/3 Objekt Browser

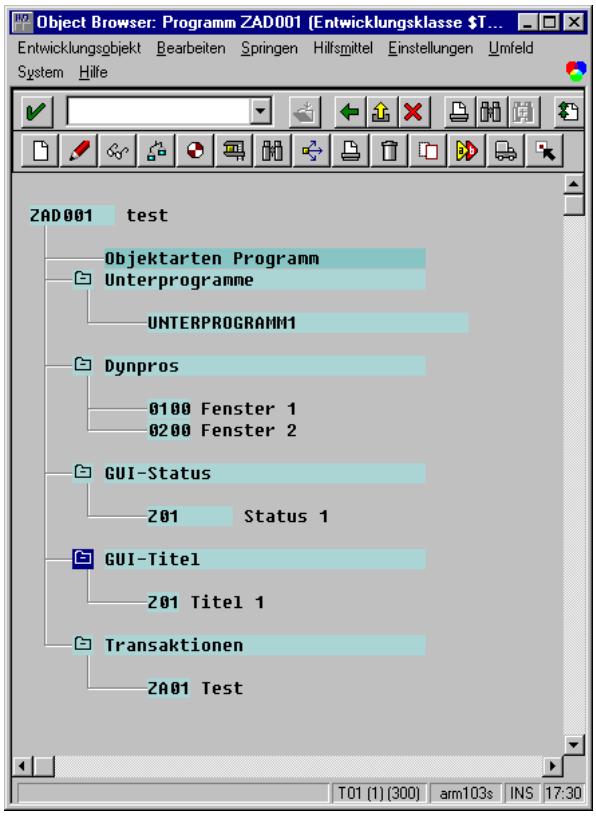


Bild: Programm ZAD001 mit einigen Programmkomponenten

Maskenlayout Designen: Der Screen-Painter

Ein DYNPRO kann grundsätzlich in die Bildschirmmaske und die Verarbeitungslogik untergliedert werden. Die Bildschirmmaske kann in R/3 inzwischen mit der Maus via Drag & Drop gezeichnet werden. Das Werkzeug dazu ist der SCREEN - PAINTER. Im Screen - Painter können die bekannten Bildschirmelemente wie Buttons, Eingabezeilen, Textelemente, Listenelemente, Checkboxes und Radiobuttons positioniert werden. Mit einem Doppelklick auf das entsprechende Dynpro erscheint folgendes Editierfenster, auf das später noch eingegangen wird:

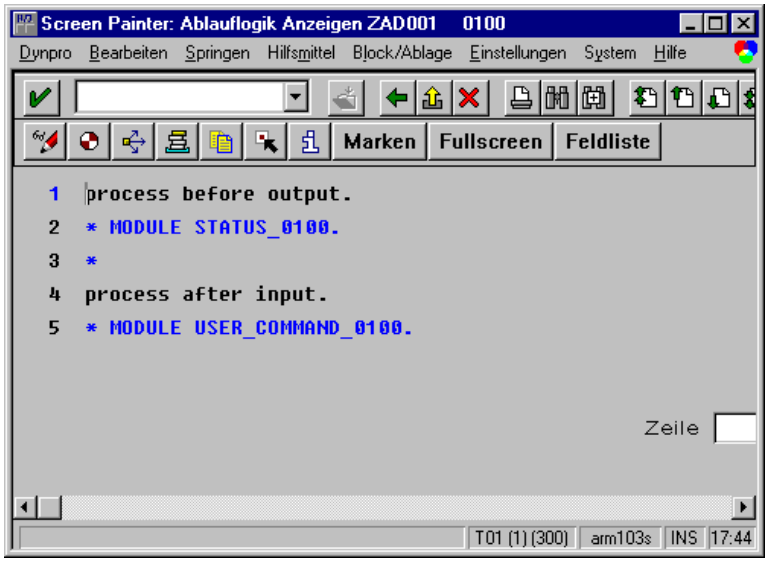


Bild: Vorgabe - Verarbeitungslogik eines Dynpros

Mit SPRINGEN / FULLSCREEN EDITOR erscheint der Screen - Painter auf dem Bildschirm.

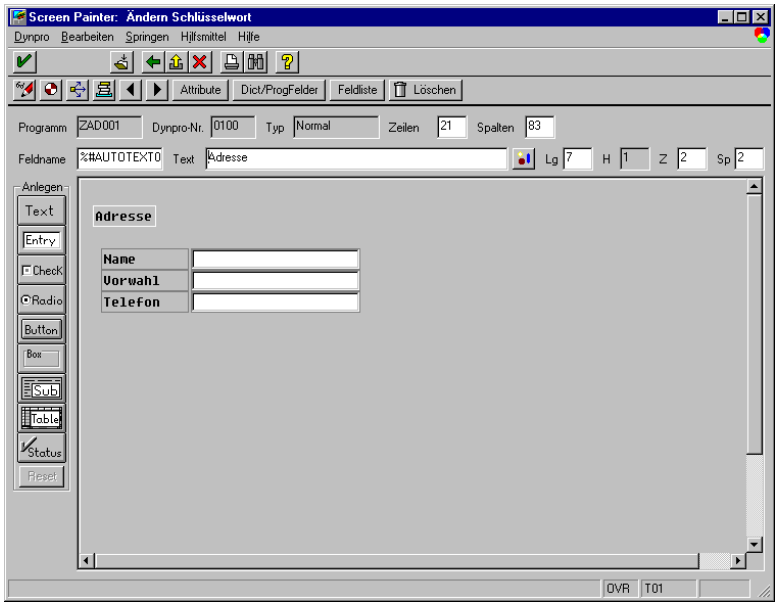


Bild: Der Screen - Painter

Die Verknüpfung der Bildschirmelemente mit den Daten eines Programms erfolgt recht komfortabel über den Namen einer Variablen, den eine Bildschirmelement repräsentiert. Eine Eingabezeile ist innerhalb eines ABAP/4 - Programms kein eigenständiges Objekt, das gesondert behandelt wird (Daten aus Eingabezeile lesen / in Eingabezeile schreiben), sondern es ist direkt mit einer Programmvariable oder einem Tabellenfeld verknüpft. Die Eingaben eines Anwenders stehen damit sofort in einer Variablen zur Verfügung und jede Änderung der verknüpften Variablen in einem ABAP/4 - Programm wird auf dem Bildschirm in der entsprechenden Eingabezeile angezeigt.

2. Datenfelder anordnen (Datenbank + Programmfelder)

Mit SPRINGEN / DICT. PROGRAMMFELDER erscheint das Tool zum

Einfügen eines neuen Feldes. Nach Eingabe des Feldnamens, dies kann ein Variablenname im aktuellen Programm oder ein Tabellenname im Data - Dictionary sein, erscheinen die gefunden Felder in einer Liste, von wo sie mit einem Doppelklick in die Bildschirmmaske übertragen und positioniert werden können.

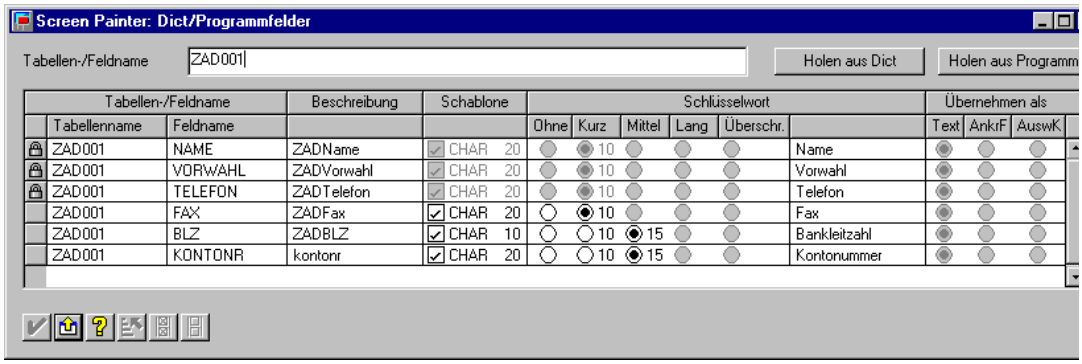


Bild: Screen - Painter => Felddefinition

Listendarstellung

Die Darstellung von Listen (Tabellen) in einem Dynpro ist nicht so unkompliziert gelöst, wie die Darstellung von einzelnen Feldern. R/3 bietet zwei verschiedene Möglichkeiten zur Listendarstellung an:

1. **Step-Loop**

Ein Step-Loop gruppiert vorhandene Bildschirmelemente wie z.B. Eingabezeilen und wiederholt sie mehrfach vertikal, so daß eine Liste entsteht. Die Datenübergabe ist bei einem Step-Loop programmtechnisch relativ aufwendig. Step-Loops haben keine elementaren Funktionalitäten wie das seitenweise Blättern oder das horizontale Scrollen implementiert, so daß diese Funktionalitäten bei der Programmierung berücksichtigt werden müssen. Da Step-Loops lange Zeit die einzige Möglichkeit zur Listendarstellung in Dynpros waren, sind sie noch sehr häufig in den R/3 – Standardtransaktionen im Einsatz. Außerdem müssen Step-Loops in Dynpros verwendet werden, die als Vorlage für ein R/3 – Internet - Template dienen sollen (im Zusammenhang mit dem ITS von R/3). An dieser Stelle sei auf die Arbeit von Herrn Cuti verwiesen, die genau diese Zusammenhänge genauer behandelt. Unter *R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench | ABAP/4 – Benutzerhandbuch* können in der R/3 – Dokumentation unter dem Stichwort *Steploop* Beispielcodes und detailliertere Informationen zur Programmierung eingesehen werden.

Im folgenden Bildschirmfoto wurden drei Eingabezeilen nebeneinander angeordnet und mit der Maus selektiert. Anschließend wurden sie über BEARBEITEN / STEPLOOP / DEFINIEREN zu einer STEPLOOP gruppiert. Die Eingabezeilen lassen sich nun nicht mehr einzeln bearbeiten (nur noch spaltenweise). Mit BEARBEITEN / STEPLOOP / AUFLÖSEN läßt sich der Vorgang wieder aufheben.

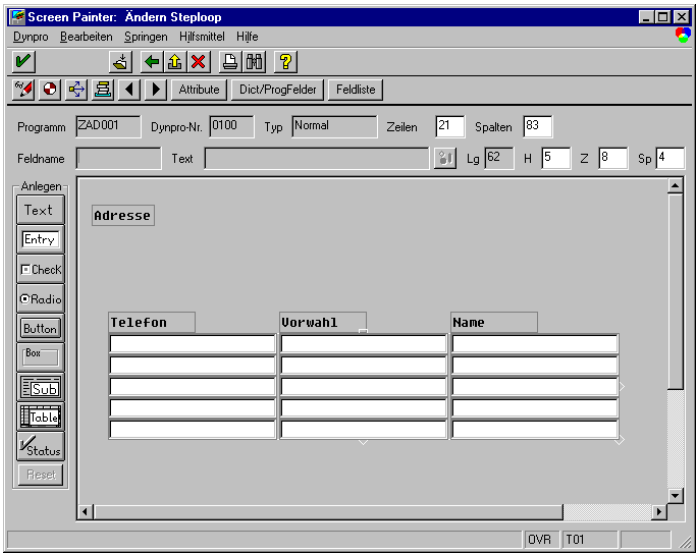


Bild: Listendarstellung mit Step-Loop

Table-Control

Ein Table-Control ist die moderne Form der Listendarstellung in Dynpros. Ein Table-Control ist ein eigenständiges Bildschirmelement (wie eine Eingabezeile), das schon elementare Funktionalitäten zur Listenbehandlung wie das seitenweise Blättern oder das horizontale Scrollen implementiert hat. Grundsätzlich gibt es keinen Grund, in eigenen Dynpros nicht ausschließlich das Table-Control für die Listendarstellung zu verwenden.

In Internet - fähigen Dynpros ist dessen Einsatz jedoch nicht möglich; da lediglich leere Felder im Internet-Browser erscheinen. Über ein entsprechendes (noch zu entwickelndes) Java–Applet ließe sich ein Table-Control theoretisch auch in einen Browser implementieren.

Während eine Eingabezeile mit einer Variablen einfachen Datentyps (String, Number, ...) verknüpft ist, wird ein Table-Control mit einer Variable vom Typ `Tableview`, einer Struktur aus dem Data - Dictionary, verknüpft.

Beispiel:

```
TABLES: SFLIGHT.

CONTROLS: FLIGHTS TYPE TABLEVIEW USING SCREEN 200.
```

3.
4. **Anwendereingaben verarbeiten**

ABAP/4 ist eine ereignisgesteuerte Programmiersprache, d.h. es werden Programmroutinen mit bestimmten Systemereignissen verknüpft und automatisch aufgerufen. Diese Systemereignisse werden normalerweise automatisch ausgelöst, ohne das der ABAP/4 - Entwickler einen Einfluß darauf hat. Es gibt einige Dutzend verschiedene Ereignisse, auf die in ABAP/4 - Programmen reagiert werden kann. Im Zusammenhang mit Dynpros gibt es jedoch nur zwei Ereignisse, die wirklich relevant sind und die gesamte logische Funktionalität eines Dynpros implementieren. Diese beiden Ereignisse heißen

PBO für *Process Before Output* und PAI für *Process After Input*.

Das Ereignis PBO wird vom System ausgelöst, bevor ein Dynpro auf dem Bildschirm angezeigt wird. Es dient damit zur Aufbereitung und Initialisierung von Programmvariablen und Bildschirmfeldern. In einem PBO - Ereignis können sehr komplexe SQL-Aufrufe stattfinden; es ist also nicht auf die simple Vorbelegung von Variablen beschränkt.

Das Ereignis PAI wird automatisch bei jedem Funktionsaufruf des Benutzers ausgelöst. Als Funktionsaufruf gelten in erster Linie das Betätigen eines Buttons, das Aufrufen eines Menüpunktes und der Aufruf eines Shortcuts bzw. einer Funktionstaste über die Tastatur. In bestimmten Situationen löst auch ein Doppelklick mit der Maus ein Ereignis aus. Jeder Funktionsaufruf ist durch ein 4-stelliges Kürzel gekennzeichnet, welches in PAI aus einer globalen Variablen abgefragt werden kann. Entsprechend dieses Kürzels können beliebige weitere Unterprogramme und Funktionsbausteine aufgerufen werden.

Die Ereignisse PBO und PAI werden iterativ ausgelöst, d.h. nachdem das Ereignis PAI ausgelöst und die Funktion ausgeführt wurde, wird danach wieder automatisch das Ereignis PBO ausgelöst, so daß wieder der Programmcode in PBO ausgeführt wird. Es ist z.B. möglich, das beim ersten Anzeigen eines Dynpros ein anderer ABAP/4 - Programmcode ausgeführt wir, als bei den folgenden, indem von dem ABAP/4 – Programm nach dem ersten PAI ein globales Flag (globale Programmvariable) entsprechend gesetzt wird, das in PBO immer abgefragt wird.

Als Vorgabelogik vergibt das System je einen Modulaufruf für PBO und PAI. Ein Modul ist ein Unterprogramm, das die Verarbeitungslogik eines Dynpros beinhaltet. Die beiden Module STATUS_0100 und USER_COMMAND_0100 existieren noch nicht und können mit einem Doppelklick auf den Modulnamen (im Editor) automatisch angelegt werden.

Bild: PBO und PAI eines Dynpros

7.
8. Reports

1. Was ist ein Report ?

Ein Report ist ein Bericht, der Daten aus Tabellen liest, auswertet und auf den Bildschirm oder den Drucker ausgibt. Ein Report wird wie ein Dynpro in ABAP/4 programmiert. Der Hauptunterschied zu Dynpros ist, daß keine Bildschirmmasken mit dem Screen - Painter erstellt werden, über die der Benutzer mit dem System kommuniziert, sondern daß ein Report die Informationen über direkte Ausgabebefehle (*Write*) auf den Bildschirm bzw. den Drucker ausgibt. Reports sind historisch gesehen die Vorgänger der Dynpros und ABAP/4 war zu Anfangszeiten der SAP AG eine reine Berichtsgenerator - Sprache, die die einfachere Erstellung von Auswertungen und Berichten ermöglichen sollte. Wenn ein Benutzer einen Report einmal aufgerufen hat, erscheint zuerst ein SELEKTIONSBILD, in dem der Benutzer Parameter für den Report eintragen kann. Dies könnte z.B. eine Kundennummer sein, um die offenen Posten eines Kunden anzuzeigen. Nach dem SELEKTIONSBILD läuft der Report ohne weiteren Benutzereingriff ab und zeigt das Ergebnis auf dem Bildschirm an. Daraus folgt, daß ein Report grundsätzlich nicht interaktiv ist, d.h. nicht auf Benutzereingaben reagieren kann. Jede interaktive Kommunikation war den Dynpros vorbehalten. Die Eigenschaft, daß ein Report nicht interaktiv ist, wurde durch die Anforderungen der Benutzer im Laufe der R/3 - Versionen immer weiter aufgeweicht. Inzwischen ist es in R/3 möglich, auch in einem Report auf Benutzereingaben zu reagieren; so können z.B. mit einem Doppelklick auf eine Zeile in einem Report detailliertere Informationen zu dieser Zeile angezeigt werden.

Selektionsbilder sind ein zentraler Bestandteil der Reportprogrammierung. Sie stellen die Hauptschnittstelle zum Benutzer dar, über den der Ablauf des Reports beeinflußt werden kann. Bei komplexen Reports mit sehr vielen Selektionsfeldern (Parametern) und / oder bei sehr häufig verwendeten Reports mit gleichen Parameterbewertungen können *Reportvarianten* angelegt werden, die einen bestimmten Report mit vorbelegten Selektionsfeldern
(Parametern) repräsentieren. Bei Verwendung von Reportvarianten muß der Anwender nicht mehr einzeln jedes Selektionsfeld bewerten, wenn er einen Report aufruft.

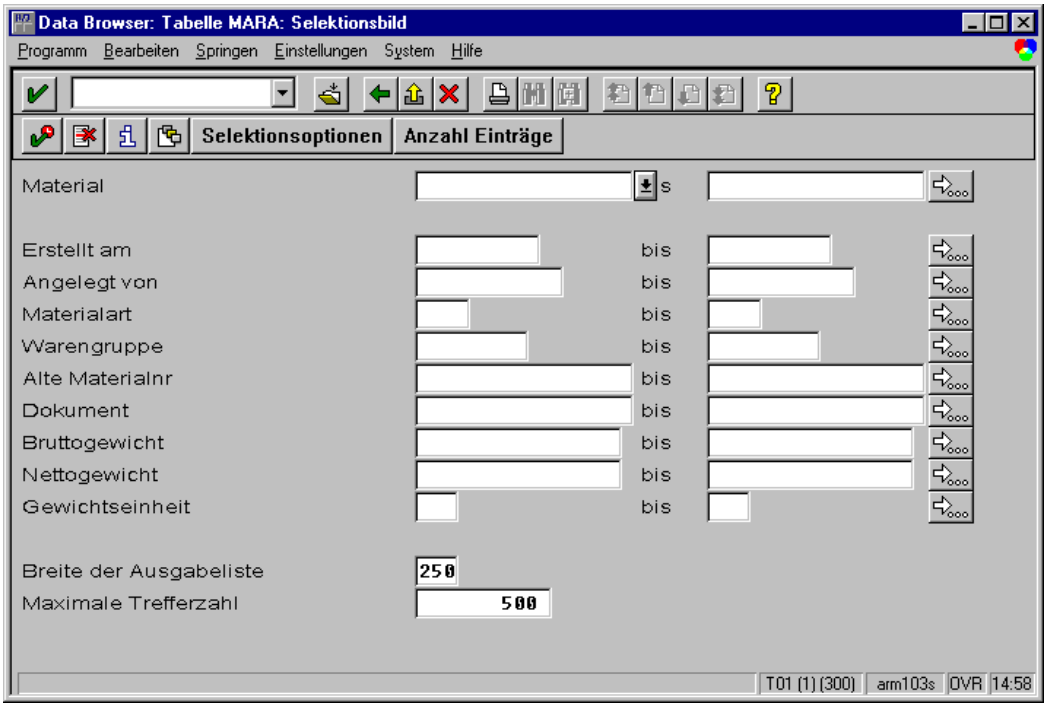


Bild: Selektionsbild für die Materialtabelle

Bild: Die Materialtabelle als Report

2.

3. Logische Datenbanken

Eine logische Datenbank ist ein R/3 - Verfahren, um allgemeine Datenbeziehungen zentral zu implementieren und mehrfach wiederzuverwenden. So muß z.B. die Master / Detail - Relation zwischen den Tabellen Kunde / Bestellung in ABAP/4 durch zwei verschachtelte SQL-Abfragen implementiert werden. Diese Implementierung in Form von zwei verschachtelten Schleifen muß in jedem Report, der auf diese Beziehung zurückgreift, erneut erfolgen. Neben dem erhöhten Aufwand der

wiederholten Programmierung, ist als Nachteil dieses Vorgehens der erhöhte Wartungsaufwand und die höhere Fehlerquote hervorzuheben. Insbesondere bei komplexen Datenbeziehungen ist die mehrfache Implementierung und Wartung von solchen ABAP/4 / SQL-Abfragen fast zwangsläufig mit Fehlern behaftet.

R/3 bietet in Form einer logischen Datenbank die Möglichkeit, diese speziellen ABAP/4 / SQL-Abfragen in einem eigenen Programm, einer logischen Datenbank, abzulegen. Beim Anlegen eines neuen Reports kann angegeben werden, ob dieser Report auf einer logischen Datenbank aufbauen soll oder nicht. Wenn eine logische Datenbank angegeben wird, werden die Daten, die der Report verarbeiten und ausgeben soll nicht mehr ausschließlich in dem Report über eigene SQL-Anweisungen gelesen, sondern sie werden über entsprechende Ereignisse von der logischen Datenbank an den Report übergeben.

Der Vorteil dieses Verfahrens ist, daß beliebig viele Reports auf eine logische Datenbank zugreifen können und Änderungen der logischen Datenbank sofort für alle diese Reports gültig werden. Die eigentliche Abfrage der Daten durch verschachtelte ABAP/4 / SQL-Aufrufe bleibt auch bei der Verwendung einer logischen Datenbank erhalten, jedoch ist sie zentral abgelegt.

Eine logische Datenbank kann auch eigene Selektionsbilder haben, wenn diese für alle Reports, die auf diese Datenbank zugreifen, relevant sind. Damit können auch die Selektionsbilder, die nach dem Aufruf eines Reports angezeigt werden, zentral mit der logischen Datenbank angelegt und gepflegt werden. Optional kann ein Report Selektionsbilder von logischen Datenbanken unterdrücken bzw. um eigene Parameter ergänzen.

Mit ENTWICKLUNG / PROGRAMMIERUMFELD / LOGISCHE DATENBANKEN wird das Werkzeug zur Erstellung einer logischen Datenbank aufgerufen. Eine logische Datenbank wird mit einem 3-stelligen Kürzel bezeichnet.

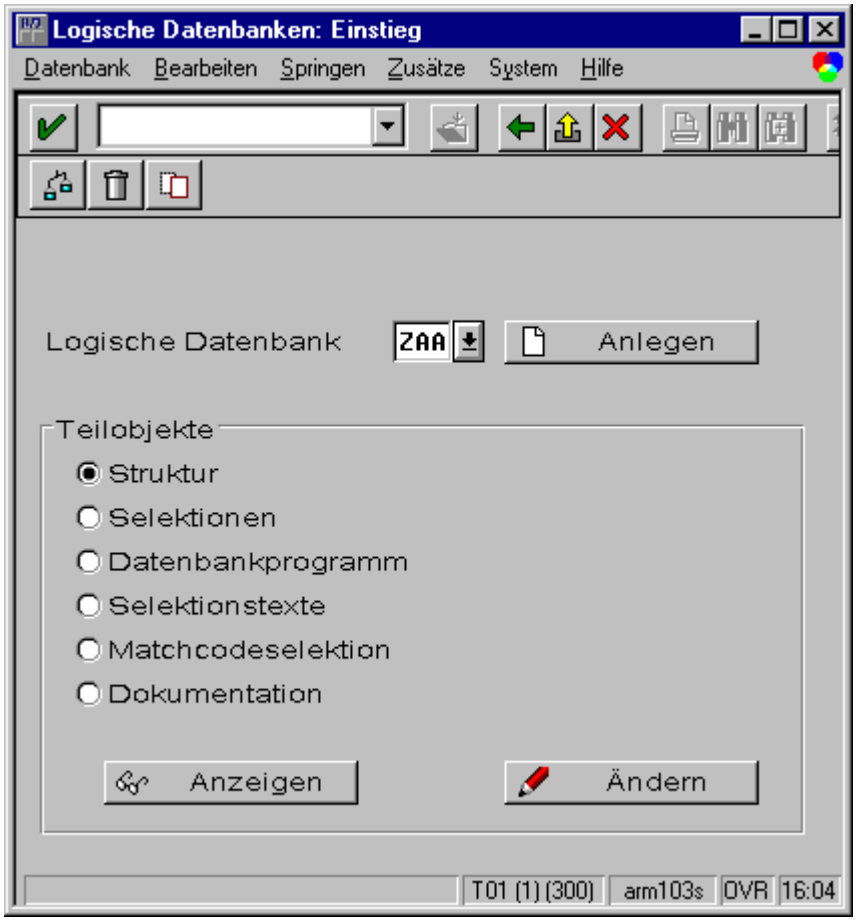


Bild: Einstieg logische Datenbanken

1.
2. **Relationen einer logischen Datenbank**

   Die relationalen Beziehungen der Tabellen einer logischen Datenbank werden übersichtlich in einem Hierarchiebaum angezeigt und können komfortabel editiert werden. Diese Beziehungen werden nicht physikalisch im Data - Dictionary angelegt, sondern dienen lediglich als Schnittstelle für Reports, die auf diese logische Datenbank zugreifen (Metainformationen). Fremdschlüssel, Indizes und sonstige Zugriffsmechanismen müssen grundsätzlich manuell im Data - Dictionary erstellt werden.

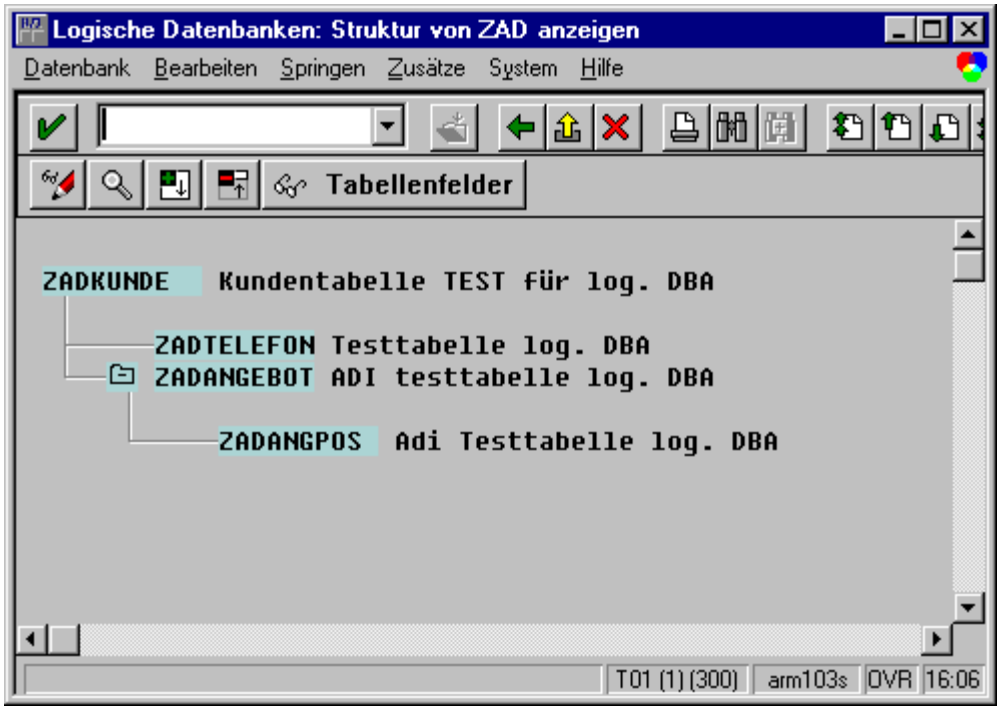


Bild: Relationen einer logischen Datenbank

Nach dem Anlegen einer logischen Datenbank wird automatisch ABAP/4 - Quellcode erzeugt, der in Abhängigkeit von den verwendeten Tabellen, Tabellenfeldern und Relationen ein Standard - Selektionsbild und Standard - Abfragen generiert. Diese Programme können manuell beliebig an eigene Anforderungen angepaßt werden. Sie ersparen einem in jedem Fall den Aufbau eines Programmgerüstes. Die Vorgaben sind mit einem * am Zeilenbeginn als Kommentar deklariert und damit deaktiviert. Durch Löschen des * wird der Code aktiviert.

3.
4. **Das Selektionsbild**

Ein Selektionsbild kann sehr komplexe Ausmaße annehmen. So ist z.B. neben der Angabe von einfachen Parameterwerten wie einer Kundennummer, die Angabe von Intervallen und Ausschlußkriterien möglich. Über sogenannte Matchcodes können auf dem Selektionsbild auch Nachschlagelisten für die Suche nach einzelnen Parametern integriert werden (ähnlich einer Combobox). Die detaillierte Dokumentation dieser Möglichkeiten würde den Rahmen dieser Arbeit sprengen, so verweist der Autor auf die gute Online - Dokumentation von R/3 zu diesem Thema. Es seien kurz die elementaren Befehle erwähnt:

- Eingabe eines Intervalls für ein bestimmtes Tabellenfeld. Die Variable `KUNDEID` ist eine Struktur und beinhaltet Anfangs- und Endwert für das Tabellenfeld `KUNDE-ID` (Tabelle `KUNDE`, Feld `ID`)

```
SELECT-OPTIONS KUNDEID FOR KUNDE-ID.
```

- Eingabe eines einzelnen Parameterwertes für das Tabellenfeld KUNDE-ID.

```
PARAMETERS KUNDEID LIKE KUNDE-ID FOR TABLE KUNDE.
```

Das Beispiel auf der nächsten Seite zeigt den automatisch generierten Code für ein Selektionsbild zu obiger logischen Datenbank (Bild: *Relationen einer logischen Datenbank*). Die einzige

aktivierte Programmzeile ist fettgedruckt. Sie generiert bei Aufruf eines Reports, der auf diese Datenbank zugreift, ein Selektionsbild, das wie folgt aussieht:

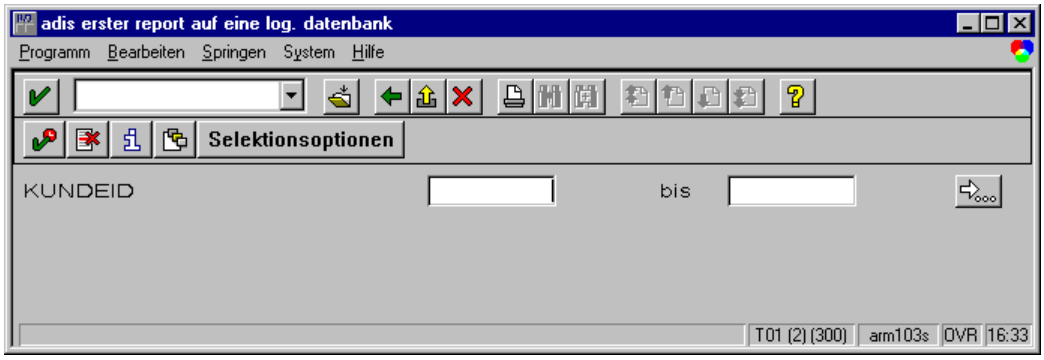


Bild: Ein Selektionsbild für einen R/3 - Report

Der Button neben der rechten Eingabezeile erscheint automatisch und öffnet ein weiteres allgemeines R/3 - Fenster für die detaillierte Angabe von Intervallen.

Codebeispiel: Automatisch generiertes Selektionsbild

```
*---------------------------------------------------------------------*
* INCLUDE DBZADSEL
*  It  will  be  automatically  included  into  the  database  program.
*---------------------------------------------------------------------*
*
* If the source code is automatically generated,
* please perform the following steps:
*   1.   Replace   ?   by   suitable   names   (at   most   8   characters).
*    2.    Activate    SELECT-OPTIONS    and    PARAMTERS    (delete    stars).
* 3. Save source code.
* 4. Edit database program
*
*   Hint:   Syntax-Check   is   not   possible   within   this   Include!
*  It  will  be  checked  during  syntax-check  of  database  program.
*
*---------------------------------------------------------------------*
SELECT-OPTIONS: KUNDEID FOR ZADKUNDE-ID.
```

```
*   Parameter   for   matchcode   selection   (DD-structure   MCPARAMS):
*   PARAMETERS   p_mc   AS   MATCHCODE   STRUCTURE   FOR   TABLE   ZADKUNDE.


* SELECT-OPTIONS: FOR ZADTELEFON-ID.
* SELECT-OPTIONS: ? FOR ZADANGEBOT-ID.
* SELECT-OPTIONS: ? FOR ZADANGPOS-ID.
```

1.
2. **Das Datenbankprogramm**

Das Datenbankprogramm, das automatisch generiert wird und die relationalen Beziehungen als ABAP/4 – Programmcode (verschachtelte Select - Abfragen) implementiert, folgt weiter unten. Auch hier fällt auf, daß grundsätzlich alle Zeilen als Kommentar deaktiviert sind und vom ABAP/4 - Programmierer aktiviert werden müssen. Dies unterstreicht die Tatsache, daß der automatisch generierte Code lediglich ein Gerüst sein soll, das die ersten Schritte erleichtert. Die eigentliche Programmlogik muß komplett manuell integriert werden. Für ein Verständnis des Ablaufs in einem ABAP/4 - Programm für eine logische Datenbank muß der ABAP/4 - Entwickler das Konzept des ereignisorientierten bzw. ereignisgesteuerten Programmablaufs sicher beherrschen. Der Code ist in einzelne EVENT - Bereiche geteilt, die automatisch vom System aufgerufen werden. Die Aufrufreihenfolge wird von den Relationen des Hierarchiebaumes der logischen Datenbank beeinflußt.

Codebeispiel: Automatisch generiertes Datenbankprogramm:

```
*---------------------------------------------------------------------*
* DATABASE PROGRAM OF LOGICAL DATABASE ZAD
*---------------------------------------------------------------------*
*
* The automatically generated subroutines (FORMs) are called by
* system routines. Therefore their names must not be changed!!!
*
* If the source code is automatically generated,
* please perform the following steps:
* 1. Replace ? by suitable ABAP statements.
* 2. Activate ABAP statements (delete stars).
* 3. Save source code.
* 4. Check syntax of database program.
* SELECT-OPTIONS and PARAMETERS will be checked automatically.
*---------------------------------------------------------------------*
*---------------------------------------------------------------------*
```

```
* Performance notes
*---------------------------------------------------------------------*
* General information about the use of logical databases is
* contained
* in the extended help information of transaction SE36.
* Please consider in particular the following aspects:
* 1. Use of internal tables:
* SELECT * FROM table INTO TABLE i_table WHERE ... .
* LOOP AT i_table.
* MOVE-CORRESPONDING table TO i_table.
* PUT table.
* ENDLOOP.
* 2. Use of OPEN/FETCH CURSOR for nested structures.
*   3.   Use   of   dynamic   selections   to   enable   further   selection   criteria
* (cf. documentation of SELECTION-SCREEN DYNAMIC SELECTIONS).
* 4. Authority checks already at PAI of selection screen.
*---------------------------------------------------------------------*
PROGRAM SAPDBZAD DEFINING DATABASE ZAD.
TABLES: ZADKUNDE,
ZADTELEFON,
ZADANGEBOT,
ZADANGPOS.
************************************************************************
*   !!!   PLEASE   DO   NOT   CHANGE   MANUALLY   (BEGIN   OF   BLOCK)   !!!!!!!!!!!!   *
*---------------------------------------------------------------------*
* Data structures for matchcode selection *
*   !!!   PLEASE   DO   NOT   CHANGE   MANUALLY   (END   OF   BLOCK)   !!!!!!!!!!!!!!   *
************************************************************************
*---------------------------------------------------------------------*
* BEFORE_EVENT will be called before event EVENT
* Possible values for EVENT: 'START-OF-SELECTION'
*---------------------------------------------------------------------*
* FORM BEFORE_EVENT USING EVENT.
* CASE EVENT.
* WHEN 'START-OF-SELECTION'
*
```

```
* ENDCASE.
* ENDFORM. BEFORE_EVENT
*---------------------------------------------------------------------*
* AFTER_EVENT will be called after event EVENT
* Possible values for EVENT: 'END-OF-SELECTION'
*---------------------------------------------------------------------*
* FORM AFTER_EVENT USING EVENT.
* CASE EVENT.
* WHEN 'END-OF-SELECTION'
*
* ENDCASE.
* ENDFORM. AFTER_EVENT
*---------------------------------------------------------------------*
* Initialize selection screen (processed before PBO)
*---------------------------------------------------------------------*
FORM INIT.
ENDFORM. INIT.
*---------------------------------------------------------------------*
* PBO of selection screen (processed always after ENTER)
*---------------------------------------------------------------------*
FORM PBO.
ENDFORM. PBO.
*---------------------------------------------------------------------*
* PAI of selection screen (processed always after ENTER)
*---------------------------------------------------------------------*
FORM PAI USING FNAME MARK.
* CASE FNAME.
* WHEN 'KUNDEID '.
* WHEN '*'.
* ENDCASE.
ENDFORM. PAI
*---------------------------------------------------------------------*
* Call event GET ZADKUNDE
*---------------------------------------------------------------------*
FORM PUT_ZADKUNDE.
SELECT * FROM ZADKUNDE
```

```
WHERE ID IN KUNDEID.
PUT ZADKUNDE.
ENDSELECT.
ENDFORM. PUT_ZADKUNDE
*---------------------------------------------------------------------*
* Call event GET ZADTELEFON
*---------------------------------------------------------------------*
FORM PUT_ZADTELEFON.
SELECT * FROM ZADTELEFON
WHERE IDKUNDE = ZADKUNDE-ID.
PUT ZADTELEFON.
ENDSELECT.
ENDFORM. PUT_ZADTELEFON
*---------------------------------------------------------------------*
* Call event GET ZADANGEBOT
*---------------------------------------------------------------------*
FORM PUT_ZADANGEBOT.
SELECT * FROM ZADANGEBOT
WHERE IDKUNDE = ZADKUNDE-ID.
PUT ZADANGEBOT.
ENDSELECT.
ENDFORM. PUT_ZADANGEBOT
*---------------------------------------------------------------------*
* Call event GET ZADANGPOS
*---------------------------------------------------------------------*
FORM PUT_ZADANGPOS.
SELECT * FROM ZADANGPOS
WHERE IDANGEBOT = ZADANGEBOT-ID.
PUT ZADANGPOS.
ENDSELECT.
ENDFORM. PUT_ZADANGPOS
*---------------------------------------------------------------------*
* Authority Check for table ZADKUNDE
*---------------------------------------------------------------------*
* FORM AUTHORITYCHECK_ZADKUNDE.
* AUTHORITY-CHECK ...
```

```
* ENDFORM. AUTHORITYCHECK_ZADKUNDE
*---------------------------------------------------------------------*
* Authority Check for table ZADTELEFON
*---------------------------------------------------------------------*
* FORM AUTHORITYCHECK_ZADTELEFON.
* AUTHORITY-CHECK ...
* ENDFORM. AUTHORITYCHECK_ZADTELEFON
*---------------------------------------------------------------------*
* Authority Check for table ZADANGEBOT
*---------------------------------------------------------------------*
* FORM AUTHORITYCHECK_ZADANGEBOT.
* AUTHORITY-CHECK ...
* ENDFORM. AUTHORITYCHECK_ZADANGEBOT
*---------------------------------------------------------------------*
* Authority Check for table ZADANGPOS
*---------------------------------------------------------------------*
* FORM AUTHORITYCHECK_ZADANGPOS.
* AUTHORITY-CHECK ...
* ENDFORM. AUTHORITYCHECK_ZADANGPOS
*---------------------------------------------------------------------*
* PUT_ZAD_MATCHCODE.
* Processed when matchcode selection is used,
* i.e. user input into PARAMETERS p_mc AS MATCHCODE STRUCTURE.
*---------------------------------------------------------------------*
* FORM PUT_ZAD_MATCHCODE.
* ENDFORM. PUT_ZAD_MATCHCODE
```

1.

2. Ein einfacher Report

   Um einen neuen Report zu erstellen, müssen folgende Schritte durchgeführt werden:

   Im Object - Browser wird im Feld PROGRAMM der Name des neuen Reports eingetragen und ANZEIGEN gewählt. Da der Report nicht existiert, wird er automatisch angelegt. In den einzutragenden Basisdaten eines ABAP/4 - Programms deklariert der Typ 1 (ONLINEPROGRAMM) einen Report. Bei LOGISCHE DATENBANK erfolgt kein Eintrag, d.h. der Datenzugriff erfolgt im Report selber.

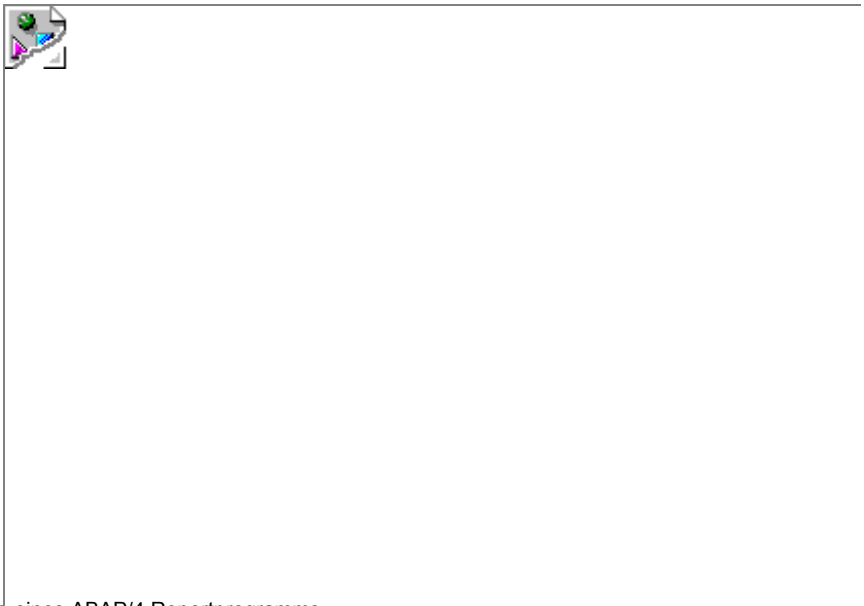


Bild: Programmattribute eines ABAP/4 Reportprogramms

Es erscheint folgender Programmcode.

Bild: Der ABAP/4 Programmeditor

Nun kann mit allen ABAP/4 - Mitteln programmiert werden. Es können SQL-Zugriffe auf Tabellen stattfinden, zusätzliche Daten durch Berechnungen erzeugt werden und auch sonst alle allgemeinen Möglichkeiten der Informationsgenerierung in einer Programmiersprache genutzt werden. Es gibt grundsätzlich keinerlei Beschränkungen (bis auf z.B. Dynpro - spezifische Sprachbestandteile), die durch den Begriff REPORT beim Benutzer evtl. assoziiert werden könnten. Es können beliebige Unterprogramme angelegt und Funktionsbausteine aufgerufen werden. Das Report - Programm verhält sich wie ein linear abgearbeitetes Programm, das in der ersten Programmzeile beginnt und in der letzten beendet wird. Es fällt auf, daß das ereignisorientierte Konzept von ABAP/4 hier *versteckt* wird. Es sind keine Event - Blöcke definiert, die automatisch vom System aufgerufen werden.

Erreicht wird dieses Verhalten durch einen kleinen *Trick* der SAP bei der ABAP/4 - Programmablaufs - Definition. Der Autor sieht folgende Definition zur Aufrufreihenfolge von ABAP/4 – Programmbestandteilen aus der R/3 – Dokumentation als so wichtig an, daß er sie an dieser Stelle zitiert:

Alle Anweisungen zwischen zwei Ereignisschlüsselwörtern oder zwischen einem Ereignisschlüsselwort und einer FORM - Anweisung (siehe Unterprogramme definieren) bilden einen Verarbeitungsblock. Tritt ein Ereignis ein, verarbeitet das System den Verarbeitungsblock hinter dem entsprechenden Ereignisschlüsselwort.

Jede Anweisung in einem ABAP/4-Report ist Bestandteil eines Verarbeitungsblocks oder eines Unterprogramms.

Alle Anweisungen, die nicht nach einem Ereignisschlüsselwort oder einem FORM-ENDFORM-Block aufgeführt werden, sind automatisch Bestandteil des Verarbeitungsblocks, der zum Standard-Zeitereignis START-OF-SELECTION gehört (Näheres über dieses Ereignis finden Sie unter START-OF-SELECTION). Dies hat folgende Konsequenzen:

Wenn Sie Anweisungen zwischen den Anweisungen REPORT oder PROGRAM und dem ersten Ereignisschlüsselwort oder FORM schreiben, werden diese Anweisungen bei START-OF-SELECTION durchgeführt.

Enthält Ihr Report kein START-OF-SELECTION, bilden diese Anweisungen den vollständigen Verarbeitungsblock START-OF-SELECTION. Enthält Ihr Report ein START-OF-SELECTION, werden diese Anweisungen am Anfang des zugehörigen Blocks eingefügt.

Geben Sie in Ihrem Programm überhaupt keine Ereignisschlüsselwörter an, bilden alle Anweisungen, die vor einer eventuell auftretenden FORM - Anweisung liegen, den Verarbeitungsblock START-OF-SELECTION.

Alle Anweisungen zwischen einer ENDFORM-Anweisung und einem Ereignisschlüsselwort oder zwischen einer ENDFORM-Anweisung und dem Ende des Programms bilden einen Verarbeitungsblock, der niemals abgearbeitet wird. Schreiben Sie keine solchen Anweisungen in Ihr Programm. Setzen Sie sämtliche Unterprogramme an das Ende des Programms.

Die Reihenfolge der Verarbeitungsblöcke in einem Programm ist nicht entscheidend. Die Verarbeitungsabfolge wird ausschließlich über Ereignisse ausgelöst. Aus Gründen der Lesbarkeit sollten Sie die Verarbeitungsblöcke aber in der Reihenfolge im Programm aufführen, in der das System sie ausführt.

Im Gegensatz zu operationalen oder Steuerungsanweisungen verarbeitet das System deklarative Anweisungen während der Programmgenerierung, nicht zur Laufzeit (Näheres zu deklarativen Anweisungen finden Sie unter Schlüsselwörter). Sie werden unabhängig von ihrer Position im Programm-Coding verarbeitet und können zu einem beliebigen Verarbeitungsblock gehören. Aus Gründen der Übersicht sollten Sie jedoch alle deklarativen Schlüsselwörter am Anfang Ihres Programms oder Unterprogramms zusammenfassen.

Unter *R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench | ABAP/4 – Benutzerhandbuch* können in der R/3 – Dokumentation unter dem Stichwort *Ereignisschlüsselwörter* Beispielcodes und detailliertere Informationen zur Programmierung eingesehen werden.

3.

4. Direkter Datenbankzugriff

Um z.B. eine Kundenliste mit entsprechenden Telefonnummern auszugeben kann folgender REPORT genutzt werden:

Codebeispiel:

Report zur Anzeige einer Kundenliste mit Telefonnummern.

```
*&---------------------------------------------------------------------*
*& Report ZADREP1 *
*& *
*&---------------------------------------------------------------------*
*& *
*& *
*&---------------------------------------------------------------------*
REPORT ZADREP1 .
TABLES: ZADKUNDE, ZADTELEFON.
DATA: TESTSTR(15) TYPE C.
*parameters kid like zadkunde-id.
SELECT-OPTIONS: KIDS FOR ZADKUNDE-ID.
SELECT * FROM ZADKUNDE WHERE ID IN KIDS.
WRITE / ZADKUNDE-NAME.
* Hier findet der direkte SQL – Zugriff auf die Daten statt
SELECT * FROM ZADTELEFON
WHERE ZADTELEFON-IDKUNDE = ZADKUNDE-ID.
WRITE: / ZADTELEFON-VORWAHL, ZADTELEFON-TELEFON.
ENDSELECT.
NEW-PAGE.
ENDSELECT.
```

5.

6. Datenbankzugriff über logische Datenbank

Die gleiche Funktionalität, die Ausgabe einer Kundenliste mit Telefonnummern, kann natürlich auch mit einer logischen Datenbank realisiert werden. Die logische Datenbank, auf der dieser Report aufbaut, wurde bereits mit ABAP/4 - Programmcode vorgestellt.

Codebeispiel:

Report zur Anzeige einer Kundenliste mit Telefonnummern unter Verwendung einer logischen Datenbank

```
*&---------------------------------------------------------------------*
*& Report ZADREP2 *
*& *
*&---------------------------------------------------------------------*
*& *
```

```
*& *
*&---------------------------------------------------------------------*
REPORT ZADREP2 .

TABLES: ZADKUNDE, ZADTELEFON.

* Hier wird das Ereignis (Event) zum Empfangen der Kundendaten
deklariert.

GET ZADKUNDE.

WRITE: / ZADKUNDE-NAME, ' ', ZADKUNDE-VORNAME.


* Hier wird das Ereignis (Event) zum Empfangen der Telefondaten
deklariert.

GET ZADTELEFON.

WRITE: / ZADTELEFON-VORWAHL, ZADTELEFON-TELEFON.


*  Ereignis  Ein  Kunde  ist  abgearbeitet  .  =>  Neue  Seite

GET ZADKUNDE LATE.

* uline.

NEW-PAGE.
```

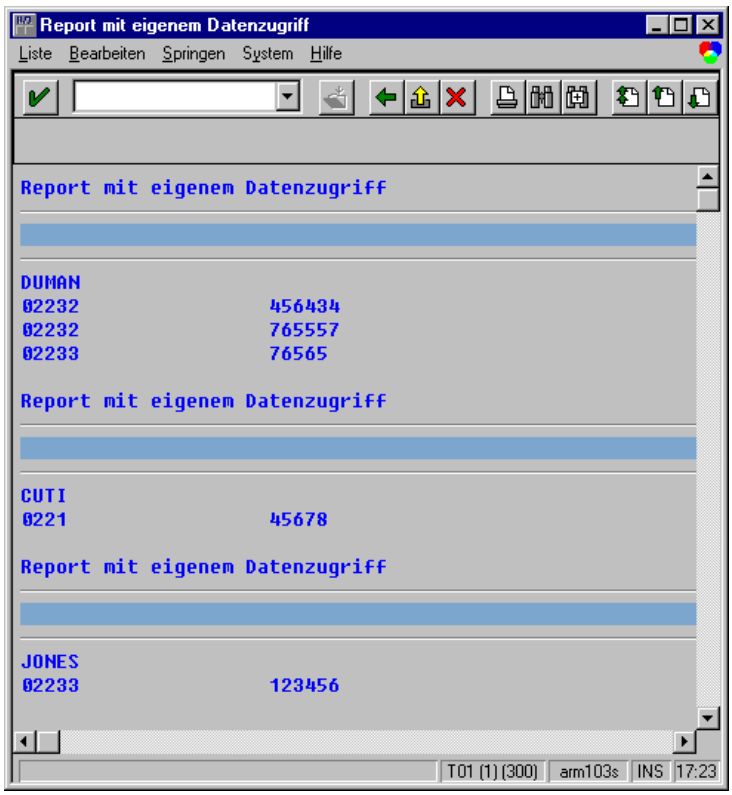


Bild: Ausgabe eines einfachen R/3 - Reports

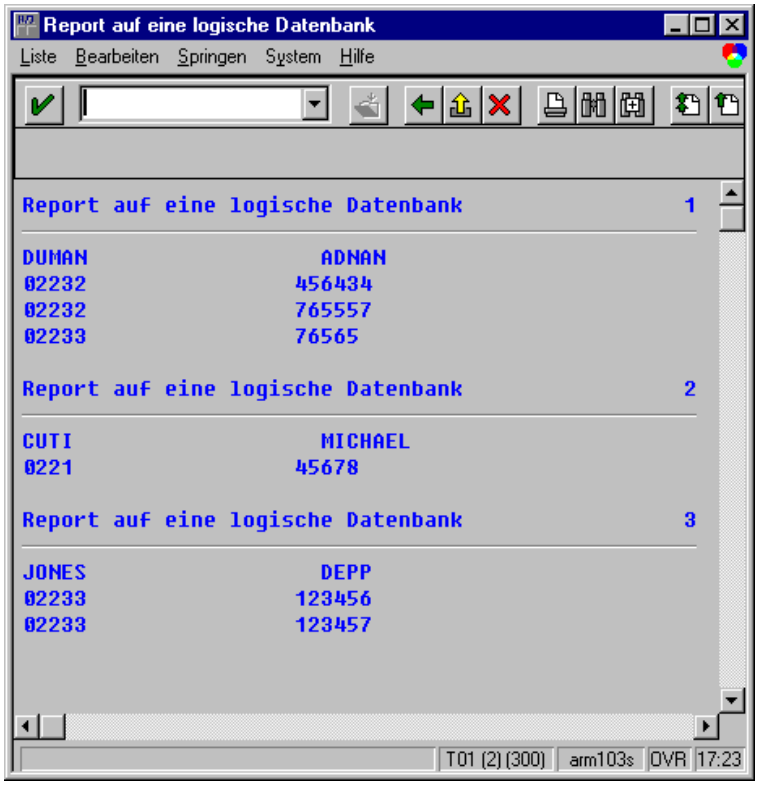


Bild: Ausgabe eines R/3 - Reports mit logischer Datenbank

7.

8. Tips & Tricks zu Reports

- In jedem Report - Ausgabefenster (siehe Bilder) kann über SYSTEM / LISTE / LISTÜBERSCHRIFT der Anwender die Spaltenbeschriftungen der Ausgabe anpassen.

- Im ABAP/4 - Editor kann während der Programmierung eines Reports mit SPRINGEN / TEXTELEMENTE ein Tool zur Anpassung und Vorgabe aller Texte im Zusammenhang mit dem zu erstellenden Report aufgerufen werden. Dies betrifft Listen- bzw. Spaltenüberschriften und die Bezeichnungen der Selektionsfelder im Selektionsbild. Insbesondere letzteres kann sehr praktisch sein, da als Vorgabebeschriftung für ein Selektionsfeld der Variablenname des PARAMETERS bzw. SELECT-OPTIONS - Befehls dient.

Namenskonventionen

Beim Anlegen von neuen Objekten im R/3 - System ist eine strenge Namenskonvention vorgegeben, die strikt eingehalten werden muß. Diese Namenskonvention gilt für fast alle ABAP/4 - Workbench - Objekte wie z.B. Tabellen und Views im Data - Dictionary, aber auch für Dynpros und Reportprogramme. Dies ist erforderlich, da mit der stetigen Weiterentwicklung von R/3 durch die SAP AG ständig neue Programme und Tabellen in das R/3 - Standardsystem aufgenommen werden. Wenn jeder R/3 - Kunde bei seinen internen Anpassungen und Weiterentwicklungen beliebige Bezeichnungen für seine Programme und Tabellen verwenden dürfte, wäre die Gefahr von Überschreibungen bei einem der regelmäßigen R/3 - Releasewechsel (Updates) sehr hoch. Die SAP AG hat daher die Namenskonvention aufgestellt, daß jeder Kunde seine ABAP/4 - Workbench - Objekte z.B. mit einem Y bzw. einem Z als erstes Zeichen zu benennen hat. Außerdem empfiehlt die SAP AG, daß jeder Entwickler seine erstellten Programme zusätzlich mit seinen Namensinitialien an zweiter und dritter Stelle benennt. Für den Autor würde dies bedeuten, daß alle Programme, die er in R/3 anlegt mit *ZAD* beginnen müssen. Die SAP - Programme beginnen naturgemäß mit *SAP* . In der Praxis führt dies leider oft zu sehr nichtssagenden Programmbezeichnungen, da die zulässigen Namenslängen bei R/3 recht knapp bemessen sind.

Bei Programmen sind dies z.B. 8 Zeichen, von denen 3 schon für die Namenskonvention vergeben werden müssen. Ab der R/3 - Version 4.X soll sich dies allerdings bereits geändert haben.

Die Namen einiger SAP - Objekte verstoßen gegen diese Namenskonvention, da diese Objekte vor der Einführung der Namenskonvention entstanden. SAP hat aus Aufwands- und Stabilitätsgründen bei diesen Objekten auf eine Umbenennung verzichtet. Die Namen dieser Objekte sind in der Ausnahmetabelle TDKZ verzeichnet und dürfen beim Anlegen von Kunden - Objekten ebenfalls nicht verwendet werden !

1.
2. **Kunden - Namensräume für ABAP/4 Dictionary Objekte**

Objekt	Namenslänge	Kundennamensraum	Beispiel
Domäne	10	Y*, Z*	YNAME, Z1234
Datenelement	10	Y*, Z*	ZNAME, YNAME12
Datenelement-Zusatznummer	4	9000 bis 9999	
Matchcodeobjekt	4	Y*, Z*	YMCO, ZMCO
Matchcode - ID	1	0 bis 9	YMCO1, ZMCO7
Pool-/Clustername	10	Y*, Z*	YPOOL, ZCLUSTER
Sperrobjekt	10	EY*, EZ*	EYNAME, EZNAME
Struktur	10	Y*, Z*, T9*, P9*	YSTRUKT, ZSTR123
transp. Tabelle	10	Y*, Z*, T9*, P9*	YTAB1, ZTAB2
Indexkennung	10	Y*, Z*	Y12, ZAB
Appends	3	Y*, Z*	ZAPPEND, YAPPEND
Felder	10	YY*, ZZ*	ZZFELD, YYFELD
Pool-/Clustertabelle	10	Y*, Z*, T9*, P9*	YPOOL, ZCLUSTER
View	10	Y*, Z*	YVIEW, ZVIEW
Helpview	10	H_Y*, H_Z*	H_YVIEW, H_ZVIEW
Kunden - Includes	10	CI_*	CI_KUNDE

Außerdem gibt SAP folgende Empfehlungen:

Für transparente Tabellen, Strukturen und Pool-/Clustertabellen sollten die Namensräume Y* und Z* verwendet werden. Die Namensräume T9* und P9* sind für spezielle Tabellen vorgesehen. T9* ist für die Pooltabellen im ATAB - Pool und P9* für kundeneigene Infotypen (HR) vorgesehen.

Kapitel 2 – Projekt Produktkatalog

Einführung

Nachdem die technischen Möglichkeiten von R/3 bzw. der ABAP/4 – Workbench ausgiebig erläutert wurden, wird in diesem Kapitel die Aufgabenstellung analysiert. Dazu werden, nachdem die Aufgabenstellung noch einmal formuliert wurde, die vorhandenen Katalogseiten der August Rüggeberg GmbH & Co analysiert und in ihre elementaren Datenstrukturen zergliedert.

Aufgabenstellung

Es soll jederzeit ohne technische Vorlaufzeiten eine Datei erstellt werden können, die alle Informationen aus einem SAP R/3 – System in geeigneter Form strukturiert enthält, um aus ihr mit entsprechender externer Software automatisiert einen Produktkatalog zu erstellen. Neben den eigentlichen Produktdaten wie Produktbezeichnungen, Preisen etc. enthält diese Datei zusätzliche Texte, Layout- und Bildinformationen, die katalogspezifisch sind. Ziel ist es, die Vorlaufzeiten und -kosten vor einer Katalogproduktion erheblich zu senken und damit öfter aktuellere Kataloge produzieren zu können. Außerdem sollen einzelne Spezialkataloge mit ausgewählten Produkten gedruckt werden können. Um die sich anbietenden Synergieeffekte zu nutzen, soll der gleiche Datenbestand auch für die Internet-Präsentation und eine CD-ROM Produktion verfügbar sein. Die Pflege der Daten soll durch maximale Nutzung von operativen Daten auf ein Minimum gesenkt werden, womit auch das Problem der Aktualität und Redundanz gelöst würden.

Als zusätzliche unabdingbare Nebenbedingung ist die Flexibilität der Katalogstruktur und der zu pflegenden Produktdaten und Zusatzdaten (Katalogtexte, Fotos, etc.) gegeben. Insbesondere diese Zusatzinformationen sollen nicht proprietär abgelegt werden, sondern evtl. später anderen Anwendungen zugänglich sein. Da der spätere Anwender kein EDV- bzw. R/3 - Fachmann sein wird (Marketing / Vertrieb), soll er ohne besondere Systemkenntnisse mit einfachen Mitteln eigene Katalog- und Produktdatenstrukturen anlegen und bearbeiten können.

Außerdem unterliegen alle Lösungen natürlich der Restriktion der Konformität mit den SAP-Richtlinien und der Releasefähigkeit von SAP R/3.

Unabhängig von R/3 und ausgehend von der Aufgabenstellung wirft ein datenorientierter Top-Down-Ansatz folgende Fragen auf:

- Welche Informationen / Daten sollen in welcher Form verfügbar gemacht

werden (Output) ?

- Welche Informationen sind zur Generierung der Outputdaten notwendig (Input) ?
- Wie müssen diese Input-Informationen verarbeitet werden (Verarbeitung) ?

Bei der Integration in R/3 stellen sich zusätzlich noch folgende Fragen:

- Sind die Input-Informationen bereits im System vorhanden ?

  Wenn ja, wie lassen sie sich möglichst transparent für den Anwender in die Lösung integrieren. Wenn nicht, muß ein möglichst standardisiertes Verfahren zur Speicherung und Pflege dieser Informationen gefunden werden.

- Kann durch eine weitere Nutzung der Ausgangsdaten, der zu integrierenden Lösung, ein zusätzlicher positiver Synergieeffekt erzielt werden ?

1.

2. Analyse der Produktdatenstruktur

Wie bereits in der Aufgabenstellung erwähnt wurde, soll eine einzige Datei (Katalogdatei) die gesamten Informationen zur Erstellung eines Produktkataloges in strukturierter Form beinhalten. Um diese Informationen genauer zu spezifizieren, muß der Aufbau eines bereits bestehenden Kataloges analysiert werden.

Die in diesem Kapitel gemachten Aussagen zur Strukturierung der Informationen in einem Produktkatalog sind grundsätzlich unabhängig von einem bestimmten Medium gültig.

Es folgen einige beispielhafte Ausschnitte des Produktkataloges der August Rüggeberg GmbH & Co. Im Anhang sind einige vollständige Katalogseiten der August Rüggeberg GmbH & Co abgebildet.

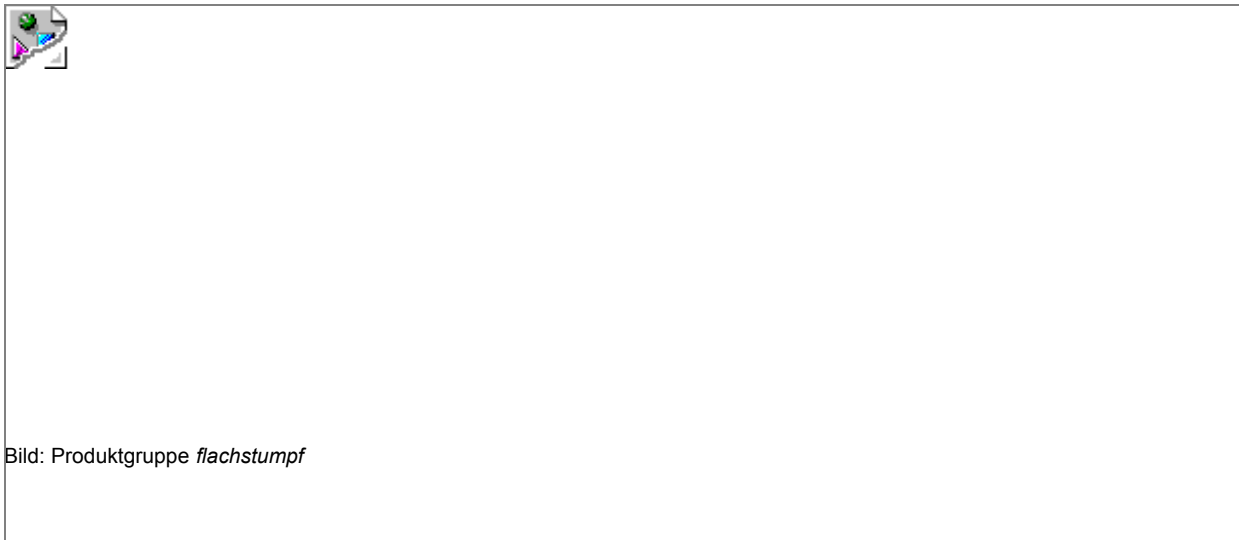


Bild: Produktgruppe *flachstumpf*

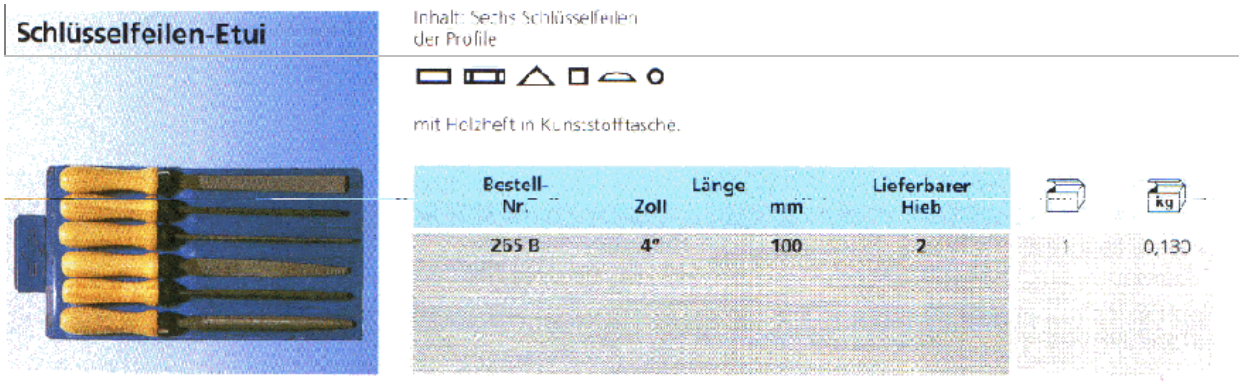


Bild: Produktgruppe *Schlüsselfeilen - Etui*

Bild: Produktgruppe *Schleifblätter*

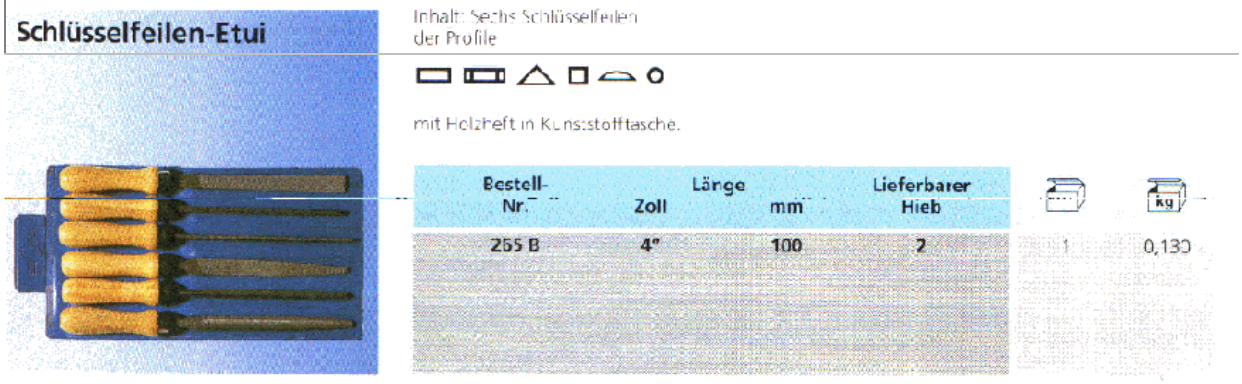


Bild: Produktgruppe *Kleinfiberschleifer*

Bild: Produktgruppen *Kissenschleifer, Klettronden, Klettrondenhalter*

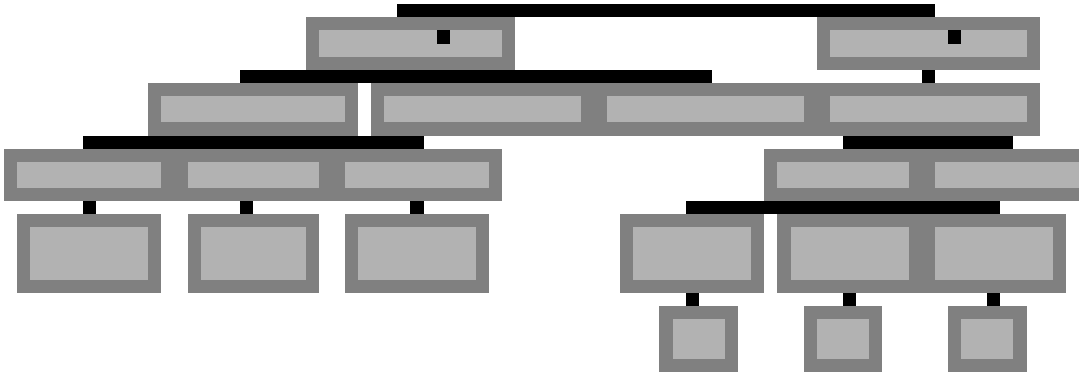
Vom Katalog ausgehend lassen sich die Informationen in der Katalogdatei in folgende Gruppen unterteilen:

DATENGRUPPE 1

Produkthierarchie, Zuordnung Produkt <=> Hierarchieebene

Die Produkte in einem Produktkatalog sollten grundsätzlich nach irgendeinem Kriterium geordnet präsentiert werden. Es macht keinen Sinn, Produkte verschiedenster Art z.B. nach Bestellnummern sortiert einfach aufzulisten, da damit ein gezieltes Vorgehen eines Kunden bei der Suche nach einem bestimmten Produkt nicht möglich ist. Es sollten Produktgruppen gebildet werden, die gleichartige Produkte zusammenfassen, wie z.B. Bohrer, Feilen, Schleifscheiben etc. Als weitere Gliederungsmaßnahme können hierarchische Beziehungen zwischen Produktgruppen gebildet werden, was wiederum die Komplexität einer großen Produktvielfalt verringert.

Beispiel:



Die Bildung von Produktgruppen und -hierarchien findet innerhalb von R/3 an verschiedenen, voneinander unabhängigen Stellen, Anwendung. Sie kann im Klassensystem zur schnellen Produktsuche oder im Vertrieb (Modul SD) für die Preiskalkulation etc. eingesetzt werden.

Grundsätzlich könnten bereits vorhandene Produkthierarchien in R/3 auch für die Verwendung im Katalog genutzt werden. Da jedoch die Hierarchiestruktur für die Katalogpräsentation eigenen Kriterien unterliegt (Kataloglayout) und völlig unabhängig von den anderen Produkthierarchien verändert werden sollte, muß sie unabhängig von anderen Produkthierarchien gespeichert werden. Ansonsten hätten Änderungen im Kataloglayout auch zwangsläufig z.B. Preisänderungen von Produkten zur Folge. Auch die Zuordnung eines Produktes zu einer bestimmten Produktgruppe im Katalog sollte keinerlei Rückwirkung auf andere Anwendungen dieses Produktes haben.

Es sollte auch die Möglichkeit bestehen, mehrere voneinander unabhängige Katalogstrukturen (Hierarchien) zur erstellen, um z.B. verschiedene Zielgruppen individuell anzusprechen.

Die Trennung der Kataloghierarchie von anderen Anwendungen hat zur Folge, das jedes Produkt manuell einzeln einer Produktgruppe und damit einer Hierarchieebene zugeordnet werden muß. Dabei kann ein Produkt auch mehreren Hierarchieebenen zugeordnet werden.

DATENGRUPPE 2

Technische Produktdaten

Die Daten dieser Datengruppe entsprechen den Produktdaten aus dem Materialstamm, die dem Kunden im Katalog präsentiert werden sollen. Neben allgemeinen Daten wie Produktbezeichnung, Einheit etc., können dies auch sehr produktspezifische Merkmale wie *Schaftdurchmesser, Winkel, Drehzahl, Länge* etc. sein. Die Anzahl solcher produktspezifischen Merkmale kann bei einer großen Produktvielfalt eines Unternehmens sehr hoch sein (mehrere Hundert).

Es darf hier also nicht von einem statischen Datensatz *PRODUKT* ausgegangen werden, der fest alle Produktdaten eines Unternehmens beinhaltet. Bei Aufnahme eines neuen Produktes in das Programm des Unternehmens können neue produktspezifische Datenfelder in den Materialstamm hinzukommen oder es können umgekehrt auch bestehende Datenfelder wegfallen.

Im allgemeinen werden Produkte mit gemeinsamen Datenfeldern (Merkmalen) in einem Katalog zu einer *Produktgruppe* zusammengefaßt (z.B. *Flachstumpf - Feilen*). Dies erlaubt eine einheitliche Darstellung von logisch zusammengehörenden Produkten im Katalog. Es kann also davon ausgegangen werden, daß die Produkte einer bestimmten Produktgruppe alle die gleichen technischen Merkmale (nicht Merkmalwerte) haben. Der *Produktgruppe* kommt bei der Katalogerstellung eine große Bedeutung zu. Sie ist als eigenständiges Objekt zu betrachten, das eigene, von den Produkten der Produktgruppe unabhängige, Merkmale haben kann.

Die technischen Produktdaten einer Produktgruppe, die im Katalog dargestellt werden sollen, müssen flexibel ein-/ausblendbar sein, d.h., es muß für jede Produktgruppe getrennt definierbar sein, welche Produktdatenfelder der Produkte in einer

Produktgruppe im Katalog präsentiert werden sollen.

DATENGRUPPE 3

Katalogspezifische Produktdaten

Für den Katalogdruck sind neben den eigentlichen technischen Produktdaten noch zusätzliche produkt- und produktgruppenspezifische Informationen nötig.

Zu den einzelnen Produkten und Produktgruppen können verschiedene Texte und Bilder zugeordnet werden, die der zusätzlichen Information des Kunden dienen. Dabei kann ein Produkt/Produktgruppe mehrere Bilder/Texte haben oder ein Bild/Text mehreren Produkten/Produktgruppen zugeordnet sein. Diese Informationen sind wie die technischen Produktmerkmale der Datengruppe 2 zu behandeln. Die Zuordnung eines Bildes zu einem bestimmten Produkt ist genauso Teil des Materialstamms, wie die Angabe einer Gewichtsangabe für dieses Produkt. Statt des Merkmals *Gewicht in gr* heißt das Merkmal dann z.B. *Produktfoto* , dem der Name eine Fotodatei zugewiesen wird.

Der Hauptunterschied dieser katalogspezifischen Informationen zu den technischen Produktdaten ist die Möglichkeit der häufigen Anpassung der verfügbaren Datenfelder. Während die verfügbaren technischen Merkmale langfristig unverändert bleiben dürften (Länge, Breite, Höhe, Tiefe, etc.) könnten die Zusatzinformationen vom Marketing eines Unternehmens
(fast) beliebig verändert werden. Es kann z.B. jederzeit ein zusätzliches

Foto oder eine Fußnote zu einer Produktgruppe hinzugefügt (bzw. entfernt) werden.

Es muß also möglich sein, schon bestehenden Produkten und Produktgruppen nachträglich beliebig ein zusätzliches Feld hinzuzufügen oder zu entfernen, da die Erstellung eines Kataloges ein iterativer Prozeß ist, der noch rückwirkende Änderungen an den Produktstrukturen notwendig werden lassen kann.

Für alle Texte der Datengruppen 1 bis 3 gilt die Restriktion der Mehrsprachfähigkeit, d.h. alle Merkmale, die landessprachenabhängige Textinformationen beinhalten können, müssen für jede Landessprache getrennt gepflegt werden können. Dabei müssen die Katalogstrukturen natürlich für alle Landessprachen einheitlich gültig sein. Ein Katalog muß jederzeit in jeder verfügbaren Sprache zu erstellen sein.

DATENGRUPPE 4

Artikelpreise

Die Preise der einzelnen Artikel in den Produktgruppen sind nicht als statische (technische) Eigenschaft eines Artikels, sondern als eigene Datengruppe zu betrachten. Die Artikelpreise dürfen einem Artikel nicht direkt wie ein technisches Maß (Länge, Breite etc.) zugeordnet werden, da dies die Preisangaben in einem *Werbemittel* (Katalog) vollkommen von der R/3 - Preisfindung abkoppeln würde, und damit zwangsläufig Inkonsistenzen entstehen würden, wenn die Preiskonditionen in R/3 geändert werden. Für jeden Artikel muß festgelegt sein, wie der Preis für diesen Artikel zu ermitteln ist, d.h. es muß ein Verweis auf die Konditionen zur R/3 - Artikelpreisfindung vorhanden sein. Diese Konditionen müssen mit denen übereinstimmen, die das R/3 - System zur operativen Preisfindung verwendet.

DATENGRUPPE 5

Layoutinformationen

Diese Informationen beschreiben die visuelle Art der Darstellung der Daten aus den Datengruppen eins bis vier in einem Zielmedium (Internet / CD-ROM / Printkatalog). Damit ist nicht die Beschreibung der endgültigen Position der Bilder und Texte auf einer Katalogseite gemeint, sondern lediglich WELCHE Informationen (Produkte / Produktgruppen), WIE dargestellt werden sollen. Jede Produktgruppe bekommt einen Layoutschlüssel (ID) zugewiesen, der von einer Publishing - Software, die die Katalogdatei verarbeitet, ausgewertet wer

den muß. Erst innerhalb dieser Publishing - Software wird anhand des Layoutschlüssels die tatsächliche visuelle Darstellung der Produktinformationen als Design - Schablone (Template) definiert. Dies beinhaltet sowohl die räumliche Anordnung auf einer Katalogseite, als auch die Darstellungsart in Form von verschiedenen Farben, Tabellentypen, anzuzeigenden Tabellenspalten, ...

DATENGRUPPE 6

Selektionsdaten, Katalogvarianten

Nachdem ein kompletter Katalog innerhalb von R/3 abgebildet worden ist (Datengruppen 1 bis 5), sollte in einem Selektionsverfahren definiert werden können, welche Produkte für den Katalogdruck exportiert werden sollen. Eine Selektion ist eine Untermenge eines bestehenden Kataloges und kann als *Teilkatalog* bezeichnet werden. Ziel dieses Selektionsverfahrens ist die Erzeugung von zielgruppenorientierten Spezialkatalogen oder saisonalen Katalogaktionen. Es können Produktgruppen und Produkte einzeln für den Export aktiviert / deaktiviert werden. Eine Selektion kann gespeichert und jederzeit wieder geladen werden, um einen Export zu wiederholen. Dabei können mehrere Selektionen unabhängig voneinander verändert und gespeichert werden.

Außerdem sollten die Produktfelder, die exportiert werden sollen, explizit einzeln selektiert werden können, da ein pauschaler Export aller Datenfelder unnötig viele Daten produziert und bei der Layouterstellung in der Publishing-Software die Übersichtlichkeit behindern kann.

3.

4. Die Database - Publishing - Software

Die Informationen, die als Output von R/3 zur Verfügung gestellt werden sollen, sind die Input Informationen der externen Publishing – Software. Um die Art und Struktur dieser Informationen verständlich zu machen, muß die Arbeitsweise dieser Database – Publishing - Anwendungen verstanden werden. Auf dem Markt werden z.Zt. ca. 20 bis 30 solcher Lösungen im Preissegment von ca. DM 1.000, 00 bis ca. DM 50.000, 00 angeboten. Der Autor hat drei dieser Publishing – Anwendungen aus verschiedenen Preissegmenten genauer betrachtet. Allen betrachteten Produkten liegt grundsätzlich das gleiche Konzept zugrunde. Alle drei Produkte erwarten eine Eingangsdatei, die die Informationen tabellenartig strukturiert beinhaltet. Jede Spalte entspricht dabei einem Datenfeld, jede Zeile einem Datensatz. Diese Tabelle beinhaltet alle Produkte mit

ihren Produktfeldern. Zusätzlich zu den Produkt- (Stammdaten) und Produktgruppenfeldern kommt für jede Hierarchieebene der Katalogstruktur eine zusätzliche Spalte hinzu, so daß jeder Datensatz exakt einer Hierarchieebene zugeordnet werden kann. Außerdem wird noch eine Spalte *Layoutschlüssel* angefügt, die die logische Verbindung zwischen den Datensätzen und dem Layout in der Publishing - Software herleitet.

Bei der Produktion der Katalogseiten liest die Publishing-Software die Tabelle von Anfang bis Ende und schaltet für jeden Datensatz anhand des Eintrags in der Spalte *Layoutschlüssel* intern das Darstellungslayout um. Solange ein Layoutschlüssel gleich bleibt, bleibt auch das Kataloglayout unverändert. Diese Darstellungslayouts werden innerhalb der Publishing-Software datenunabhängig erstellt (*Vorlagen* bzw. *Templates*). Bei der Erstellung der Layouts werden an den Positionen der späteren Produktdaten Platzhalter eingefügt, die mit den Spalten der Eingangsdatei logisch (über ihre Namen oder Positionen) verknüpft werden. Über die Bewertung der Spalte *Layoutschlüssel* in der Ursprungsdatenbank kann das zu verwendende Kataloglayout für eine bestimmte Produktgruppe gesteuert werden.

Bei einigen leistungsfähigeren Publishing-Produkten kann statt einer simplen *flachen* Eingangsdatei, die durch die Auflösung der Hierarchiestruktur extrem groß werden kann, auch eine relationale Datenbank angegeben werden, deren Tabellen die gleichen Informationen in normalisierter Form zur Verfügung stellen. Auf sie kann z.B. über ODBC zugegriffen werden kann. Eine wesentlich kleinere Eingangsdatei (keine Redundanzen) und eine schnellere Verarbeitung sind der Vorteil dieser Methode.

Als Endergebnis werden, je nach Publishing - Lösung, z.B. Dateien im Adobe PDF- bzw. HTML - Format erzeugt oder professionelle Publishing - Anwendungen von Fremdanbietern
(Quark - Xpress via Apple - Script) automatisiert ferngesteuert.

Es folgen kurz die analysierten Programme mit ihren Haupteigenschaften. Alle Produkte erlauben eine komfortable grafische Erstellung der Layout - Templates mit den bekannten Zeichenwerkzeugen (Kreis, Linie, Rahmen, ...). Bei Auswahl eines bestimmten Produktes empfiehlt der Autor insbesondere auf Details wie z.B. die *automatische Tabellenspalten - Anpassung* zu achten.

InBetween

Anbieter: Building – Systems, Köln.

InBetween ist eine reine Apple - Lösung, die als Apple – Script - Client die professionelle Publishing - Lösung *Quark - Xpress* automatisiert fernsteuert.

Vorteile:

Günstiger Preis, relative einfache Bedienung.

Nachteile:

Langsamer Produktionsvorgang durch Apple - Script, Verwaltung von vielen komplexen Kataloglayouts wird unübersichtlich.

Leaflet

Anbieter: Fa. Salomon, Köln.

Leaflet hebt sich durch den stark SQL - orientierten Datenzugriff auf beliebige ODBC - Datenbanken vom simplen *nächsten Datensatz lesen* – Prinzip ab. Leaflet ist nur für Windows NT verfügbar.

Vorteile:

Zugriff auf ODBC - Datenbanken, sehr schnelle Seitenproduktion, automatisierte HTML - Seitenproduktion für Internet-Präsentation.

Nachteile:

Für komplexere Layouts sind SQL - Kenntnisse notwendig.

CataloX

Anbieter: Fa. Schnittstelle, Stuttgart.

CataloX ist ein stark Workflow - orientiertes System, das z.B. auch Daten - Änderungen in der Layout - Software wieder zurück in die Ursprungsdatenbank schreiben kann.

Vorteile:

Sehr hohe Integration zwischen den beteiligten Systemen möglich.

Nachteile:

Hoher Implementierungsaufwand, hoher Beraterbedarf.

Die Datenfelder für ein Layout - Template können in ihrer Anzahl und Art sehr unterschiedlich ausfallen. Dies hängt grundsätzlich von zwei Faktoren ab:

Da für jede Produktgruppe, die spezifische Produktdatenfelder (Datengruppe 2) hat, mindestens ein eigenes Layout - Template definiert werden muß, steigt die Anzahl der Layout - Templates mit der Vielfalt der Produkte bzw. Produktgruppen.

Um die Attraktivität eines Produktkataloges zu steigern werden üblicherweise verschiedene Darstellungs - Designs in einem Katalog verwendet. Dies ist ausschließlich von der Phantasie der Layout - Designer abhängig.

1.
2. **Zusammenspiel zwischen Publishing – Software und Datengruppen**

Wie das Layout in seiner endgültigen Fassung optisch aussieht, wird in der Publishing - Software definiert. Dort werden alle Layouts visuell mit Spalten - und Gestaltungsdefinitionen so abgelegt, wie sie auch wirklich im Druck erscheinen sollen. So könnten in einer Produktgruppe, die die gleichen Produktdatenfelder (Datengruppe 2) wie eine andere Produktgruppe hat, zwei Bilder statt nur einem angegeben werden (Datengruppe 3), oder die Tabellendarstellung hätte ein völlig anderes visuelles Design, obwohl sie genau die gleichen Informationen beinhaltet. Während die Verwendung von zwei Bildern Auswirkungen auf die Produktgruppenstruktur hat (Name des zweiten Bildes), wird das visuelle Design komplett in der externen

Bearbeitungssoftware erstellt, ohne das ein gesondertes Feld in der R/3 – Umgebung erforderlich ist. Für jede einzelne Darstellungsform wird in der Publishing-Software ein eigenes Layout angelegt und mit einem Schlüssel (ID) versehen. Dieser Layoutschlüssel wird in R/3 einer Produktgruppe zugeordnet. Jeder Produktgruppe kann ein eigenes Layout zugewiesen werden, das auf die Produktgruppe (Anordnung von Bilder + Zusatztexten) und die Artikel der Produktgruppe angewendet wird.

Hier ist anzumerken, daß eine Standardisierung der Darstellungsformen (Bildanordnung etc.) unausweichlich ist. Die Folge dieser Standardisierung ist ein sehr einheitliches bzw. *langweiliges* Kataloglayout, dem die maschinelle Fertigung anzusehen ist. Bei günstigen Zubehörteilen (Schrauben etc.) oder sehr kurzlebigen Angeboten (Last – Minute - Reisen etc.) ist diese Einschränkung sicher kein Problem. Unternehmen, bei denen der Katalog ein wichtiger Werbe- und Informationsträger ist (Rüggeberg, Otto - Versand), legen jedoch zu recht sehr viel Wert auf eine individuelle und abwechslungsreiche Präsentation ihrer Produkte.

Diese Forderung kann durch die produktgruppenabhängige Layoutzuweisung erfüllt werden. Theoretisch könnte jede Produktgruppe ihr eigenes, individuell gestaltetes, Layout bekommen. Dadurch kann die vermeintliche Einschränkung der Darstellungsmöglichkeiten durch die Layout - Standardisierung auf ein Minimum beschränkt werden. Mit steigender Zahl an definierten Layouts steigt die Individualisierung der Katalogpräsentation, aber auch der Pflegeaufwand der Layouts, die in der externen Publishing-Software erstellt werden müssen. Es gilt, das wirtschaftliche Optimum zwischen Individualisierung und Standardisierung der Produktgruppen - Layouts zu finden. Letztendlich wird das verfügbare Budget das ausschlaggebende Kriterium sein. Die Möglichkeiten der visuellen Darstellung der Daten werden ausschließlich von der verwendeten Publishing-Software beeinflußt.

5.

6. Lösungsansätze

Bei der R/3-unabhängigen Entwicklung einer proprietären Lösung werden die benötigten Datenstrukturen und Algorithmen individuell entwickelt. Es werden eigene Pflegemasken und Bearbeitungsfunktionen implementiert und außer dem technischen Dateiformat der verwendeten Datenbank gibt es keinerlei Standards, die den Zugriff anderer Software auf die Daten ermöglichen.

Eine Voraussetzung für das integrative Konzept von R/3 ist, daß die Informationen nicht nur technisch in einer Standardform abgelegt werden, wie dies bei einer relationalen Datenbank der Fall ist (Datenbankformat: z.B. DBase, Access, Oracle, Sybase, etc.), sondern auch semantisch gewissen Standards unterliegen. Erst die einheitliche Ablage bestimmter Informationen ermöglicht deren modulübergreifende Nutzung.

Es ist zu erwähnen, daß die alleinige Verwendung von R/3 zur Abbildung von Softwarelösungen nicht automatisch die Nutzung eines Standards und der R/3-Integration bedeutet. Auch innerhalb von R/3 kann eine proprietäre Softwarelösung abgebildet werden, die alle Nachteile von Individualsoftware in sich birgt. Es können eigene Datenstrukturen angelegt und Pflegetransaktionen erstellt werden, die völlig unabhängig vom Gesamtsystem arbeiten. Es mag durchaus Fälle geben, wo dies sinnvoll ist, jedoch sollte bei alternativen Möglichkeiten die proprietäre Lösung die zweite Wahl sein.

Die Lösung soll möglichst standardisierte Ablageformen für Informationen innerhalb von R/3 nutzen. Dadurch soll die Option auf anderweitige, zukünftige Nutzungsmöglichkeiten der Daten erhalten bleiben.

Kapitel 3 – Abbildung der Informationsstrukturen in R/3

Das R/3 - System bietet mehrere flexible Möglichkeiten zur Ablage von anwendungsbezogenen Daten. Jede dieser Möglichkeiten hat Vor- und Nachteile, die nun unabhängig von der Aufgabenstellung bewertet werden.

7. Das Klassensystem

Das Klassensystem ist eine gute Möglichkeit für die Speicherung zusätzlicher Daten im R/3 - Stamm. Die hohe Performance - Effektivität und der geringe Anpassungsaufwandes (Customizing) sprechen eindeutig für eine häufige Verwendung des Klassensystems. Außerdem können neue Felder auch vom geübten Anwender hinzugefügt und verändert werden
(wenn er dazu berechtigt ist). Die Mehrsprachfähigkeit ist ebenfalls bereits enthalten. Die Anwendung des R/3-Klassensystems wird an entsprechender Stelle in dieser Arbeit ausführlich beschrieben.

1. Vorteile

- Maximale Flexibilität

  Zusätzliche Felder (Merkmale) können ohne Datenverluste und – anpassungen (von vorhandenen Daten) jederzeit aufgenommen werden

- Geringer Änderungsaufwand

  Die Aufnahme und Anpassung von Merkmalen findet in Standard - Dialogtransaktionen statt. Diese sind auch vom geübten Anwender zu bedienen. Es sind keinerlei Programmierarbeiten für die Anpassung von Dialog - Transaktionen etc. notwendig

- Orientierung an Standardlösung

  Das Klassensystem ist eine Standard – Funktionalität von SAP R/3. Damit stehen die Daten auch anwendungsübergreifend im gesamtem R/3 – System zur Verfügung

-
- Hohe Performance

  Die Speicherung der Daten, die in Merkmalen des R/3 – Klassensystems eingegeben werden, erfolgt sehr effektiv in eigenen Merkmaltabellen. Dabei wird nicht für jedes Merkmal ein eigenes Tabellenfeld (Tabellenspalte) generiert, sondern jedes zusätzliche Objekt - Merkmal entspricht einer Tabellen zeile . Dadurch wächst der Speicher- und Performancebedarf eines Objekt - Datensatzes (Stammsatz in Stammtabelle) nicht mit der Anzahl der Merkmale eines Objektes an, sondern es erhöht sich lediglich die Anzahl der Datensätze (Tabellenzeilen) in der Merkmaltabelle. Der Nachteil der höheren Anzahl an Datensätzen in der Merkmaltabelle kann durch die Verwendung von entsprechenden Tabellen – Indizes ausgeglichen werden, um Performance – Probleme zu vermeiden.

- Mehrsprachigkeit

  Die Merkmalbezeichnungen und Merkmalbewertungen sind mehrsprachfähig. Dies ist für viele multinationale Unternehmen eine Basisfunktionalität und ein echtes k.o. – Kriterium bei der Bewertung von Software – Lösungen

1. Nachteile

- Die Merkmale des Klassensystems haben max. 30 Zeichen Kapazität
- Die Pflege der Daten, die in Merkmalen des Klassensystems abgelegt werden, findet grundsätzlich unabhängig vom Kontext der Anwendung in standardisierten Pflegemasken des Klassensystems statt. Das integrative Konzept von R/3 kommt bei der Integration der Pflegedialoge des Klassensystems in die Pflegedialoge der Anwendungsdaten nicht voll zur Geltung und es wird eine gewisse Abstraktionsfähigkeit vom Anwender gefordert, was entsprechende Schulungen zur Folge hat.

  Die Funktion *Zusatzfelder* in der Dokumentenverwaltung zeigt einen Ansatz zur Lösung dieses Problems (siehe Dokumentenverwaltung / Zusatzfelder). Dies ist jedoch als Ausnahme zu bewerten.

1.

2. Zusätzliche Tabellen

R/3 bietet die Möglichkeit im Data - Dictionary eigene Tabellen anzulegen. Diese Tabellen können den Anforderungen beliebig angepaßt werden. Damit eignet sich diese Möglichkeit für die Ablage aller denkbaren Informationen, die in strukturierter Form vorliegen und damit natürlich auch für die Speicherung der Daten aus den Datengruppen 1 bis 6. Die Anlage von zusätzlichen Tabellen entspricht dem Einsatz einer relationalen Datenbank. Es müssen Tabellen mit ihren Feldern definiert und Relationen zwischen den Tabellen gebildet werden. Relationen zwischen den Tabellen werden über Fremdschlüssel abgebildet. Die Normalisierung und Indizierung von Tabellen gehört ebenfalls zu den auszuführenden Aufgaben. Da die Tabellen dem R/3 System neu hinzugefügt werden, existieren auch noch keine Dialogtransaktionen zur Anzeige und Pflege dieser Tabellen. Die einzige Möglichkeit die Daten in den Tabellen zu bearbeiten, ist die Verwendung der Werkzeuge aus dem Data - Dictionary. Für die Benutzung durch einen Anwender sind diese jedoch völlig ungeeignet.

Daraus folgt, daß für die Anzeige, Pflege und Verarbeitung von Daten in eigenen Tabellen entsprechende individuelle Dialoge und Reports erstellt werden müssen. Dies ist innerhalb von R/3 grundsätzlich nur mit den Werkzeugen der ABAP/4 - Workbench und der Programmiersprache ABAP/4 möglich. Damit entspricht die Verwendung zusätzlicher Tabellen innerhalb von R/3 einer Individualprogrammierung und es können die Vor- und Nachteile von Individualsoftware zur Bewertung herangezogen werden. Die Verwendung zusätzlicher Tabellen entspricht einer proprietären Lösung bzw. einer Erweiterung des R/3-Standards. Daraus lassen sich auch folgende Eigenschaften einer solchen Lösung ableiten.

1. Vorteile

- Individuelle Dialogtransaktionen ermöglichen sehr komfortable und einfache Datenpflege, da sie auf genau einen Anwendungsfall und sogar zielgruppenorientiert angepaßt werden können.

- Es können individuelle Funktionalitäten optimal integriert werden.

- Individuallösungen sind im allgemeinen performanter als Standardlösungen, da sie weniger Overhead bieten.

- Funktionalitäten sind unabhängig von einem Anbieter zu implementieren.

1. **Nachteile**

- Evtl. Releasewechsel - Probleme

  Wenn bei einer Eigenentwicklung auf R/3 - Funktionsbausteine zugegriffen wird, die bei einem R/3 – Releasewechsel verändert werden, können Programmfehler oder sogar Datenverluste die Folge sein. Jeder Releasewechsel erfordert deswegen einen zusätzlichen Aufwand für die Überprüfung der weiteren Funktionsfähigkeit der Eigenentwicklungen.

- Geringere Offenheit

  Individuallösungen sind weniger offen als Standardlösungen. Die eingepflegten Daten stehen nicht immer ohne weiteres anderen Anwendungen zur Verfügung.

- Höherer Zeitaufwand

Die Entwicklung von individuellen Dialogmasken ist relativ programmieraufwendig.

- Höhere Kosten

  Die Fachkräfte für die Entwicklung von Dynpros etc. sind Mangelware und dementsprechend teuer. Für die Pflege werden langfristig höhere Personal - Ressourcen gebunden.

- Mangelnde Flexibilität

  Änderungen in den Anforderungen oder den Datenstrukturen haben immer auch Entwicklungsarbeiten zur Folge, was mit relativ hohem Kosten- und Zeitaufwand verbunden ist.

- Strategische Vorgabe

  Ein Unternehmen, daß die Entscheidung für den Einsatz von SAP R/3 getroffen hat, hat damit auch eine langfristige strategische Richtung für die EDV vorgegeben. Zu viele Individualentwicklungen würden diese Vorgabe unterlaufen und sollten immer gut begründet sein.

- Abhängigkeit vom Hersteller

  Jeder Softwareanwender ist auf Support durch den Softwarehersteller angewiesen. Bei Individualsoftware ist die Abhängigkeit eines Anwender - Unternehmens vom Hersteller der Individualsoftware viel größer als bei Standardsoftware, da bei Standardsoftware das nötige Know - how notfalls auch von Dritten zugekauft werden kann. Bei Individualsoftware ist diese Ausweichmöglichkeit des Anwender - Unternehmens nicht gegeben.

1.

2. Append - Strukturen

Append - Strukturen erlauben die Erweiterung von bestehenden Tabellen um weitere Felder, ohne die Tabellen direkt im SAP Data - Dictionary zu verändern. Es können z.B. R/3 Standardtabellen um zusätzliche Felder ergänzt werden. Dies ist möglich, weil das Anlegen von Tabellen im SAP - Data - Dictionary nicht auch dem sofortigen Anlegen einer Tabelle in der zugrundeliegenden relationalen Datenbank entspricht. Eine Tabelle wird erst dann physikalisch in der Datenbank angelegt, wenn sie im SAP Data - Dictionary *aktiviert* wird. Bei der Aktivierung wird die endgültige physikalische Tabellenstruktur aus den verschiedenen Tabellenbeschreibungs - Verfahren (DDL), die das SAP- Data - Dictionary zur Verfügung stellt, generiert und anschließend erst wird die Tabelle in der Datenbank angelegt.

Append - Strukturen sind Teil der SAP DDL (Data Definition Language), d.h. sie ändern (ergänzen) nicht direkt die physikalischen (SAP-Begriff => transparenten) Datenbanktabellen, sondern beschreiben lediglich die Ergänzungen. Bei Aktivierung einer Tabelle führt R/3 normalerweise ein SQL - ALTER TABLE aus, das die Tabellenstruktur physikalisch anpaßt. Bei einem R/3 - Releasewechsel werden Append - Strukturen automatisch berücksichtigt.

Append-Strukturen würden sich z.B. für die Erweiterung des Materialstammsatzes eignen, wenn diese Felder als elementarer Bestandteil des Materialstammsatzes einzuordnen sind und keine ständigen Anpassungen zu erwarten sind.

1. Vorteile

- Releasefähigkeit
  Bei einem Releasewechsel werden die Append-Strukturen bei Standardtabellen mit berücksichtigt, d.h. es sind keine Vorkehrungen zur Sicherung und nachträglichen Bereitstellung der Daten zu treffen.

- Die Datentypen und -längen der Felder sind frei definierbar, d.h. sie können auf den Anwendungsfall angepaßt werden

1.

2. Nachteile

- Die zusätzlichen Felder werden nicht automatisch vom System erkannt, d.h. es stehen nicht automatisch Pflegedialoge für diese zusätzlichen Felder zu Verfügung. Für die Pflege der zusätzlichen Felder müssen eigene Dialogtransaktionen entwickelt werden. In einigen R/3 Anwendungen besteht die Möglichkeit, die Standarddialoge um die zusätzlichen Felder zu erweitern. Dies ist z.B. im Materialstamm der Fall.

- Jede Änderung bzw. Ergänzung von Feldern hat zwangsläufig eine manuelle Anpassung der Dialoge zur Folge. Damit ist die Verwendung von Append-Strukturen nur für langlebige Anpassungen geeignet.

- Der Umgang mit Data - Dictionary und Dialogentwicklung erfordert teueres Know-how und/oder relativ hohe Programmierkapazität

- Bei einer sehr hohen Zahl von Datenfelder (einige Hundert) können sowohl die Kapazitäten der zugrundeliegenden Datenbank erreicht werden, als auch Performance - Probleme auftreten. Letzeres ist besonders bei Tabellen mit hohem Zugriffsvolumen wie der

Materialtabelle MARA zu berücksichtigen.

- Keine Mehrsprachigkeit

Zusammenfassend läßt sich feststellen, daß die Aufnahme neuer Felder über Append - Strukturen durchaus seine Berechtigung haben kann. Umzusetzen ist es jedoch nur durch einen Entwickler. Der Aufwand schränkt die Anwendungsgebiete dieses Verfahrens jedoch stark ein. Es ist ideal für Daten im Materialstamm geeignet, die z.B. eine Produkteigenschaft beschreiben, die nicht in den Standard – SAP - Vorgaben enthalten ist, aber dringend dort mit gepflegt werden soll. Durch die fehlende Mehrsprachigkeit ist die Anwendung in erster Linie auf eher technische Merkmale beschränkt.

1.
2. **Beispiel: Aufnahme eines zusätzlichen Tabellenfeldes im Materialstamm.**

Die Pflegedialoge des Materialstamms bestehen aus einer Sammlung von Subscreens, die abhängig von der verwendeten Anwendersicht (Einkauf, Verkauf, etc.) und anderen Faktoren (Werk etc.) dynamisch während der Laufzeit eines Programms zu einem individuellen Pflegedialog zusammengestellt werden. Für diese Anwendersichten müssen eigens SUBSCREENS erstellt werden. SUBSCREENS sind Bausteine, die sich flexibel zu Dynpros zusammenbauen lassen (Subscreens werden auch als FRAMES in Internet-Browsern verwendet).

Im Customizing kann im Unternehmens – IMG, dem zentralen R/3 - Customizing Werkzeug, unter LOGISTIK ALLGEMEIN / GRUNDDATEN LOGISTIK / MATERIAL / BENUTZEDEFINIERTEN MATERIALSTAMM KONFIGURIEREN die Logik der Darstellung der Felder in den Subscreens des Materialstamms parametrisiert werden. Dabei werden die Zustände (Nur Lesen, Editierbar, Ausgeblendet) der Felder in Abhängigkeit von verschiedenen Faktoren (Werk, Anwendersicht, etc.) definiert.

Die Aufnahme eines neuen Subscreens ist nicht als einfacher Parameter möglich, sondern es muß das SAP - Standardprogramm, daß die Dialogführung steuert, komplett kopiert werden. In diese Ablauflogik der Kopie können dann eigene Subscreens mit eingebunden werden.

Die genauen Schritte der Einbindung sind in der SAP - Dokumentation dargestellt. Alternativ zur Einbindung eines neuen Materialfeldes über Subscreens können eigene Dialogtransaktionen für die Pflege der neuen Datenfelder erstellt werden. Der Aufwand entspricht ungefähr dem ersten Weg, jedoch ist die Form der Integration ein anderer.

Dokumentation:

R/3 Bibliothek | Einführungswerkzeuge | Einführungsleitfaden (IMG) | Logistik Allgemein | Grunddaten Logistik: Materialstamm | Benutzereigenen Materialstamm konfigurieren

1.
2. **Werbemittel**

Die Komponente *Werbemittel* ist für den Aufbau einer Werbemittelstruktur und Produktzuordnung ausgelegt. Um den R/3 – Standard zu wahren sollten diese Informationen in jedem Fall auch dort gepflegt werden. Der eigentliche Trumpf der Komponente *Werbemittel* ist jedoch die Funktionalität der automatischen Preisermittlung der Produkte. Über eine BAPI - Funktion können alle Produktpreise

einfach ausgelesen werden, ohne mit der komplexen Preisfindungsfunktionalität von R/3 in Berührung zu kommen.

Leider kann die Funktionalität des Klassensystems nicht auf die *Werbemittel* angewendet werden, da sich die *Werbemittel* – Objekte nicht klassifizieren lassen. Dieser Umstand muß über Dummy – Artikel umgangen werden. Auf die Verwendung von Dummy – Artikeln wird später noch näher eingegangen.

1. **Vorteile:**

- Releasefähigkeit

  Die Komponente *Werbemittel* ist eine R/3 – Standardfunktionalität.

- Automatische Preisermittlung

  Die Produktpreise werden automatisch nach der R/3 – Preiskonditionierung errechnet. Damit ist die Konsistenz der Preisinformationen gewährleistet.

1. **Nachteile:**

- Keine direkte Verwendungsmöglichkeit des Klassensystems

  Es müssen Dummy - Artikel verwendet werden.

- Keine Programmierschnittstellen für schreibenden Zugriff vorhanden

  Es existieren BAPI – Funktionsbausteine, um die Katalogstruktur, die Katalogzusatzinformationen der Komponente *Werbemittel* und die Produkte mit deren Preisen von einer externen Software auszulesen. Das Zurückschreiben dieser evtl. veränderten Daten ist z.Zt. bis zur Version 4.0 nicht ohne weiteres möglich. Auf diese Problemstellung wird in Kapitel 5 näher eingegangen.

1.
2. Anwendungsbezogene Bewertung der technischen Möglichkeiten

Bei der anwendungsbezogenen (Datengruppen 1 bis 6) Bewertung der verschiedenen Abbildungsverfahren läßt sich feststellen, daß es nicht eine Ideallösung gibt, die alle Anforderungen gleichzeitig abdeckt. Der goldene Mittelweg ist eine Verteilung der Daten aus den verschiedenen Datengruppen auf die einzelnen Teilsysteme des R/3 - Gesamtsystems.

Nachdem die relevanten technischen Möglichkeiten mit Vor- und Nachteilen dargestellt wurden, muß für jede Datengruppe, die bei der Analyse der Aufgabenstellung ermittelt wurde, eine möglichst optimale Ablage in R/3 festgelegt werden.

1. **Datengruppe 1**

Produkthierarchie, Zuordnung Produkt <=> Hierarchieebene

Für diese Datengruppe sind folgende Kriterien relevant:

- Releasefähigkeit.
- Anwendungsübergreifende Verfügbarkeit der Daten.

Die Komponente *Werbemittel* ist die ideale Ablage für diese Information, da es auch genau für diese Informationen konzipiert wurde. Die Daten stehen anwendungsübergreifend zur Verfügung, so daß z.B. der Internet - Produktkatalog oder der Internet - Online - Store darauf zugreifen können. Außerdem existiert eine standardisierte Schnittstelle (BAPI), die die Informationen für externe Software - Anwendungen verfügbar macht.

1.
2. **Datengruppe 2**

Technische Produktdaten

Für diese Datengruppe sind folgende Kriterien relevant:

- Releasefähigkeit
- Anwendungsübergreifende Verfügbarkeit der Daten
- Mehrsprachigkeit
- Flexible und schnelle Ergänzung/Änderung von Feldern, am besten durch den Anwender
- Möglichst geringe Belastung der Performance, da eine hohe Zahl von Datenfeldern zu erwarten ist (mehrere Hundert)

Das R/3 - Klassensystem bietet sich an, diese Daten abzubilden. Dabei wird die Einschränkung der kurzen Merkmalbewertungen (30 Zeichen) über Verweise auf Dokumente im DVS umgangen. Näheres dazu wird im Konzept in Kapitel 4 erläutert.

1. **Datengruppe 3**

Katalogspezifische Produktdaten

Für diese Datengruppe sind folgende Kriterien relevant:

- Releasefähigkeit
- Anwendungsübergreifende Verfügbarkeit der Daten
- Mehrsprachigkeit
- Möglichst geringe Belastung der Performance, da eine hohe Zahl von relativ großen Datenobjekten zu erwarten ist (mehrere Tausend). Es können Bilder mit bis zu 100 MB und mehr zur Nutzung kommen

Alle Bilder und Texte mit beschreibendem Charakter werden im R/3 - Dokumenten - Verwaltungssystem (DVS) abgelegt. Auch ihre Pflege wird zentral im DVS durchgeführt. Die Zuordnung der einzelnen Dokumente, d.h. der Bilder und Texte an die entsprechenden Produkte und Produktgruppen geschieht über zusätzliche Merkmale des Klassensystems, die auf die Dokumente verweisen. Näheres dazu wird im Konzept in Kapitel 4 erläutert.

1. **Datengruppe 4**

Artikelpreise

Für diese Datengruppe sind folgende Kriterien relevant:

- Releasefähigkeit
- Automatische Preisfindung unter Berücksichtigung der R/3 – Preis – Konditionierung

In der Komponente *Werbemittel* ist die automatische Preisfindung bereits integriert. Zu jedem Produkt wird automatisch der Preis entsprechend den im *Werbemittel* eingetragenen Preis - Konditionen ermittelt.

1. **Datengruppe 5**

Layoutinformationen

Für diese Datengruppe sind folgende Kriterien relevant:

- Releasefähigkeit
- Flexible und schnelle Ergänzung / Änderung von Feldern, im Idealfall durch den Anwender, der auch die Produktkatalogdaten pflegt

Die Informationen dieser Datengruppe sind grundsätzlich produktgruppenbezogen. Eine Produktgruppe entspricht in R/3 einer Hierarchieebene der *Werbemittel*. Damit ist es naheliegend, diese Informationen auch bei einer *Werbemittel* - Hierarchieebene abzulegen, indem sie dort z.B. im Feld *Langtext* strukturiert eingegeben werden.

Die geforderte Flexibilität und Bedienbarkeit erfordert jedoch den Einsatz des Klassensystems. Für jede Produktgruppe (Hierarchieebene) wird eine eigene Klasse angelegt, der die Merkmale dieser Produktgruppe zugeordnet werden. Anschließend werden die *Werbemittel* - Hierarchieebenen in *ihrer* Klasse klassifiziert.

Da sich in R/3 eine einzelne *Werbemittel* - Hierarchieebene nicht klassifizieren läßt, muß hier jedoch ein wenig *getrickst* werden. Die Zuordnung einer *Werbemittel* – Hierarchieebene an eine Klasse geschieht nicht direkt über die Klassifizierung einer *Werbemittel* – Hierarchieebene, sondern über Dummy - Artikel, die je eine *Werbemittel* – Hierarchieebene repräsentieren und statt dieser in den entsprechenden Klassen klassifiziert werden. D.h. es wird für jede *Werbemittel* – Hierarchieebene ein eigener Artikel im R/3 Materialstamm angelegt, der durch ein eindeutiges Kriterium mit dieser Werbemittel – Hierarchieebene logisch verknüpft werden ist. Am naheliegendsten ist die Verwendung von übereinstimmender Artikelbezeichnung und Werbemittel – Hierarchieebenen – Bezeichnung.

Näheres dazu wird im anschließenden Konzept erläutert.

1. **Datengruppe 6**

Selektionsdaten, Katalogvarianten

Für diese Datengruppe sind folgende Kriterien relevant:

- Releasefähigkeit.
- Produktive und einfache Pflege und Verwaltung der verschiedenen Kataloge.

Diese Daten werden mit hoher Wahrscheinlichkeit in keiner anderen Anwendung Verwendung finden, so daß hier eine Speicherung in eigens angelegten Tabellen sinnvoll erscheint. Die Forderung nach einer produktiven und einfachen Pflege und Verwaltung der verschiedenen Kataloge erfordert sowieso höchst wahrscheinlich die Entwicklung von eigens auf diesen Anwendungsfall zugeschnittenen Dialogen, so daß dieser Nachteil der eigenen Tabellen kaum ins Gewicht fällt.

1. **Zusammenfassung:**

Die hier dargestellte Lösung zur Ablage der Informationen aus den Datengruppen 1 bis 6 orientiert sich stark an der Nutzung der von R/3 gebotenen Möglichkeiten der standardisierten Datenablage.

Die sich daraus ergebenden Vorteile sind:

a. ein geringerer Aufwand für die Implementation, da sofort mit dem Aufbau der Daten - Strukturen begonnen werden kann (mit SAP-Mitteln).

b. die Daten stehen evtl. zukünftigen anderen Anwendungen zur Verfügung

c. garantierte Releasefähigkeit

d. Skalierbarkeit und Flexibilität für die Zukunft

e. Übertragbarkeit auf andere SAP R/3 - Systeme und Datenbestände

Kapitel 4 – Das Abbildungskonzept in SAP R/3

In diesem Abschnitt wird der gesamte Vorgang der Datenpflege schrittweise mit allen Zusammenhängen dargestellt. Dabei wird auch der konzeptionelle Hintergrund entsprechend erläutert.

1. **Anlage eines neuen *Werbemittel*s**

   Nach dem Anmelden am R/3 - System kann über das Menü LOGISTIK / MATERIALWIRTSCHAFT / WERBEMITTEL in die Werbemittelplanung verzweigt werden.

Mit WERBEMITTEL ANLEGEN wird eine neues, leeres *Werbemittel* angelegt.

Bild: *Werbemittel* anlegen

Folgende Grunddaten müssen für ein neues *Werbemittel* gepflegt werden.

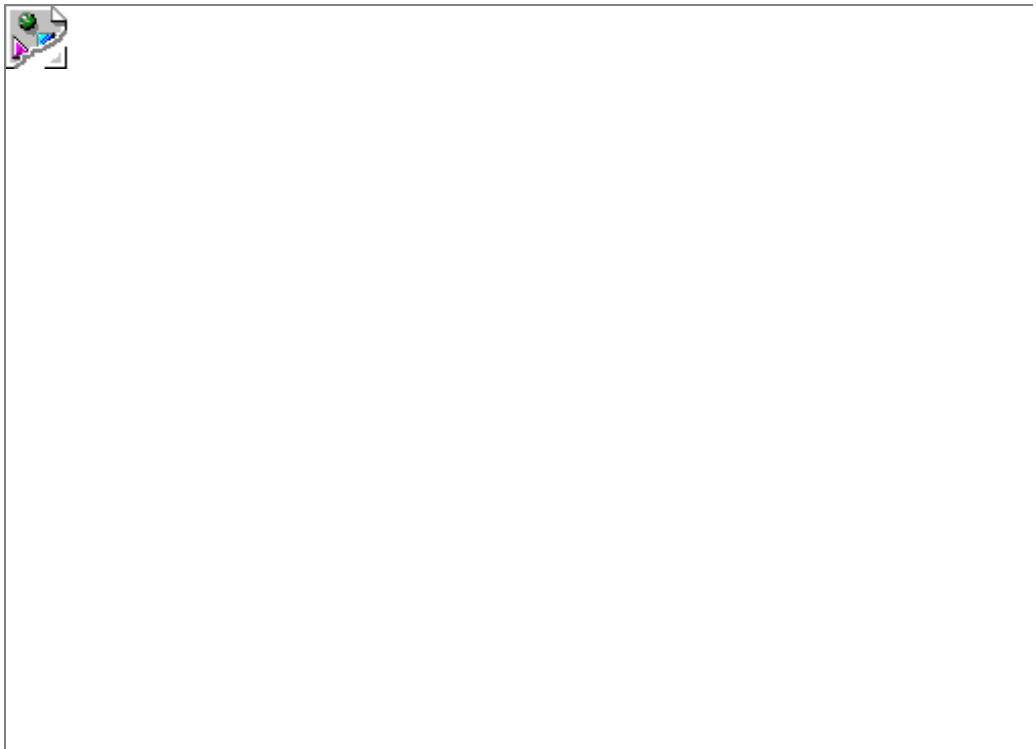


Bild: *Werbemittel*: Grunddaten

Die Angaben in dem Bereich PREISRELEVANTE DATEN dienen der späteren Preisermittlung der Produkte, d.h. die Angaben an dieser Stelle bestimmen die Preise, die im Produktkatalog erscheinen sollen. Im Bereich VARIANTEN können verschiedene Varianten eines Kataloges angelegt werden, die in ihrer Struktur und ihren Artikelzuordnungen voll übereinstimmen, jedoch eine andere Währung oder Landessprache (oder beides) haben.

2. **Aufbau der Kataloghierarchie in *Werbemittel*.**

Im Modul *Werbemittel* wird die Katalogstruktur in Form eines hierarchischen Baumes aufgebaut. Es ist zu beachten, daß Informationsseiten wie *Einführungsseite 1* als Hierarchieknoten abgebildet werden können. Die endgültige Verwendung dieser Informationen ist abhängig von der Interpretation in der Publishing - Software. Dort kann, über ein entsprechendes Template, z.B. eingestellt werden, daß die Produktgruppe *Einführungsseite 1* einseitig dargestellt werden soll. Der Inhalt der *Einführungsseite 1* kann z.B. als Dokument im R/3 – Dokumentenverwaltungssystem abgelegt werden. Selbstverständlich kann eine solche Seite auch aus mehreren Dokumenten zusammengesetzt sein. Über Klassen - Merkmale der Werbemittel – Hierarchieebenen (indirekt über Dummy - Artikel) kann auf Dokumente im DVS verweisen werden (Bewertung der Merkmale mit Dokumentnamen).

An dieser Stelle ist noch einmal hervorzuheben, daß die aufzubauende Hierarchiestruktur die Katalogstruktur bzw. *Werbemittelstruktur* darstellt und nicht die *tatsächliche* Produkthierarchie. In der Praxis wird sich die Katalogstruktur naturgemäß stark an der Produktstruktur orientieren, so daß fast immer eine Übereinstimmung zwischen Werbemittelhierarchie und Produkthierarchie vorhanden sein wird. Bei der Berücksichtigung von Sonderinformationen wie die *Einführungsseite 1* kommt dieser Unterschied jedoch voll zur Geltung.

Die Interpretationsfähigkeit der einzelnen Hierarchieknoten durch die Publishing - Software ermöglicht eine sehr hohe optische Flexibilität des Endproduktes. Im folgenden sind in dem Begriff *Produktgruppe* auch immer die *Sonderseiten* enthalten.

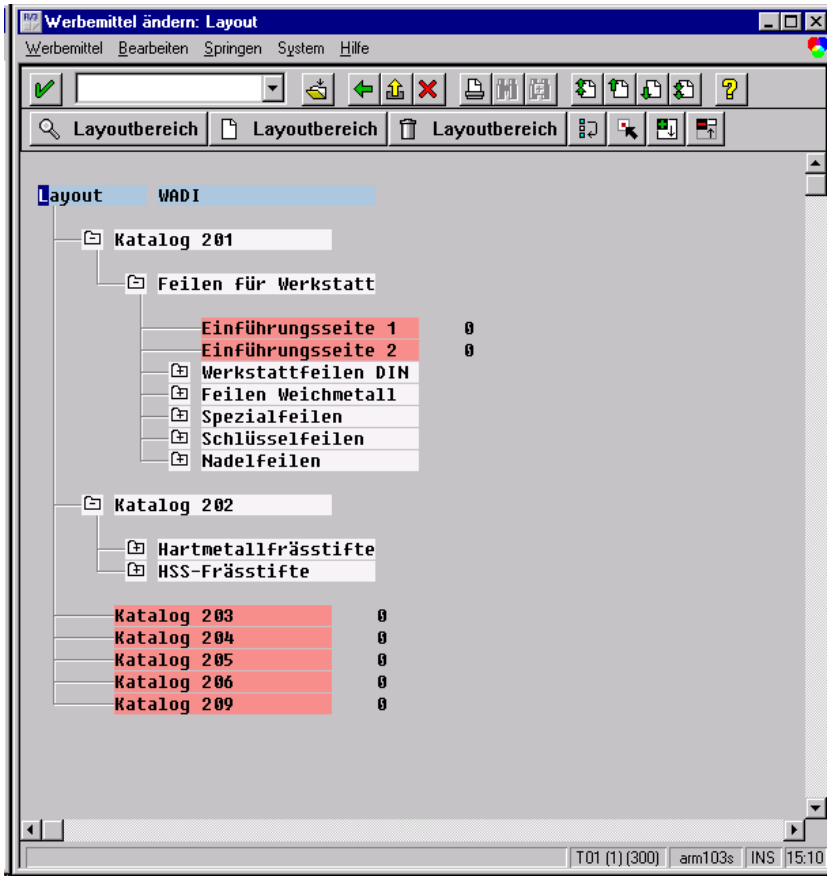


Bild: Katalogstruktur August Rüggeberg GmbH & Co.

3.

4. Aufnahme einer Produktgruppe

Um eine neue Produktgruppe anzulegen sind folgende Schritte notwendig:

- Cursor auf der Produktgruppe positionieren, für die eine Untergruppe angelegt werden soll; z.B. FEILEN FÜR DIE WERKSTATT.

- Menü BEARBEITEN / LAYOUTBEREICH ANLEGEN wählen und UNTERGEORDNET ALS ERSTER anklicken. Es ergibt sich folgendes Dialogfenster, das mit OK (grüner Haken) bestätigt wird:

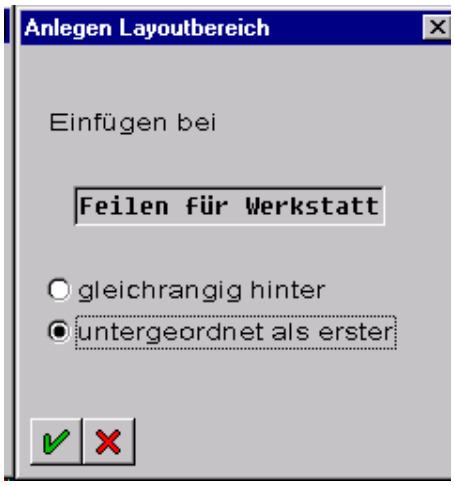


Bild: Neue Hierarchieebene in *Werbemittel* aufnehmen

Es erscheint das Bearbeitungsfenster für eine Produktgruppe. In BEREICH wird der Kurzname der Produktgruppe eingetragen. Dieser Name dient lediglich der internen Benennung der Produktgruppe. Die *offizielle* Produktgruppenbezeichnung, wie sie im Katalog erscheint, wird im nächsten Schritt vergeben.

Bei Produktgruppen, die am äußersten Ende der Hierarchiestruktur hängen, können in diesem Dialogfenster die Produkte dieser Produktgruppe einzeln aufgenommen werden. Bei Produktgruppen, die noch weitere Produktgruppen unter sich haben, ist keine Artikelzuordnung möglich und es erscheinen keine entsprechenden Eingabefelder POSITIONEN.

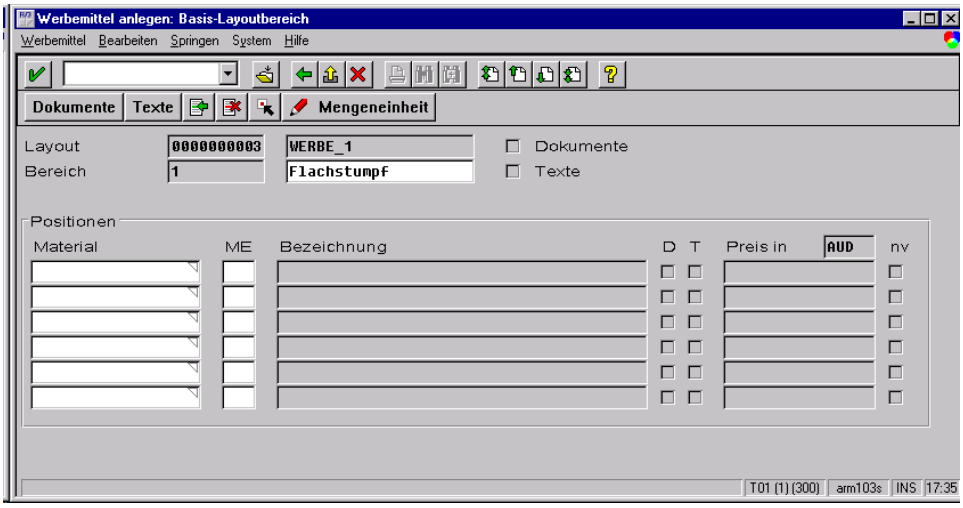


Bild: Daten einer *Werbemittel* - Hierarchieebene

Mit SPRINGEN / TEXTE / ZUM LAYOUTBEREICH können die erweiterten Texte einer Produktgruppe editiert werden. Für jede eingestellte Sprachvariante (siehe *Werbemittelvariante*) erscheint ein eigener Textblock, wo für jede Landessprache eine ÜBERSCHRIFT und ein LANGTEXT eingetragen werden kann. Der Text in der ÜBERSCHRIFT ist die Produktgruppenbezeichnung, die auch im Katalog erscheint.

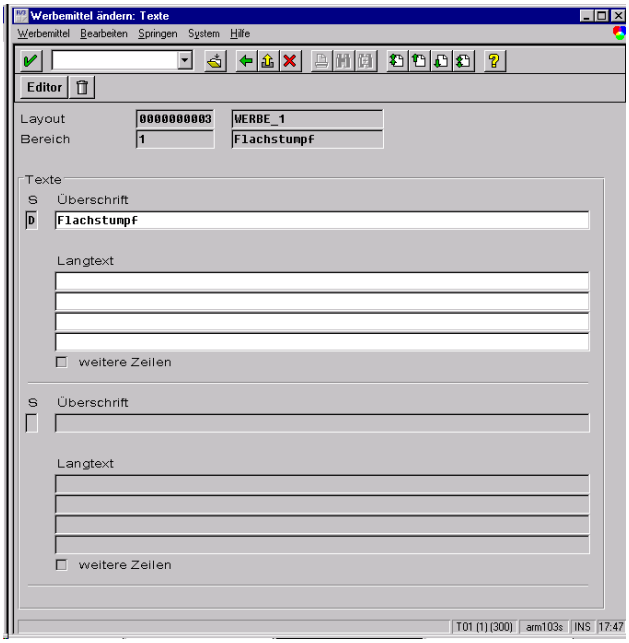


Bild: Mehrsprachige Texte einer *Werbemittel* – Hierarchieebene.

1.

2. Zuordnung der Artikel in die Produkthierarchie im *Werbemittel*.

Den einzelnen Produktgruppen werden die entsprechenden Artikel einzeln zugeordnet. Es können nur den jeweils untersten Hierarchieknoten Artikel hinzugefügt werden.

Bild: Artikel der Produktgruppe *Katalog 201 / Feilen für die Werkstatt / Werkstattfeilen DIN / Flachstumpf*

3.

4. Aufnahme einer Klassengruppe PRODKAT

Aus organisatorischen Gründen werden alle Klassen, die im Zusammenhang mit den Werbemitteln angelegt werden, in einer Klassengruppe *PRODKAT* zusammengefaßt. Diese Klassengruppe muß vor dem Anlegen einer Klasse bereits existieren. Klassengruppen werden im CUSTOMIZING angelegt.

Über WERKZEUGE / BUSINESS ENGINEERING / CUSTOMIZING gelangt man in das R/3 Customizing. Mit GRUNDFUNKTIONEN / UNTERNEHMENS-IMG / ANZEIGEN wird das Unternehmens-IMG, das zentrale R/3 Customizing – Werkzeug,

angezeigt.

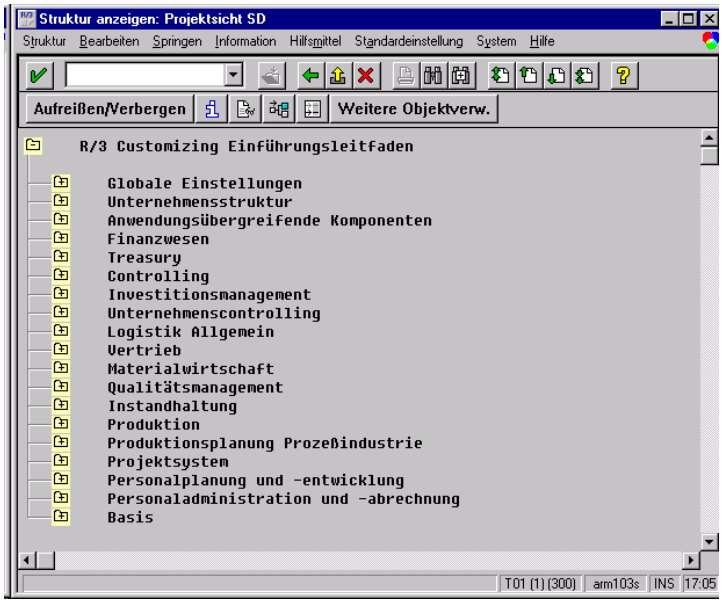


Bild: Unternehmens-IMG

Durch das schrittweise Aufklappen der entsprechenden Hierarchieknoten mit der Maus gelangt man an den Knoten KLASSENGRUPPEN DEFINIEREN.

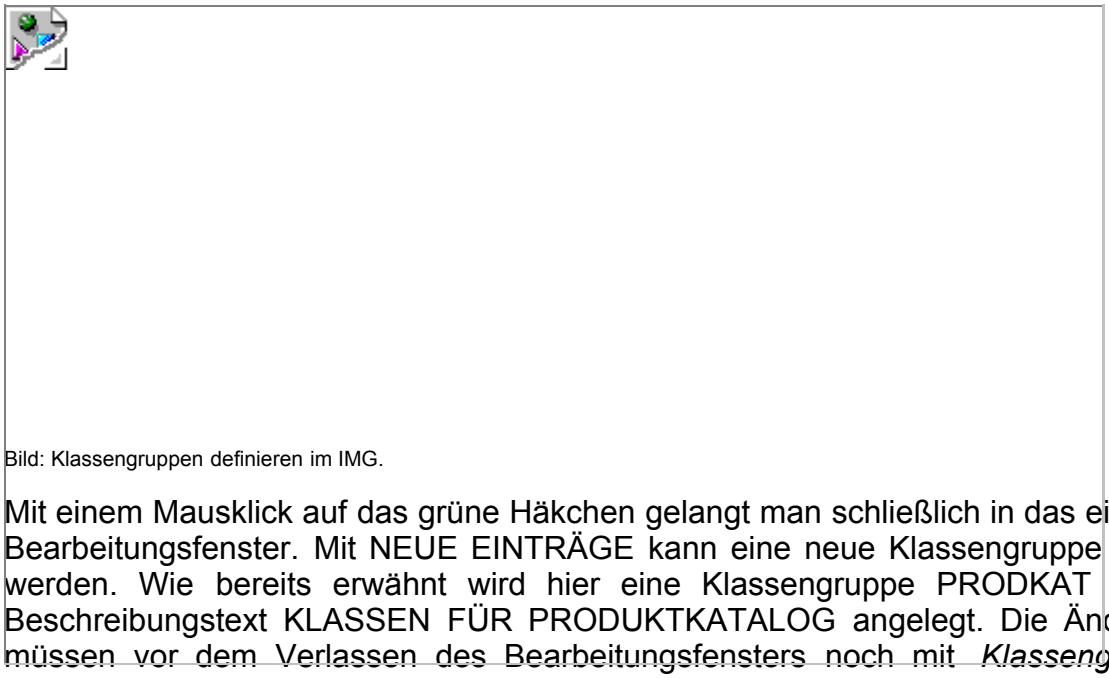


Bild: Klassengruppen definieren im IMG.

Mit einem Mausklick auf das grüne Häkchen gelangt man schließlich in das eigentliche Bearbeitungsfenster. Mit NEUE EINTRÄGE kann eine neue Klassengruppe angelegt werden. Wie bereits erwähnt wird hier eine Klassengruppe PRODKAT mit dem Beschreibungstext KLASSEN FÜR PRODUKTKATALOG angelegt. Die Änderungen müssen vor dem Verlassen des Bearbeitungsfensters noch mit *Klassengruppen / Sichern* gesichert werden.

Bild: Klassengruppe aufnehmen

5.

6. Aufnahme einer Klasse *WMG_BASIS*

Es wird eine Klasse mit dem Namen *WMG_BASIS* angelegt, von der alle Produktgruppen - Klassen abgeleitet (vererbt) werden.

Klassenname : WMG_BASIS

Klassenart : 001 (MATERIALKLASSE)

Bezeichnung : Basisklasse Produktgruppen

Klassengruppe: PRODKAT

Dies erleichtert die Anlage von allgemeingültigen Merkmalen, die ausnahmslos in allen Produktgruppen vorhanden sein müssen. Für das Merkmal LAYOUTSCHLÜSSEL trifft diese Forderung zu. An dieser Stelle erfolgt eine Begriffsdefinition, die für das Verständnis der folgenden Zusammenhänge unabdingbar ist:

Hinweis:

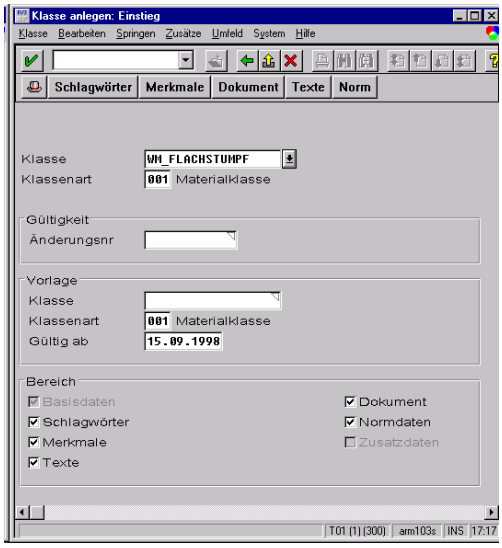
Produktgruppe bezeichnet das Objekt *Produktgruppe* (und nicht die Artikel in einer Produktgruppe). Diese Objekte wird durch eine Hierarchieebene in den *Werbemitteln* dargestellt. Die Produkte, die in einer Produktgruppe zusammengefaßt werden, werden *Produktgruppen - Position* genannt.

7.

8. Aufnahme von Klassen für Produktgruppen aus *Werbemittel*

Für jede Produktgruppe (auch Gruppen wie *Einführungsseite 1*), die spezifische Merkmale hat, muß eine eigene Klasse im Klassensystem angelegt werden. Theoretisch kann jede Produktgruppe mindestens ein spezifisches Merkmal haben, so daß ungefähr so viele Klassen im Klassensystem angelegt würden, wie Produktgruppen (Hierarchieebenen) in der Werbemittelhierarchie existieren. Bei Vorhandensein von zwei oder mehr Produktgruppen, die exakt dieselben Merkmale verwenden, ist eine gemeinsame Klasse für diese Gruppe von Produktgruppen ausreichend. An dieser Stelle ist genau zwischen den Klassen einer Produktgruppe und den Klassen der Produktgruppen - Positionen zu unterscheiden. Produktgruppen haben andere Merkmale als Produktgruppen - Positionen, so daß sie auch jeweils eigene Klassen haben. Hier werden die Klassen und Merkmale der PRODUKTGRUPPEN gemeint.

Als Klassenart ist immer 001 = MATERIALKLASSE zu wählen. Als Klassengruppe wird immer PRODKAT eingetragen. Alle Klassen, die im Zusammenhang mit den *Werbemitteln* angelegt werden, werden durch ein *WMx_* am Namensanfang von den Klassen unterschieden, die im *produktiven* Einsatz sind. Dies ist eine feste Namenskonvention dieses Konzeptes, die unbedingt eingehalten werden sollte. Für die Produktgruppe FLACHSTUMPF wird somit eine Klasse *WMG_FLACHSTUMPF* angelegt. WM bedeutet dabei *Werbemittel*, das G drückt die Zugehörigkeit zu einer Produktgruppe aus. Ist die verfügbare Länge des Klassennamens nicht ausreichend, sind sinngemäße Abkürzungen zulässig.



Einstiegsbild: Neue Klassen anlegen

Im Feld BEZEICHNUNG einer Klasse muß der korrekte Name der Produktgruppe eingetragen werden, wie er auch im Modul *Werbemittel* als KURZNAME im Eingabefeld BEREICH verwendet wird, inkl. der exakten Schreibweise (Groß-/Kleinschreibung, Leerzeichen etc.). Die Übereinstimmung ist ein Verknüpfungsindikator zwischen der Hierarchieebene im Modul *Werbemittel* und einer Klasse des Klassensystems.

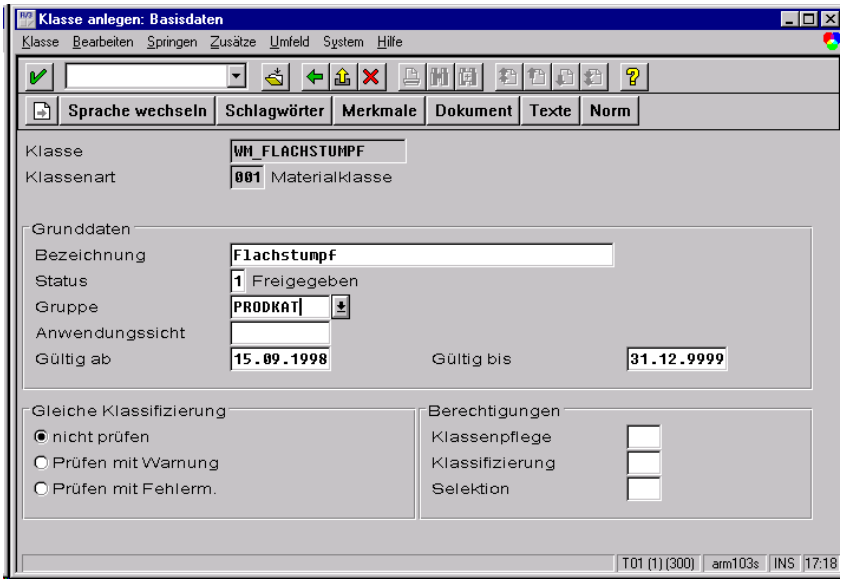


Bild: Basisdaten einer Klasse

9.

10. **Produktgruppenklassen der Basisklasse unterordnen (vererben)**

Im Klassensystem kann mit ZUORDNUNG / PFLEGEN / KLASSE ZU KLASSEN jede angelegte Produktgruppenklasse der Klasse *WMG_BASIS* untergeordnet werden. Die Unterordnung alle *Werbemittel* bezogenen Produktgruppenklassen unter die Klasse WMG_BASIS ist eine feste Konvention.

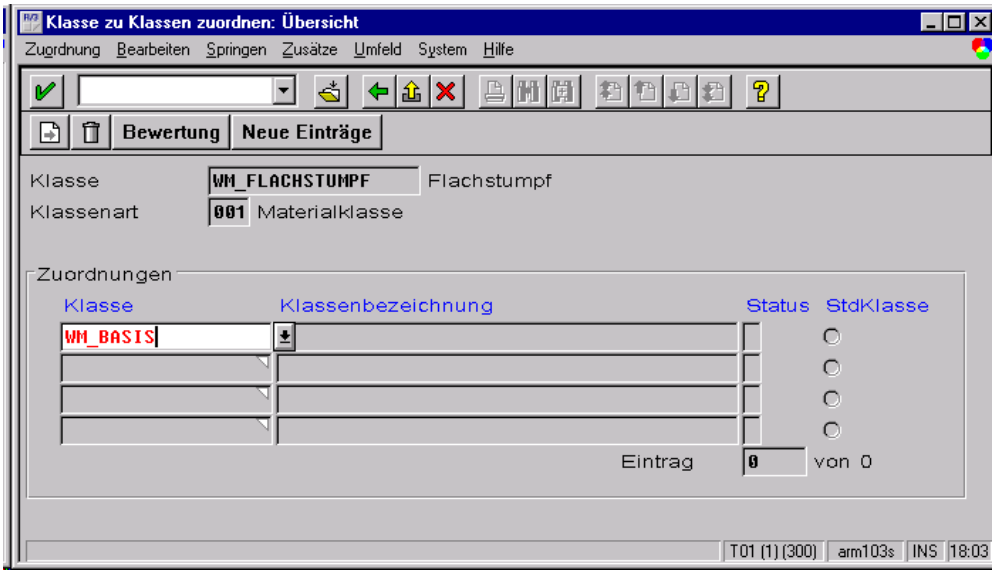


Bild: Klassenhierarchie aufbauen

Hinweis:

Im Bild ist erkennbar, daß einer Klasse unter *Zuordnungen* mehrere übergeordnete Klassen zugeordnet werden können. Dies erlaubt den Aufbau von *Klassennetzen*.

11. Aufnahme einer Klasse *WMA_BASIS*

Es wird eine Klasse mit dem Namen *WMA_BASIS* angelegt, von der alle Produktgruppen – Positions - Klassen abgeleitet (vererbt) werden.

Klassenname : WMA_BASIS

Klassenart : 001 (MATERIALKLASSE)

Bezeichnung : Basisklasse Produktgruppen - Positionen

Klassengruppe: PRODKAT

Dies erleichtert die Anlage von allgemeingültigen Merkmalen, die ausnahmslos für alle Produktgruppen - Positionen (d.h. Materialien) vorhanden sein müssen.

12.

13. Aufnahme von Klassen für Produktgruppen - Positionen.

Für jede Produktgruppe, die sich am äußersten Ende des Hierarchiebaumes befindet und deren Positionen spezifische Merkmale haben, muß eine Klasse im Klassensystem angelegt werden. Theoretisch können die Positionen jeder Produktgruppe mindestens ein spezifisches Merkmal haben, so daß so viele Klassen im Klassensystem angelegt würden, wie es *äußere* (Blätter des Hierarchiebaumes) Produktgruppen in der *Werbemittelhierarchie* gibt.

Als Klassenart ist immer 001 = MATERIALKLASSE zu wählen. Als Klassengruppe wird immer PRODKAT eingetragen. Alle Klassen, die im Zusammenhang mit den *Werbemitteln* für die Produktkatalog - Positionen angelegt werden, werden durch ein *WMA_* am Namensanfang von anderen Klassen unterschieden. Dies ist eine feste Namenskonvention dieses Konzeptes, die unbedingt eingehalten werden sollte. Für

die Produktgruppen - Positionen der Produktgruppe *FLACHSTUMPF* wird somit eine Klasse *WMA_FLACHSTUMPF* angelegt. Ist die verfügbare Länge des Klassennamens nicht ausreichend, sind sinngemäße Abkürzungen zulässig.

14. Aufnahme der Merkmalgruppe PRODKAT.

Analog zur Klassengruppe PRODKAT werden aus organisatorischen Gründen alle Merkmale, die im Zusammenhang mit den *Werbemitteln* angelegt werden, in einer Merkmalgruppe *PRODKAT* angelegt. Diese Merkmalgruppe muß vor dem Anlegen eines Merkmals bereits existieren. Merkmalgruppen werden analog zu den Klassengruppen im CUSTOMIZING angelegt. Die *Klassengruppe* PRODKAT und die *Merkmalgruppe* PRODKAT sind zwei völlig verschiedene Gruppen, die lediglich aus Gründen der Vereinheitlichung gleich benannt werden.

Über WERKZEUGE / BUSINESS ENGINEERING / CUSTOMIZING gelangt man in das R/3 Customizing.

Mit GRUNDFUNKTIONEN / UNTERNEHMENS-IMG / ANZEIGEN wird das Unternehmens-IMG angezeigt. Durch das schrittweise Aufklappen der entsprechenden Hierarchieknoten mit der Maus gelangt man an den Knoten MERKMALGRUPPEN DEFINIEREN.

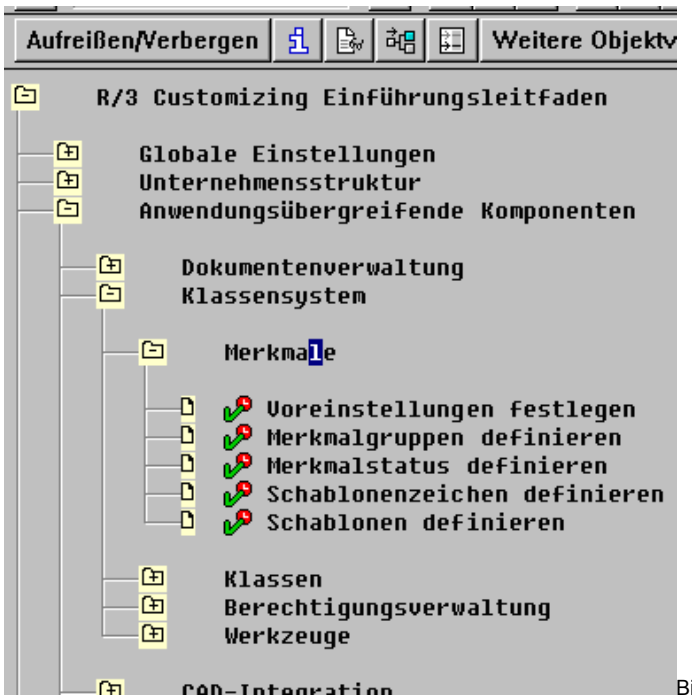


Bild: Merkmalgruppen definieren im IMG.

Mit einem Mausklick auf das grüne Häkchen gelangt man schließlich in das eigentliche Bearbeitungsfenster. Mit NEUE EINTRÄGE kann eine neue Merkmalgruppe angelegt werden. Wie bereits erwähnt, wird hier eine Merkmalgruppe PRODKAT mit dem Beschreibungstext MERKMALE FÜR PRODUKTKATALOG und eine Merkmalgruppe PRODKAT_M mit dem Beschreibungstext MERKMALE FÜR PRODUKTKATALOG-MERKMALE angelegt. Der Merkmalgruppe PRODKAT werden alle Merkmale der Produktgruppen, die im Zusammenhang mit der *Werbemittel*planung angelegt werden, zugeordnet. Der Merkmalgruppe PRODKAT_M werden alle *internen Merkmale* zugeordnet. Interne Merkmale werden im weiteren Verlauf näher erläutert.

Die Änderungen müssen vor dem Verlassen des Bearbeitungsfensters noch mit *Merkmalsgruppen sichern* gesichert werden.

Bild: Merkmalgruppen im IMG definieren

15. Aufnahme einer Klasse *WMM_BASIS*

Bei einer großen Produktvielfalt eines Unternehmens wie der August Rüggeberg GmbH & Co kann sich die Anzahl der verschiedenen Produktmerkmale auf mehrere Hundert summieren. Außerdem erhöht ein individuell auf Produktgruppen abgestimmtes Kataloglayout auch die Anzahl der Merkmale, die zur Beschreibung des entsprechenden Layouts notwendig sind. Im allgemeinen dürfte ein einfaches Layout z.B. mit folgenden Merkmalen beschrieben werden können:

LAYOUTSCHLUESSEL, BILD 1, BILD 2, GRUPPENTEXT, TABELLENSPALTE_1, TABELLENSPALTE_2, TABELLENSPALTE_3, FUSSNOTE, TEXT_1, TEXT_2

Um die Übersicht über die angelegten Merkmale nicht zu verlieren und langfristig eine Strukturierung und Organisation der Merkmale zu erreichen, müssen diese Merkmale von Beginn an planmäßig verwaltet werden. Die Existenz einer Klassenart MERKMALE und der Klasse 012 MERKMALKLASSE in SAP R/3 bestätigt die Notwendigkeit der Organisation von angelegten Merkmalen. Die Verwaltung von vielen heterogenen Merkmalen erfordert im Prinzip die gleichen Schritte, wie die Verwaltung jeder anderen Objektart. Die Merkmale werden nach gleichartigen Merkmalen gruppiert und in einer Klasse zusammengefaßt. Anfangs erscheint die Verwaltung von Merkmalen mit Hilfe von anderen Merkmalen ein wenig verwirrend, was sich jedoch bei der Umsetzung schnell als selbsterklärend auflöst. Aus organisatorischen Gründen wird eine Klasse mit dem Namen *WMM_BASIS* angelegt, von der alle Klassen zur Verwaltung der Produktgruppenmerkmale abgeleitet (vererbt) werden. Dies erleichtert die Anlage von Merkmalen, die ausnahmslos in allen Klassen zur Merkmalverwaltung vorhanden sein müssen.

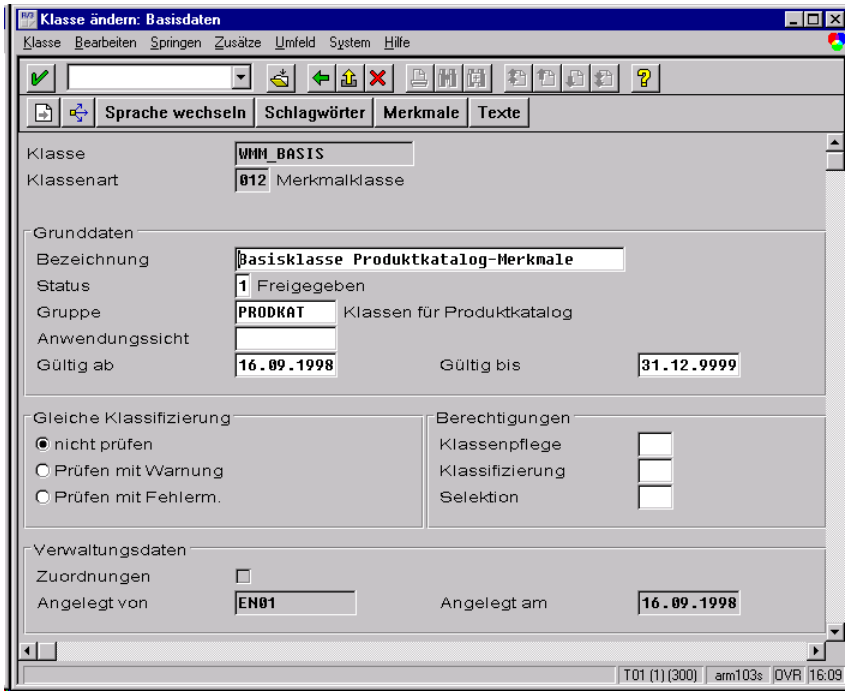
Die Existenz einer Basisklasse für alle anderen Klassen ist auch für die Anlage von *internen Merkmalen* notwendig (näheres dazu folgt später).

Klassenname : WMM_BASIS

Klassenart : 012 (MERKMALKLASSE)

Bezeichnung : Basisklasse Produktkatalog - Merkmale

Klassengruppe : PRODKAT



16.

17. Aufnahme der internen Merkmale und Zuordnung an Klasse *WMM_BASIS*

Der Begriff INTERNES MERKMAL ist eine Definition des Autors. Interne Merkmale sind Merkmale, die nicht direkte Produktivinformationen (wie die Produktgruppenmerkmale) beinhalten, sondern intern von dieser Lösung, respektive dem Export - Programm, als Steuerzeichen verwendet und interpretiert werden (Steuerdaten). Dementsprechend treten diese Merkmale dem Anwender gegenüber gar nicht als Merkmal auf. Interne Merkmale sind funktionsbezogen und können bei funktionalen Erweiterungen ergänzt werden, d.h. es können bei Bedarf zusätzliche interne Merkmale hinzukommen.

Das *interne* Merkmal MERKMALWERT ist ein Merkmal der Produktgruppenmerkmale und der Produktgruppen - Positionsmerkmale . Es wird der Klasse WMM_BASIS zugewiesen, so daß es automatisch für alle Merkmale der Anwendung verfügbar ist, da für die Produktgruppenmerkmale und Produktgruppen - Positionsmerkmale die feste Konvention gilt, daß diese in der Klasse WMM_BASIS klassifiziert sein müssen. Die internen Merkmale selber werden nicht klassifiziert. Das interne Merkmal MERKMALWERT hat die beiden Werte STANDARD und DVS_VERWEIS und steuert die Interpretation eines Produktgruppen - Merkmalwertes. Bei STANDARD wird der Wert im Merkmal übernommen, bei DVS_VERWEIS enthält das betroffene Merkmal den Namen eines Dokumentes aus dem R/3 Dokumentenverwaltungssystem.

Im Klassensystem kann mit MERKMAL / ANLEGEN ein neues Merkmal angelegt werden.

Das interne Merkmal MERKMALWERT hat folgende Eigenschaften:

Merkmalname : MERKMALWERT

Merkmalbezeichnung : Merkmalwert

Merkmalgruppe : PRODKAT

Status : 1

Eingabe erforderlich : JA

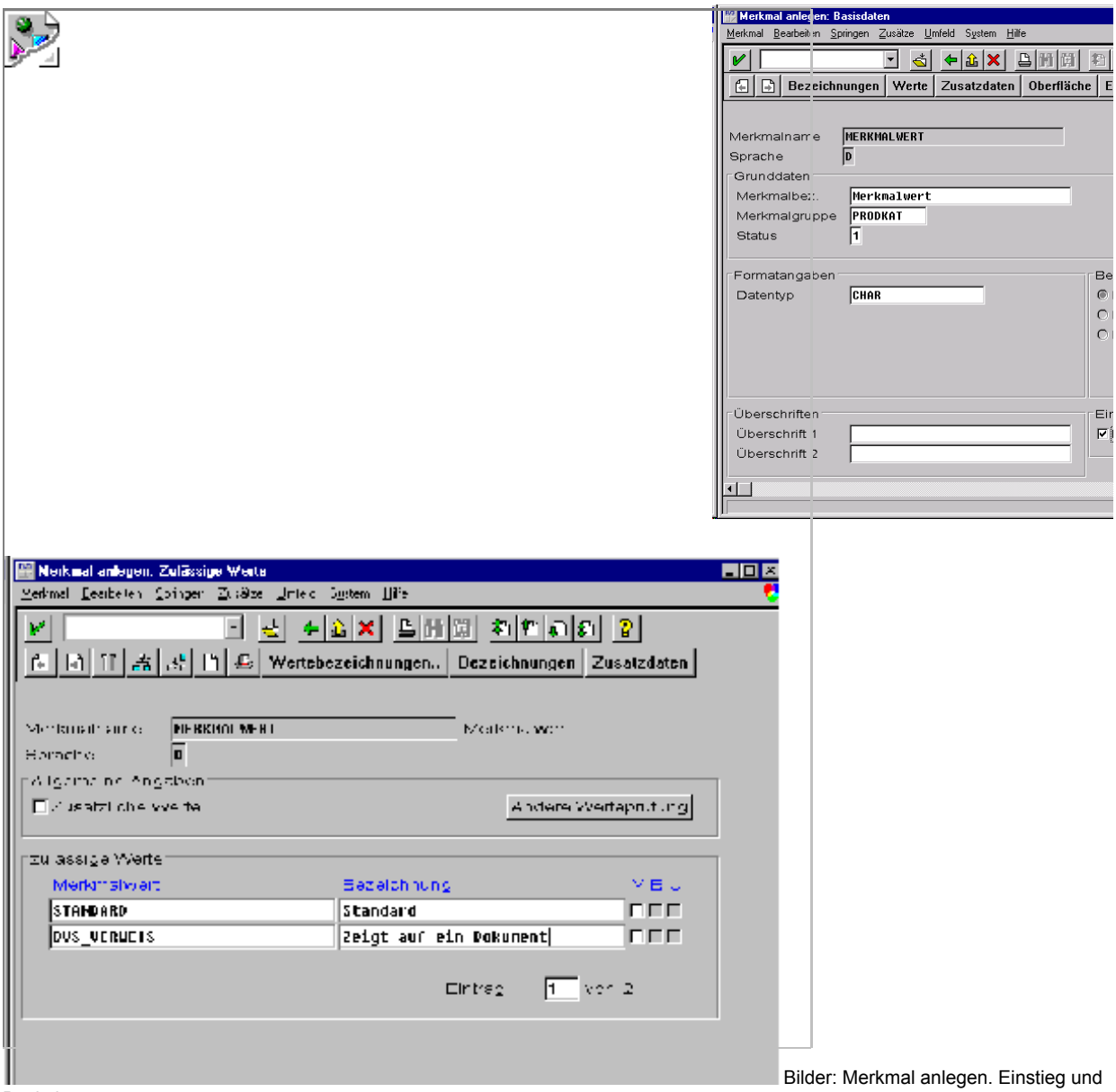
Datentyp : CHAR

Anzahl Stellen : 30

Bewertung : Einwertig

Werte : STANDARD; DVS_VERWEIS

Klassifizierung : keine



Bilder: Merkmal anlegen. Einstieg und Basisdaten

Bild: Werte eines Merkmals definieren

18. Merkmal *LAYOUTSCHLUESSEL* an Klasse *WMG_BASIS*

Da jede Produktgruppe definitiv einen Layoutschlüssel zugewiesen bekommt, über den die Publishing - Software die zu verwendende Layout - Vorlage (visuelle Darstellung) für die entsprechende Produktgruppe identifiziert, wird dieses Merkmal

der Klasse WMG_BASIS zugewiesen. Damit wird dieses Merkmal an alle, von der Klasse WMG_BASIS abgeleiteten, Klassen weitergegeben. Da laut Konventionen alle Produktgruppenklassen von dieser Klasse abgeleitet werden müssen, ist die Datenkonsistenz gewährleistet.

Das Merkmal LAYOUTSCHLUESSEL hat folgende Eigenschaften:

Merkmalname : LAYOUTSCHLUESSEL

Merkmalbezeichnung : Layoutschlüssel

Merkmalgruppe : PRODKAT

Status : 1

Eingabe erforderlich : JA

Datentyp : CHAR

Anzahl Stellen : 30

Bewertung : Einwertig

Klassifizierung : WMM_BASIS

Merkmalwerte : Der Wertevorrat des Merkmals LAYOUTSCHLUESSEL entspricht

den Namen (Schlüsseln) der verfügbaren Layoutvorlagen

(Templates) in der verwendeten Publishing - Software.

Bei der Eingabe der Werte für dieses Merkmal ist auf eine exakte Übereinstimmung der Schreibweise mit der Publishing-Software zu achten. LAYOUTSCHLUESSEL ist ein zu interpretierendes Merkmal, das keine direkten Kataloginformationen enthält, sondern ausschließlich durch die Publishing-Software ausgewertet wird. Durch die Zuordnung an die Klasse WMG_BASIS existiert es automatisch in allen Produktgruppen.

19. Aufnahme der Merkmale einer Produktgruppe und Zuordnung in Klassen.

Jedes spezifische Merkmal einer Produktgruppe muß einmal zentral angelegt werden. Merkmale, die in verschiedenen Produktgruppen verwendet werden, müssen nur einmal aufgenommen werden. Als Merkmal werden sowohl direkte Produktgruppeninformationen (Bilder, Texte) als auch interpetationsfähige Produktgruppeninformationen (Steuerinformationen) angelegt. Die Merkmale dieser Klasse beschreiben die Eigenschaften der PRODUKTGRUPPE, nicht die Eigenschaften der Artikel in dieser Produktgruppe (Produktgruppen - Positionen) .

Die Produktgruppe wird somit als eigenständiges Objekt betrachtet und hat somit auch eigene Merkmale. Da in dieser Anwendung eine rein katalogorientierte Sichtweise auf die Produktgruppen im Vordergrund steht, werden die Merkmale einer Produktgruppe ausschließlich den Charakter von Layout - Beschreibungsmerkmalen haben (Ausnahmen bestätigen auch hier die Regel).

Beispiel:
LAYOUTSCHLUESSEL, BILD 1, BILD 2, GRUPPENTEXT, TABELLENSPALTE_1, TABELLENSPALTE_2, TABELLENSPALTE_3, FUSSNOTE, TEXT_1, TEXT_2

20. **Merkmal *MATERIALKLASSENVORLAGE* an Klasse *WMG_BASIS***

Das Merkmal MATERIALKLASSENVORLAGE ist ein internes Merkmal und speichert für jede Produktgruppenklasse den Namen der Klasse, in der die Produkte einer Produktgruppe (Produktgruppen - Positionen) klassifiziert werden sollen. Die Klassifizierung der Produktgruppen - Positionen (Materialien) bestimmt wiederum die Merkmale, die ein Material einer Produktgruppe hat. In R/3 wird grundsätzlich jedes Objekt und damit auch jedes Material einzeln klassifiziert. Durch die Zuweisung einer Standardklasse für eine Produktgruppe kann dieser Vorgang automatisiert werden, indem mit der Zuordnung eines Materials in eine Produktgruppe (Hierarchieebene der *Werbemittel*) dieses Material auch in der entsprechenden Klasse klassifiziert wird. Der Wert dieses Merkmals ist immer der Name einer WMA_xxxxxxx - Klasse.

Das Merkmal MATERIALKLASSENVORLAGE hat folgende Eigenschaften:

Merkmalname : MATERIALKLASSENVORLAGE

Merkmalbezeichnung : Klasse für Produktgruppen - Positionen

Merkmalgruppe : PRODKAT

Status : 1

Eingabe erforderlich : JA

Datentyp : CHAR

Anzahl Stellen : 30

Bewertung : Einwertig

Klassifizierung : keine

Werte : Der Wertevorrat des Merkmals MATERIALKLASSENVORLAGE

entspricht den Namen der verfügbaren Produktgruppenpositions –

Klassen. Dies sind alle Klassen, deren Namen laut

Namenskonvention mit WMA_ beginnen.

Bei der Eingabe der Werte für dieses Merkmal ist auf eine exakte Übereinstimmung der Schreibweise der Klassennamen zu achten.

21.

22. Aufnahme einer Warengruppe PRODKAT

Aus organisatorischen Gründen werden alle Materialien, die im Zusammenhang mit den *Werbemitteln* angelegt werden in einer *Warengruppe* PRODKAT angelegt. Die anzulegenden Materialien werden als Dummy - Artikel ausschließlich internen Zwecken dienen (siehe unten). Diese Warengruppe muß vor dem Anlegen eines

Materials bereits existieren. Warengruppen werden im CUSTOMIZING angelegt.

Hinweis:

Eine *Warengruppe* dient, analog zu der *Klassengruppe* und der *Merkmalgruppe*, der

besseren Verwaltung und Übersichtlichkeit des Materialstamms. Mit diesen Gruppenarten sind keine direkten R/3 Funktionalitäten verknüpft (bis auf z.B. warengruppenbezogene Auswertungen etc.). Die *Warengruppe* ist nicht mit der *Materialgruppe* zu verwechseln. Die *Materialgruppe* faßt Materialien mit gleichen Preiskonditionen zusammen.

Über WERKZEUGE / BUSINESS ENGINEERING / CUSTOMIZING gelangt man in das R/3 Customizing.

Mit GRUNDFUNKTIONEN / UNTERNEHMENS-IMG / ANZEIGEN wird das Unternehmens-IMG, das zentrale R/3 Customizing - Werkzeug angezeigt. Durch das schrittweise Aufklappen der entsprechenden Hierarchieknoten mit der Maus gelangt man an den Knoten WARENGRUPPEN DEFINIEREN.

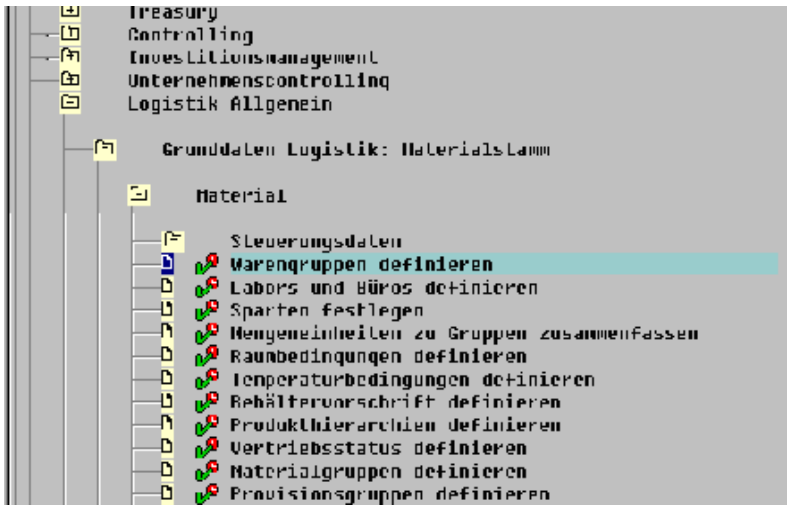


Bild: Warengruppen definieren im IMG.

Mit einem Mausklick auf das grüne Häkchen gelangt man schließlich in das eigentliche Bearbeitungsfenster. Mit NEUE EINTRÄGE kann eine neue Warengruppe angelegt werden. Wie bereits erwähnt wird hier eine Warengruppe PRODKAT mit dem Beschreibungstext PRODUKTKATALOG angelegt. Die Änderungen müssen vor dem Verlassen des Bearbeitungsfensters noch mit *Tabellensicht sichern* gesichert werden.

Bild: Warengruppen definieren

23. Dummy - Material für jede Hierarchieebene von *Werbemittel*.

In den bisherigen Schritten wurde u.a. die Hierarchieebene aufgebaut, und die Produktgruppen- und Produktgruppenmerkmale bestimmt, indem für jede Produktgruppe eine Klasse erstellt und diesen die entsprechenden Merkmale zugewiesen wurden. Um diese Produktgruppenmerkmale auch bewerten zu können, müßte jede Werbemittel - Hierarchieebene in der entsprechenden Klasse klassifiziert werden. R/3 erlaubt jedoch nicht die Klassifizierung einer einzelnen Hierarchieebene eines *Werbemittel*s. Dieser Umstand wird hier umgangen, indem für jede *Werbemittel* - Hierarchieebene ein eigenes Material im Materialstamm angelegt wird. Dieses Material wird in der Klasse der korrespondierenden Werbemittel - Hierarchieebene klassifiziert, so daß die Merkmale der entsprechenden Produktgruppe dem Material zugeordnet werden. Die Bewertung der Produktgruppenmerkmale findet somit indirekt über die Bewertung der Merkmale des Dummy - Materials statt, das eine Produktgruppe repräsentiert.

Über LOGISTIK / MATERIALWIRTSCHAFT / MATERIALSTAMM kann der R/3 Materialstamm gepflegt werden. Mit MATERIAL / ANLEGEN ALLGEMEIN / SOFORT wird ein neues Material aufgenommen. Alle Materialien, die im Zusammenhang mit den *Werbemittel*n angelegt werden, werden durch ein *WMH_* am Namensanfang von den Materialien unterschieden, die im *produktiven* Einsatz sind. Dies ist eine feste Namenskonvention, die unbedingt eingehalten werden sollte. Für die Produktgruppe *FLACHSTUMPF* wird somit ein Material *WMH_FLACHSTUMPF* angelegt. Ist die verfügbare Länge des Materialnamens nicht ausreichend, sind sinngemäße Abkürzungen zulässig.

Für die Materialpflege sind nur die Grunddaten und die Klassifizierung relevant.

Als Grunddaten sind lediglich die Angaben WARENGRUPPE und KURZTEXT (dies ist die Materialbezeichnung) notwendig. Bei Warengruppe wird immer PRODKAT eingetragen. Im Feld KURZTEXT eines Materials muß der Text *Produktgruppe:* gefolgt von dem korrekten Namen der Produktgruppe eingetragen werden, wie er auch im Modul *Werbemittel* als KURZNAME im Eingabefeld BEREICH einer Hierarchieebene verwendet wird, inkl. der exakten Schreibweise (Groß-/Kleinschreibung, Leerzeichen etc.). Die Übereinstimmung ist ein Verknüpfungsindikator zwischen der Hierarchieebene im Modul *Werbemittel* und eines Dummy - Materials. Alle sonstigen Felder haben keinen Einfluß auf die Verarbeitung eines Dummy - Materials und

können mit beliebigen Standardwerten gefüllt werden.

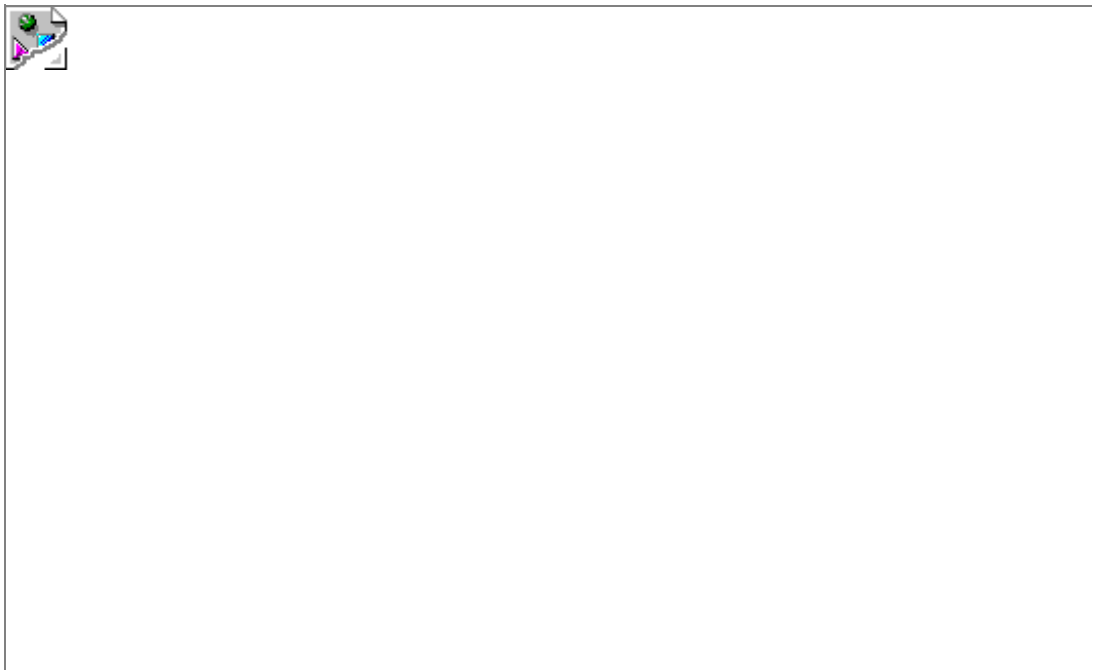


Bild: Grunddaten Material

Anschließend muß das neue Dummy - Material in der Klasse *seiner* Produktgruppe klassifiziert werden. Die Klassifizierung erfolgt immer in einer WMG_ - Klasse (Produktgruppenklassen).

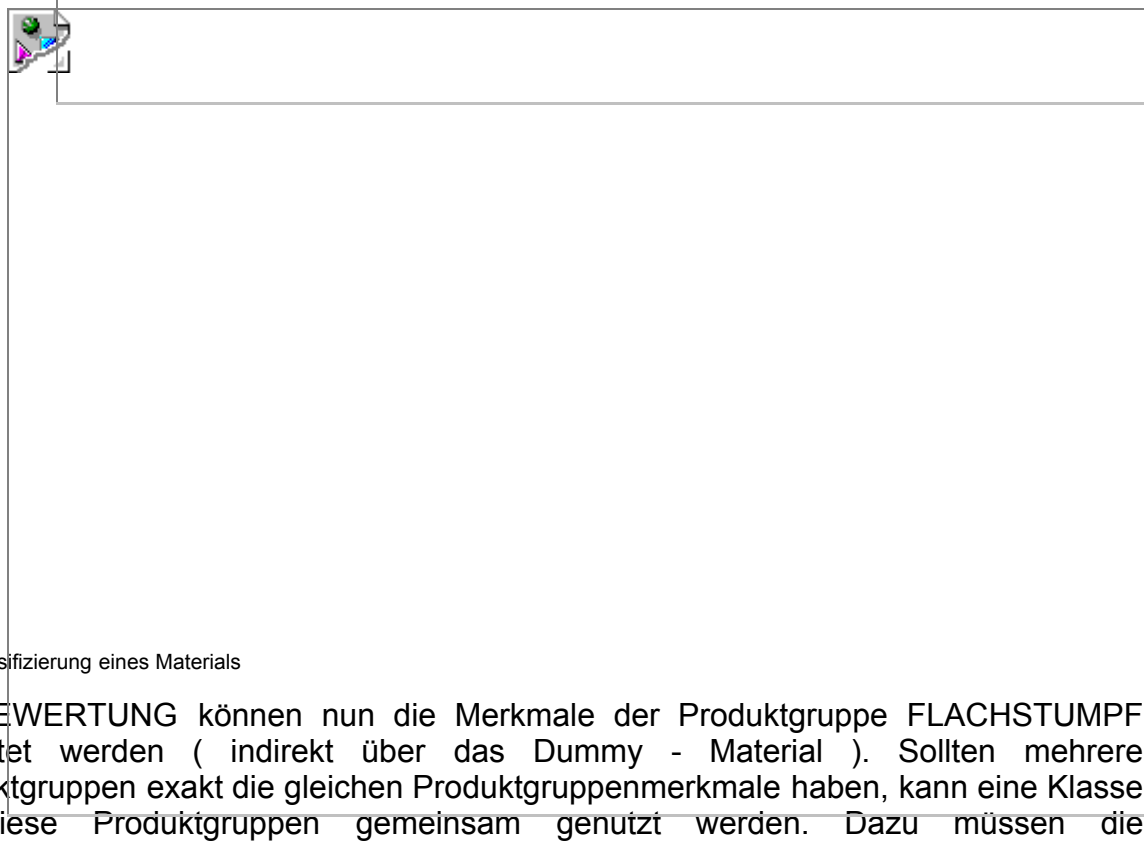


Bild: Klassifizierung eines Materials

Mit BEWERTUNG können nun die Merkmale der Produktgruppe FLACHSTUMPF bewertet werden (indirekt über das Dummy - Material). Sollten mehrere Produktgruppen exakt die gleichen Produktgruppenmerkmale haben, kann eine Klasse für diese Produktgruppen gemeinsam genutzt werden. Dazu müssen die Dummy-Artikel dieser Produktgruppen lediglich in der gleichen Klasse klassifiziert werden. In dem Merkmal *MATERIALKLASSENVORLAGE* wird der Name der Klasse angegeben, in der die Produktgruppen - Positionen klassifiziert werden.

24.

25. Aufnahme einer Klasse *WMD_BASIS*

In der R/3 - Dokumentenverwaltung werden alle Bilder und Texte (ab 30 Zeichen) eines Produktkataloges abgelegt. Über entsprechende Merkmale (Merkmale mit *internem Merkmalwert*: MERKMALWERT = DVS_VERWEIS) der Produktgruppen und Produktgruppen – Positionen erfolgt die Zuordnung zu einzelnen Dokumenten. Bei einer hohen Zahl von Dokumenten (mehrere Tausend), ist eine einfache listenartige Ablage nicht mehr praktikabel. Wie schon bei den Merkmalen, ist auch bei den Dokumenten eine strukturierte Dokumentenablage zu empfehlen. Auch hier kann das Allround - Talent des R/3 - Systems, das Klassensystem, mit einer eigenen Klassenart DOKUMENT und der Klasse 017 = DOKUMENTENVERWALTUNG aufwarten.

Analog zu den bisherigen Klassenstrukturen wird eine Klasse *WMD_BASIS* angelegt, von der alle Klassen zur Verwaltung der *Werbemittel* - Dokumente abgeleitet (vererbt) werden. Dies erleichtert die Anlage von Merkmalen, die in allen Klassen zur Dokumentenverwaltung enthalten sein sollen.

Und auch hier ist die Existenz einer Basisklasse für die Anlage von *internen Merkmalen* zur Dokumentenverwaltung notwendig.

Klassenname : WMD_BASIS

Klassenart : 001 (DOKUMENTENVERWALTUNG)

Bezeichnung : Basisklasse Produktkatalog - Dokumente

Klassengruppe : PRODKAT

26.

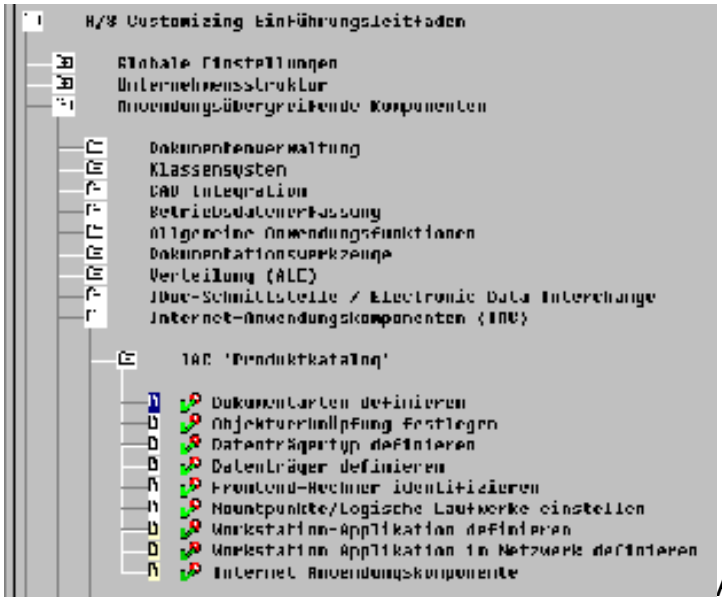
27. Konfiguration des R/3 - Dokumentenverwaltungsystems (DVS)

Um die eingegebenen Dokumente medienunabhängig (insbesondere *internetfähig*), in das R/3 System einzupflegen, sollten vor der Dateneingabe einige Konfigurationen überprüft werden. Die R/3 Internet - Anwendungkomponenten (IAC's) PRODUKTKATALOG und ONLINE-SHOP greifen u.a. auf das R/3 - DVS zu, um die zu den im Internet dargestellten Produkten (Materialien) passenden Bilder und Sounds (Klangeffekte) im Internet - Browser darzustellen. Die folgenden Schritte decken die erforderlichen Konfigurationen für die *internetfähige* Ablage der Daten in R/3 ab.

28. Dokumentenarten anlegen

Eine Dokumentart steuert u.a. das Eingabeverhalten der Pflegemasken eines Dokumentes im DVS. Es ist zu empfehlen, für ähnliche Dokumente eine eigene Dokumentart anzulegen, die in ihrem Eingabeverhalten den Erfordernissen zur Pflege einer bestimmten Dokumentart angepaßt sind. Außerdem haben nachträgliche Änderungen an einer *Dokumentart*, die nur bestimmte Dokumente betreffen sollen, keine unerwünschten Nebenwirkungen auf die Dokumente anderer Dokumentarten (z.B. *Zusatzfelder*, näheres Kapitel 1). Wenn dies schon vor der Aufnahme einer größeren Anzahl von Dokumenten berücksichtigt wird, können später viele Ärgernisse erspart bleiben.

Dokumentenarten werden im Customizing angelegt:



ANWENDUNGSÜBERGREIFENDE KOMPONENTEN / INTERNET - ANWENDUNGSKOMPONENTEN / IAC-PRODUKTKATALOG / DOKUMENTARTEN DEFINIEREN

Bild: Dokumentarten definieren. IMG

Mit einem Klick auf das grüne Häkchen erscheint das Fenster zur Pflege der Dokumentarten:

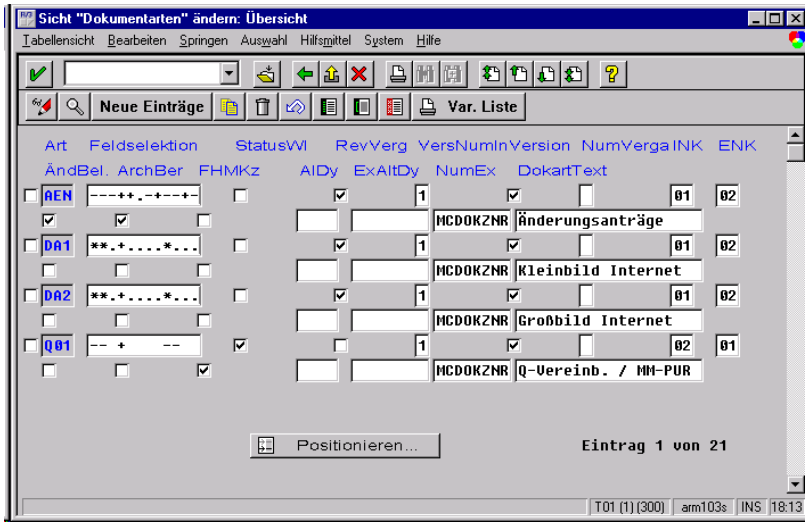


Bild: Dokumentarten definieren im IMG. Dokumentarten

Mit NEUE EINTRÄGE wird eine neue Dokumentart aufgenommen. Der 3-stellige Name der Dokumentart kann frei vergeben werden. Hier wird einfach von DA1 bis DAx hochgezählt.

Die beiden erwähnten R/3 – IAC's *Produktkatalog* und *Online - Shop* unterstützen je Produkt ein Kleinbild (i.d.R. ein *thumbnail*), ein Großbild und einen Sound. Dementsprechend wird für jede dieser Informationen eine eigene *Dokumentart* aufgenommen. Die Dokumentart DA4 ist nicht zwingend erforderlich. Sie wird jedoch im Rahmen dieses Konzeptes aus organisatorischen Gründen mit aufgenommen. Die IAC's unterstützen die Dokumentart DA4 nicht.

DA1: Kleinbild

DA2: Großbild

DA3: Sound

DA4: Produktkatalog – Text

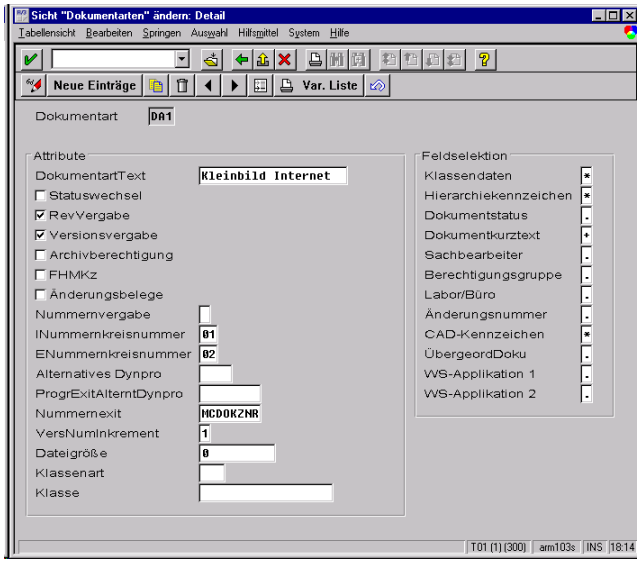


Bild: Dokumentarten definieren. Basisdaten

29.

30. R/3 - Objektverknüpfung definieren

Nachdem eine neue Dokumentart aufgenommen wurde, muß definiert werden, mit welchen R/3 - Objekttypen Dokumente dieser Dokumentart *verknüpft* werden können (*Dokumentverknüpfung* siehe Kapitel 1). Wenn Dokumente mit der Dokumentart DA1 z.B. mit einem *Material* oder einem *Debitor* verknüpft werden sollen, so muß dies explizit im Customizing eingestellt werden. Einige Standardverknüpfungen werden mit der Aufnahme einer neuen Dokumentart automatisch eingetragen.

Im Customizing - IMG unter ANWENDUNGSÜBERGREIFENDE KOMPONENTEN / INTERNET-ANWENDUNGSKOMPONENTEN / IAC-PRODUKTKATALOG / OBJEKTVERKNÜPFUNG FESTLEGEN können die Verknüpfungen eingetragen werden. Die im Bild hervorgehobenen Einträge wurden vom Autor manuell nachgetragen. Die anderen Objektverknüpfungen sind standardmäßig vorgegeben. Die R/3 - Objekte TWGLV und WLBM sind Objekte aus dem Modul *Werbemittel* und entsprechen einer Hierarchieebene (Produktgruppe) und einer Produktgruppen - Position.

Mit diesen Ergänzungen ist es also möglich Dokumente der Dokumentart DAx mit den erwähnten Werbemittelobjekten zu verknüpfen. Dies ist notwendig, da die Verknüpfung eines Dokumentes mit einer *Werbemittel* - Produktgruppe bzw. -position die Darstellung der Bilder im Internet-Browser steuert.

Die Objektnamen TWGLV und WLBM erfährt man aus der Fehlermeldung, die bei dem Versuch angezeigt wird, ein Dokument mit einem nicht im IMG entsprechend registrierten R/3 - Objekt zu verknüpfen.

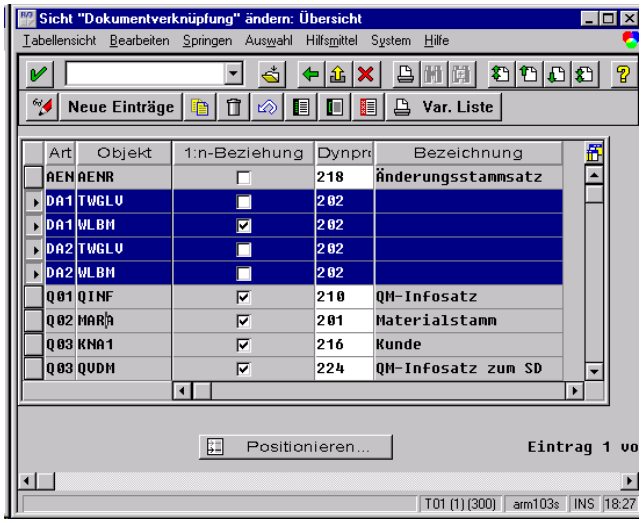


Bild: Objektverknüpfungen im IMG

31. Workstation-Applikation anlegen

Die IAC's klassifizieren ein Dokument anhand von Workstation - Applications (WSApp). Eine WSApp ist eine Applikation, die zur Bearbeitung eines Dokumentes automatisch aufgerufen wird. Je nach WSApp wird ein Dokument als *Kleinbild (thumbnail)*, als *Großbild* oder als *Sound* eingeordnet und interpretiert.

Folgende WSApp's müssen definiert werden:

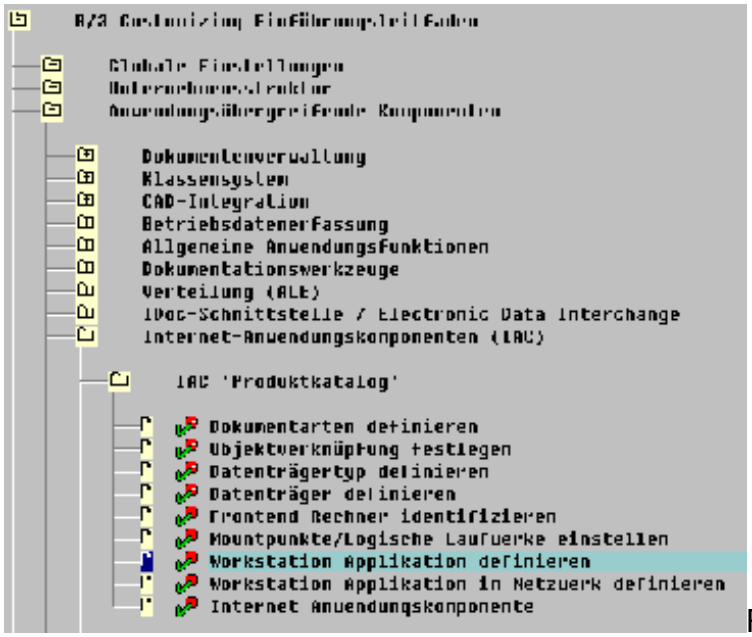
LIM: Großbild

SIM: Kleinbild

SND: Sound

Die Bezeichnungen LIM, SIM und SND sind frei wählbar. Diese WSApps werden auch nicht wirklich verwendet, sondern dienen lediglich als Dummys für die Verwendung durch die IAC's.

ANWENDUNGSÜBERGREIFENDE KOMPONENTEN / INTERNET - ANWENDUNGSKOMPONENTEN / IAC-PRODUKTKATALOG / WORKSTATION-APPLIKATION DEFINIE



REN

Bild: Workstation Applikation definieren im IMG.

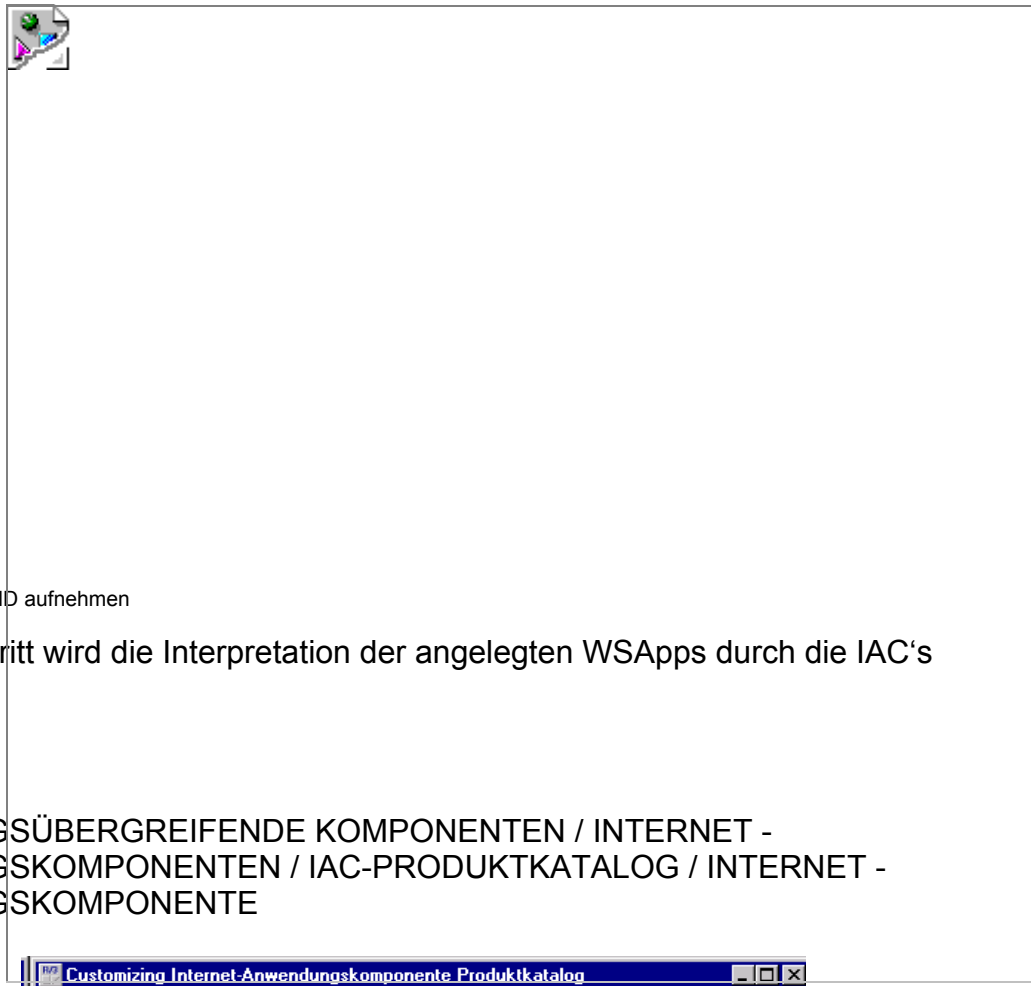


Bild: LIM, SIM und SND aufnehmen

Im letzten Schritt wird die Interpretation der angelegten WSApps durch die IAC‘s eingestellt.

IMG:

ANWENDUNGSÜBERGREIFENDE KOMPONENTEN / INTERNET - ANWENDUNGSKOMPONENTEN / IAC-PRODUKTKATALOG / INTERNET - ANWENDUNGSKOMPONENTE

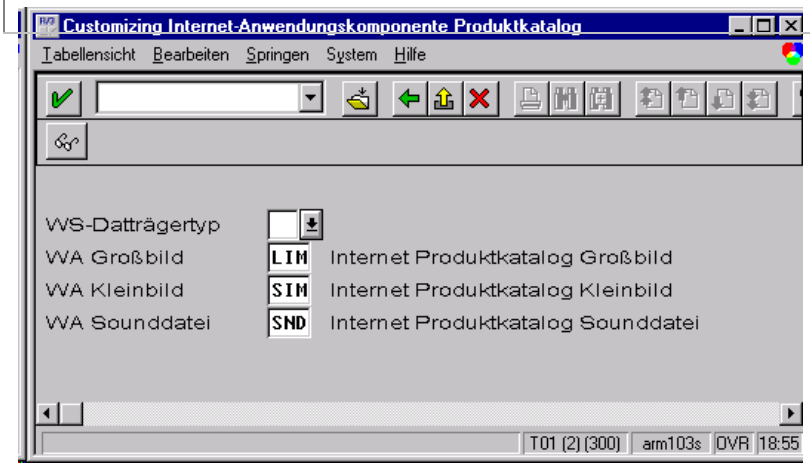


Bild: Zuordnung Workstation - Applikation zu Internet – Produktkatalog

Die Vorgehensweise der SAP bei der Implementierung und Integration der IAC‘s in R/3 kann als Musterbeispiel für die anwendungsbezogene Interpretation von Daten in R/3 herangezogen werden. Der Autor orientierte sich beim Entwurf dieses Konzeptes an den Prinzipien der IAC - Integration.

32. **Aufnahme der Dokumente in die Dokumentenverwaltung**

Nachdem die für diesen Anwendungsfall relevanten Einstellungen des DVS durchgeführt wurden, können die Dokumente in das DVS aufgenommen werden. Mit LOGISTIK / ZENTRALE FUNKTIONEN / DOKUMENTENVERWALTUNG gelangt man in die R/3 - Dokumentenverwaltung.

Die Dokumente werden über Verweise in den Merkmalen referenziert. Die Originaldateien der Dokumente sollten über das Dateisystem erreichbar sein.

33.

34. Verwendete R/3 – Dialog - Transaktionen

LOGISITIK / MATERIALWIRTSCHAFT / WERBEMITTEL

Produkthierarchiepflege, Materialzuordnung

LOGISITIK / MATERIALWIRTSCHAFT / MATERIALSTAMM

Materialpflege, Klassenzuordnung, Merkmalbewertung, Dummy-Artikel

ZENTRALE FUNKTIONEN / KLASSENSYSTEM

Merkmale pflegen, Klassen pflegen, Objektzuordnung

ZENTRALE FUNKTIONEN / DOKUMENTENVERWALTUNG

Bilder + Langtexte aufnehmen + pflegen, Dokumentensuche

35.

36. Das Export - Programm

Wie bereits mehrfach erwähnt wurde, ist es für die Lösung der Aufgabenstellung erforderlich, alle relevanten Daten in Form einer einzigen Datei verfügbar zu machen. Um diese Datei zu generieren muß ein eigenes Export – Programm geschrieben werden, das die Daten im R/3 – System anhand der logischen Zusammenhänge des im Kapitel 4 vorgestellten Abbildungskonzeptes *zusammensucht* , entsprechend *aufbereitet* und *strukturiert* in eine physikalische Datei in einem offenen Datenformat (z.B. ASCII) ausgibt. Über ODBC könnte auch eine relationale Datenbank aufgebaut werden, die die Daten in normalisierter Form ausgibt (z.B. in eine Access-Datenbank). Die Struktur und das Format der Export – Datei sind ausschließlich von der externen Database – Publishing – Software abhängig.

Voraussetzung für das Verständnis des Konzeptes aus Kapitel 4 ist die Erkenntnis, daß sich das Export – Programm an, im Prinzip zwar logischen, aber dennoch vom Autor *willkürlich* definierten Kriterien im R/3 – System orientiert und Daten anwendungsbezogen zusammenfaßt. Die Entscheidung, welche Daten nach welchen Kriterien miteinander verknüpft werden sollen, erfordert sowohl ein Verständnis für die technischen Zusammenhänge innerhalb von R/3, als auch für die betriebswirtschaftlichen Zusammenhänge der abzubildenden Informationen.

Die Informationen wurden zuerst in ihre logischen *Bestandteile* zerlegt (Datengruppen 1 bis 6) und anschließend getrennt in R/3 verteilt abgelegt, wobei immer die Nutzung des R/3 – Standards Vorrang hatte. In dieser *verteilten* Form werden die Informationen

im operativen Tagesgeschäft von den verschiedenen Abteilungen des Unternehmens gepflegt. Keine Redundanz, maximale Aktualität und anwendungsübergreifende Verfügbarkeit der Daten sind die drei Hauptgründe für dieses Vorgehen.

Vor einer Katalog - Produktion kehrt das Export - Programm den *Zerlegungsprozess* der Informationen um. Anhand der im Export - Programm fest codierten Kriterien des Abbildungskonzeptes werden die Daten im R/3 – System zusammengesucht und in jeder gewünschten Form ausgegeben. Das Export - Programm ist die eigentliche Schnittstelle zu weiteren Anwendungen. Es könnte auch ein Programm geschrieben werden, daß statt der Ausgabe in eine ASCII-Datei, HTML-Seiten generiert, die sofort im Internet präsentiert werden können.

Für die Entwicklung eines solchen Export - Programms ist der Zugriff auf, in R/3 – Standard – Dialogen, gepflegte Informationen aus ABAP/4 erforderlich. Der Autor hat noch im Rahmen dieser Arbeit bereits *Ausschau* im R/3 – System nach den wahrscheinlich erforderlichen Funktionsbausteinen aus dem Bereich *Klassensystem, Werbemittel, Materialstamm* und *Dokumentenverwaltung* gehalten. Vordergründig waren keine unlösbaren Zugriffsprobleme erkennbar, wobei sehr wohl einige wahrscheinlich notwendige *Tricks* bei der Programmierung absehbar waren (siehe später am Beispiel *Batch - Input* und *Produktgruppe anlegen, Kap. 5*).

Das Export - Programm kann in ABAP/4 als *Dynpro* bzw. *Report* oder mit einer beliebigen *externen Entwicklungsumgebung* programmiert werden.

Zum letzteren Verfahren geht Kapitel 5 näher ein.

Der Autor hat im Rahmen dieser Arbeit kein Export - Programm entwickelt, um den Umfang der Arbeit nicht weiter wachsen zu lassen. Es war zu diesem Zeitpunkt jedoch schon absehbar, daß die Firma Rüggeberg GmbH & Co an einer weiteren Zusammenarbeit mit dem Autor interessiert ist und das Projekt auch zu Ende geführt werden wird.

Kapitel 5 - Die externe Softwarelösung

37. Wozu eine externe Lösung ?

Das Konzept aus dem letzten Kapitel ermöglicht die strukturierte Ablage aller Informationen für einen Produktkatalog in R/3 - Standardfunktionalitäten. Für die Datenpflege hat dies den Vorteil, daß sie für den Materialstamm im operativen Tagesgeschäft der einzelnen Fachabteilungen eines Unternehmens, innerhalb der gewohnten Bildschirmmasken, durchgeführt werden kann.

1. Folgen der Lösung für die Datenpflege der August Rüggeberg GmbH & Co

Für die August Rüggeberg GmbH & Co bedeutet dies, daß die Einführung der *automatisierten Produktdatenaufbereitung für die Katalogproduktion* keinen Einfluß auf die Datenpflege der davon grundsätzlich unbeteiligten Abteilungen wie *technische Entwicklung* (technische Merkmale, Datengruppe 2), *Vertrieb* und *Arbeitsvorbereitung* (Produktpreise, Datengruppe 4) hat. Diese arbeiten wie gewohnt weiter in ihren R/3 – Standardmasken. Die Aufnahme und Bewertung von neuen technischen Merkmalen und Preiskonditionen wird für diese völlig

transparent auch in die Katalogproduktion übernommen.

Für die Abteilung *Marketing* jedoch, die die Katalogdaten der Datengruppen 1, 3, 5 und 6 in ihrem gesamten Kontext zueinander pflegen muß, stellt sich die Datenpflege als recht komplex dar. Durch die überwiegende Nutzung des R/3 Standards und die dadurch verteilte Ablage der Daten im System muß hier auch die Datenpflege entsprechend *verteilt* stattfinden. So werden übergreifend das *Klassensystem* , die *Dokumentenverwaltung* , das Modul *Werbemittel* und der *Materialstamm* verwendet.

Dies erfordert einen relativ hohen zeitlichen Pflegeaufwand der Katalogdaten und ein entsprechendes Abstraktionsvermögen des Anwenders, weil er in verschiedenen, voneinander unabhängigen R/3 - Dialog – Transaktionen, logisch zusammenhängende Informationen pflegen muß. Außerdem muß der Anwender das erarbeitete Konzept der Katalogdatenablage in R/3 beherrschen (Kap. 4) und dessen Konventionen manuell bei der Katalogpflege einhalten, was naturgemäß die Gefahr einer hohen Fehlerquote in sich birgt. Die Einhaltung der Daten - Konventionen, die mit dem Konzept ausgearbeitet wurden, ist jedoch für ein fehlerfreies Funktionieren des Export - Programms eine unabdingbare Voraussetzung.

Bei einigen Praxisversuchen zeigte sich, daß die Katalogdatenpflege mit den R/3 – Standard – Dialogen für die Abteilung *Marketing* nicht praktikabel ist. Das Auffinden von evtl. Tippfehlern in Eingaben, die für das Export - Programm einen logischen Bezug zwischen verschiedenen Daten herleiten, würde bei entsprechend großem Materialstamm und Katalogumfang einer *Suche im Heuhaufen* gleichen. Bei einer mehrwöchigen Pause wird jeder durchschnittliche Anwender natürlicherweise die detaillierten Zusammenhänge des Konzeptes wieder vergessen und vor einer erneuten Datenpflege erst wieder einen mehrtägigen Lernprozeß durchmachen müssen, in der seine Motivation und Produktivität eingeschränkt ist.

2. Die Lösung des Problems *Datenpflege*

Die Probleme bei der *Pflege und Verwaltung der Kataloge* lassen sich, wenn das Abbildungskonzept aus Kapitel 4 in der Form beibehalten werden soll, nur durch die Entwicklung einer eigenen Softwareanwendung für die Abteilung *Marketing* lösen. Diese Anwendung muß die Katalogdaten und -funktionalitäten zentral in anwendungsbezogenen Dialogmasken darstellen. Der dem Unternehmen vom Autor vorgetragene Vorschlag zur Entwicklung einer solchen Software - Anwendung, wurde im Hause Rüggeberg, insbesondere von der EDV - Abteilung, ausführlich diskutiert. Eine solche Softwareanwendung ist Individualsoftware und mit den bereits erwähnten Nachteilen von Individualsoftware behaftet. Nachdem der Autor das wichtigste Gegenargument, nämlich die Abhängigkeit vom Entwickler (d.h. dem Autor), entkräften konnte, indem er das von ihm entworfene Abbildungskonzept aus Kapitel 4 vorstellte, erteilte die EDV-Abteilung ihre Zusage. Die Orientierung am R/3 – Standard ermöglicht die Nutzung der Lösung notfalls auch ohne die zu entwickelnde Anwendung für die Datenpflege. Die einzige Bedingung der EDV-Abteilung der August Rüggeberg GmbH & Co war, daß die Programmierung des Export - Programms in ABAP/4 erfolgen sollte, um die Behebung von evtl. auftretenden Problemen oder Anpassungen notfalls im Hause durchführen zu können. Da keine direkten technischen Interdependenzen zwischen dem *Export - Programm* und der *Datenpflege - Anwendung* bestehen, konnte der Autor dies zusagen.

Für die Entwicklung einer eigenen Softwarelösung zur Pflege und Verwaltung der Kataloge sprechen unter anderem folgende Gründe:

- Höhere Produktivität bei der Katalogpflege durch zentrale Bereitstellung aller für die Katalogpflege relevanten Funktionen
- Einfachere Katalogpflege und damit ein geringerer Schulungsaufwand für den Anwender
-
- Die Datenintegrität und –konventionen des Konzeptes zur Ablage der Informationen in R/3 werden automatisch eingehalten
- Geringere Fehlerquote bei der Datenpflege
- Bestimmte Funktionalitäten lassen sich nur mit einer Zusatzentwicklung realisieren (z.B. verschiedene Katalogversionen, Teilkataloge, Katalogvergleiche, etc.)

1. Die Verwendung externer Entwicklungsumgebungen

Grundsätzlich kann mit den Mitteln der ABAP/4 – Workbench eine Dialog – Transaktions – Anwendung entwickelt werden, die die gegebenen Anforderungen ausreichend erfüllt. Seit der R/3 - Version 3.1h verfolgt die SAP AG jedoch eine neue Strategie bei der Erweiterung des R/3 - Systems. Bis zu dieser Version stand die maximale Integration und Funktionsabdeckung durch R/3 im Vordergrund. Ab der Version 3.x öffnet die SAP AG ihre R/3 - Software für andere, externe Softwaresysteme, indem sie offene, standardisierte Schnittstellen implementiert und propagiert, über die Fremdanbieter ihre Softwaresysteme an R/3 anbinden und gemeinsame Daten nutzen können.

Da die ABAP/4 – Workbench und die mit ihr erstellten Anwendungen trotz der modernen Technologie, die sie verwenden (Client / Server etc.), im Handling und in ihrer optischen Darstellung ein wenig *unzeitgemäß* wirken (Version 3.1h), bietet es sich an, die Anwendung in einer modernen Entwicklungsumgebung, wie z.B. Visual Basic von Microsoft oder Borland Delphi von Inprise, zu entwickeln. Der einzige Nachteil, der evtl. durch die Verwendung solcher externer Tools entstehen kann, ist die Plattformabhängigkeit. Die sehr hohe Verbreitung von Windows als Front - End Plattform relativiert diesen Nachteil jedoch wieder. Für die hier behandelte Aufgabenstellung kann dieser Nachteil vollends vernachlässigt werden.

Als einzige Anforderung an die zu verwendende externe Entwicklungsumgebung ist unter 32-Bit-Windows die Unterstützung von ActiveX – Komponenten, bzw. des Microsoft Standards COM / DCOM.

Die Vorteile, der durch die Nutzung externer Tools zu den bereits erwähnten Pluspunkten einer Zusatzentwicklung hinzukommen, sind für den Anwender:

- Produktivere, komfortablere und einfachere Bedienung der Anwendung durch die Implementierung von Drag & Drop und anderen üblichen Windows – Funktionalitäten, die von der R/3 – Workbench nicht unterstützt werden
- Die R/3 – Komplexität wird gegenüber dem Anwender vollständig gekapselt

Für den Entwickler ergeben sich ebenfalls große Vorteile, die in der Praxis oft sogar die eigentliche Entscheidungsgrundlage für die Entwicklung mit einer externen Entwicklungsumgebung sein dürften:

- Höhere Produktivität bei der Programmierung durch Nutzung einer modernen Entwicklungsumgebung

- Höhere Produktivität bei der Programmierung durch Nutzung zusätzlicher externer Entwicklungstools (Case – Tools, etc.)

- Orientierung an der zukünftigen strategischen Ausrichtung der SAP AG, externe Softwaresysteme stärker einzubinden und zu unterstützen.

Im Rahmen dieser Arbeit wurde die grundsätzliche Konzeption einer solchen Anwendung erarbeitet. Der Autor hat sich bei der Wahl der Entwicklungsumgebung für Borland (Inprise) Delphi entschieden, da er bereits Erfahrungen mit diesem Tool hat. Das Ergebnis ist ein Windows – Programm (Prototyp), das sich Online in ein R/3 – System einloggt und die R/3 – Daten für den Anwender transparent und anwendungsbezogen verfügbar macht.

Als Name für die Software - Anwendung wählt der Autor *CATMAKER*, das von *CATalogMAKer* hergeleitet wurde. Es sei hier noch einmal darauf hingewiesen, daß durch die Erstellung einer externen Lösung alle bisher gemachten Angaben ihre Gültigkeit beibehalten, dies gilt insbesondere für die Einhaltung des Abbildungskonzeptes, d.h. des R/3 - Standards.

1.
2. Prototyp CatMaker

1. Das Hauptfenster

Nach dem Programmstart erscheint das Hauptfenster, das die Hauptfunktionen zentral verfügbar macht. Der Anwender muß sich für den Aufruf nicht noch parallel mit der SAPGUI in R/3 einloggen. *CATMAKER* arbeitet als autonomer R/3 – Client. Jedoch muß der Windows - PC natürlich als R/3 – Client eingerichtet sein.

Bild: Das Hauptfenster von *CatMaker*

2.

3. Katalogstruktur definieren

Bevor die eigentlichen Produktdaten eingetragen werden können, muß eine Katalogstruktur aufgebaut werden. Im Menü *Katalogstruktur definieren* kann ein Hierarchiebaum komfortabel aufgebaut werden. Die Reihenfolge der Einträge in dem Hierarchiebaum kann intuitiv mit der Maus verändert werden. *CatMaker* speichert diesen Hierarchiebaum in der R/3 – Komponente *Werbemittel* ab. Bei der Aufnahme einer neuen Hierarchieebene werden alle Schritte gemäß dem Konzept aus Kapitel 4 automatisch in R/3 durchgeführt. Dies betrifft z.B. die Erstellung aller notwendigen Klassen und Klassifizierungen.

Im Bereich *Produktgruppenmerkmale* werden die Merkmale einer *Produktgruppe* eingetragen. Die R/3 – interne Nutzung von Dummy - Artikeln wird gegenüber dem Anwender vollständig gekapselt.

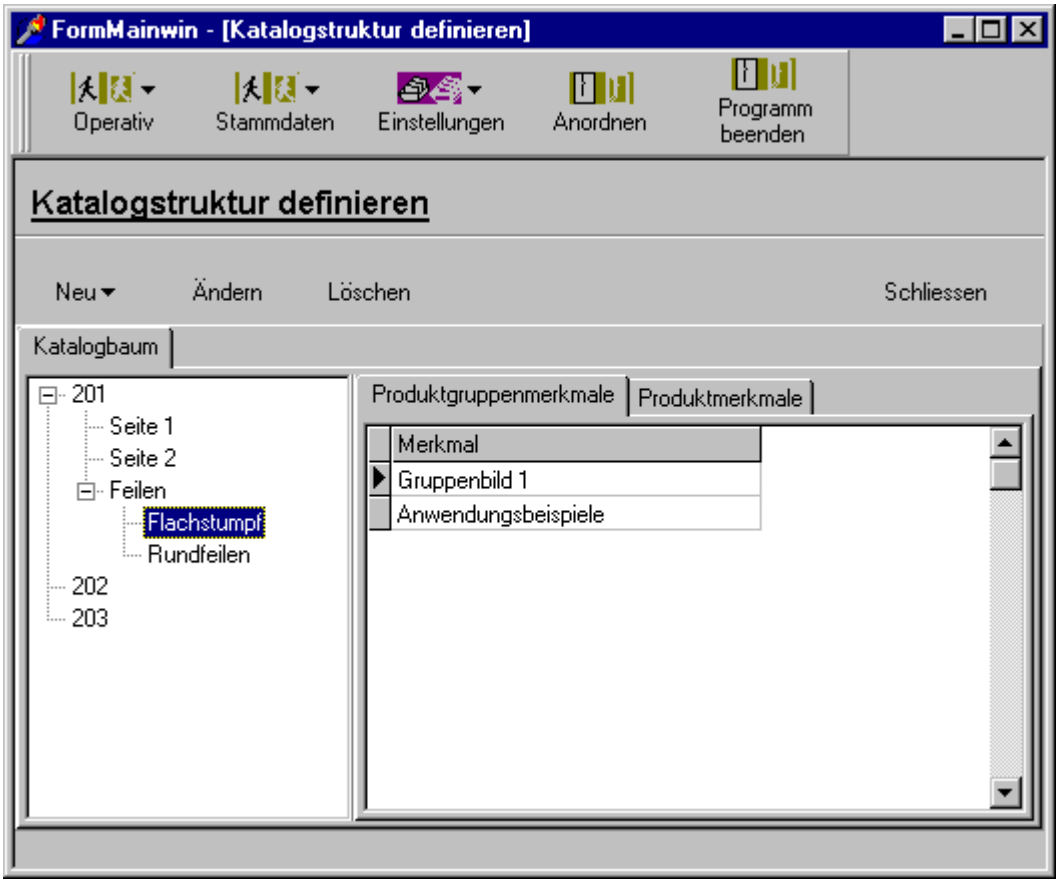


Bild: Aufbau eines neuen Kataloges

Im Bereich *Produktmerkmale* werden die Merkmale der Produktgruppen - Positionen eingetragen. Auch hier werden die notwendige Klassengenerierung und Klassifizierung transparent für den Anwender durchgeführt. Die zentrale Präsentation der *Produktgruppenmerkmale* und *Produktmerkmale* neben der Kataloghierarchie bedeutet für den Anwender eine intuitive Katalogdatenpflege, ohne sich mit den technischen oder konzeptionellen Details der Katalogproduktion befassen zu müssen. Er kann sich voll auf die eigentliche Datenpflege konzentrieren.

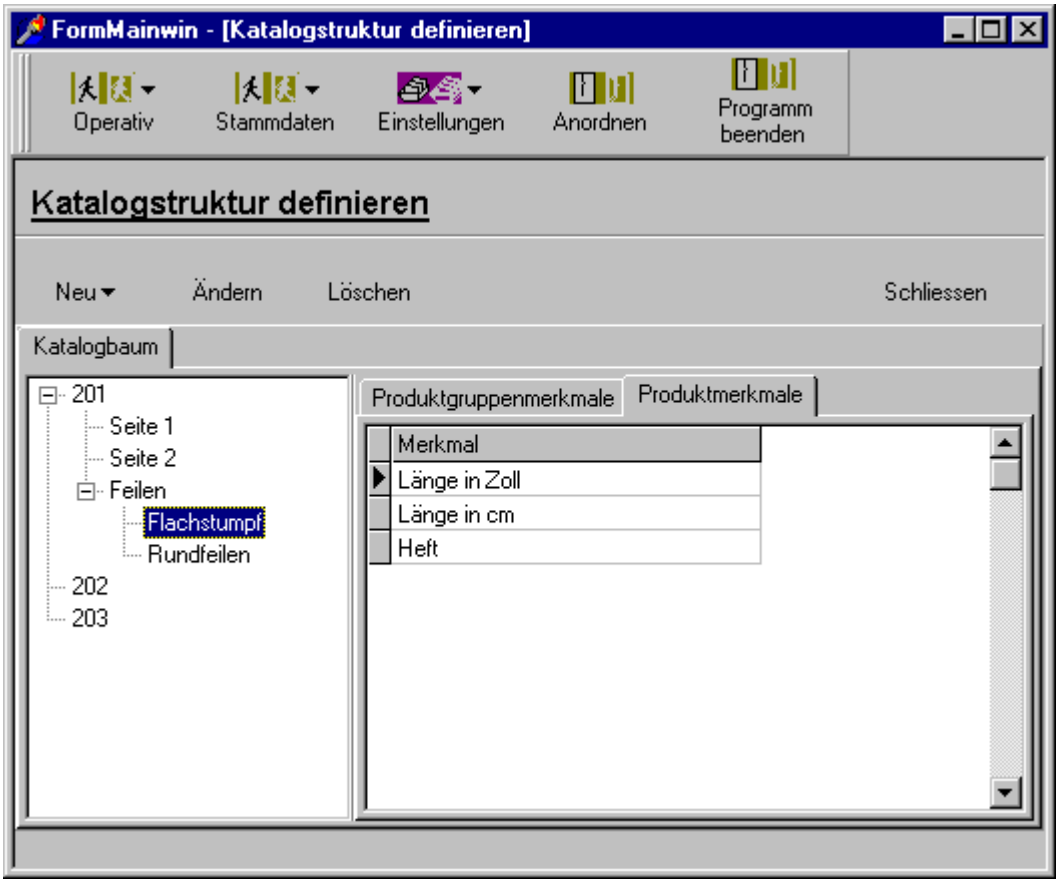


Bild: Artikelstruktur definieren

4.

5. Merkmale definieren

Die Merkmale, die im Katalog verwendet werden, müssen vorher einmal zentral aufgenommen werden. Diese Merkmale werden als Merkmale des R/3 Klassensystems im R/3 – System gespeichert. Die Merkmalseingabe beschränkt sich auf die Daten, die für *CatMaker* relevant sind. Die Klassifizierung der Merkmale zur Merkmalverwaltung und die Erstellung und Zuordnung der *internen Merkmale* geschieht automatisch.

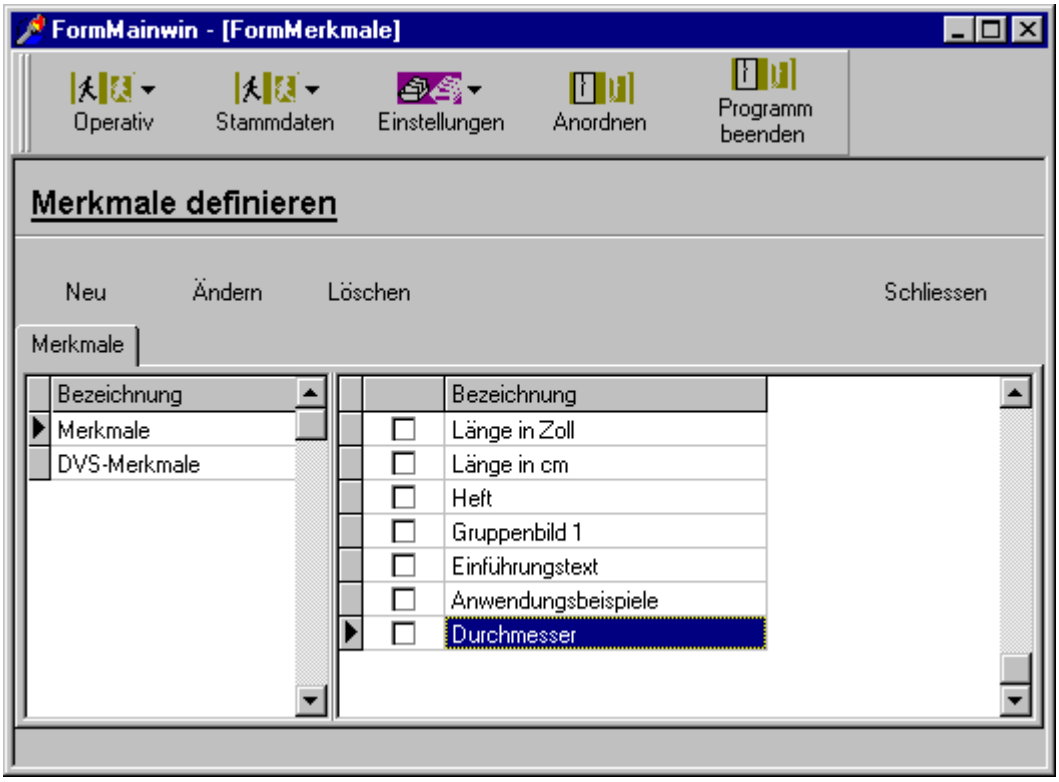
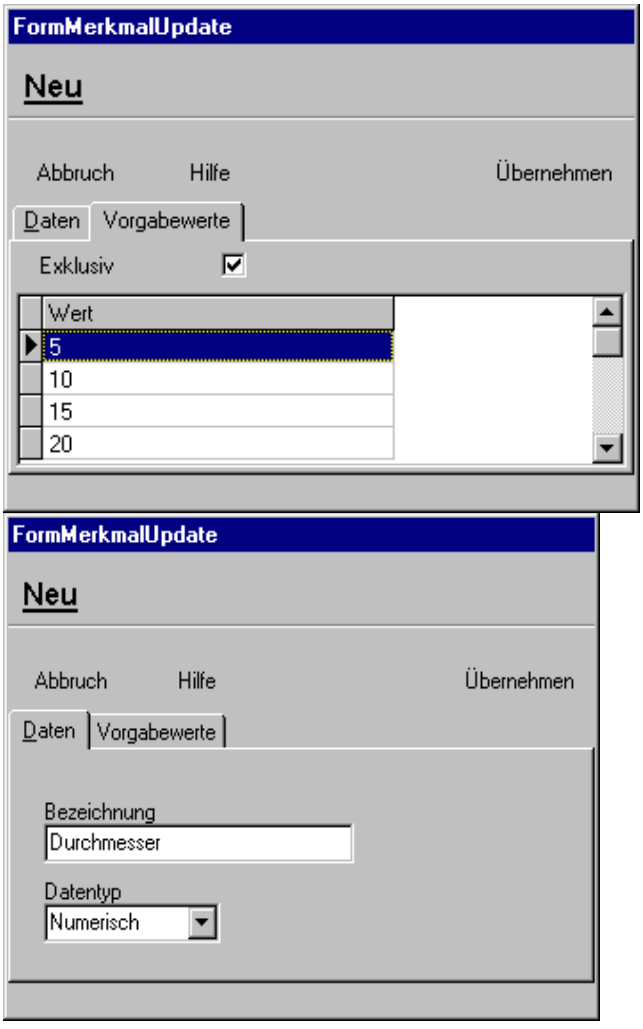


Bild: Merkmale



Bilder: Neues Merkmal aufnehmen

Artikelliste

Die Artikelliste ist die R/3 – Materialliste. Eine Pflege der Artikelliste ist in *CatMaker* grundsätzlich nicht vorgesehen. Dies hätte einen sehr hohen Overhead zur Folge, da die gesamte R/3 – Funktionalität zur Materialstammpflege mit Zugriffsberechtigungen und Anwendungssichten etc. berücksichtigt werden müßte, was einen kaum abschätzbaren Aufwand bedeutet. Die Artikelliste kann eingesehen und nach individuellen Kriterien durchsucht werden.

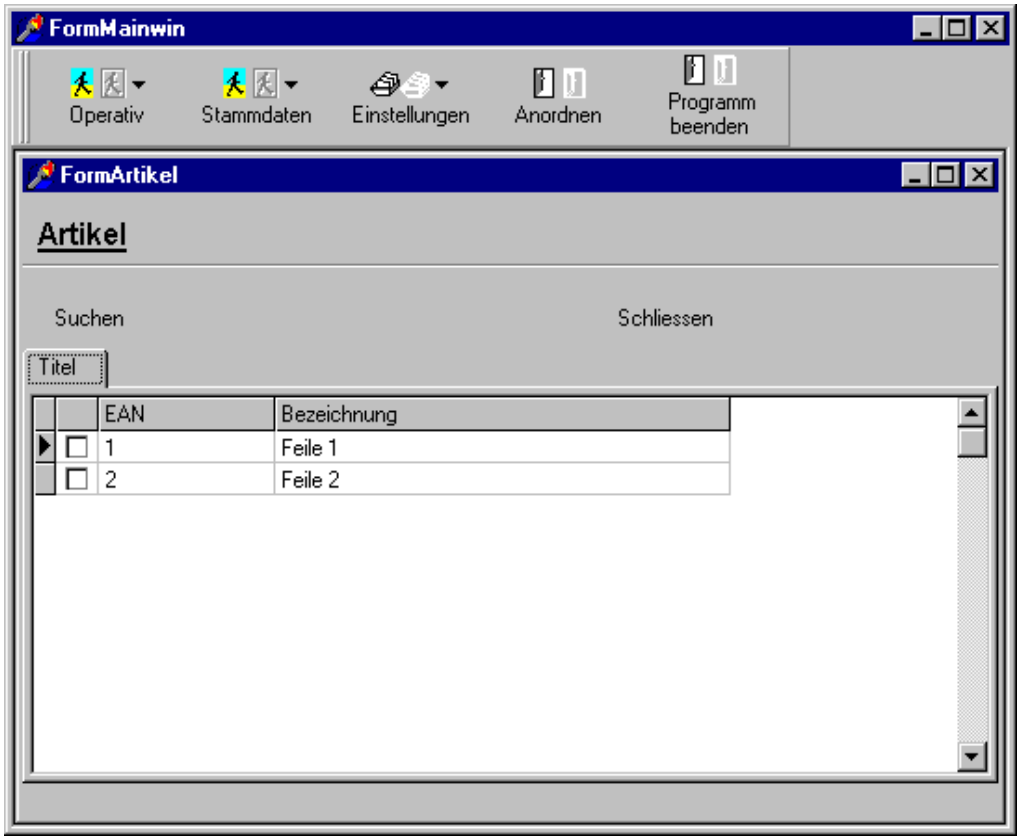


Bild: Die Artikelliste

6.

7. Dokumente

Die Dokumente aus der R/3 – Dokumentenverwaltung erscheinen genau wie die Artikelliste in einer übersichtlichen Liste. Eine einfache Dokumentenpflege zur Aufnahme neuer Bilder und Texte ist hier vorgesehen. Die automatische Zuordnung neuer Dokumente (Texte, Bilder etc.) an die entsprechenden Dokumentarten und damit die sofortige Verfügbarkeit für die R/3 IAC's (R/3 - Internet - Komponenten), erfolgt für den Anwender transparent.

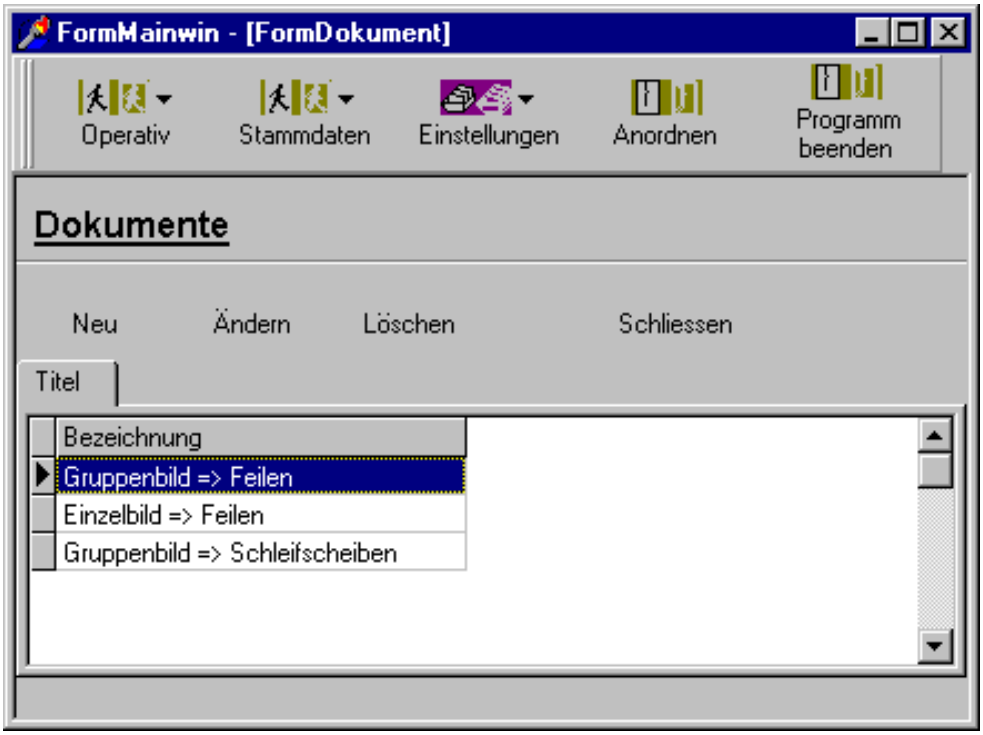


Bild: Dokumentenliste

8.

9. Katalogdaten Pflegen

Das Fenster *Katalogdaten Pflegen* erlaubt die einfache Bewertung der Merkmale eines Kataloges, dessen Struktur (Produktgruppen und Hierarchie) in *Katalogstruktur definieren* aufgebaut wurde. Die verfügbaren Merkmalwerte werden in Comboboxen zur Auswahl angeboten.

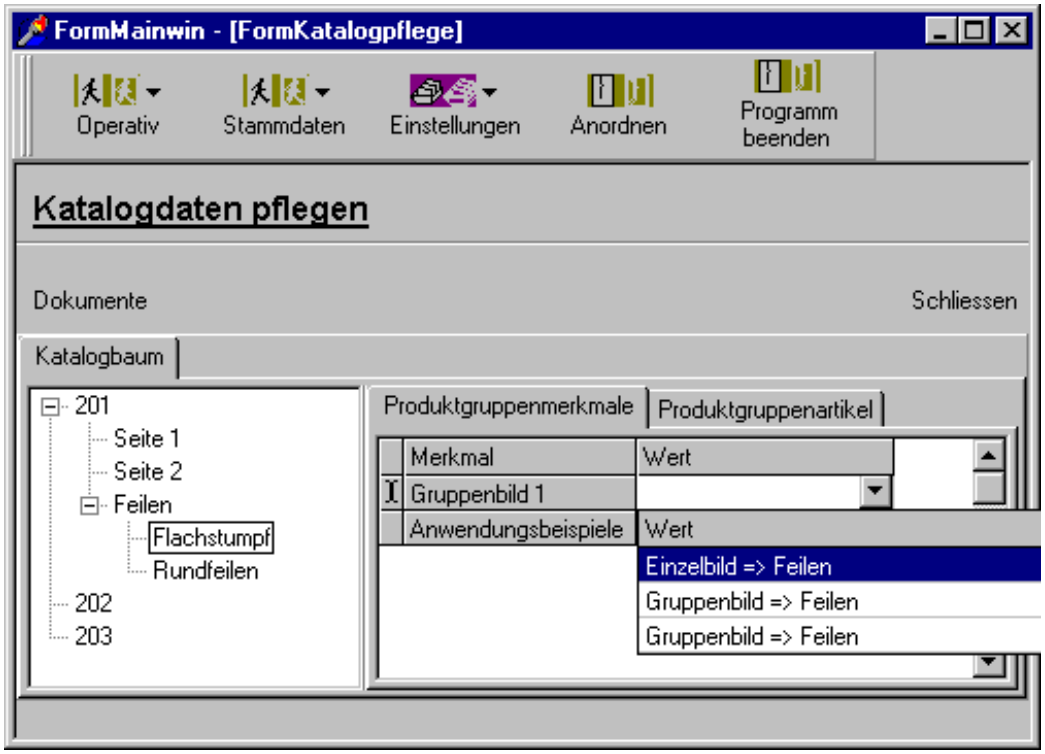


Bild: Bewertung eines Merkmals

Da bei der Erstellung eines Kataloges ständige Änderungen und Anpassungen zu erwarten sind, ist die Zuordnung von Artikeln zu Produktgruppen durch eine produktive Drag & Drop – Bedienung implementiert. Dabei werden die Artikel mit der Maus einfach aus der Artikelliste in den Bereich *Produktgruppenartikel* von *Katalogdaten pflegen* gezogen. Auch hier werden die komplexen *logischen Daten - Beziehungen* automatisch in R/3 berücksichtigt.

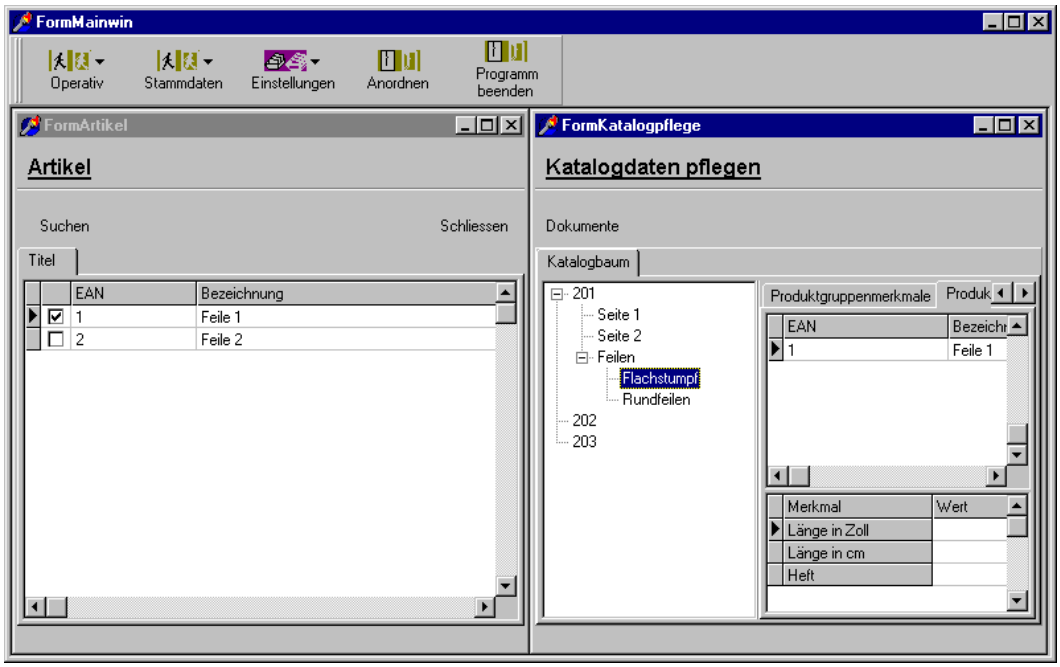


Bild: Produkte in Produktgruppe zuordnen

3.

4. Anbindung externer Softwaresysteme an SAP R/3.

Die Anbindung eines Softwaresystems an ein R/3 - System bedeutet in der Regel, daß R/3 Daten mit anderen Softwaresystemen austauscht und / oder Funktionalitäten gegenseitig genutzt bzw. erweitert werden. Dabei gibt es grundsätzlich zwei organisatorische Möglichkeiten:

1. Der Inside - Out Ansatz

R/3 ruft Funktionen eines externes Softwaresystems auf und greift auf Datenbestände eines externes Softwaresystems zu, d.h. R/3 ist der *Client*, der die Dienste einer *Server - Anwendung* aufruft.

- R/3 steuert die Anwendung
- Applikationslogik ist in R/3
- Programmierung in ABAP, HTML - Business

Einsatzgebiet:

- R/3 ist alleiniges System
- Alle Daten liegen in R/3
- Transaktionssicherheit wichtig

1. Der Outside - In Ansatz

Ein externes Softwaresystem ruft R/3 – Funktionen auf und greift auf R/3 – Daten zu. R/3 ist der *Server*, der Dienste für eine *Client-Anwendung* bereitstellt.

- R/3 wird von außen angesteuert
- Applikationslogik außerhalb von R/3
- Programmierung in C/C++, VB, Java, HTML, CFML etc.

Einsatzgebiet.

- Heterogene IT - Landschaft
- Daten in verschiedenen Systemen
- R/3 – Integration ohne detaillierte R/3 – Kenntnisse möglich.

CatMaker nutzt den *Outside - In* Ansatz, da es die gesamte Applikationslogik beinhaltet und R/3 lediglich zur standardisierten Datenablage verwendet, d.h. lediglich Funktionsbausteine zum Lesen und Schreiben der Daten aufruft. Die Schwierigkeiten, die dabei auftreten, hängen in erster Linie mit der hohen Komplexität und Integration von R/3 zusammen. Um z.B. die Materialstammdaten aus einem R/3 - System zu lesen, kann bzw. darf ein Programm nicht einfach auf die relationale Datenbank, auf der R/3 aufsetzt, zugreifen. Dies hat gleich mehrere Gründe, von denen jeder einzelne als k.o. - Kriterium ausreicht:

- Die Daten eines Materials (und aller anderen R/3 - Objekte) sind nicht übersichtlich in einer einzigen Tabelle (man bedenke z.B. die Merkmale des Klassensystems), sondern in vielen miteinander stark verknüpften Tabellen abgelegt. Um diese Informationen wieder logisch zusammenzuführen muß die entsprechende R/3-Logik vom externen Programm nachvollzogen werden. Dies entspräche faktisch einer Nachprogrammierung von R/3 - Algorithmen, was sicher nicht im Sinne des Erfinders ist.
- Das R/3 Data - Dictionary kann (theoretisch) eine Tabelle physikalisch anders in der zugrundeliegenden Datenbank ablegen, als sie im R/3 Data - Dictionary angelegt wurde.
- Die SAP AG behält sich vor, jederzeit Tabellenstrukturen etc. zu ändern bzw. weiterzuentwickeln. Im besten Fall funktioniert ein externes Programm dann nicht mehr, im zweitbesten Fall liest es fehlerhafte Daten, aber im schlimmsten Fall schreibt es fehlerhafte Daten in die internen Tabellen von R/3, was das gesamte R/3 - System zum Zusammenbruch bringen kann.

Als Ergebnis dieser Punkte ist festzuhalten, daß jeder Zugriff auf R/3 - Daten über das R/3 - Data - Dictionary von ABAP/4 erfolgen sollte (muß), egal ob lesend oder schreibend (Ausnahmen bestätigen die Regel). Ergänzend gilt, daß der schreibende Zugriff auf komplexe Daten, d.h. auf Daten, die in mehreren miteinander verknüpften Tabellen abgelegt werden, nur über die für diesen Zweck vorhandenen Funktionsbausteine geschehen sollte (die ebenfalls in ABAP/4 geschrieben sind).

Wenn ein externes Programm Daten mit einem R/3 System austauschen soll, gibt es grundsätzlich folgende Möglichkeiten.

1.
2. Batch - Input

Batch - Input ist ein Verfahren zur dialogfreien Übernahme großer Datenmengen in das SAP - System. Es können Daten zum Zeitpunkt der Systemeinführung (Altdatenübernahme), aber auch danach (Änderungen) in das R/3 - System übernommen werden. Damit die Konsistenz der R/3 - Datenbasis garantiert ist, müssen die maschinell übernommenen Daten die gleichen Prüfungen durchlaufen wie die manuell erfaßten:

- Es finden die gleichen Prüfungen wie bei der manuellen Eingabe statt.
- Es werden die gleichen Fehlermeldungen und Warnungen ausgegeben.
- Die Daten werden in gleicher Weise gebucht und in der Datenbank fortgeschrieben.

Das Prinzip des Batch - Input entspricht der Fernsteuerung der R/3 – Standard - Dialogfenster. Über ein Script - ähnliches Steuerungsprotokoll, das in ABAP/4 programmiert wird, können Daten aus einer (ASCII-) Eingangsdatei mit den Eingabefeldern der R/3 – Dynpros assoziiert werden. Die Übernahme der Daten erfolgt dann automatisiert. Für die gängigsten Dialogtransaktionen sind bereits Batch - Input – Programme in R/3 vorhanden (z.B. Materialstamm RMDATIND), die aus einer vorgegebenen ASCII-Datei Daten in das R/3 – System importieren können.

Batch – Input erlaubt nur den schreibenden Zugriff auf R/3. Ein großer Vorteil des Batch – Input ist der Zugriff auf grundsätzlich alle R/3 – Funktionalitäten, die in Form von R/3 – Standard - Transaktionen verfügbar sind, ohne das dafür die entsprechenden Funktionsbausteine notwendig sind.

Beispiel: *Produktgruppe anlegen*

Es existiert ein Funktionsbaustein (BAPI) BAPI_GETLAYOUT, der die Hierarchie - Struktur (z.B. für *CATMAKER*) eines *Werbemittel*s zurückliefert. Es existiert jedoch kein Gegenstück BAPI_PUTLAYOUT, die die Speicherung von extern veränderten *Werbemittel* - Daten in R/3 erlaubt. Diese fehlende Funktionalität kann durch eine geschickte Batch – Input – Programmierung ausgeglichen werden, indem in *CATMAKER* dynamisch Batch – Input - Scripts generiert werden, die in R/3 überspielt und ausgeführt werden (z.B. für die Funktion NEUE PRODUKTGRUPPE). Für einfache Funktionen kann dies synchron zu den Anwendereingaben geschehen, bei komplexeren Operationen kann (muß) dies aus Performance – Gründen über eine SPEI

CHERN – Funktion gelöst werden, da das Batch – Input – Verfahren *relativ langsam* arbeitet. Die Implementation dieser Funktionalität wird in einem separaten Abschnitt dieses Kapitels ausführlicher behandelt, um die R/3 - Anbindung an externe Systeme beispielhaft zu demonstrieren.

Bei der Erstellung eigener Batch – Input – Programme ist die R/3 – Transaktion SHDB sehr praktisch. Die Transaktion SHDB ist ein MAKRO – REKORDER, der alle Schritte bei der Bedienung einer beliebigen R/3 - Dialog – Transaktion aufzeichnet und als ABAP/4 Batch – Programm verfügbar macht. Dieses kann manuell verändert und beliebig oft aufgerufen werden. Der große Vorteil bei diesem Verfahren ist, daß nicht mehr mühsam alle technischen Bezeichnungen der Bildschirm - Elemente (für die namentliche Ansteuerung im Batch -

Programm) einer Dialog – Transaktion manuell ermittelt werden müssen, sondern automatisch im generierten Batch – Input - Programm eingefügt werden.

Dokumentation:

R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench | Basis – Programmierschnittstellen | Daten mit Batch – Input übernehmen

1. **RFC - Funktionsaufrufe**

   ABAP/4 – Funktionsbausteine können RFC - fähig (Remote – Function - Call) sein, d.h., sie können von jeder Programmiersprache aus aufgerufen werden, die ebenfalls RFC - fähig ist. Programmiersprachen wie Visual Basic oder Borland Delphi sind RFC - fähig, so daß sie direkt Funktionsbausteine in R/3 aufrufen können. In R/3 müssen die Funktionsbausteine explizit als RFC - fähig deklariert werden. Dies geschieht in den Basisdaten eines Funktionsbausteines. Dort steht u.a. eine Checkbox *RFC* zur Auswahl. Es können also eigene Funktionsbausteine in R/3 erstellt werden, die von externen Anwendungen aufrufbar sind.

2. **BAPI**

BAPI ist die Abkürzung für *Business Application Programming Interface*. BAPI‘s sind die jüngste und langfristig wichtigste offene Schnittstelle in R/3. Sie kapseln R/3 Geschäftsobjekte als standardisierte und von SAP propagierte Schnittstelle an externe Softwaresysteme. Technisch gesehen sind BAPI‘s ganz *normale* RFC - fähige Funktionsbausteine, die eine definierte, betriebswirtschaftliche Funktion erfüllen.

Der Unterschied zu den *einfachen* Funktionsbausteinen ist nicht technisch, sondern konzeptionell begründet.

- In einem BAPI dürfen keine Bildschirmausgaben stattfinden, d.h. von einem BAPI dürfen keine R/3 - Bildschirmdialoge und –meldungen ausgegeben werden. Das gilt für das BAPI selbst und für alle Funktionsbausteine, die von dem BAPI aufgerufen werden.

- BAPI‘s dürfen nur betriebswirtschaftliche Funktionen beinhalten. Es wäre nicht zulässig, Funktionen wie OPEN_MATERIALTABLE und CLOSE_MATERIALTABLE in der aufrufenden Anwendung verwenden zu müssen, um ein Material zu lesen. Diese Funktionsaufrufe müssen in betriebswirtschaftlich orientierten Funktionsbausteinen (=> GET_MATERIAL) für den Aufrufer transparent erfolgen.

- Datenbankveränderungen dürfen nur über die Verbuchung durchgeführt werden (Mit COMMIT WORK).

- Die Datenübergabe ist nur über die Funktionsbaustein – Parameter erlaubt, d.h. Set- und Get - Parameter und das globale Memory dürfen nicht zur Übergabe von Werten benutzt werden.

- Bei Bedarf muß jedes BAPI eigene Berechtigungsprüfungen durchführen.

- BAPI‘s dürfen keinen Status haben, d.h. der Verlauf ihrer Abarbeitung darf nicht abhängig von ihren vorherigen Aufrufen sein.

- Die Bezeichnungen von BAPI‘s beginnen immer mit *BAPI*. In der ABAP/4 – Workbench kann mit dem Suchbegriff *BAPI** gezielt nach BAPI‘s gesucht werden

1.
2. **Business - Objekte**

Die Business – Objekt - Technologie und -Programmierung basiert auf dem Konzept der *Business - Objekte* . Reale Objekte, wie z. B. ein Mitarbeiter oder ein Kundenauftrag, werden in betriebswirtschaftlichen Anwendungssystemen, wie dem R/3 - System, als Business - Objekte abgebildet. SAP – Business - Objekte können als *Black Box* interpretiert werden, die R/3-Daten und Geschäftsprozesse kapseln und auf diese Weise die Struktur- und Implementierungsdetails der zugrundeliegenden Daten verbergen. Um diese Kapselung zu erzielen, sind die SAP – Business - Objekte als Entitäten aus mehreren Schichten angelegt.

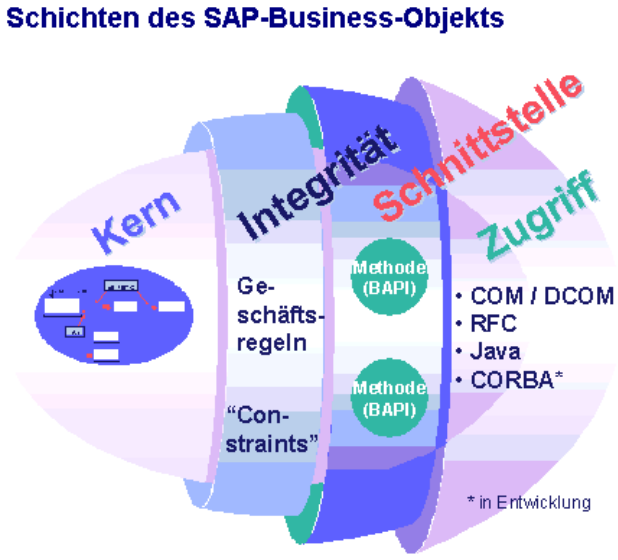


Bild: Aufbau eines Business - Objekts

- Die innerste Schicht des SAP – Business - Objekts ist der Kern, der die eigentlichen Daten des Objekts darstellt. In R/3 sind dies die *transparenten* Tabellen des R/3 – Data - Dictionary.

- Die zweite Schicht, die Integritätsschicht, stellt die betriebswirtschaftliche Logik des Objekts dar. Sie umfaßt Geschäftsregeln und Einschränkungen (*Constraints*), die für die Business - Objekte gelten. In R/3 sind dies in erster Linie die Funktionsbausteine, über die der Zugriff auf die Tabellen erfolgt. Sie Lesen und Beschreiben die Tabellen und führen die erforderlichen Prüfungen durch.

- Die dritte Schicht, die Schnittstellenschicht, beschreibt die Implementierung und Struktur des SAP – Business - Objekts und definiert die Schnittstelle des Objekts zur Außenwelt. In R/3 sind dies die Metadaten im *BOR* (*Business Object Repository*).

- Die vierte und äußerste Schicht eines Business - Objekts ist die Zugriffsschicht. Sie definiert die Technologien, mit denen der externe Zugriff auf die Objektdaten möglich ist, z. B. COM / DCOM (Component Object Model / Distributed Component Object Model). SAP bietet verschiedene plattformabhängige Tools für den Zugriff auf das R/3 - BOR an. Dies sind z.B. C++ und JAVA Klassenbibliotheken und diverse ActiveX – Komponenten.

Zugriff auf Business - Objekte

Wie in der Grafik zu sehen ist, trennt die *Schnittstellenschicht* die Daten eines Business - Objekts von den Anwendungen und Technologien, mit denen auf diese Daten zugegriffen wird. Nach außen zeigen die SAP – Business - Objekte nur ihre Schnittstelle, die aus einer Reihe klar definierter Methoden besteht. Anwendungen können auf die Daten des Business - Objekts nur über dessen Methoden zugreifen. Ein Anwendungsprogramm, das ein SAP – Business - Objekt und seine Daten nutzen möchte, benötigt nur die Informationen zur Ausführung der Objektmethoden. Auf diese Weise kann ein Anwendungsprogrammierer mit den SAP – Business - Objekten arbeiten und ihre Methoden aktivieren, ohne die zugrundeliegenden Implementierungsdetails des Objekts zu kennen oder in Betracht ziehen zu müssen.

Ein Beispiel für eine Methode, die für das Business - Objekt *Material* ausgeführt werden kann ist die Methode *Existenz des Materials prüfen*: *EXISTENCECHECK*.

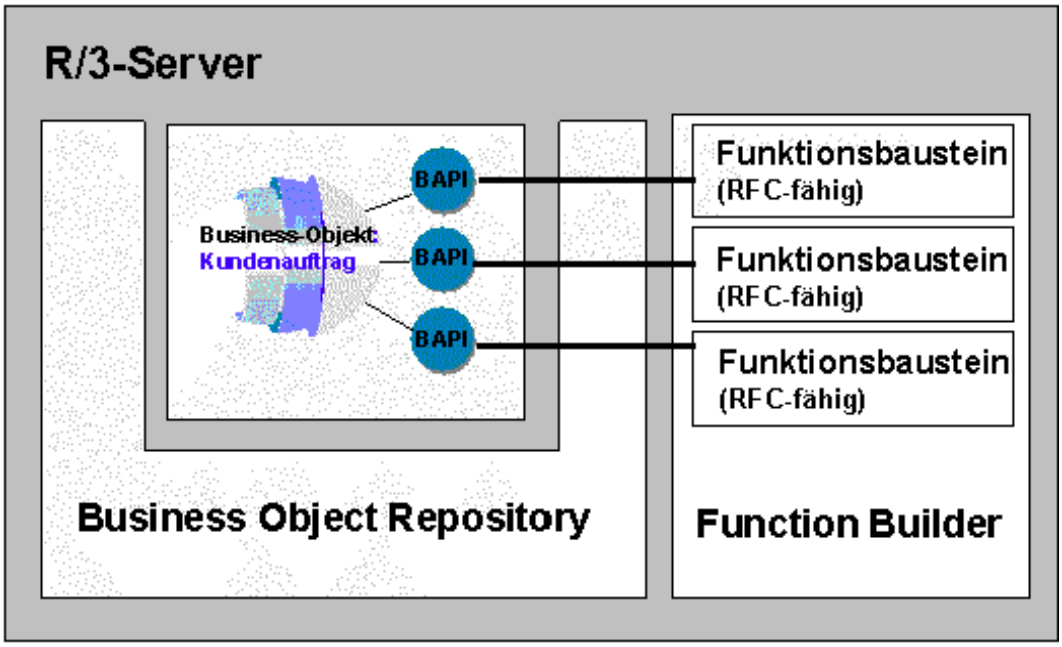


Bild: Kapselung eines Business Object in R/3

1.
2. **Das Business Object Repository (*BOR*)**

   Alle SAP – Business - Objekttypen und ihre Methoden werden im R/3 Business Object Repository (BOR) identifiziert und beschrieben. Das BOR identifiziert und beschreibt die verfügbaren SAP – Business - Objekttypen und ihre BAPI's, d.h. es ist eine zentrale Metadatenbank für die R/3 – Business – Objekte.

   Logisch ist das *BOR* lediglich eine Abstraktionsschicht, die die BAPI's (*normale* RFC - Funktionsbausteine) nach außen als Objekte kapselt. Im BOR werden die Business - Objekte mit ihren Attributen und Methoden gebildet (Interface) und mit den dahinterliegenden RFC - Funktionsbausteinen verknüpft (Implementation). Für die Praxis bedeutet daß, daß der Aufruf einer Objektmethode dem (simplen) Aufruf eines RFC - Funktionsbausteins entspricht

   Um die Informationen im BOR darzustellen, kann das Tool *BAPI Browser* verwendet werden. Der BAPI - Browser ist ein Navigationswerkzeug, das die BOR - Informationen (Objekte, Methoden, Attribute) anwendungsbezogen und hierarchisch anzeigt und auch dokumentiert. Die Dokumentationen entsprechen dabei denen der Methoden - Funktionsbausteine. In der Praxis ist der BAPI – Browser ein unverzichtbares Programmiertool.

   Dokumentation:

BAPI Einführung *caalbd.hlp*

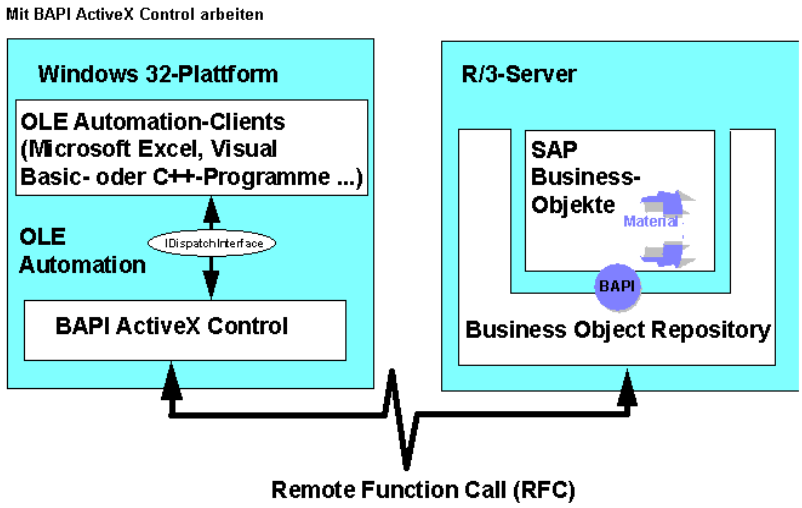
BAPI Programmierleitfaden *caalcd.hlp*

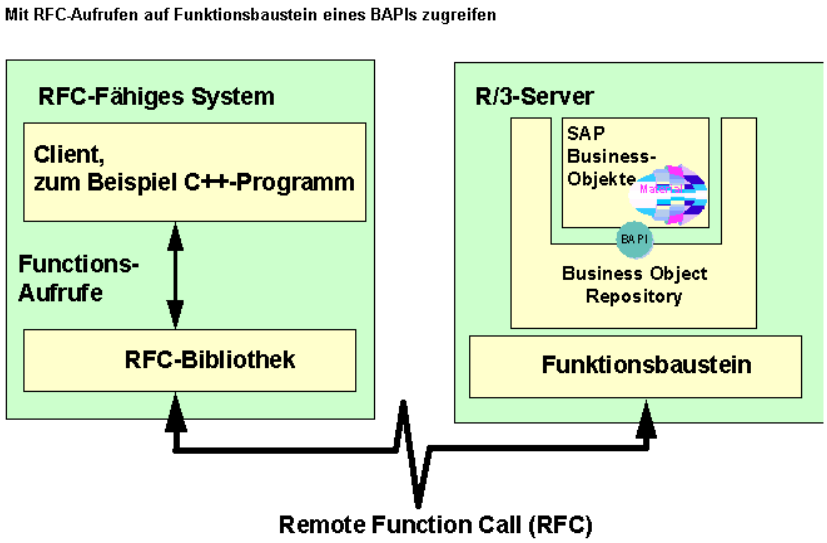
(Download von www.SAP.com/bapi)

BAPI‘s verwenden

BAPI‘s sind als Methoden der SAP – Business - Objekte im Business Object Repository (BOR) definiert und werden als Funktionsbausteine implementiert. Durch die Trennung der Definition eines *Business Objektes* von seiner tatsächlichen Implementierung stehen zwei Ansätze zur Verfügung, über die auf BAPI‘s zugegriffen werden kann. Unabhängig vom verwendeten Ansatz sind folgende Schritte für einen BAPI – Aufruf notwendig: 1.) Suchen des benötigten SAP – Business - Objekts und des BAPI‘s mit dem BAPI - Browser durchgeführt. 2.) Ermitteln der Parameterinformationen zur Schnittstelle des BAPI‘s.

3. **BAPI über BOR aufrufen (Objektmethode).**





4. **Direkte RFC - Aufrufe**

an BAPI Funktionsbaustein.

5.
6. **Tools von SAP für den BAPI – Zugriff**

SAP bietet auf ihrer Homepage WWW.SAP.COM/BAPI kostenlose Downloads von Dokumentationen und Software – Tools für die Anwendung von BAPI‘s an.

- **ActiveX – Komponenten**

  Unter 32 – Bit Windows stehen diverse ActiveX – Controls zur Verfügung, die eine einfache programmgesteuerte Anmeldung an R/3 und den Zugriff auf das BOR bzw. die Business – Objekte ermöglichen. Der Autor hat die Komponenten aus dem Internet geladen und getestet und kann diese uneingeschränkt empfehlen. Sowohl das problemlose Handling, als auch die Performance sind schlichtweg als gut zu bezeichnen. Allerdings ist ein gewisse Erfahrung in den *Bereichen Objektorientierte Programmierung* und *ActiveX*, *COM / DCOM* unabdingbar. Während die Dokumentationen der BAPI – Konzepte und RFC – Zusammenhänge sehr ausführlich und gut verständlich sind, beschränken sich die Informationen zu den Tools oft fast nur auf die reine Aufzählung von Zugriffs - Objekten, Methoden und Attributen.

  Prinzip:

  Es existiert ein Zugriffsobjekt *FUNCTION*, das die Attribute *EXPORTS* und *IMPORTS* für die Funktionsbaustein – Parameter und die Methode *CALL* für den Aufruf hat. Um einen Funktionsbaustein (BAPI) in R/3 aufzurufen wird eine Objektinstanz von *FUNCTION* erzeugt, die einen bestimmten Funktionsbaustein mit ihren Parametern repräsentiert.

- **Klassenbibliotheken**

  C++ und JAVA Klassenbibliotheken kapseln die Komplexität des BOR – und RFC – Zugriffs in überschaubare Objekte.

- **Fremdanbieter**

  Hersteller von Entwicklungsumgebungen implementieren den transparenten Zugriff auf R/3 – Objekte in ihre Produkte. Die bekanntesten sind Microsoft (Visual Studio), Inprise (Delphi SAP / Connect) und IBM (Visual Age). Dabei werden in der Regel die von SAP zur Verfügung gestellten Komponenten produktspezifisch integriert.

1.
2. **Delphi - Anbindung an SAP R/3.**

   Um die praktische Anwendung der bisher vorgestellten Funktionalitäten zu demonstrieren wird in diesem Abschnitt die Implementierung der Funktion *Neue Produktgruppe* in *CatMaker* detailliert dargestellt. In R/3 entspricht das Anlegen einer Produktgruppe in *CatMaker*, dem Anlegen einer neuen Hierarchieebene in der R/3 - Komponente *Werbemittel*. Dieses Beispiel eignet sich besonders gut zur Demonstration, weil es nicht nur einfach eine vorhandene BAPI aufruft, sondern eine fehlende BAPI – Funktionalität implementiert. Während die Verwendung einer BAPI als recht trivial zu bezeichnen ist (Aufruf über BAPI - Bezeichnung), soll in diesem Fall ein eigener ABAP/4 Funktionsbaustein programmiert und *von außen* aufgerufen werden. Da der Funktionsbaustein die

Aufnahme einer Produktgruppe als Batch - Input implementiert, wird nebenbei auch dieses Verfahren dokumentiert.

1. **Was ist Delphi ?**

   Delphi ist eine sogenannte RAD (Rapid Application Development) Windows – Entwicklungsumgebung, die die komponentenbasierte Softwareentwicklung unterstützt. Der Hersteller Inprise (ehemals Borland) bietet auch eine direkte R/3 – Anbindung an, die jedoch für diese Arbeit zu kostspielig war (*SAP/Connect for Delphi*, z.Zt. DM 45.000). Da die von SAP kostenlos verfügbaren ActiveX – Komponenten ebenfalls sehr gut zu verwenden sind, war dies in der Praxis jedoch kein allzu großes Handicap. Es sind lediglich größere R/3 – Kenntnisse, als bei der Nutzung von *SAP/Connect for Delphi*, notwendig, welche jedoch bei einer ernsthaften R/3 – Anbindung grundsätzlich vorhanden sein sollten.

   Nach der Installation der selbstentpackenden ActiveX – Komponenten von SAP und anschließendem Import in Delphi stehen die Komponenten einsatzbereit in der Delphi -

   zur Verfügung.

   Bild: Delphi 4.0

2.
3. **Erstellung des ABAP/4 Batch – Input - Programms**

   Die Funktion *Produktgruppe anlegen* wird implementiert, indem ein ABAP/4 – Batch – Input Programm als RFC - Funktionsbaustein in R/3 programmiert und von Delphi aufgerufen wird. Als Parameter erhält der

Funktionsbaustein den Namen des Werbemittels und der anzulegenden Produktgruppe. Das Beispiel demonstriert auch eindrucksvoll die Flexibilität und Leistungsfähigkeit des Batch - Input's. Die Performance des Batch – Input's ist für eine solche Funktionalität überhaupt kein beachtenswertes Problem.

Im ersten Schritt muß das Batch – Input - Programm, daß die eigentliche Erstellung der neuen Werbemittel - Hierarchieebene implementiert, programmiert werden. Dazu wird der Vorgang der Erstellung einer Werbemittel - Hierarchieebene manuell durchgeführt und dabei mit der Transaktion SHDB wie ein Makro aufgenommen und unter dem Namen *NeuProdGrp* gespeichert:

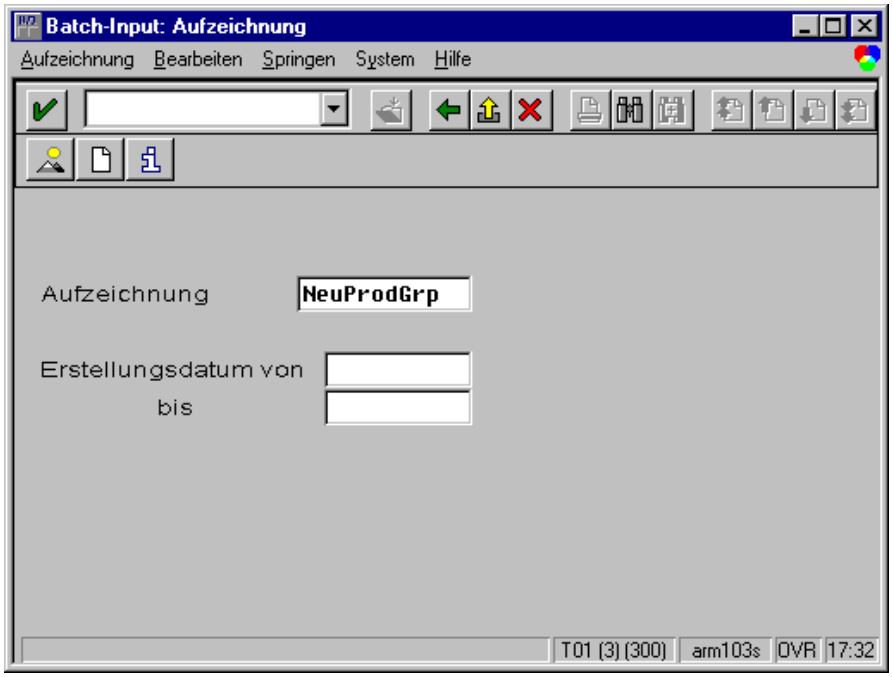


Bild: Transaktion SHDB. Batch – Input - Aufzeichnung

Die aufzuzeichnende Transaktion muß vor dem Aufzeichnungsbeginn angegeben werden. Es ist zu beachten, daß oft gleiche Dynpros in Abhängigkeit ihrer Funktion mit verschiedenen Transaktionscodes aufzurufen sind (z.B. Neues Werbemittel WWM1 / Werbemittel Ändern WWM2).

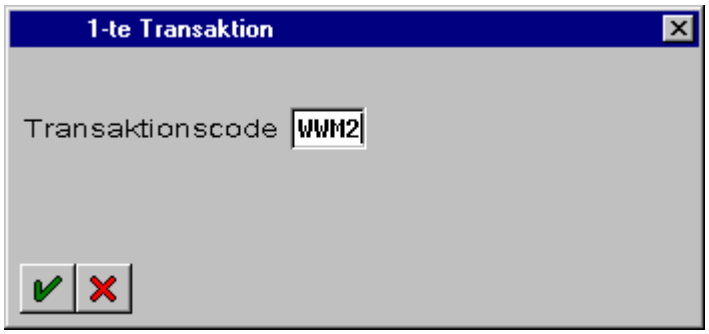


Bild: Batch - Input Aufzeichnung starten

Nach dem Beginn der Aufzeichnung wird die Transaktion aufgerufen und es müssen manuell alle Schritte in den R/3 - Standard - Dialogen zur Aufnahme einer neuen Werbemittel - Hierarchieebene durchgeführt werden:

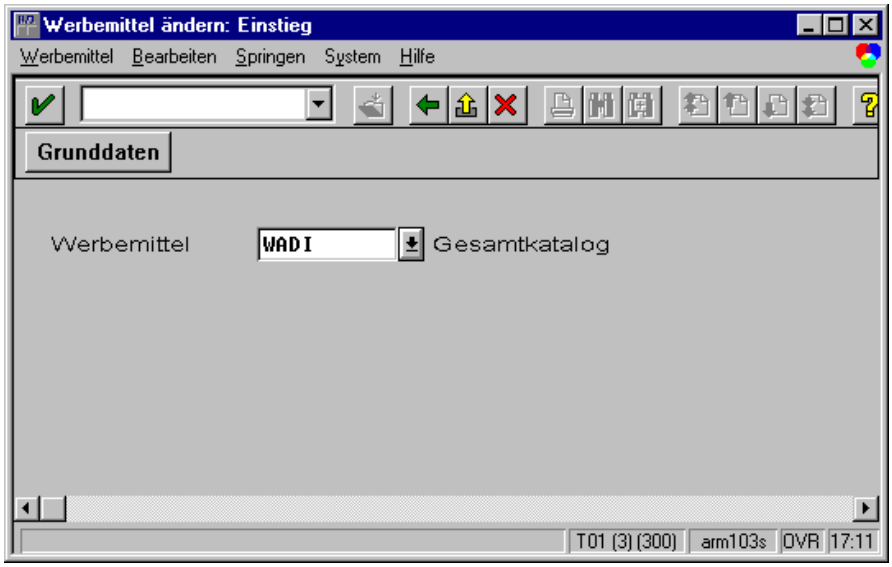


Bild: Werbemittel aufrufen

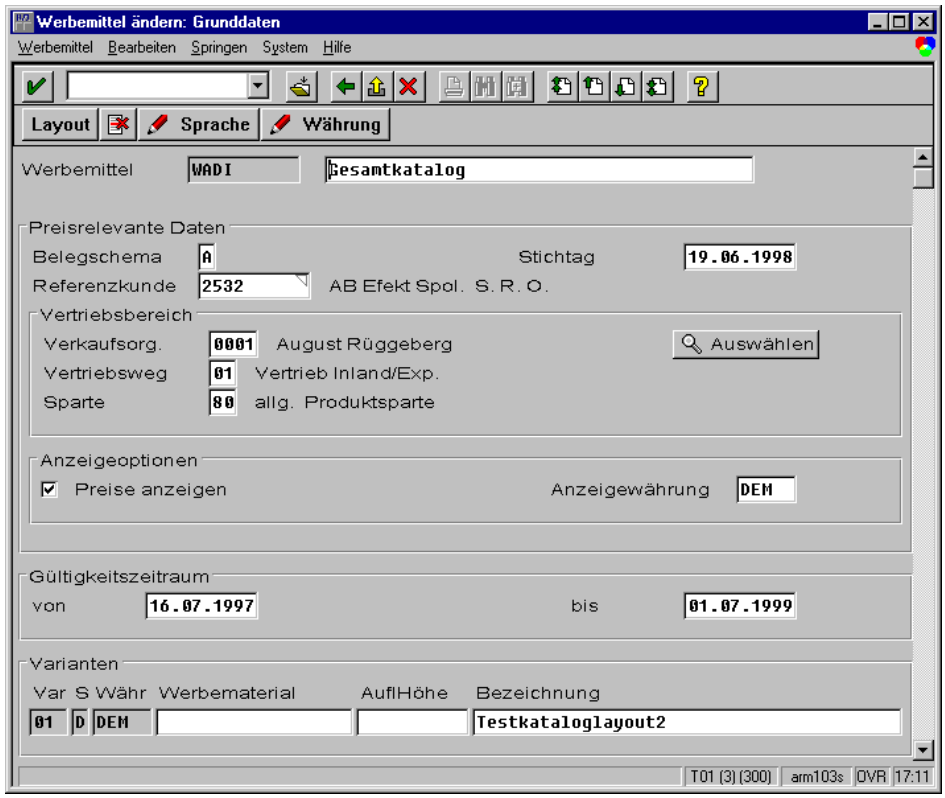


Bild: In das Layout wechseln (Button *Layout*)

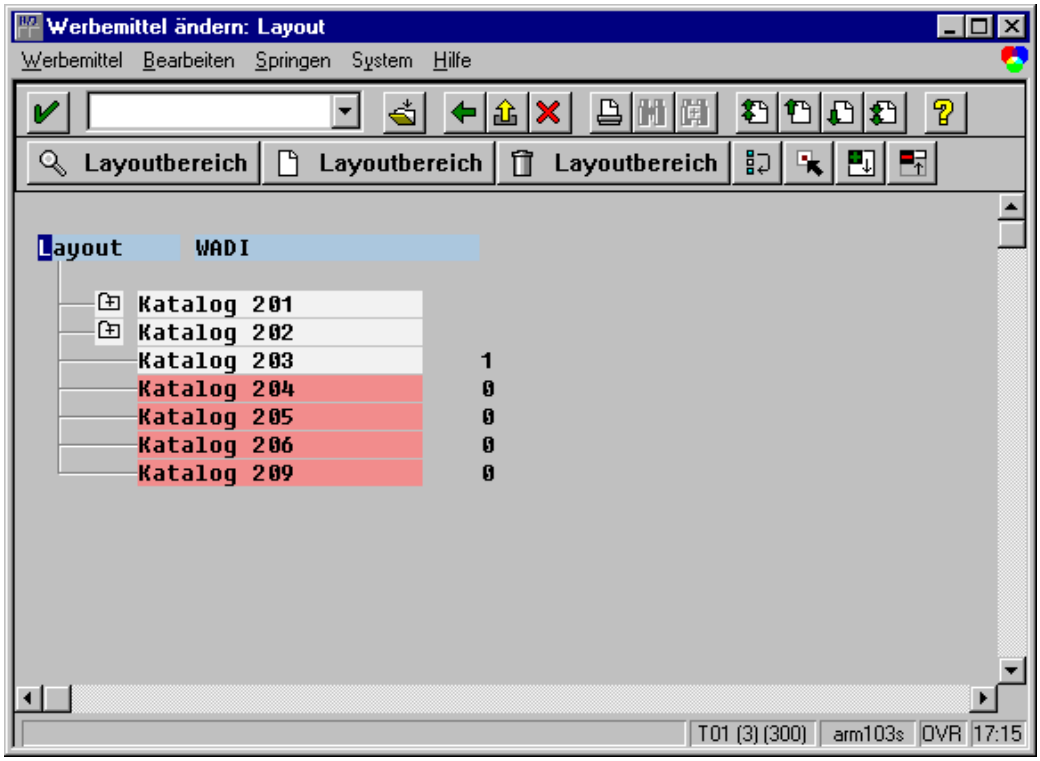


Bild: Cursor auf Layout positionieren und *Layoutbereich (*anlegen) anklicken

Bild: Den Namen des neuen Layoutbereiches (*Neue Produktgruppe*) eingeben

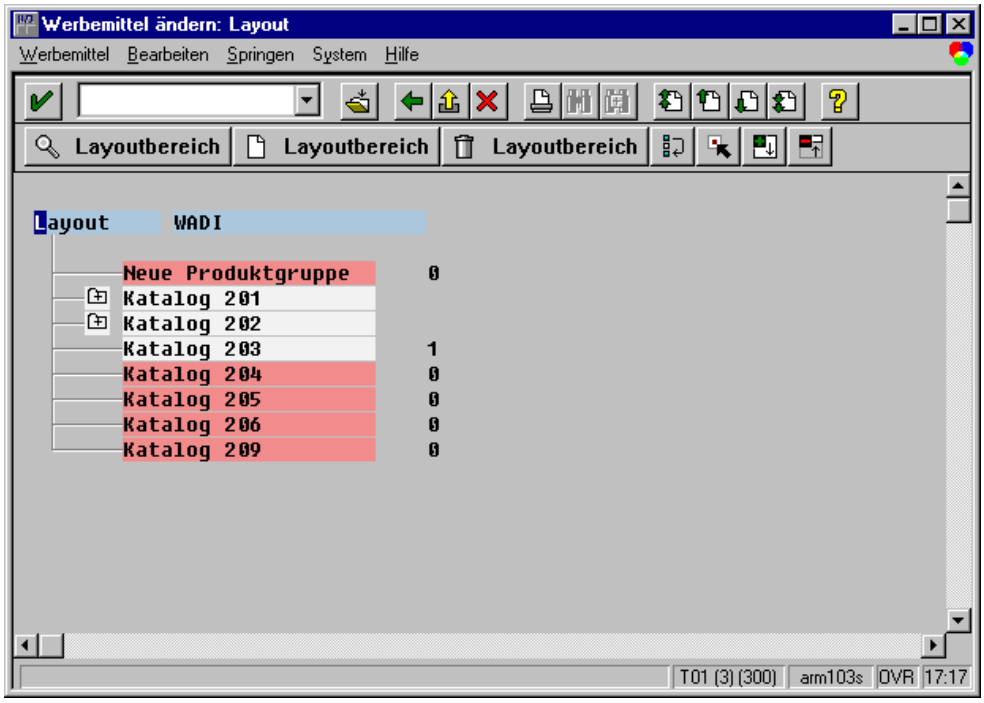


Bild: *Werbemittel / sichern* wählen

Mit dem *Werbemittel / S ichern* wird die Transaktion und die Batch - Aufzeichnung automatisch beendet und es werden die einzelnen Batch - Befehle in den einzelnen Bildschirmmasken angezeigt:

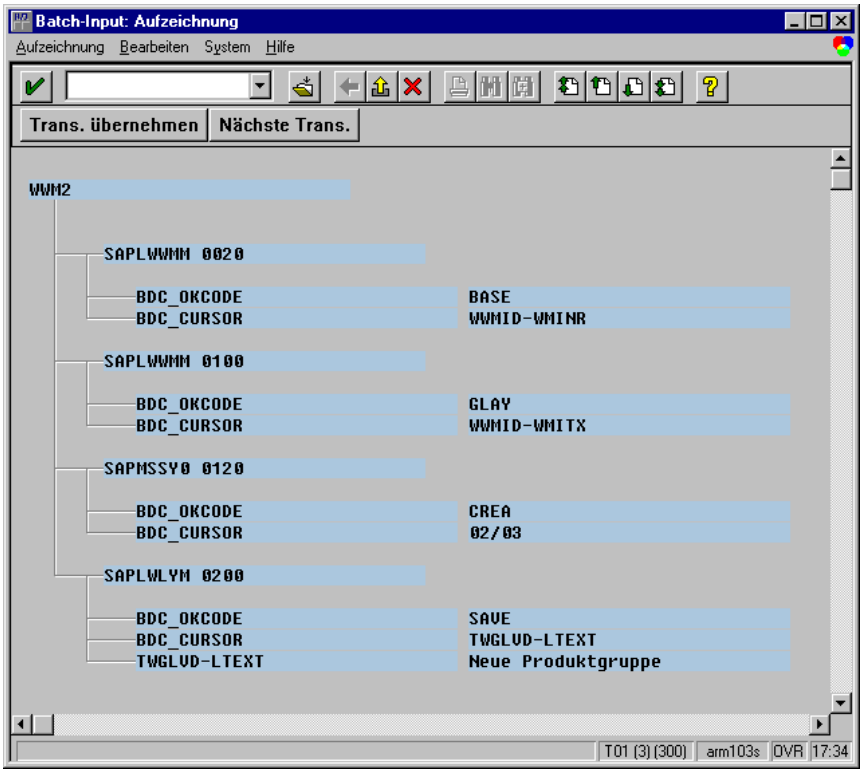


Bild: Eine Batch - Aufzeichnung

Nach der Übernahme der Transaktion kann automatisch ABAP/4 – Programmcode generiert werden, der die Batch - Schritte implementiert. Diesem ABAP/4 Programm kann eine Transaktionsnummer vergeben werden. Jeder Aufruf dieser Transaktion würde eine Produktgruppe *Neue Produktgruppe* erzeugen.

1.

2. Automatisch generiertes ABAP/4 Batch - Input Programm

(Die fettgedruckten Zeilen sind Kommentare des Autors)

```
Implementation als Report-Programm
```

report ZADNGRP no standard page heading.

Hier wird eine einfache Batch-Bibliothek includiert (R/3 Standard)

include bdcrecxx.

Ein ABAP/4 Event. Nach dem Selektionsbild.

start-of-selection.

Eine Batch-Input-Gruppe wird erzeugt (näheres dazu siehe R/3-Doku)

perform open_group.

Hier werden die einzelnen Schritte bei der Aufnahme einer Produktgruppe in die interne Tabelle BDCDATA eingefügt. Jeder Aufruf fügt eine Zeile hinzu und entspricht genau einem Schritt.

perform bdc_dynpro using 'SAPLWWMM' '0020'.

perform bdc_field using 'BDC_OKCODE'

'BASE'.

perform bdc_field using 'BDC_CURSOR'

'WWMID-WMINR'.

perform bdc_dynpro using 'SAPLWWMM' '0100'.

perform bdc_field using 'BDC_OKCODE'

'GLAY'.

perform bdc_field using 'BDC_CURSOR'

'WWMID-WMITX'.

perform bdc_dynpro using 'SAPMSSY0' '0120'.

perform bdc_field using 'BDC_OKCODE'

'CREA'.

perform bdc_field using 'BDC_CURSOR'

'02/03'.

perform bdc_dynpro using 'SAPLWLYM' '0200'.

perform bdc_field using 'BDC_OKCODE'

'SAVE'.

perform bdc_field using 'BDC_CURSOR'

'TWGLVD-LTEXT'.

perform bdc_field using 'TWGLVD-LTEXT'

'Neue Produktgruppe'.

perform bdc_transaction using 'WWM2'.

Die Batch-Input-Gruppe wird geschlossen

perform close_group.

3.

4. Die Standard - Batch – Bibliothek BDCREGXX

Um zu verstehen, wie die Batch - Steuerung arbeitet, wird hier auch der Quellcode der includierten Batch – Input - Bibliothek aufgelistet und dokumentiert.

```
***INCLUDE BDCRECXX.

PARAMETERS:

GROUP(12) OBLIGATORY, group name of session

USER(12) OBLIGATORY, user for start session in batch

KEEP AS CHECKBOX, ' ' = delete session if finished

'X' = keep session if finished

HOLDDATE LIKE SY-DATUM.
```

BDCDATA ist die Standard-Struktur für den Batch-Input. Hier wird eine interne Tabelle BDCDATA angelegt, die die einzelnen Batch-Schritte aufnimmt.

Näheres dazu siehe R/3 Dokumentation *Batch - Input*

```
* Batchinputdata of single transaction

DATA: BEGIN OF BDCDATA OCCURS 0.

INCLUDE STRUCTURE BDCDATA.

DATA: END OF BDCDATA.

*---------------------------------------------------------------------
```

```
* create batchinput session *
*---------------------------------------------------------------------*
```

Mit dem Funktionsbaustein BDC_OPEN_GROUP wird eine sogenannte BATCH-MAPPE geöffnet. Batch-Mappen erlauben u.a. eine Protokollierung der Batch-Aufrufe.

```
FORM OPEN_GROUP.
SKIP.
WRITE: /(20) 'Create group'(I01), GROUP.
SKIP.
* open batchinput group
CALL FUNCTION 'BDC_OPEN_GROUP'
EXPORTING CLIENT = SY-MANDT
GROUP = GROUP
USER = USER
KEEP = KEEP
HOLDDATE = HOLDDATE.
WRITE: /(30) 'BDC_OPEN_GROUP'(I02),
(12) 'returncode:'(I05),
SY-SUBRC.
ENDFORM.
```

Hier das Gegenstück zu OPEN_GROUP

```
*---------------------------------------------------------------------*
* end batchinput session *
*---------------------------------------------------------------------*
FORM CLOSE_GROUP.
* close batchinput group
CALL FUNCTION 'BDC_CLOSE_GROUP'.
WRITE: /(30) 'BDC_CLOSE_GROUP'(I04),
(12) 'returncode:'(I05),
SY-SUBRC.
ENDFORM.
```

Der Aufruf einer neuen Bildschirmmaske bei dem manuellen Aufzeichnungsvorgang wird mit BDCDATA-DYNBEGIN = X markiert. Die beiden anderen Tabellenfelder identifizieren die aufgerufene Bildschirmmaske über Programmname und Dynpronummer

```
*---------------------------------------------------------------------*
* Start new screen *
```

```
*---------------------------------------------------------------------*

FORM BDC_DYNPRO USING PROGRAM DYNPRO.

CLEAR BDCDATA.

BDCDATA-PROGRAM = PROGRAM.

BDCDATA-DYNPRO = DYNPRO.

BDCDATA-DYNBEGIN = 'X'.

APPEND BDCDATA.

ENDFORM.
```

Hier wird ein manueller Bearbeitungsschritt (des Anwenders) aufgenommen. FNAM bezeichnet das Dynpro-Bildschirmelement und FVAL den einzugebenden Wert.

```
*---------------------------------------------------------------------*

* Insert field *

*---------------------------------------------------------------------*

FORM BDC_FIELD USING FNAM FVAL.

CLEAR BDCDATA.

BDCDATA-FNAM = FNAM.

BDCDATA-FVAL = FVAL.

APPEND BDCDATA.

ENDFORM.
```

5.

6. RFC – Funktionsbaustein *Z_CREATEGROUP* erstellen

Aus dem automatisch generierten ABAP/4 Code kann nun (manuell) ein RFC - fähiger Funktionsbaustein erzeugt werden. Dazu muß in der ABAP/4 – Workbench unter *Funktionsbibliothek* der Name des neuen Funktionsbausteins (*Z_CREATEGROUP*) eingetragen und *Anlegen* gewählt werden. Die RFC - Fähigkeit muß explizit angewählt werden. Es erscheint folgendes Bild:

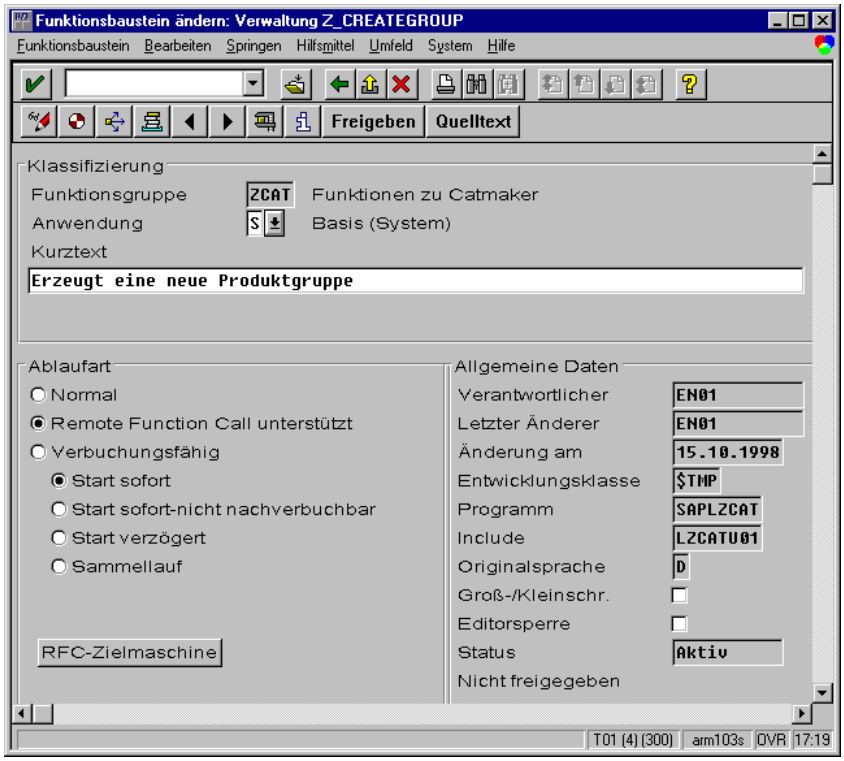


Bild: Grunddaten Daten des Funktionsbausteins *Z_CREATEGROUP*

Mit *Springen / Import - Export Schnittstelle* können die Parameter des Funktionsbausteins definiert werden.

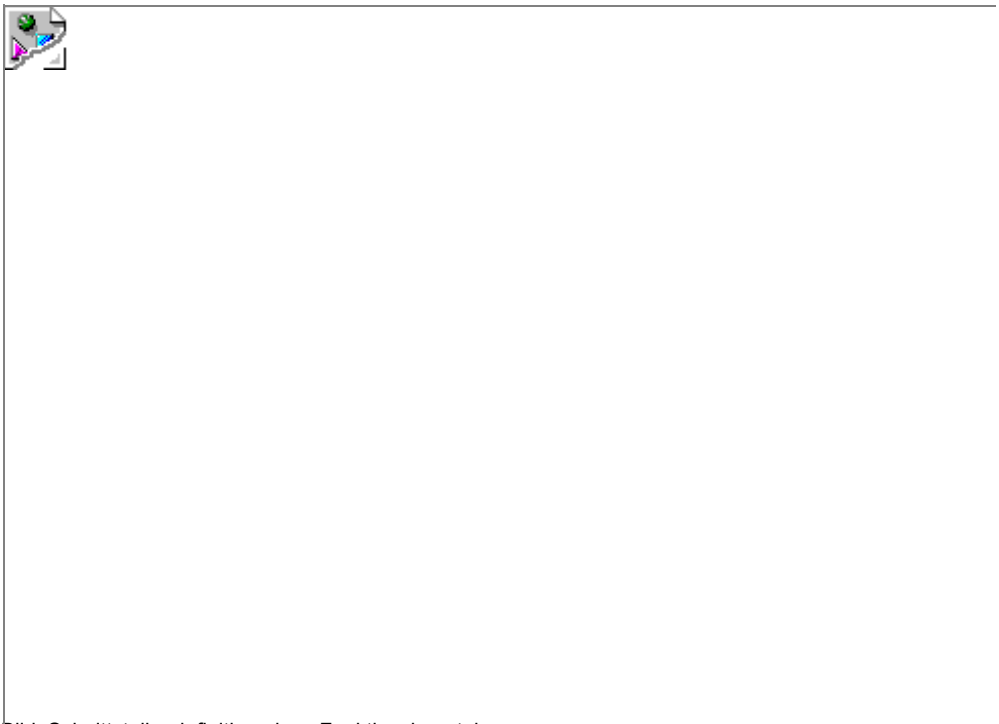


Bild: Schnittstellendefinition eines Funktionsbausteins

GROUP ist der Name des Parameters. *ZCAT_GROUP* ist der Datentyp von *Group*.

Der Datentyp *ZCAT_GROUP* wurde im R/3 - Data - Dictionary als Domäne angelegt und ist eine *Struktur*. Die Struktur besteht aus zwei Feldern und bezeichnet den Namen des Werbemittels in der eine Produktgruppe aufgenommen werden soll und den Namen der

aufzunehmenden Produktgruppe. Beide Felder sind vom Datentyp CHAR (50).

```
BEGIN OF ZCAT_GROUP.

WNAME C(50)

GNAME C(50)

END OF ZCAT_GROUP.
```

7.

8. Implementation von Z_CREATEGROUP

Mit *Springen / F.Baustein Quelltext* kann der ABAP/4 – Programmcode implementiert werden. Der Autor hat die beiden vorherigen Programmcodes zusammengefaßt und in den Funktionsbaustein eingefügt:

```
FUNCTION Z_CREATEGROUP.

*---------------------------------------------------------------------

**Lokale Schnittstelle:

* IMPORTING

* VALUE(GROUP) LIKE ZCAT_GROUP STRUCTURE ZCAT_GROUP

*---------------------------------------------------------------------

Deklaration einer internen Tabelle BDCDAT vom
Batch-Input-Standardtyp BDCDATA. BDCDATA ist standardmäßig im
Data - Dictionary definiert.

DATA BDCDAT LIKE BDCDATA OCCURS 50 WITH HEADER LINE.

Alle Datensätze in Tabelle BDCDAT löschen ( falls vorhanden )

REFRESH BDCDAT. Neues DYNPRO starten

* Werbemittel starten und auswählen

CLEAR BDCDAT.

BDCDAT-PROGRAM = 'SAPLWWMM'.

BDCDAT-DYNPRO = '0020'.

BDCDAT-DYNBEGIN = 'X'.

APPEND BDCDAT.

Einstiegsbild Werbemittel. Werbemittel auswählen

CLEAR BDCDAT.

BDCDAT-FNAM = 'WWMID-WMINR'.

BDCDAT-FVAL = GROUP-WNAME.
```

```
APPEND BDCDAT.
CLEAR BDCDAT.
BDCDAT-FNAM = 'BDC_OKCODE'.
BDCDAT-FVAL = '=BASE'.
APPEND BDCDAT.
```

Neue Bildschirmmaske. GRUNDDATEN WERBEMITTEL

```
CLEAR BDCDAT.
BDCDAT-PROGRAM = 'SAPLWWMM'.
BDCDAT-DYNPRO = '0100'.
BDCDAT-DYNBEGIN = 'X'.
APPEND BDCDAT.
```

In Werbemittel-Layout wechseln

```
CLEAR BDCDAT.
BDCDAT-FNAM = 'BDC_OKCODE'.
BDCDAT-FVAL = '=GLAY'.
APPEND BDCDAT.
```

Neue Bildschirmmaske WERBEMITTEL-LAYOUT

```
CLEAR BDCDAT.
BDCDAT-PROGRAM = 'SAPMSSY0'.
BDCDAT-DYNPRO = '0120'.
BDCDAT-DYNBEGIN = 'X'.
APPEND BDCDAT.
```

Cursor auf oberste Ebene positionieren

```
CLEAR BDCDAT.
BDCDAT-FNAM = 'BDC_CURSOR'.
BDCDAT-FVAL = '02/02'.
APPEND BDCDAT.
```

Neue Werbemittel-Hierarchieebene anlegen

```
CLEAR BDCDAT.
BDCDAT-FNAM = 'BDC_OKCODE'.
BDCDAT-FVAL = 'CREA'.
APPEND BDCDAT.
```

Neue Bildschirmmaske. WERBEMITTEL-Hierarchieebene anlegen

```
CLEAR BDCDAT.
```

```
BDCDAT-PROGRAM = 'SAPLWLYM'.

BDCDAT-DYNPRO = '0200'.

BDCDAT-DYNBEGIN = 'X'.

APPEND BDCDAT.
```

Hierarchieebenen – Name eintragen

```
CLEAR BDCDAT.

BDCDAT-FNAM = 'TWGLVD-LTEXT'.

BDCDAT-FVAL = GROUP-GNAME.

APPEND BDCDAT.
```

Sichern

```
CLEAR BDCDAT.

BDCDAT-FNAM = 'BDC_OKCODE'.

BDCDAT-FVAL = 'SAVE'.

APPEND BDCDAT.
```

Die Transaktion WWM2 aufrufen und Tabelle BDCDATA mitgeben.

Mode E = KEINE BILDSCHIRMAUSGABEN.

Update S = SYNCHRONE VERBUCHUNG

```
CALL TRANSACTION 'WWM2' USING BDCDAT

MODE 'E'

UPDATE 'S'.

ENDFUNCTION.
```

9.

10. Z_CREATEGROUP mit Batch - Bibliothek

Bei Verwendung der Batch - Input – Bibliothek wird der Programmcode des Funktionsbausteins *Z_CREATEGROUP* wesentlich kürzer und übersichtlicher. Hier noch einmal die Implementierung von *Z_CREATEGROUP* mit Verwendung der Batch - Bibliothek:

Hier wird die Bibliothek mit den Batch – Routinen includiert. Der Autor hat diese vom SAP Original *BDCREGXX* in die neu angelegte Bibliothek *ZADBATCH* kopiert, da das Original wegen einiger Header – Deklarationen nicht Funktionsbaustein – tauglich war.

INCLUDE ZADBATCH.

FUNCTION Z_CREATEGROUP.

***---**

```
*Lokale Schnittstelle:

* IMPORTING

* VALUE(GROUP) LIKE ZCAT_GROUP STRUCTURE
ZCAT_GROUP

*---------------------------------------------------------------------

REFRESH BDCDATA.

PERFORM BDC_DYNPRO USING 'SAPLWWMM' '0020'.

PERFORM BDC_FIELD USING 'WWMID-WMINR'
GROUP-WNAME.

PERFORM BDC_FIELD USING 'BDC_OKCODE' '=BASE'.

PERFORM BDC_DYNPRO USING 'SAPLWWMM' '0100'.

PERFORM BDC_FIELD USING 'BDC_OKCODE' '=GLAY'.

PERFORM BDC_DYNPRO USING 'SAPMSSY0' '0120'.

PERFORM BDC_FIELD USING 'BDC_CURSOR' '02/02'.

PERFORM BDC_FIELD USING 'BDC_OKCODE' '=CREA'.

PERFORM BDC_DYNPRO USING 'SAPLWLYM' '0200'.

PERFORM BDC_FIELD USING 'TWGLVD-LTEXT'
GROUP-GNAME.

PERFORM BDC_FIELD USING 'BDC_OKCODE' '=SAVE'.

CALL TRANSACTION 'WWM2' USING BDCDATA

MODE 'E'

UPDATE 'S'.

ENDFUNCTION.
```

11.

12. Aufruf des Funktionsbausteins *Z_CREATEGROUP aus ABAP/4*

Der Funktionsbaustein Z_CREATEGROUP kann in ABAP/4 aufgerufen werden
(Sourcecode - Ausschnitt).

Programmname *ZADCAT05*

```
PROGRAM ZADCAT05 .

Deklaration einer Variablen für den Parameter von Z_CREATEGROUP

DATA: GROUP1 LIKE ZCAT_GROUP.

...

GROUP1-WNAME = 'WADI'.

GROUP1-GNAME = 'Neue Produktgruppe'.

CALL FUNCTION 'Z_CREATEGROUP'

EXPORTING

GROUP = GROUP1

EXCEPTIONS

OTHERS = 1.

ENDCASE.

ENDMODULE.
```

4.
5. **Delphi – RFC - Demo**

Der Aufruf des Funktionsbausteins Z_CREATEGROUP aus Delphi wird in einer kleinen Demonstrations - Anwendung demonstriert. Beim Start der Demonstration erscheint folgendes Programmfenster.

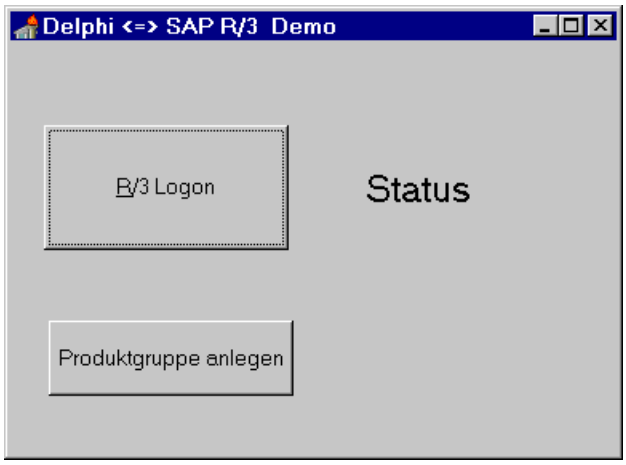


Bild: Delphi - Demo

Mit dem Button *R/3 – Logon* erfolgt die Anmeldung am R/3 – System (wird über *Status* angezeigt). Der Anmeldevorgang ist komplett im ActiveX – Control *SAPLogonControl* von SAP implementiert. *SAPLogonControl* ist eine Komponente, die als anklickbarer *Button* im Fenster erscheint. Nach dem Anklicken erscheint automatisch das R/3 – Standard - Anmeldefenster, in dem die Anmeldedaten eingetragen werden müssen. Diese sind die technischen Daten für die R/3 – Anmeldung (System, Host etc.) und die üblichen User – Daten (Name, Paßwort). Die technischen Daten können auch als Vorgabe eingestellt werden, so daß sie nicht bei jeder Anmeldung abgefragt werden.

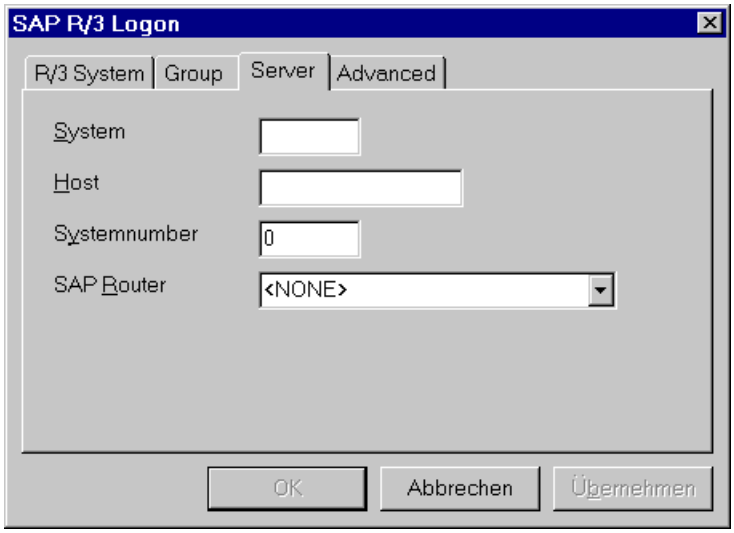


Bild: Das R/3 - Logon

6.

7. Delphi - Anmeldung an R/3 und Aufruf eines RFC - Funktionsbausteins

Es folgt die gesamte Implementierung der Delphi – Demo zur Erstellung einer neuen Produktgruppe (inkl. R/3 - Anmeldung).

```
unit Unit1;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms,
Dialogs,

StdCtrls, comobj, OleCtrls, SAPLogonCtrl_TLB;

type

TForm1 = class(TForm)

Label1: TLabel;

SAPLogonControl1: TSAPLogonControl;

Button2: TButton;

procedure LogonControl1Logon(Sender: TObject; Connection: Variant);

procedure Button1Click(Sender: TObject);

procedure SAPLogonControl1Logon(Sender: TObject;

const Connection: IDispatch);

procedure SAPLogonControl1Logoff(Sender: TObject;

const Connection: IDispatch);

procedure SAPLogonControl1Click(Sender: TObject);

procedure Button2Click(Sender: TObject);

private

{ Private-Deklarationen }

public
```

```
{ Public-Deklarationen }
```

Hier wird die Variable für die aktuelle R/3 – Verbindung angelegt (Global)

```
conn:variant;
end;


var
Form1: TForm1;
implementation
{$R *.DFM}
```

KLICK AUF *R/3 – LOGON* - BUTTON

Die If-Abfrage prüft nur, ob der Aufruf des Befehls LOGON erfolgt ist. Das Ergebnis eines erfolgreichen Aufrufs (Anmeldung erfolgt / nicht erfolgt) wird über die beiden nachfolgenden Events gemeldet.

Die globale Variable *CONN* repräsentiert die aktuelle R/3 – Verbindung.

```
procedure TForm1.SAPLogonControl1Click(Sender: TObject);
begin
conn := saplogoncontrol1.newconnection;
if conn.logon then
begin
// der Aufruf des Befehls LOGON wurde erfolgreich ausgeführt
end else
begin
// der Aufruf des Befehls war fehlerhaft
end;
end;
```

LOGON. Der Status wird auf *Anmeldung erfolgt* gesetzt

```
procedure TForm1.SAPLogonControl1Logon(Sender: TObject;
const Connection: IDispatch);
begin
label1.caption := 'Anmeldung erfolgt !';
end;
```

LOGOFF. Der Status wird auf *Abmeldung* gesetzt

```
procedure TForm1.SAPLogonControl1Logoff(Sender: TObject;
const Connection: IDispatch);
begin
label1.caption := 'Abgemeldet';
```

```
end;
```

KLICK AUF *PRODUKTGRUPPE ANLEGEN*- BUTTON

```
procedure TForm1.Button2Click(Sender: TObject);

var func, functions, group1:variant;

begin
```

Es wird eine Instanz des Zugriffsobjektes *SAPFunctions* erzeugt

```
functions := CreateOLEObject ('SAP.Functions');
```

Die aktuelle Verbindung wird *functions* zugewiesen

```
functions.Connection := conn;
```

Der RFC-Funktionsbaustein *Z_CREATEGROUP* wird instanziert.

```
func := Functions.Add ('Z_CREATEGROUP');
```

Der erste Parameter (*GROUP*) wird referenziert

```
group1 := func.exports.item(1);
```

Da der Parameter *Group* eine Struktur ist, müssen die Felder der Struktur einzeln über die Methode *VALUE* und ihre Namen angesprochen werden.

Bei einfachen Parametern genügt die Angabe von *VALUE* (ohne Feldname dahinter)

```
group1.value('wname') := 'wadi';

group1.value('gname') := 'Delphi-Group';
```

Der RFC-Funktionsbaustein wird aufgerufen

```
If Func.Call then begin label1.caption := 'Funktionsaufruf OK'; end
else label1.caption := 'Fehler bei Funktionsaufruf';end;

end.
```

8.

9. Delphi - Ermittlung aller Attribute eines Klassensystem - Merkmals

Der Vollständigkeit halber wird hier auch noch die Verwendung eines R/3 – Standard – Funktionsbausteins in Delphi demonstriert.

Der Funktionsbaustein *CARD_CHARACTERISTIC_READ* liefert die Eigenschaften eines Klassensystem – Merkmals zurück (inkl. Wertevorrat).

```
procedure TForm1.Button1Click(Sender: TObject);

var row, tab, func, functions:variant;

i:integer;

begin
```

Es wird eine Instanz des Zugriffsobjektes *SAPFunctions* erzeugt

```
functions := CreateOLEObject ('SAP.Functions');
```

Die aktuelle Verbindung wird *functions* zugewiesen

```
functions.Connection := conn;
```

Der R/3 Standard - RFC-Funktionsbaustein *CARD_CHARACTERISTIC_READ* wird instanziert.

```
func      :=      Functions.Add      ('CARD_CHARACTERISTIC_READ');
```

Der zweite Parameter (*Merkmalname*) wird über seine Position in der Parameterliste bewertet. Das Merkmal Test wurde von dem Autor vorher in R/3 angelegt.

```
func.exports.item(2).value := 'TEST';
```

Der Parameter *WITH_VALUES* wird über seinen Namen bewertet. Merkmalwerte zurückgeben

```
func.exports('WITH_VALUES') := 'X';

If Func.Call then begin
```

Merkmalname ausgeben

```
listbox1.items.add( func.tables.item(1).name );
```

Ausgabe der verfügbaren Merkmalwerte des Merkmals *TEST* in einer Listbox

```
for i := 1 to func.tables.rowcount do

begin

row := func.tables.item(1).rows.item(i);

listbox1.items.add( row.value(1) );

end else

label1.caption := 'Fehler bei Aufruf !';

end;

end;
```

10.

11. Bemerkungen zur Delphi - Anbindung

Ein Klick auf den Button *Neue Produktgruppe* erzeugt sofort eine neue Hierarchieebene in der R/3 Komponente *Werbemittel*. Natürlich könnte die Implementation des ABAP/4 - Funktionsbausteins *Z_CREATEGROUP* ebenfalls komplett in Delphi erfolgen. Es kann in Delphi eine interne Tabelle mit der Struktur BDCDATA erzeugt werden, die mit den einzelnen Batch - Schritten zeilenweise gefüllt wird. Anschließend kann diese Tabelle (genau wie in ABAP/4) beim Delphi - Aufruf des Funktionsbausteins CALL TRANSACTION als Parameter übergeben werden. Jedoch zeigen die Code - Beispiele, daß der Zugriff über die SAP - ActiveX - Komponenten zwar problemlos und performant erfolgen kann, jedoch ist er sehr Code - Aufwendig.

Alle R/3 - Funktionalitäten sind in Objekten gekapselt, die vor ihrer Verwendung einzeln instanziert werden müssen. Dieses Prinzip geht so weit, daß sogar jeder einzelne Funktionsbaustein - Parameter ein eigenes

Objekt ist, das vor seiner Verwendung instanziert werden muß. Die teilweise stark ausgeprägte Objekthierarchie der SAP - ActiveX - Komponenten kann für sehr einfache Funktionen lange Ausdrücke erzwingen, die die Lesbarkeit des Quellcodes beeinträchtigen.

Außerdem sollte die RFC - Schnittstelle (bzw. Batch – Schnittstelle) trotz der guten Performance - Erfahrungen des Autors nicht über Gebühr beansprucht werden. Tabellen als Funktionsbaustein - Parameter mit großen Datenmengen und sehr häufige RFC - Aufrufe können zu Performance - Engpässen führen, die die Vorteile einer externen Entwicklungslösung wieder relativieren.

Die vom Autor hier demonstrierte *Arbeitsteilung* (eigentlich das bekannte *Client / Server - Prinzip*) zwischen ABAP/4 und Delphi umgeht das Performanceproblem durch Minimierung der RFC - Aufrufe, erhöht die Lesbarkeit des Delphi - Quellcodes und zwingt nebenbei den Entwickler zu einer sorgsamen Planung der Software - Architektur. Für den Anwender erscheinen solche technische Details völlig transparent (bis auf das gute Laufzeitverhalten der Anwendung).

12.

13. SAP - BAPI – Beispiel: Termine eines SAP – Users ermitteln

Die beiden folgenden SAP - Beispiele demonstrieren die Verwendung der R/3 – Business - Objekte mit dem *BAPI - Control* in Visual Basic von Microsoft. Der Hauptunterschied zur Verwendung der normalen RFC – Funktionsbaustein besteht in der Möglichkeit des syntaktisch direkten Aufrufs der Funktionsbausteine über ihre Methodennamen, ohne eine Objektinstanz FUNCTION zu verwenden.

Objektvariablen deklarieren:

```
Dim oBAPICtrl As Object ' BAPI-Control-Objekt für BOR Zugriff

Dim oConnection As Object ' SAP R/3 Verbindungsobjekt

Dim boSAPusr As Object ' SAP-User, dessen Termine ermittelt werden

Dim boScheduler As Object ' Schedule - Objekte

Dim boSchedule As Object ' Schedule des Users boSAPusr

Dim oAppointments As Object ' Liste der Termine

Dim datToDay as Date ' eine Datumsvariable
```

Business - Objekte werden erzeugt:

```
Set oBAPICtrl = CreateObject(SAP.BAPI.1)
```

Verbindungsobjekt holen:

```
oConnection = oBAPICtrl.Connection
```

R/3 – Benutzerdaten eintragen und an das R/3 – System einloggen:

```
oConnection.User = MyUserName

oConnection.Client = 000

oConnection.Logon
```

SAP User – Objekt holen:

```
Set boSAPusr = oBAPICtrl.GetSAPObject(USR01, Plattner)
```

```
Get the calendar application object:

Set boScheduler = oBAPICtrl.GetSAPObject(SCHEDULER)
```

Kalender des SAP – Users holen:

```
Set boSchedule = boScheduler.ScheduleGet (SCHEDULE, boSAPusr)
```

Termine des SAP – Users holen

```
datToDay = Date

Set oAppointments = boSchedule.AppointmentsGet(datToDay,datToDay)
```

In einer Schleife die Termine aus der Terminliste lesen und in einer Listbox ausgeben

```
Dim str as String

For Each boAppointment In oAppointments

str = Format(boAppointment.TimeFrom, hh:mm

str = str & Format(boAppointment.TimeTo, hh:mm )

str = str & boAppointment.Room &

str = str & boAppointment.Note

lstAppointments.AddItem(str)

Next
```

Business – Objekte wieder freigeben:

```
Set boSAPusr = Nothing

Set boScheduler = Nothing

Set boSchedule = Nothing

Set oAppointments = Nothing
```

Aus R/3 ausloggen und Ressourcen des BAPI – Controls freigeben:

```
oConnection.Logoff

Set oBAPICtrl = Nothing
```

14.

15. SAP - BAPI – Beispiel: Kundenbestellung erzeugen

(ohne An- und Abmeldeprozedur)

Objektvariablen deklarieren:

```
Dim boOrder As Object ' BAPI-Control-Objekt für Zugriff auf das R/3 - BOR

Dim oPartners As Object ' Parameter OrderPartners einer - BAPI Objektmethode

Dim oHeader As Object ' Parameter OrderHeaderIn einer - BAPI Objektmethode

Dim oItemsIn As Object ' Parameter OrderItemsIn einer - BAPI Objektmethode

Dim oReturn As Object ' Parameter Return einer BAPI - Objektmethode
```

Verbindung mit dem Business – Objekt CustomerOrder. Dabei wird eine leere Instanz des Objektes erzeugt (leeres Schlüsselfeld = Objektinstanz-ID) => Business – Objekt erzeugen:

```
Set boOrder = oBAPICtrl.GetSAPObject(CustomerOrder)
```

Business – Objekt – spezifische Strukturen und Tabellen holen:

```
Set oPartners = oBAPICtrl.DimAs(boOrder, CreateFromData, OrderPartners)

Set oHeader = oBAPICtrl.DimAs(boOrder, CreateFromData, OrderHeaderIn)

Set oItemsIn = oBAPICtrl.DimAs(boOrder, CreateFromData, OrderItemsIn)
```

Bestellkopfdaten füllen:

```
oHeader.Value(DOC_TYPE) = TA 'Terminated order

oHeader.Value(SALES_ORG) = 0001 'Sales organization

oHeader.Value(DISTR_CHAN) = 01 'Sales channel

oHeader.Value(DIVISION) = 01 'Business area

oHeader.Value(PO_NUMBER) = 'Customer purchase order number

oHeader.Value(PRICE_DATE) = Now 'Date
```

Kundendaten übergeben:

```
oPartners.Rows.Add

oPartners.Value(1, PARTN_ROLE) = AG

oPartners.Value(1, PARTN_NUMB) = 0000010096 'Kundennummer
```

Bestellposition einfügen:

```
oItemsIn.Rows.Add

oItemsIn.Value(1,REQ_QTY) = 0000000010000 'Menge

oItemsIn.Value(1,MATERIAL) = BERLINER 'Produkt

oItemsIn.Value(1,COND_VALUE) = 1432 'Betrag
```

BAPI - Methode aufrufen (Kundenbestellung erzeugen):

```
boOrder.CreateFromData OrderHeaderIn:=oHeader, _

OrderPartners:=oPartners, _

OrderItemsIn:=oItemsIn, _

Return:=oReturn
```

Ressourcen für Business – Objekt CustomerOrder freigeben:

```
Set boOrder = Nothing
```

Schlußbemerkungen

Zusammenfassend läßt sich sagen, daß diese Arbeit sowohl für die August Rüggeberg GmbH & Co als auch für den Autor als voller Erfolg zu verbuchen ist. In einer guten Zusammenarbeit konnte eine praktikable Lösung ausgearbeitet werden, die im Anschluß an diese Arbeit gemeinsam mit dem Autor, im Rahmen seiner freiberuflichen Tätigkeit, implementiert werden wird. Die in dieser Arbeit gestellte Aufgabenstellung war nur ein erster konkreter Schritt der August Rüggeberg GmbH & Co auf dem Weg zum Aufbau einer zentralen, medienunabhängigen Datenbank, die nicht nur zur Katalogproduktion dient, sondern über das Internet weltweit eine

Wissens- und Expertendatenbank für die Mitarbeiter und Partner des Unternehmens zur Verfügung stellen soll. Die ständige Erweiterung der Datenbank um *Anwendungen der Produkte*, der Aufbau eines internen *Data Warehouse* und die Pflege und Auswertung von *Marktdaten über Mitbewerber* zeigen die noch auszuschöpfenden Potentiale auf. Eine vom Autor gerne gewünschte, längerfristige Zusammenarbeit mit der August Rüggeberg GmbH & Co zeichnet sich, bei weiterem erfolgreichem Projektverlauf, deutlich ab.

Der Autor konnte sein Wissensspektrum während dieser Arbeit stark erweitern. Dies beschränkt sich nicht nur auf R/3 – spezifische technische Details, sondern meint vielmehr die Konzepte und Prinzipien bei der Abbildung und Integration von komplexen Geschäftsprozessen in Softwaresystemen.

Zum Abschluß möchte der Autor als Empfehlung an R/3 interessierte Leser bemerken, daß es bei der Einarbeitung und Erkundung des R/3 – Systems kein *Geheimnis* gibt. Fundierte Kenntnisse über relationale Datenbanksysteme, die gute Beherrschung einer beliebigen Programmiersprache, die Aufnahmefähigkeit für komplexe Zusammenhänge und die Bereitschaft, viel Zeit am PC zu verbringen, sind das Rezept. Als Fallstrick zeigte sich immer wieder die zu intensive Betrachtung einzelner technischer Details. Die teilweise sehr hohe Integration von R/3 erschwert die Handhabbarkeit zusätzlich. Für den Anfang ist eine zuerst oberflächliche Einarbeitung in das Gesamtsystem nützlicher, als die isolierte und detaillierte Betrachtung einzelner Komponenten. Nach den ersten Lichtblicken stellte sich beim Autor die, eigentlich selbstverständliche, Erkenntnis durch, daß auch SAP *nur mit Wasser kocht* . Also, Zähne zusammenbeißen und durch.

Adnan Duman

Oktober 1998, Hürth bei Köln

Anhang

1. **Katalogseiten der August Rüggeberg GmbH & Co**

Literaturverzeichnis

Internet: WWW.SAP.COM/BAPI

SAP R/3 Rev. 3.1h Online Dokumentationen, insbesondere:

- R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench.
- R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench | ABAP/4 - Benutzerhandbuch.
- R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench | ABAP/4 - Dictionary.
- R/3 – Bibliothek | Basis | ABAP/4 – Development – Workbench | Basis – Programmierschnittstellen | Daten mit Batch – Input übernehmen
- BAPI Einführung *caalbd.hlp*

- BAPI Programmierleitfaden *caalcd.hlp*