Obeco GmbH

KONZEPT UND AUFBAU

Policom – Datenbankstruktur

Stand: Juni 2026

Obeco GmbH
Arenberger Str. 230a, 56077 Koblenz

Policom – Datenbankstruktur

Dieses Dokument beschreibt Konzept und Aufbau der Policom-Datenbankstruktur. Es zeigt, wie eine konsequente Standardisierung von Datenbank, Oberfläche und Programmcode die Entwicklung kommerzieller Anwendungen beschleunigt und wartbarer macht. Der Text setzt keine speziellen Kenntnisse in Programmierung oder Datenbankdesign voraus und eignet sich daher auch für Anwendungsarchitekten, Führungskräfte und Entscheider in Entwicklungsabteilungen.

Am Anfang war …

Meine ersten Programmierschritte machte ich 1977/78 während meines BWL- und Statistikstudiums in Mannheim. Erste praktische Erfahrungen mit der Programmierung sammelte ich Anfang der 1980er-Jahre in der Qualitätssicherung der Robert Bosch GmbH in Erbach im Odenwald – damals noch auf einer Tandem Nonstop (HP) unter FORTRAN und COBOL. Ab 1985 begann ich, mit SQL-Datenbanken auf dem Host und dem PC (dBASE) zu arbeiten, und erstellte für meinen Arbeitgeber und später für meine Kunden Datenbankanwendungen (MS SQL Server).

Mitte der 1990er-Jahre stellte ich fest, dass sich große Teile meiner Arbeit wiederholten und meine Produktivität darunter litt: Meiner Erfahrung nach wiederholten sich in jedem Projekt 30 bis 50 % der Arbeiten für Standardfunktionen. Um meine Arbeit kostengünstiger (ohne sich wiederholende Programme) und effektiver (schneller) anbieten zu können, entstand das Konzept und die Struktur, die ich im Folgenden beschreibe. Unsere Postulate sind im Text besonders gekennzeichnet, damit Sie sie leichter wiederfinden.

Einführung

Grundlage aller Programmierarbeiten in kommerziellen Anwendungen, die Daten in Datenbanken speichern, sind nach wie vor zwei Komponenten:

- das **Datenbankdesign** – wo werden die Daten gespeichert?
- die **Datenbankoperationen** – wie wird mit ihnen umgegangen?

Datenbankdesign

Datenbanken werden mithilfe des relationalen Datenbankmodells von Codd definiert. Zum weiteren Verständnis genügt es zu wissen, dass es Tabellen gibt (z. B. KUNDEN, ARTIKEL), in denen Daten gespeichert werden. Jede Tabelle hat einen sogenannten Primärschlüssel (KUNDENNUMMER), der einen Kunden eindeutig identifiziert. Tabellen können aufeinander verweisen: So hat die Tabelle AUFTRÄGE zunächst einen Primärschlüssel (AUFTRAGSNUMMER), aber auch einen Fremdschlüssel (KUNDENNUMMER), der in die Tabelle KUNDEN verweist und bestimmt, zu welchem Kunden ein Auftrag gehört.

Datenbankoperationen

Um mit diesen Tabellen arbeiten zu können, benötigt man Methoden, um die Daten zu lesen und zu schreiben. Es gibt folgende atomare Methoden:

- einen vorhandenen Satz lesen (**read**)
- einen neuen Satz anlegen (**write**)
- einen vorhandenen Satz verändern (**update**)
- einen vorhandenen Satz löschen (**delete**)

Lässt man die Teile der jeweiligen Businesslogik (wann darf ein Satz wie geändert werden) zunächst beiseite – wir kommen später darauf zurück –, reduziert sich jede Operation in der Datenbank auf diese vier Standardfunktionen (RWUD).

Verfahren bisher

Im Weiteren betrachten wir alle Phasen, wie man diese Informationen gewinnt, als gegeben, um uns auf den Umgang mit den Daten zu konzentrieren. Der nächste Schritt in der Anwendungsentwicklung besteht darin, die Datenbankstruktur zu definieren und die Anwendungslogik umzusetzen:

- **Schritt 1:** Die Tabellen, Primär- und Fremdschlüssel sowie Eigenschaften werden bestimmt und gemäß dem Codd'schen Modell umgesetzt und angelegt.
- **Schritt 2:** Die Programmlogik wird erarbeitet und in den Bereich Interaktion (was passiert auf dem Bildschirm) und den Bereich Reaktion (welcher Programmcode wird ausgeführt) gegliedert.
- **Schritt 3:** Die Bildschirmeingabe wird realisiert. Sie beschreibt, welche Daten angezeigt werden und welche Operationen darauf möglich sind.
- **Schritt 4:** Der Programmcode, der die gewünschte Operation ausführt, wird programmiert. Dabei unterscheiden wir zwischen der Businesslogik (Programmlogik der jeweiligen Anwendung) und den Standardfunktionen (Lese- und Schreiboperationen).

Nachteile

Die herkömmliche Vorgehensweise hat folgende Nachteile:

- Auch bei starker Standardisierung müssen große Teile der Benutzeroberfläche immer wieder neu programmiert werden.
- Das Entwerfen der Anzeigeformulare, der Auswahl- und Eingabefelder sowie der Operationen ist aufwendig.
- Das Anpassen dieser Komponenten bei Änderungen und Ergänzungen ist zeitintensiv.
- Die Standardfunktionen für neue Tabellen müssen immer wieder neu programmiert werden.

Ursachen

Die Ursachen liegen einerseits in der mangelnden Abstraktion der verwendeten Komponenten. Da jede Tabelle eine individuelle Struktur hat, ist es erforderlich, individuelle Programmteile für den Umgang mit dieser Tabelle zu erstellen. Jede Änderung an der Tabellenstruktur zieht eine

Änderung der Programmstruktur nach sich, was wiederum einen neuen Zyklus aus Test, Auslieferung, Installation und Verteilung des geänderten Programms bedeutet.

Zudem führt ein freier Umgang mit den Benutzeroberflächen dazu, hier noch einen Button hinzuzufügen und dort noch ein Event einzubauen. Das macht ein Formular im Laufe der Zeit unübersichtlich und schwer zu warten.

Ziele

Eine effiziente Programmarbeit erreichen wir, indem wir möglichst viele der genutzten Komponenten vereinheitlichen und so mehrfach verwendbar machen. Dies erreichen wir, indem wir:

- die Datenbanken standardisieren
- die Oberflächen standardisieren
- die Codebasis standardisieren

Datenbank

Ein großes Problem beim Datenbankdesign ist immer die Frage nach dem richtigen Primärschlüssel: Wer ist es, woher kommt er, bleibt er Primärschlüssel? Haben Sie sich schon einmal in eine fremde, neue Datenbank einarbeiten müssen? Wie viel Zeit vergeht, bis Sie die Struktur erfasst haben und sich zurechtfinden?

Oft ist im ersten Ansatz eines Projekts das Auffinden des richtigen Primärschlüssels sehr zeitaufwendig. Beim traditionellen Datenbankdesign ist es jedoch erforderlich, da dieser Schlüssel in die Beziehungen zu anderen Tabellen eingeht und daher zuerst festgelegt werden muss, um die Relationen weiter zu definieren. Ein nachträgliches Ändern ist sehr aufwendig.

> **Postulat:** In Erweiterung des Codd’schen Datenbankmodells wird jede Tabelle um einen von der Anwendung vergebenen Primärschlüssel ergänzt. Der bisher benutzte Primärschlüssel kann zusätzlich als eindeutiger Schlüssel weiterbestehen. Besitzt eine Tabelle einen Fremdschlüssel, wird der Name des Fremdschlüssels aus dem Tabellennamen der Herkunftstabelle abgeleitet.

So führt die neue Vorgehensweise dazu, dass bereits nach dem Sammeln der Attribute zu einer Tabelle (Entität) ein Grundkonzept vorliegt und in der Datenbank angelegt werden kann. Die weiteren Feinheiten des Entwurfs lassen sich zu einem späteren Zeitpunkt nacharbeiten.

> **Beispiel:** Die Tabelle KUNDEN hat den Primärschlüssel REFNR. Die Tabelle AUFTRÄGE hat ebenfalls den Primärschlüssel REFNR; der Fremdschlüssel (ursprünglich die Kundennummer) ist jetzt das Feld REFNR_KUNDEN. Aus den Schlüsselnamen erschließt sich die Struktur unmittelbar, etwa:

- **firmen:** REFNR (Primärschlüssel), REFNR_MANDANTEN, REFNR_WAEHRUNGEN, REFNR_MWST

- **auftraege:** REFNR (Primärschlüssel), REFNR_FIRMEN, REFNR_ANSCHRIFTEN, REFNR_ARTIKEL
- **waehrungen:** REFNR
- **mwst:** REFNR

Mit dieser Logik erschließt sich jedem Datenbankdesigner sofort die Struktur der gesamten Datenbank. Er „sieht“ an der Tabellenstruktur – und später im Programm – welche Daten aus anderen Tabellen benutzt werden. Eine kryptische Namensgebung verschwindet zugunsten der Klarheit von Sprache und Inhalt. Das wird zu einem großen Vorteil bei Datenbanken, die mehrere Hundert Tabellen umfassen und von denen der Datenbankdesigner nur einen Ausschnitt zur Verfügung gestellt bekommt. Im Programmcode – den man unter Umständen ohne Datenbankdiagramm erhält – sieht man sofort, welche Felder (Attribute) einer Tabelle aus anderen Tabellen stammen.

Oberfläche

Sie haben es bestimmt auch schon erlebt: Sie öffnen ein Formular und werden von der Anzahl der Eingabemöglichkeiten und der Funktionsvielfalt erschlagen. Jedes Formular unterscheidet sich vom anderen – oder das ganze Programm besteht aus einem einzigen Formular, das alle realisierten Funktionen versammelt. Fragen Sie sich:

- Hat dieser Entwickler an den Endbenutzer gedacht?
- Wie viel Zeit muss man für Schulungen aufwenden, bis der Anwender diese Logik versteht?
- Wie viel Zeit muss der Entwickler aufbringen, um ein solches Programm zu pflegen?
- Ist das veränderte Eventverhalten einer neuen Programmversion noch kalkulierbar?

> **Postulat:** So ist es einfacher – die Oberfläche wird in zwei Funktionsgruppen aufgeteilt, die die Navigations- und Anzeigeoperationen von den Änderungsoperationen trennen. Das vereinfacht die gesamte Formularverwaltung für den Programmierer und hilft dem Anwender, das Formular schneller zu verstehen.

Navigieren und Anzeigen

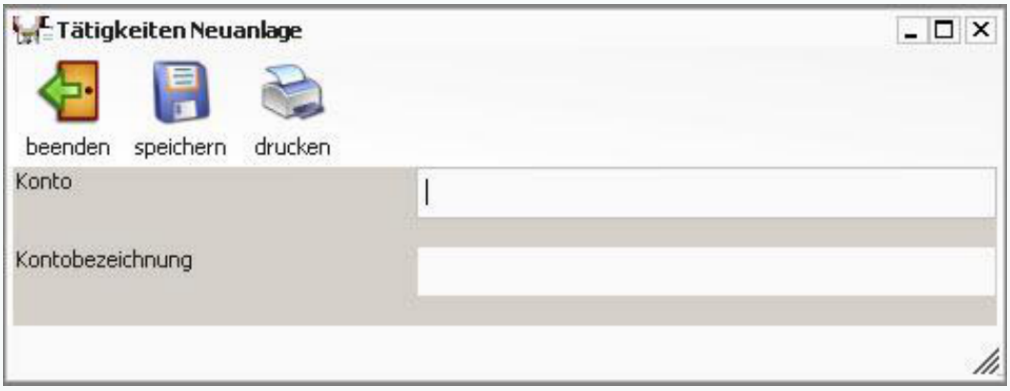
Zunächst wird der Weg des Anwenders zu „seinen“ Daten bestimmt. In grafischen Umgebungen erfolgt dies über die Menüs oder eine Buttonleiste. Die Anzeige der gesuchten Daten erfolgt tabellarisch:

Anzeigeformular: Die Daten werden tabellarisch dargestellt, Funktionen liegen auf der Buttonleiste.

Das Formular zeigt die Daten lediglich an. Änderungen an den Daten oder andere Funktionen (Drucken, Abrechnen ...) werden über Funktionstasten oder Buttons aufgerufen. Die Standardfunktionen sind leicht erkennbar – jeder weiß sofort, welche Aufgabe sie erfüllen. Alle Formulare arbeiten nach demselben Prinzip: Wer ein Formular bedienen kann, kann auch alle anderen bedienen. Unserer Erfahrung nach ist der Funktionsumfang pro Arbeitsschritt in einem Geschäftsprozess auf fünf bis sieben Funktionen pro Formular beschränkt. So bleibt das Formular immer übersichtlich und lässt sich einfach erklären.

Dateneingabe

Werden Daten neu erfasst oder geändert, erfolgt dies in einem separaten Eingabeformular:



Eingabeformular: Es zeigt nur die im Kontext erforderlichen Funktionen.

Auch dieses Formular zeigt nur die in diesem Kontext erforderlichen Funktionen: Entweder Sie verlassen das Formular oder Sie speichern Ihre Änderungen. Lediglich das Ausdrucken des Datensatzes zur Dokumentation oder Weitergabe ist zusätzlich vorgesehen. So kann sich der Anwender ganz auf diese eine Interaktion konzentrieren, ohne durch in diesem Moment überflüssige Informationen abgelenkt zu werden – der Workflow wird eindeutig.

Kernel

Die meisten Einsparungen an doppelter Programmierarbeit ergeben sich bei der Anwendung dieser Methodik im Programmcode. Die folgende Struktur bildet die Standardfunktionalitäten im Kernel unserer Anwendung ab. Im Kernel sind alle Standardfunktionen zur Navigation, Anzeige und Dateneingabe implementiert.

Navigation

Da alle Tabellen einer einheitlichen Konvention für Primär- und Fremdschlüssel folgen, ist die Navigation zwischen den Tabellen vereinheitlicht. In der Sprache der Entwickler: Ein Stück Programm realisiert die gesamte Navigation der Anwendung. Das Navigationsmodul wird einmal programmiert und in allen Anwendungen genutzt.

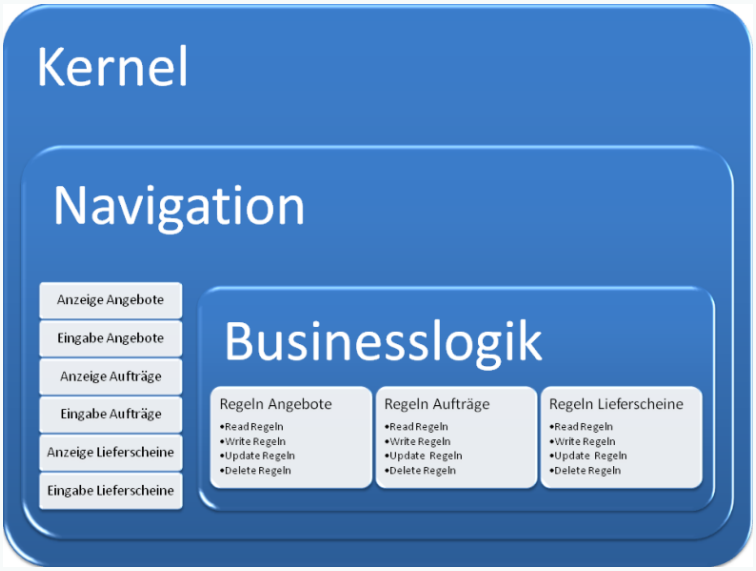
> **Beispiel:** Sie befinden sich in der Tabelle der Kunden und möchten zur Tabelle der Aufträge wechseln (Button „Aufträge“ auf dem Formular Kunden). Das Modul Navigation erhält als Eingabe den aktuellen Satz (Kunde A) und die Zieltabelle (Aufträge von Kunde A) und weiß somit, welche Daten anzuzeigen sind. Es ruft das Anzeige- oder Eingabeformular mit den ermittelten Daten auf.

Datenanzeige

Da alle Anzeigeformulare einer einheitlichen Logik folgen, wird auch der Programmcode zur Datenanzeige vereinheitlicht. Er wird nur einmal realisiert und kann im Programm an jeder Stelle und in allen Anwendungen dieser Aufbaustruktur genutzt werden. In Fortführung des obigen Beispiels erhält das Anzeigeformular von der Navigation die Information über die Zieltabelle sowie den Wert des Fremdschlüssels, verfügt damit über alle benötigten Informationen und zeigt die gesuchten Daten an.

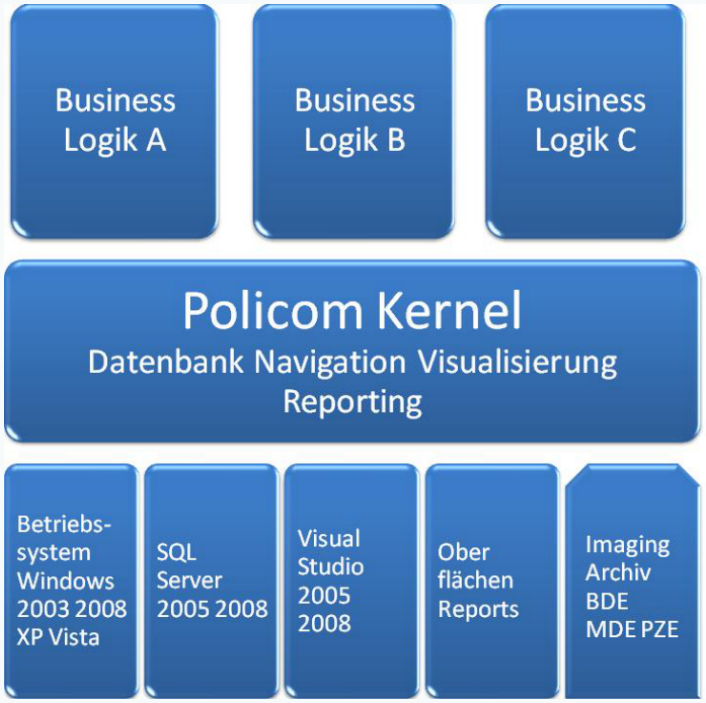
Dateneingabe

Da alle Eingabeformulare ein einheitliches Format haben, wird auch der Programmcode zur Dateneingabe vereinheitlicht. Als Ergebnis entsteht ein von der Datenbank unabhängiger Programmcode. Dieser transportiert die Daten zwischen Formular und Datenbank, ohne die tatsächliche Datenstruktur zu kennen.



Aufbau des Kernels: Navigation und Businesslogik mit den Anzeige- und Eingabeformularen.

Im Kernel sind alle Funktionen zur Navigation zwischen den Formularen implementiert. Anzeige- und Eingabeformular werden von der Navigation mit den Daten versehen, die sie zur Generierung der Anzeige benötigen. Vor jeder Datenoperation wird in die Businesslogik verzweigt, um zu prüfen, ob es Geschäftsregeln für diesen Vorfall gibt. Falls ja, werden diese ausgeführt. Ist die Regelprüfung erfolgreich, wird die Verarbeitung fortgesetzt; andernfalls wird die Steuerung an die aufrufende Komponente zurückgegeben. Aus Sicht des Betriebssystems stellt sich diese Struktur wie folgt dar:



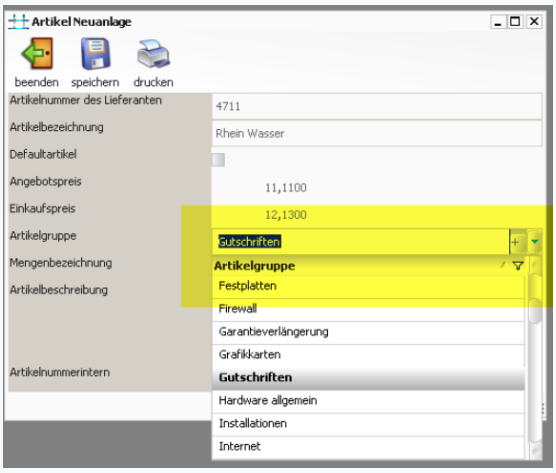
Systemsicht: Der Policom-Kernel setzt auf Betriebssystem, SQL Server und Entwicklungswerkzeugen auf und trägt die Businesslogik.

Dynamisierung

In jeder Anwendung werden nicht immer alle Daten einer Tabelle angezeigt, sondern nur die, die wir sehen möchten. Wie erreichen wir es, dass genau diese Daten angezeigt werden und nicht alle? Da Navigation, Anzeigeformular und Eingabeformular anwendungsneutral formuliert sind, kommt die Information zur Steuerung von Navigation und Anzeige aus der Datenbank.

Navigation: In der Datenbank werden zunächst der Aufbau der Navigation (woher – wohin) und die auf den Formularen benötigten Buttons in einer Tabelle hinterlegt. Navigation und Buttons werden dann zur Laufzeit dynamisch generiert. Das ermöglicht es, den Programmablauf sogar mitten im Betrieb vor Ort beim Kunden zu ändern oder zu ergänzen – ohne eine Zeile zu programmieren.

Anzeige- und Eingabeformular: Auch die Beschreibungen der Daten, die auf dem Anzeige- oder Eingabeformular darzustellen sind, liegen in der Datenbank. Die Navigation gibt dem jeweiligen Formular die benötigten Informationen aus der Datenbank mit, sodass das Formular den Bildschirmaufbau dynamisch generieren kann. Auf den Eingabeformularen werden Auswahllisten für Daten aus anderen Tabellen verwendet, um den Zugriff zu beschleunigen. Da der Verweis zwischen den Tabellen standardisiert ist und die benötigten Schlüssel bekannt sind, lässt sich auch die Funktionalität dieser Auswahllisten standardisieren:



Eingabeformular mit standardisierter Auswahlliste für Daten aus einer anderen Tabelle (Artikelgruppe).

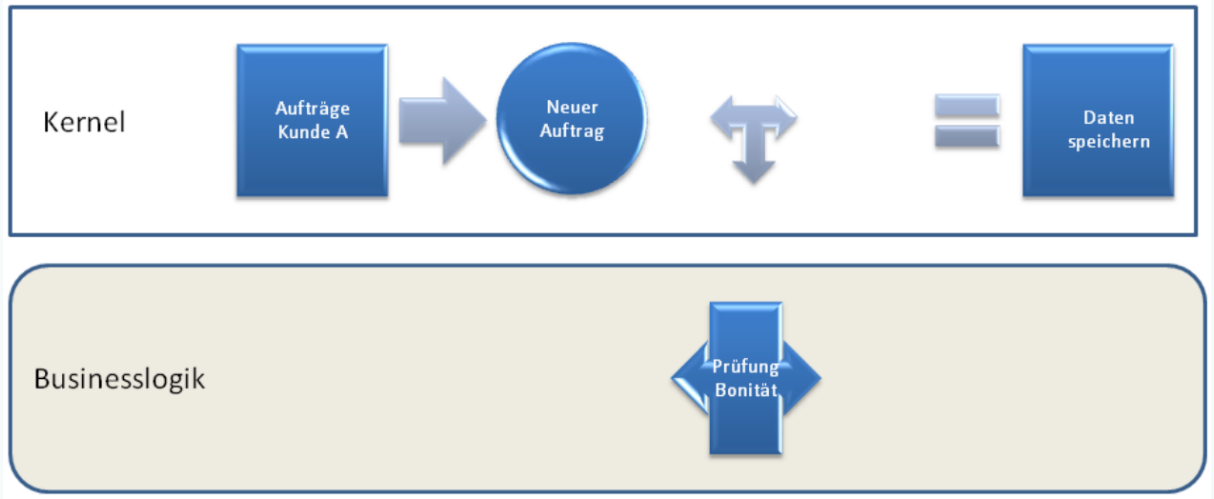
Auch hier ist es möglich, die angezeigten Formulare mitten im Betrieb vor Ort beim Kunden zu ändern oder zu ergänzen – ohne eine Zeile zu programmieren. Diese Struktur erlaubt es, Änderungen an der Datenbankstruktur vorzunehmen, ohne das Programm ändern zu müssen (dies gilt für ergänzende Felder, die keiner logischen Feldprüfung unterzogen werden müssen).

Businesslogik

Jede Anwendung hat ihre eigene Logik, die Businesslogik. Sie ist von Betrieb zu Betrieb verschieden und muss – wie bisher auch – kundenspezifisch angepasst werden. Wie wird dieser Programmcode integriert? Der Standardcode von Navigation, Anzeigeformular und Eingabeformular stellt zwischen den Formularen sowie vor und nach Datenbankinteraktionen Schnittstellen bereit. Diese werden von der Businesslogik genutzt, um die Verarbeitung zu

steuern und zu regeln.

Beispiel: Im Anzeigeformular der Aufträge von Kunde A wird der Button „Neuer Auftrag“ geklickt. Die Navigation ruft den Programmcode zur Generierung des Erfassungsformulars auf und verzweigt zuvor in die Businesslogik. Dort wird geprüft, ob der Kunde im Limit liegt und der neue Auftrag erfasst werden kann. Falls ja, wird die weitere Verarbeitung freigegeben.



Zusammenspiel von Kernel und Businesslogik: Vor dem Speichern verzweigt die Verarbeitung in die Bonitätsprüfung.

Dieselbe Vorgehensweise kommt beim Speichern des neuen Auftrags zum Einsatz: Die Navigation ruft die Standardoperation zum Speichern der Daten auf, die in die Businesslogik verzweigt, um die erforderlichen Prüfungen vorzunehmen (z. B. ob die Mindestbestellmenge berücksichtigt wurde). Die Businesslogik arbeitet ihre Regeln ab und gibt den Satz entweder zum Speichern frei oder zur Überarbeitung an das Erfassungsformular zurück. Auf dieselbe Weise werden Funktionen implementiert, die die Standardkomponenten nicht abdecken: Ist der Navigation eine bestimmte Funktion oder ein Formular nicht bekannt, verzweigt sie in die Businesslogik, die die weitere Verarbeitung übernimmt.

Synergien

Von erheblichem Vorteil dieser Struktur sind die Synergien, die zwischen den Anwendungen entstehen.

Beispiel: Kunde A benötigt eine spezielle Filteroperation, um im Anzeigeformular seine Daten zu selektieren. Wird diese Änderung im Kernel implementiert, steht die Funktion automatisch allen anderen Kunden zur Verfügung. Umgekehrt lassen sich so auch größere Module (etwa die Archivierung) kostengünstig auf mehrere Schultern verteilen.

Prototypen

In einer frühen Projektphase ist es möglich, dem Kunden bereits mit dem ersten Datenbankentwurf und einem groben Ablaufplan ein Look-and-Feel der Anwendung zu

vermitteln – noch bevor die erste Zeile programmiert wurde. So lassen sich Schwachstellen im Entwurf oder im Workflow frühzeitig erkennen und beheben.

Migration

Eine bestehende Datenbank in das neue Format zu migrieren, ist mit wenigen Änderungen an der vorhandenen Struktur machbar: Es werden lediglich die neuen Primärschlüssel hinzugefügt und die Beziehungen angepasst.

Code-Qualität

Von weiterer Bedeutung sind die Standardkomponenten in Wartung und Betrieb. Einmal getestet und verifiziert, kann man sich auf stabilen Code verlassen. Das reduziert den Testumfang in der Realisierungsphase erheblich.

Zusammenfassung

Zusammengefasst umfasst das Konzept folgende Aktionen:

- die Datenbankstruktur normieren
- die Oberflächen normieren
- die Navigation standardisieren
- die Datenanzeige standardisieren
- die Datenänderungen standardisieren

Über die Obeco GmbH

Die Obeco GmbH entwickelt seit 1993 Lösungen für den IT-Bereich. Unsere Schwerpunkte sind der System Support, die Produktentwicklung sowie der betriebliche Datenschutz und der IT-Grundschutz. Unsere Kunden kommen aus Industrie, Handel und Dienstleistung; wir arbeiten für Kleinbetriebe und Mittelständler mit bis zu 2.500 Arbeitsplätzen.

Policom

Angepasst an Ihre individuellen Betriebsabläufe liefern wir mit Policom ein modular aufgebautes System, das Ihre Anforderungen an die Automatisierung Ihrer Geschäftsprozesse erfolgreich umsetzt. Policom wächst mit Ihnen und passt sich flexibel ändernden Betriebsabläufen an. Eine benutzerfreundliche Bedienoberfläche erleichtert auch in komplexem Kontext die Einarbeitung und die zielsichere Anwendung. Wir bieten effiziente Lösungen für:

- den Vertrieb (CRM)
- die Auftragsabwicklung (Lieferschein- und Rechnungsschreibung, Mahnwesen)
- den Einkauf
- die Lagerhaltung

- Stammdaten und Stücklisten
- die Produktion (BDE, MDE, PZE, MES)
- die Projektsteuerung
- Workflow und Archivierung

Nutzen Sie folgende Vorteile von Policom:

- Durch die Kombination standardisierter Module mit kundenspezifischen Erweiterungen verbinden wir die Vorteile von Standardsoftware mit denen der Individualentwicklung.
- Das gewährleistet eine kurze Time-to-Production und damit eine zügige, zielgenaue Realisierung Ihrer Anforderungen.
- Durch die enge Verknüpfung mit Ihren Betriebsabläufen sichern wir Ihre Wettbewerbsvorteile.

Wir setzen ausschließlich aktuelle Microsoft-Entwicklungswerkzeuge ein, die eine zuverlässige Integration in Ihre Umgebung gewährleisten. Bei Bedarf lizenzieren Sie die Policom-Kernelkomponenten und entwickeln individuelle Erweiterungen mit Ihren eigenen Mitarbeitern – das garantiert weitreichende Freiheiten bei der Anpassung von Funktionalität und Oberfläche.

> Sie möchten mehr über Policom erfahren? Wir beraten Sie gerne unter 0261-9634466-0 oder info@obecosupport.de.