

Policom Datenbankstruktur

Konzept und Aufbau



Impressum:

Obeco GmbH
Nassauer Ring 26b
56422 Wirges

Telefon : 02602-919-6870

Fax: 02602-919-6877

Mail: info@obeco.de

Web: www.obeco.de

Geschäftsführer: Joachim Berger, Amtsgericht Montabaur, HRB 21022

Inhaltlich verantwortlich gemäß § 55 Absatz 2 RStV: Joachim Berger (Anschrift wie oben)

Umsatzsteuer-Identifikationsnummer gemäß § 27 a Umsatzsteuergesetz: DE159643101

Kto.-Nr. 650005, BLZ 505 300 05, Cronbank Dreieich

Stand: 24.01.2011 16:59:00

© Obeco GmbH alle Rechte vorbehalten

Inhaltsverzeichnis:

Inhaltsverzeichnis.....	3
Am Anfang.....	4
Einführung.....	5
Datenbankdesign.....	5
Datenbankoperationen.....	5
Verfahren bisher.....	5
Nachteile.....	6
Ursachen.....	6
Ziele.....	6
Datenbank.....	7
Oberfläche.....	9
Navigieren und Anzeigen.....	9
Dateneingabe.....	10
Kernel.....	11
Zum Ende.....	16
Obeco GmbH.....	16
Policom.....	16

Am Anfang


... tat ich meine ersten liebevollen Programmierschritte in den Jahren 1977/78 während meines BWL/Statistik Studiums in Mannheim. Die ersten praktischen Erfahrungen mit der Programmierung habe ich Anfang der 80er Jahre für die Qualitätssicherung der Robert Bosch GmbH in Erbach im Odenwald gemacht. 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 dann für meine Kunden Datenbankanwendungen (MS SQL-Server).

Mitte der Neunziger Jahre begann ich festzustellen, dass große Teile meiner Arbeiten sich wiederholen und meine Produktivität verschlechterten. Meiner Erfahrung nach wiederholten sich in jedem Projekt 30-50% der Arbeiten für Standardfunktionen!

Um meine Arbeit kostengünstiger (keine sich wiederholenden Programme) und effektiver (schneller) anbieten zu können entstand das Konzept und die Struktur wie ich sie im Folgenden beschreiben werde.

Dieser Text erfordert keine speziellen Kenntnisse in der Programmierung oder im Datenbankdesign und ist daher auch für Anwendungs-Architekten, Führungskräfte und Entscheider in Entwicklungsabteilungen geeignet.



Unsere Postulate sind besonders gekennzeichnet um sie im Text besser zu finden.

Einführung

Grundlage aller Programmierarbeiten in kommerziellen Anwendungen die Daten in Datenbanken speichern sind nach wie vor 2 Komponenten:

- Das Datenbankdesign (wo werden die Daten gespeichert)
- Die Datenbankoperationen (wie wird mit ihnen umgegangen)

Datenbankdesign

Datenbanken werden mit Hilfe des relationalen Datenbankmodells von Codd definiert. Zum weiteren Verständnis genügt es zu wissen, dass es Tabellen (z.B. KUNDEN, ARTIKEL,) gibt in denen Daten gespeichert werden. Jede Tabelle hat einen sogenannten Primärschlüssel (KUNDENUMMER), der einen Kunden eindeutig identifiziert.

Tabellen können aufeinander verweisen: So hat die Tabelle AUFTRÄGE zunächst einen Primärschlüssel (AUFTRAGSNUMMER) aber auch einen Fremdschlüssel (KUNDENUMMER) 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 atomaren Methoden:

1. einen vorhandenen Satz lesen (read)
2. einen neuen Satz anlegen (write)
3. einen vorhanden Satz verändern (update)
4. einen vorhandenen Satz löschen (delete)

Lässt man die Teile der jeweiligen Businesslogik (wann darf ein Satz wie geändert werden) zunächst zur Seite (wir werden später darauf kommen) reduziert sich jede Operation in der Datenbank auf diese 4 Standardfunktionen (RWUD).

Verfahren bisher

Im Weiteren werden wir alle Phasen wie man diese Informationen gewinnt als gegeben betrachten 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, Eigenschaften werden bestimmt und gemäß dem Codd-schen Model umgesetzt und angelegt.

Schritt 2

Die Programmlogik wird erarbeitet und in den Bereich Interaktion (was passiert auf dem Bildschirm) und in 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. Hierbei unterscheiden wir zwischen der Businesslogik (Programmlogik der jeweiligen Anwendung) und 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 dieser Vorgehensweise 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 produzieren.

Jede Änderung an der Tabellenstruktur hat eine Änderung der Programmstruktur zur Folge, was wiederum in einen neuen Zyklus für Test / Auslieferung / Installation und Verteilung des geänderten Programmes führt.

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

Ziele

Eine effiziente Programmarbeit erreichen wir indem wir möglichst viele der benutzen Komponenten vereinheitlichen und damit eine mehrfache Verwendbarkeit erreichen.

Dies erreichen wir indem wir:


1. die Datenbanken standardisieren
2. die Oberflächen standardisieren
3. die Codebasis standardisieren

Datenbank

Ein großes Problem beim Datenbankdesign ist immer die Frage nach dem richtigen Primärschlüssel, wer es ist, woher er kommt, ob er Primärschlüssel bleibt.

Haben Sie sich schon in eine fremde und neue Datenbank einarbeiten müssen? Wie viel Zeit vergeht bis Sie die Struktur aufgenommen haben und sich zu Recht zu finden?

Oft ist im ersten Ansatz des Projektes das Auffinden des richtigen Primärschlüssels sehr zeitaufwendig. Dies ist jedoch erforderlich da beim traditionellen Datenbankdesign 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.



In Erweiterung des Codd'schen¹ Datenbankmodells wird jede Tabelle ergänzt um einen vom der Anwendung vergebenen Primärschlüssel. Der bisher benutzte Primärschlüssel kann zusätzlicher als eindeutiger Schlüssel weiterbestehen. Besitzt eine Tabelle einen Fremdschlüssel wird der Name des Fremdschlüssels aus dem Tabellennamen der Herkunftstabelle abgeleitet.

Im oben beschriebenen Umfeld führt die neue Vorgehensweise dazu, dass nach der Sammlung der Attribute zu einer Tabelle (Entität) ein Grundkonzept der Tabelle vorliegt und in der Datenbank angelegt werden kann.

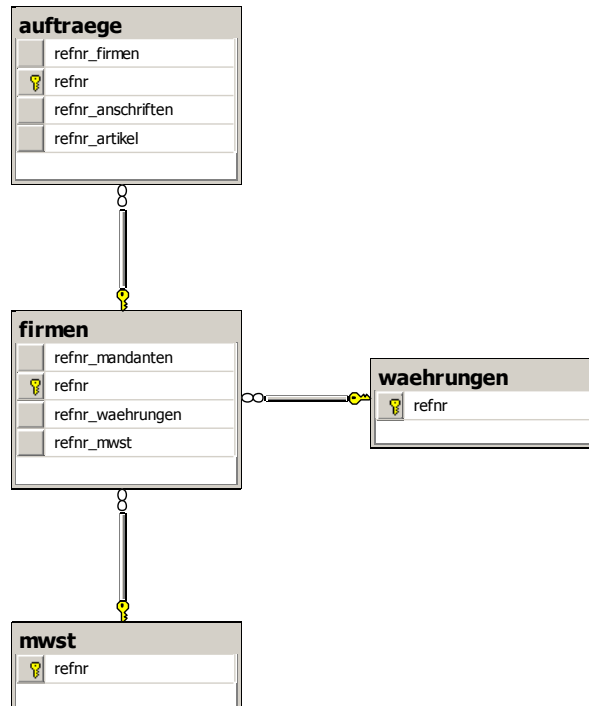
Die weiteren Feinheiten des Entwurfs können zu einem späteren Zeitpunkt nachgearbeitet werden.

¹ In Anlehnung an eine ähnliche Konstellation sprechen wir bei Codd von der „normalen relationalen Datenbank Theorie“ und dem hier vorgestellten Konzept von der „speziellen relationalen Datenbank Theorie.“

Beispiel:

Die Tabelle KUNDEN hat den Primärschlüssel REFNR.

Die Tabelle AUFTRÄGE hat auch den Primärschlüssel REFNR. Der Fremdschlüssel (ursprünglich die Kundennummer) ist jetzt das Feld RFNR_KUNDEN:



2

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.

Dies wird zu einem großen Vorteil bei Datenbanken, die mehrere Hundert Tabellen umfassen und der Datenbankdesigner nur einen Ausschnitt zur Verfügung gestellt bekommt. Im Programmcode – den man u.U. ohne Datenbankdiagramm bekommt sieht man sofort welche Felder (Attribute) einer Tabelle aus anderen Tabellen kommen.

² Es werden nur Schlüsselfelder angezeigt, alle anderen Felder sind ausgeblendet

Oberfläche

Sie haben es bestimmt auch schon erlebt: Sie öffnen ein Formular und werden von der Anzahl an Eingabemöglichkeiten und der angebotenen Funktionsvielfalt erschlagen. Jedes Formular unterscheidet sich von anderen. Oder auch – das ganze Programm besteht aus einem Formular, das alle Funktionen versammelt, die realisiert wurden.

- Hat dieser Entwickler an den Endbenutzer gedacht?
- Wie viel Zeit muss man für Schulung verwenden bis der Anwender diese Logik versteht?
- Wie viel Zeit muss der Entwickler aufbringen um so ein Programm zu pflegen?
- Ist das veränderte Eventverhalten einer neuen Programmversion noch kalkulierbar?³

So ist es einfacher:

Die Oberfläche wird aufgeteilt in 2 Funktionsgruppen, die die Navigation- und Anzeigeoperationen von den Änderungsoperationen trennt. Dies 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. Dies erfolgt in graphischen Umgebungen über die Menus oder eine Buttonleiste. Die Anzeige der gesuchten Daten erfolgt tabellarisch:



Konto	Beschreibung
6100	Beratungsleistung allgemein
6200	Besprechungen
6400	Planung und Controlling
6430	Controlling
8100	Eigenverwaltung

³ In einer windowsbasierten Umgebung spricht man von Events = Ereignissen. Diese treten immer dann ein wenn man z.B. die Maus bewegt oder einen Button klickt. Durch diese Events wird der Programmcode ausgelöst. Bei neuen Ausgaben der Programmierwerkzeuge ändert sich u.U. dieses Verhalten, oder die Abwärtskompatibilität ist doch nicht so 100%tig wie versprochen.

In diesem Fall muss oft die ganze Umgebung neu programmiert werden um das alte Verhalten der Anwendung wiederherzustellen. Dies sind nichtkalkulierbare Zusatzarbeiten, die kein Kunde gern bezahlt.

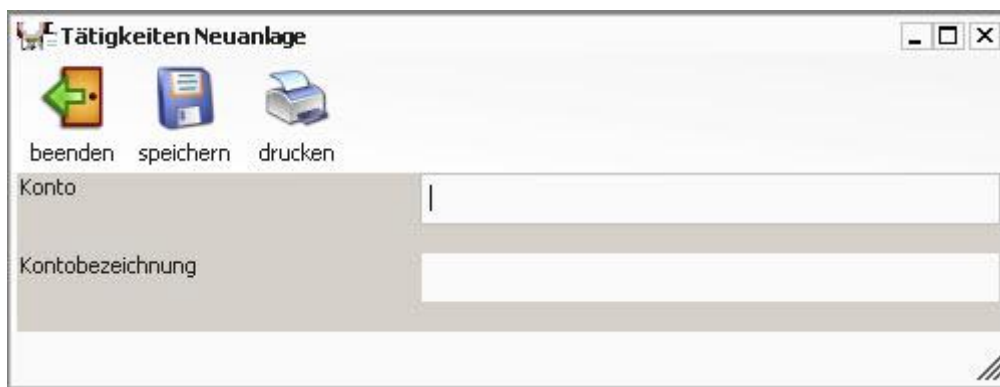
Das Formular zeigt die Daten lediglich an. Änderungen an den Daten oder andere Funktionen (Drucken, Abrechnen, ...) werden über Funktionstasten (oder Buttons) aufgerufen.

Die Standardfunktionen RWUP sind leicht erkennbar – jeder weiß sofort welche Aufgaben sie erfüllen. Alle Formulare arbeiten nach dem gleichen Prinzip. Wer somit ein Formular bedienen kann, der kann auch alle anderen Formulare bedienen.

Unserer Erfahrung nach ist der Funktionsumfang pro Arbeitsschritt in einem Geschäftsprozess auf 5 – 7 Funktionen pro Formular beschränkt. Das Formular bleibt somit immer übersichtlich und lässt sich einfach erklären

Dateneingabe

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



The screenshot shows a software window titled 'Tätigkeiten Neuanlage'. At the top, there are three icons: a green arrow pointing left, a blue floppy disk, and a printer. Below these icons are the labels 'beenden', 'speichern', and 'drucken'. The main area of the form contains two input fields: 'Konto' and 'Kontobezeichnung'. The 'Konto' field is currently empty, and the 'Kontobezeichnung' field is also empty. The window has standard Windows-style window controls (minimize, maximize, close) in the top right corner.


Auch dieses Formular zeigt nur die Funktionen, die in diesem Kontext erforderlich sind: Entweder Sie möchten das Formular verlassen oder Ihre Änderungen speichern. Lediglich das Ausdrucken des Datensatzes zur Dokumentation oder Weitergabe ist hier vorgesehen.

Dies macht es dem Anwender leicht sich genau auf diese eine Interaktion zu konzentrieren und nicht durch in diesem Moment überflüssige Informationen in anderen Bereichen des Formulars abgelenkt zu werden. Der Workflow wird eindeutig.

Kernel

Die meisten Einsparungen an doppelter Programmierarbeit ergeben sich bei 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 über eine einheitliche Konvention Ihrer Primär- und Fremdschlüssel verfügen, wird die Navigation zwischen Tabellen vereinheitlicht.

In der Sprache der Entwickler: ein Stück Programm realisiert die gesamte Navigation der Anwendung. Das Programmmodul zur Navigation wird einmal programmiert und in allen Anwendungen benutzt.

Beispiel:

Sie befinden sich in der Tabelle der Kunden und möchten zur Tabelle der Aufträge (Button Aufträge auf dem Formular Kunden). Das Modul Navigation bekommt als Input den aktuellen Satz (Kunde A) und die Zieltabelle (Aufträge Kunde A) und weist somit welche Daten anzuzeigen sind. Es ruft das Anzeige- oder Eingabeformular mit den ermittelten Daten auf. Dieses kann das Anzeige- oder Eingabeformular generieren.

Datenanzeige




Da alle Anzeigeformulare einer einheitlichen Logik folgen, wird der Programmcode zur Datenanzeige vereinheitlicht.

Der Programmcode zur tabellarischen Anzeige der Daten wird wieder lediglich einmalig realisiert und kann im Programm an jeder Stelle und in allen Anwendungen dieser Aufbaustruktur benutzt werden.

Beispiel:

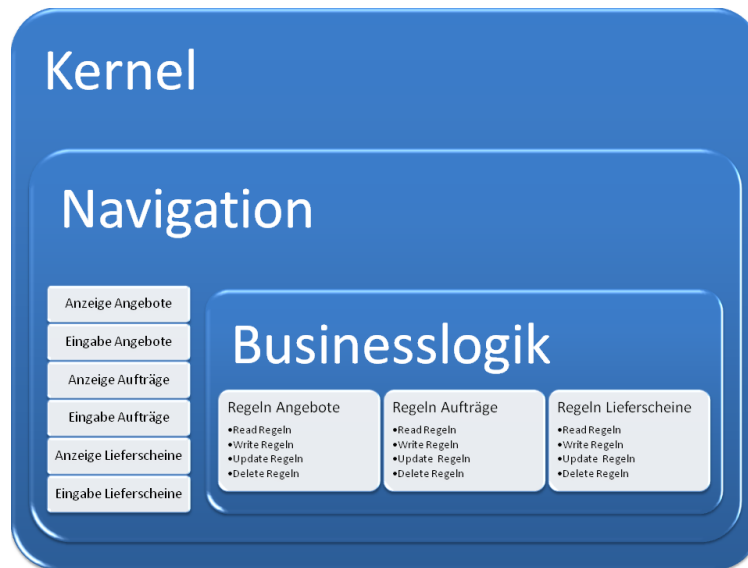
In Folge des obigen Beispiels erhält das Anzeigeformular von der Navigation die Information der Zieltabelle sowie den Wert des Fremdschlüssels. Das Programm hat alle benötigten Informationen und zeigt die gesuchten Daten auf dem Formular an.

Dateneingabe



Da alle Eingabeformulare ein einheitliches Format haben, wird der Programmcode zur Dateneingabe vereinheitlicht.

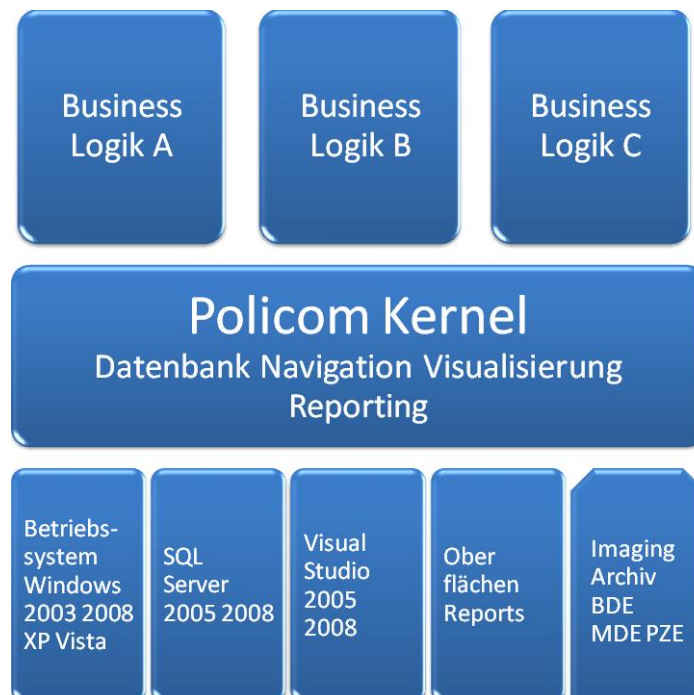
Als Ergebnis dieser Änderungen entsteht ein von der Datenbank unabhängiger Programmcode. Dieser transportiert, ohne die tatsächliche Datenstruktur zu kennen, die Daten zwischen Formular und Datenbank.



Im Kernel sind alle Funktionen zur Navigation zwischen den Formularen implementiert. Das Anzeigeformular und das 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 dort nachzuprüfen, ob es Geschäftsregeln für diesen Vorfall gibt.

Falls ja werden diese ausgeführt. Ist die Regelprüfung erfolgreich wird mit der Verarbeitung fortgefahren. Falls nicht wird die Steuerung an die aufrufende Komponente zurückgegeben.

Aus Sicht des Betriebssystems sieht diese Struktur wie folgt aus:



Dynamisierung

In jeder Anwendung werden nun nicht immer alle Daten einer Tabelle angezeigt, sondern nur die die wir sehen möchten. Wie erreichen wir es jetzt, dass dennoch genau die Daten angezeigt werden die wir sehen möchten, und nicht alle?

Da die Navigation, das Anzeigeformular und das Eingabeformular anwendungsneutral formuliert sind kommt die Information zur Steuerung der Navigation und der Anzeige aus der Datenbank.

Navigation

In der Datenbank wird zunächst der Aufbau der Navigation (woher - wohin) und die auf den Formularen benötigten Buttons in einer Tabelle hinterlegt.

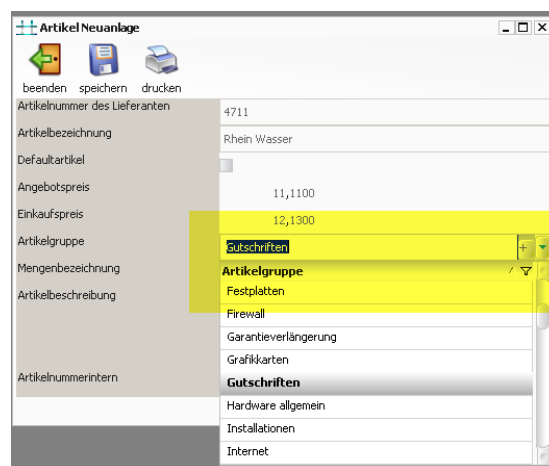
Die Navigation und die jeweiligen Buttons auf den Formularen werden dann beim Programmablauf dynamisch generiert. Dies ermöglicht es uns auch mitten im Betrieb vor Ort beim Kunden den Programmablauf zu ändern oder zu ergänzen – ohne eine Zeile zu programmieren.

Anzeige und Eingabeformular

Auch die Beschreibungen der Daten die auf dem Anzeige- oder Eingabeformular anzuzeigen sind, sind in der Datenbank hinterlegt. Die Navigation gibt dem Anzeigeformular bzw. dem Eingabeformular die benötigten Informationen aus der Datenbank mit. Die Formulare können den Bildschirmaufbau dynamisch generieren.

Auf den Eingabefeldern werden Auswahllisten für Daten aus anderen Tabellen benutzt um den Zugriff zu beschleunigen. Auch die Steuerung dieser Auswahlfelder ist automatisiert. Ihr Inhalt wird ebenfalls durch die Datenbank gesteuert:

Da der Verweis zwischen den Tabellen standardisiert ist und die benötigten Schlüssel bekannt sind, kann auch die Funktionalität der Auswahllisten in den Eingabefeldern standardisiert werden:




Wieder ist es möglich auch mitten im Betrieb vor Ort beim Kunden die angezeigten Formulare zu ändern oder zu ergänzen – ohne eine Zeile zu programmieren.

Diese Struktur ermöglicht es Änderungen an der Datenbankstruktur vorzunehmen ohne Programmänderung zu ermöglichen.⁴

Businesslogik

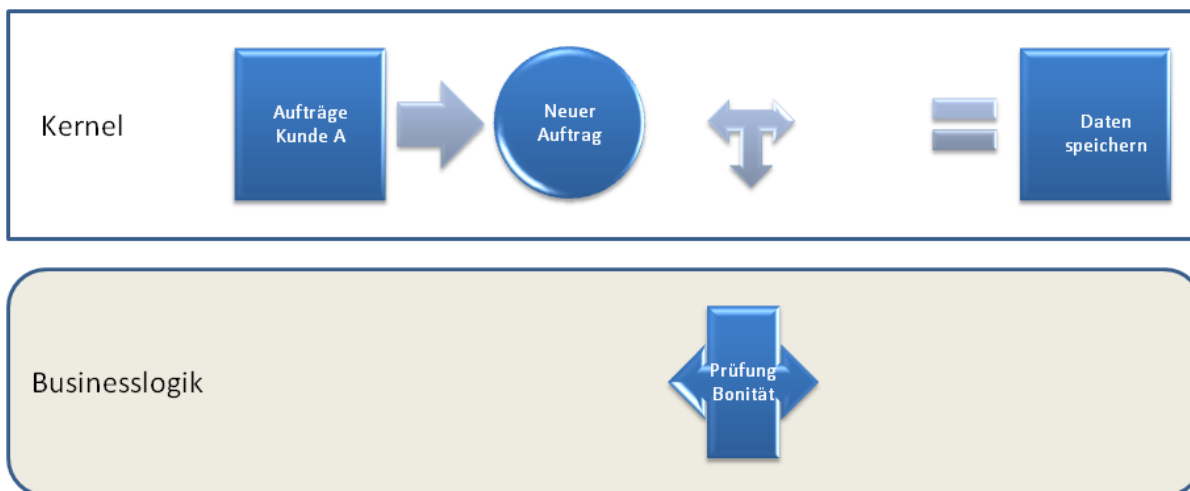
Jede Anwendung hat ihre eigene Logik, die Businesslogik. Sie ist von Betrieb zu Betrieb verschieden. Sie muss wie bisher auch kundenspezifisch angepasst werden. Wie wird dieser Programmcode integriert?



Der Standardcode der Navigation, des Anzeigeformulars und des Eingabeformulars stellt zwischen den Formularen und vor bzw. nach Datenbankinteraktionen Schnittstellen bereit. Diese Schnittstellen werden von Businesslogik benutzt um die Verarbeitung zu steuern und zu regeln.

Beispiel:

In dem Anzeigeformular der Aufträge von Kunden A wird der Button „Neuer Auftrag“ geklickt. Die Navigation ruft den Programmcode zur Generierung des Erfassungsformulars auf. Zuvor verzweigt es in die Businesslogik. Dort wird überprüft ob der Kunde im Limit liegt und der neue Auftrag erfasst werden kann. Falls ja wird die weitere Verarbeitung freigegeben.



Die gleiche Vorgehensweise kommt beim Speichern des neuen Auftrages zur Anwendung. Die Navigation ruft die Standardoperation zum Speichern der Daten auf. Diese verzweigt in den Businesslogik um die erforderlichen Prüfungen zu machen (z.B. wurde die Mindestbestellmenge berücksichtigt). Diese arbeitet ihre Regeln ab und gibt den Satz zum Speichern frei oder gibt ihn dem Erfassungsformular zur Überarbeitung zurück.

⁴ Dies ist nur möglich für ergänzende Felder, die nicht mit einer logischen Feldprüfung unterzogen werden müssen.

Schnittstellen Kernel – Businesslogik

Für den Aufruf eines Tabellen-Formulars sind folgende Schnittstellen vorgesehen, an dem ggf. Businesscode ergänzt werden kann:

- Tabellenformular
 1. Prüfung Security ob Zugriff erlaubt
 2. Funkkontext vor dem Aufruf des Formulars um festzustellen ob in dem jeweiligen Satzzusammenhang die gewünscht Operation erlaubt bzw. sinnvoll ist.
 3. Nach dem Einlesen der Daten
 4. Vor dem Füllen der Tabellenkomponente
 5. Beim Initialisieren des Layouts
 6. Vor dem Anzeigen des Formulars
 7. Beim Schließen des Formulars
- Eingabeformular
 1. Vor dem Speichern der Daten
 2. Nach dem Speichern der Daten
 3. Div. Schnittstellen beim Betreten oder Verlassen der Eingabefelder zur Steuerung und Prüfung des Inhaltes

In gleicher Weise werden Funktionen implementiert, die von den Standardkomponenten nicht abgedeckt werden. 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 in dem Anzeigeformular seine Daten zu selektieren. Falls man diese Änderung im Kernel implementiert, steht diese Funktion automatisch alle anderen Kunden zur Verfügung.

Im Umkehrschluss lassen sich somit auch größere Module (Archivierung) kostengünstig auf mehrere Schultern verteilen.

Prototypen

in einer frühen Projektphase ist es möglich dem Kunden mit dem ersten Datenbankentwurf und einem groben Ablaufplan ein Look-and-Feel der Anwendung zu ermöglichen.

Dies bevor die erste Zeile programmiert wurde!

Daher ist es möglich Schwachstellen im Entwurf oder dem Workflow frühzeitig zu erkennen und zu beheben.

Migration

Eine Datenbanken 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 der Wartung und im Betrieb. Einmal getestet und verifiziert, kann man sich auf stabilen Code verlassen. Dies reduziert den Testumfang in der Realisierungsphase erheblich.

Zum Ende

... möchte ich die Aktionen nochmals zusammen:

1. Datenbankstruktur normieren
2. Oberflächen normieren
3. Navigation standardisieren
4. Datenanzeige standardisieren
5. Datenänderungen standardisieren

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 der Industrie, dem Handel und dem Dienstleistungsbereich. Wir arbeiten für Kleinbetriebe und Mittelständler mit bis zu 2500 Arbeitsplätzen.

Policom

Angepasst an Ihre individuellen Betriebsabläufe liefern wir ein modular aufgebautes System, das Ihre Anforderungen an die Automatisierung Ihrer Geschäftsprozesse erfolgreich umsetzt.

Policom wächst mit Ihnen und gleicht sich flexibel ändernden Betriebsabläufen an. Eine benutzerfreundliche Bedienoberfläche erleichtert auch in komplexem Kontext die einfache Einarbeitung und zielsichere Anwendung.

Wir bieten effiziente Lösungen

- im Vertrieb (CRM)
- der Auftragsabwicklung (Lieferschein- und Rechnungsschreibung, Mahnwesen)
- Einkauf
- Lagerhaltung
- Stammdaten und Stücklisten

-
- der Produktion (BDE, MDE, PZE, MES)
 - in der Projektsteuerung
 - Workflow und Archivierung

Nutzen Sie folgende Vorteile von Policom:

- Durch die Kombination von standardisierten Modulen mit kundenspezifischen Erweiterungen, verbinden wir die Vorteile von Standardsoftware mit denen der Individualentwicklung.
- Dies gewährleistet eine kurze Time-to-Production Situation, die Ihnen eine zügige und ziel-treffende Realisation Ihrer Anforderungen ermöglicht.
- Durch die enge Verknüpfung mit Ihren Betriebsabläufen garantieren wir den Erhalt Ihrer Wettbewerbsvorteile.

Wir setzen ausschließlich aktuelle Microsoft-Entwicklungswerkzeuge ein, die Ihnen eine zuverlässige Integration in Ihre Umgebung garantieren. Bei Bedarf lizensieren Sie die Policom-Kernelkomponenten. Individuelle Erweiterungen des Systems werden dann von Ihren Mitarbeitern implementiert. Dies garantiert weit gehende Freiheiten in der Anpassung von Funktionalität und Oberfläche.