Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorthesis

Patrick Weber

Konsolidierung und Migration einer Kundendatenbank mit anschließendem Entwurf einer webbasierten Datenbankanwendung

Fakultät Technik und Informatik
Department Informations- und Elektrotechnik

Faculty of Engineering and Computer Science
Department of Information and Electrical Engineering

Patrick Weber

Konsolidierung und Migration einer Kundendatenbank mit anschließendem Entwurf einer webbasierten Datenbankanwendung

Bachelorthesis eingereicht im Rahmen der Bachelorprüfung
im Studiengang Informations- und Elektrotechnik
am Department Informations- und Elektrotechnik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Wilfried Wöhlke.
Zweitgutachter : Prof. Dr. Andreas Suhl

Abgegeben am 17.02.2015

Patrick Weber

Thema der Bachelorthesis

Konsolidierung und Migration einer Kundendatenbank mit anschließendem Entwurf einer webbasierten Datenbankanwendung

Stichworte

Datenbank, APEX, Konsolidierung, Gebrauchstauglichkeit, externe Verlinkung, SQL Developer, ETL Prozess, Master-Detail Beziehungen

Kurzzusammenfassung

Diese Arbeit beschreibt die Konzeption der Konsolidierung und Migration einer bestehenden Kundendatenbank in ein Oracle Schema. Zusätzlich wird eine webbasierte Datenbankanwendung zur Datenverwaltung, unter Berücksichtigung praktischer und theoretischer Design Aspekte entwickelt.

Patrick Weber

Title of the paper

Consolidation and migration of a customer database followed by designing a web based data application

Keywords

Database, APEX, consolidation, usability, external link, SQL Developer,
ETL process, master-detail relationship

Abstract

This paper describes the conception of consolidation and migration of an existing customer database to an Oracle schema. Additionally a web based data application for data administration will be developed in due consideration of practical and theoretical design aspects.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1 Einleitung

1.1 Motivation

In der heutigen Welt ist die Informationsverarbeitung durch Datenbanksysteme (DBS) im Alltagsleben, wie z.B. in den Bereichen Administration, Gesundheit, Verkehr, Bildung und Industrie von bedeutendem Stellenwert. Datenanwendungsprogramme, die eine Schnittstelle zwischen der Datenbank und dem DBS bilden, werden benötigt, um Datensätze abrufen, bearbeiten oder erstellen zu können.

Waren in der Vergangenheit noch spezielle Anwendungsprogramme erforderlich, die separat auf jedem Personal Computer installiert und konfiguriert werden mussten (On-Premise Modell[1]), hat die Entwicklung von webbasierten Datenanwendungsprogrammen (Web Applikationen) in den vergangenen Jahren rapide zugenommen. Für Entwickler und Anwender erschließen sich hierdurch neue Möglichkeiten im Bereich des B2B[2] und des Cloud Computings. Der Anwender benötigt nur noch einen beliebigen Internet Browser, um Plattform unabhängig mit dem Datenanwendungsprogramm zu arbeiten. Die erforderliche Software und die Daten liegen auf einem Server, der entweder vom eigenen Unternehmen oder durch Dienstleistung Dritter verwaltet wird. Aktualisierungen bzw. Modifikationen des Anwenderprogramms lassen sich von Entwicklerseite schneller durchführen.

Neben den allgemeinen Funktionalitäten, eines webbasierten Datenanwendungsprogramms, ist der Bedienkomfort der grafischen Benutzeroberfläche, sowie eine intuitive Bedienung nicht zu vernachlässigen, da letztendlich anhand dieser Kriterien die Alltagstauglichkeit bemessen wird und den Erfolg eines Produkts auszeichnet.

[1] Die Lizenz einer Software wird erworben und vom Kunden auf eigene Hardware installiert
[2] Business-to-business: Eine Geschäftsbeziehung zwischen zwei Unternehmen

1.2 Aufgabenstellung und Zielsetzung

Die Bachelorthesis entsteht beim Elektronikkonzern, die Philips GmbH mit Sitz in Hamburg, die im Bereich Healthcare[3] Remoteservice[4] anbietet. Zuständig für die Konfiguration und Überwachung der Infrastruktur sind die Mitarbeiter des Remote Service Network (RSN), die täglich mit Kunden im Gespräch stehen. Hierzu müssen die Kundeninformationen schnell auffindbar und einfach zu verwalten sein. Dies erfordert neben geeigneten Datenbankstrukturen, auch zeitgemäße Anwendungen zur Datenverwaltung. Im Rahmen konzerninterner Leitlinien sind Bestrebungen angedacht worden, zukünftig verstärkt auf Software as Service (SaaS)[5] Lösungen zu setzen, um IT Kosten zu reduzieren und insbesondere den Wartungsaufwand zu verringern.

Dieser Gedanke ist für eine bestehende Kundendatenbank des Remoteservice aufgegriffen worden, da die bisherige Datenverwaltung über eine Anwendung in MS Access getätigt wird. Aufgrund veränderter Marktsituationen und nicht eingehaltener Standards der Datenpflege (Data Governance) entspricht das bestehende Verwaltungsprogramm nicht mehr den zukünftigen Ansprüchen und erweist sich als nicht mehr nachhaltig.

Das Ziel dieser Arbeit ist der Entwurf einer webbasierten Datenbankanwendung mit Hilfe der Entwicklungsumgebung Oracle Application Express (APEX). Hierfür ist es erforderlich die bestehende Kundendatenbank in ein Oracle Schema zu migrieren. Dabei sind die Daten zu konsolidieren und das Datenbankmodell unter der Einschränkung, dass die Anzahl von Datenbankrelationen nicht verändert werden soll, geeignet anzupassen.
Des Weiteren soll die Webanwendung in geeigneter Weise mit Hilfe externer Verweise den Zugriff auf Kundendokumente, die auf einem Netzlaufwerk abgelegt sind, ermöglichen und mit dem PRS Portal (Philips Remote Service Portal) verknüpft werden.

Die Serverkonfiguration und die Verwaltung der APEX Entwicklungsumgebung sind nicht Gegenstand dieser Arbeit.

[3] Gesundheitswesen
[4] Fernwartungsservice
[5] Software as a Service: Infrastruktur und Software werden durch externe IT Dienstleister verwaltet, dabei fallen für die Nutzung Gebühren an

1.3 Aufbau der Arbeit

Die Arbeit ist in neun Kapitel gegliedert.

In Kapitel 2 werden wichtige Begriffe und Grundlagen zu Datenbanken und Modellierungskonzepten vorgestellt, sowie die genutzten Werkzeuge zur Analyse und Gestaltung von Datenbanken.

Das dritte Kapitel stellt die Entwicklungsumgebung APEX vor und gibt einen ersten Überblick, über den Aufbau einer Webanwendung und den darin enthaltenen Komponenten.

Kapitel 4 befasst sich mit der Analyse des Ist-Zustands und diskutiert Lösungsansätze zur Vorbereitung der Datenbankmigration und Anpassung des Datenbankmodells. Anschließend werden verschiedenen Konzepte bezüglich der Funktionen und der Logik der Webanwendung erarbeitet.

Das fünfte Kapitel umfasst Designaspekte unter Nutzererfahrungen zur Skizzierung eines vorläufigen Layouts einer Benutzeroberfläche.

In Kapitel 6 erfolgt die Realisierung eines Prototyps der Webanwendung, der in Kapital 4 und 5 erarbeiteten Konzepte, sowie der Test einzelner Funktionen.

Unter Einbeziehung der Anwender wird in Kapitel 7 ein Testszenario abgearbeitet, sowie die Akzeptanz und Bedienbarkeit des Prototyps bewertet.

Das Kapitel 8 beschreibt den Übergang vom Prototyp zur Inbetriebnahme der Webanwendung und schließt das Laden der MS Access Daten ins Oracle Schema ein.

Das neunte Kapitel skizziert rückblickend die Aufgabenstellungen und fasst die Ergebnisse der Arbeit zusammen. Ein Ausblick für mögliche Erweiterungen und ein Feedback über die Erfahrungen zur APEX Entwicklungsumgebung werden gegeben.

2 Grundlagen

2.1 Allgemeine Begriffe

2.2 Datenbank

Eine Datenbank ist eine Sammlung von einem oder mehreren Objekten mit dem Ziel umfangreiche Datenmengen verwalten und speichern zu können, damit Informationen in Abhängigkeit der Fragestellung schnell wieder gewonnen werden können.
Bei einem Objekt handelt es sich um eine Relation (Tabelle), welche logisch strukturierte Informationen enthält, die als Tupel (Datensatz) bezeichnet werden. Die Informationen können aus Datums- und Zeitwerten, alphanumerischen Zeichen oder auch aus Bildern bestehen. René Schreiner definiert eine Datenbank vereinfacht mit folgendem Satz: „Eine Datenbank ist eine selbstständige und auf Dauer ausgelegte Datenorganisation, welche einen Datenbestand sicher und flexibel verwalten kann.“ [1 S. 2]

2.2.1 Datenbasis

Eine Datenbasis enthält die gesamten Datenbestände, d.h. alle Relationen, die in physikalische Dateien auf Festplatten bzw. Magnetplattenspeichern abgelegt werden.

2.2.2 Datenbankmanagementsystem

Mit einem Datenbankmanagementsystem (DBMS) lassen sich Datenbanken erstellen, verwalten, abfragen und modifizieren. Es beinhaltet Funktionen und Mechanismen, die Zugriffsrechte, die Datensicherheit und Integrität überprüfen. Durch Schnittstellen des DBMS ist es möglich, dass externe Anwendungen auf Informationen der Datenbank zugreifen können. DBMS werden je nach Hersteller über eine Kommandozeile mit Anweisungen oder einer grafischen Benutzeroberfläche bedient. DBMS gibt es als hierarchische, objektorientierte oder relationale Systeme, wobei letzteres zum heutigen Standard geworden ist. Zu den bekanntesten relationalen Datenbankmanagement Systemen (RDBMS) zählen unter anderem:

- Oracle Database

- Microsoft SQL Server
- SAP MaxDB
- IBM DB2
- MS Access

2.2.3 Datenbanksystem

Als Datenbanksystem wird ein DBMS zusammen mit einer oder weiteren Datenbanken bezeichnet.

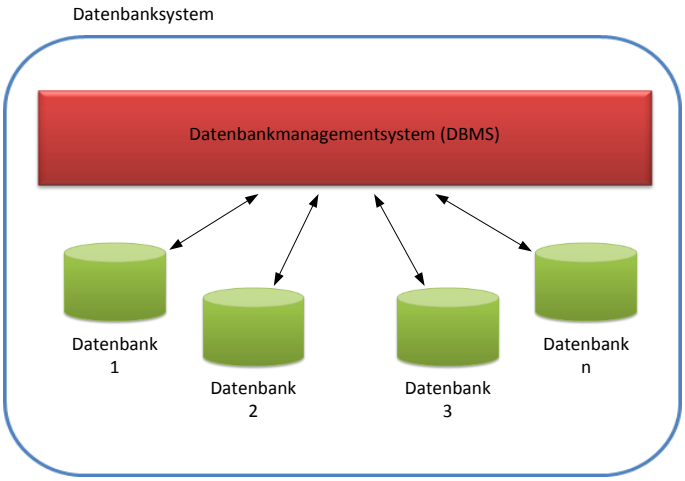


Abbildung 2.1: Zusammensetzung eines Datenbanksystems

2.2.4 Datenbankschema

Ein Datenbankschema ist als logischer Container einer relationalen Datenbank zu verstehen, welche das Relationenschema enthält, d.h. alle Datenbankobjekte, wie Indizes, Relationen, Sichten, Abfragen, Trigger usw. Jedes Datenbankobjekt kann nur einmal einem Datenbankschema zu geordnet werden. Unter Oracle wird anstatt der Bezeichnung „Datenbankschema“ der Begriff „Schema“ verwendet.

2.2.5 Sequenz

Die Sequenz erzeugt fortlaufende Integer Zahlen, die jeweils nur einmal vorkommen dürfen. Die Werte werden zur Generierung von Primärschlüsseln verwendet. Eine Sequenz kann mehreren Relationen zugeordnet werden, da sie unabhängig arbeitet. Oftmals wird aufgrund der besseren Übersicht für jede Relation eine eigene Sequenz erzeugt.

Abbildung 2.2 zeigt das allgemeine Prinzip, wie eine Sequenz den Primärschlüssels Werte zuweist. Ausgehend von einem Startwert wird pro Zeile um einen Schritt hochgezählt und der aktuelle Wert in das nächste leere Zeilenfeld eingetragen. Dabei „merkt“ sich die Methode *CurrentVal()* jeweils den zuletzt vergebenen Wert, der beim erneuten Ausführen der Sequenz als neuer Startwert gesetzt ist.

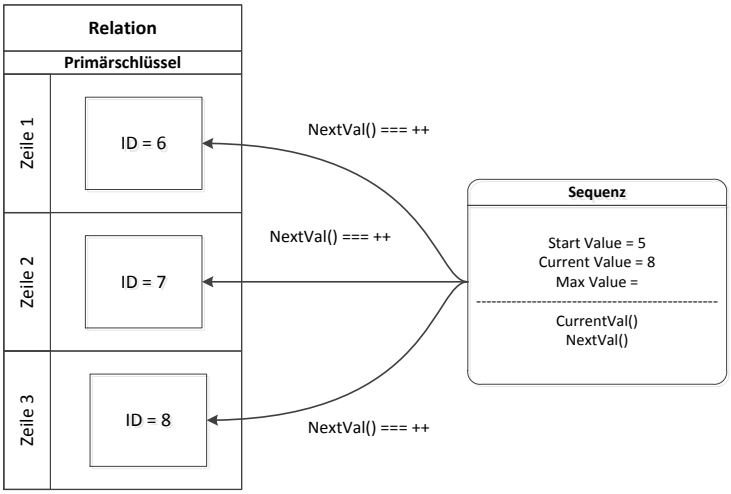


Abbildung 2.2: Prinzip einer Sequenzausführung (angelehnt an [2])

2.2.6 Trigger

Ein Trigger ist eine Funktion innerhalb eines DBMS, die bestimmte Aktionen ausführt, wenn Attributwerte innerhalb einer Relation modifiziert, gelöscht oder eingefügt werden, d.h. es können auch mehrere Trigger verschiedene Operationen gleichzeitig ausführen.
Der Trigger Befehl kann entweder vor oder nach einem Aktualisierungsprozess ausgelöst werden, es existieren drei Arten von Trigger:

- DML Trigger:

  Wird auf eine Relation angewandt und durch die Befehle `INSERT`, `UPDATE` oder `DELETE` ausgelöst.

- Instead of Trigger:

  Wird auf Sichten ausgeführt und durch die DML Anweisungen `INSERT`, `UPDATE` oder `DELETE` ausgelöst.

- DDL Trigger:

  Wird durch Systemereignisse und den Anweisungen `CREATE`, `ALTER` oder `DROP` ausgelöst.

Abbildung 2.3 zeigt einen schematischen Aufbau, wie über eine Anwendung DML Anweisungen ausgeführt werden, welchen den entsprechenden Trigger auslösen.

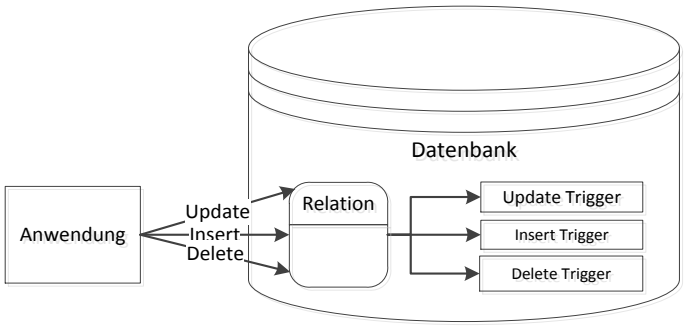


Abbildung 2.3: Trigger Aktionen (in Anlehnung an [3])

2.2.7 Bindevariable (Bind Variable)

Bindevariablen sind dynamische Parameter, die einem Platzhalter entsprechen, der durch einen konkreten Wert ersetzt wird. Sie werden oftmals verwendet, um der Gefahr einer SQL-Injection[6] vorzubeugen.

2.3 Datenbankmodelle

Mit Hilfe von Datenbankmodellen werden im ersten Schritt einer Datenbankplanung die Spezifikationen anschaulich umgesetzt und detailliert mit der Zielgruppe diskutiert.

2.3.1 Hierarchisches Modell

Eine der ältesten Datenbankmodelle ist das hierarchische Modell, welches im Bank- und Versicherungswesen das größte Nutzungspotential vorweisen konnte. Es versucht die reale Welt als hierarchische Baumstruktur wiederzugeben. Ausgehend von einer Wurzel werden die Beziehungen der Daten über Kanten miteinander verbunden und gliedern sich in Teilbäume. Jeder Knoten, d.h. jedes Datenobjekt, besitzt exakt einen Vorgänger und mindestens einen Nachfolger. Die untersten Datenobjekte in der Hierarchie werden als Blätter bezeichnet.

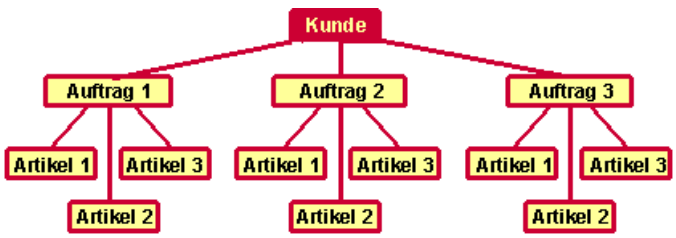


Abbildung 2.4: Beispiel eines hierarchischen Modells [4]

In der Praxis bedeutet es, dass die Daten in einer sequentiellen Datei gespeichert werden. Anhand eines Beispiels einer Kursverwaltung, die Angestellte einer Firma und das Teil-

[6] Ausnutzung einer Sicherheitslücke durch Einschleusung von SQL Anweisungen zur Datenbankmanipulation oder Datenabgriff.

nahmedatum enthält, wird in Abbildung 2.5 der Aufbau einer hierarchischen Datei gezeigt. Der Angestellte nimmt die höchste Hierarchiestufe ein, gefolgt vom belegten Kurs und des Datums.

1	121
	Meier
1.1	100
	abc
1.1.1	1.1.2005
1.2	102
	def
1.2.1	10.3.2005
1.2.2	1.2.2006
2	134
	Steffen
2.1	100
	abc
2.1.1	3.4.2007
3	155
	Huber
3.1	105
	xyz
3.1.1	4.4.2008
3.1.2	1.8.2008

Abbildung 2.5: Struktur einer hierarchisch aufgebauten Datei [1 S. 10]

Der wesentliche Nachteil des hierarchischen Modells ist, dass es nicht möglich ist mehrere Ebenen innerhalb eines Baumes zu verknüpfen und auch keine Referenzen zwischen verschiedenen Bäumen erstellt werden können. Die Suche nach bestimmten Daten kann nur über einen Pfad erfolgen, der bekannt sein muss. Dadurch gilt das Modell als starr und unflexibel, sodass aus heutiger Sicht das hierarchische Modell für Entwickler kaum noch eine Relevanz hat.

2.3.2 Netzwerkmodell

Im Gegensatz zum hierarchischen Datenmodell, kann ein Datenobjekt beim Netzwerkmodell mehrere Vorgänger besitzen. Dadurch können verschiedene Suchpfade verwendet werden. Das Netzwerkmodell ist in den 1960er Jahren im Zusammenhang mit der Programmiersprache „Cobol“ entwickelt worden und gilt heutzutage als veraltet.

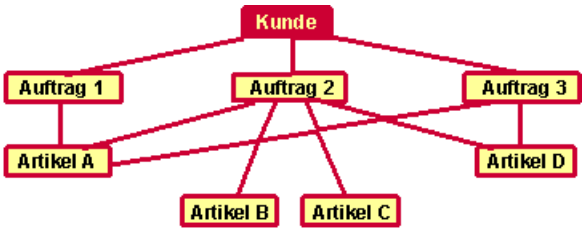


Abbildung 2.6: Beispiel eines Netzmodells [4]

2.3.3 Objektorientiertes Modell

Objektorientierte Modelle sind in den 1980er Jahren erstmals entwickelt worden, aber bis heute in der Praxis sehr selten verbreitet. Das Datenbankmodell greift auf das Konzept der objektorientierten Programmierung zurück, d.h. es besteht aus einem Operationsteil, der alle Funktionen (Methoden) enthält und einem Strukturteil, der alle Objekteigenschaften und Beziehungen des Modells beschreibt.

2.3.4 Relationales Modell

Die Grundlagen des relationalen Datenbankmodells sind in den 1960er und 1970er Jahren durch Edgar F. Codd entwickelt worden und bilden die Basis für RDBMS.

Das relationale Modell besteht aus einer Ansammlung von Objekttypen, die miteinander in Beziehung stehen. Dabei handelt es sich um zweidimensionale Tabellen, die als Relation bezeichnet werden. Eine präzise Definition aus mathematischer Sicht wird durch Gerhard Stegemann gegeben, der eine Relation folgendermaßen definiert:
„Eine Relation ist eine Teilmenge des kartesischen Produktes von Mengen.“ [5]

Die wesentlichen Merkmale einer relationalen Datenbank sind, dass Informationen auf mehrere Relationen verteilt und durch Abfragen wieder verlustfrei zusammengeführt werden können. Die korrekte Zuordnung zwischen den einzelnen Relationen erfolgt über Fremdschlüssel. Damit die unterschiedlichen Sachverhalte fehlerfrei zusammengesetzt werden können, ist es erforderlich eine Datenbankstruktur zu verwenden, die nach Regeln der Normalisierung modelliert worden ist.

Die Regeln und die Bedeutung der Normalisierung werden in Kapitel 2.5.2 näher erläutert.

Die relationalen Datenbankmodelle haben sich gegenüber anderen Konzepten durchsetzen können, da sie einerseits große Flexibilität bezüglich der Erweiterungen und Änderungen bieten und andererseits die reale Welt am genausten nachbilden können.

Der weitere Verlauf dieser Arbeit befasst sich ausschließlich mit dem relationalen Datenbankmodell.

2.4 Begriffe konzeptioneller und relationaler Datenbankmodelle

In Abhängigkeit der Betrachtungsweise werden für physische Modelle auf Ebene der konzeptionellen und relationalen Datenbankmodelle unterschiedliche Bezeichnungen während der Analyse- und Entwurfsphase verwendet.

2.4.1 Entitätstyp und Entitäten

In einem konzeptionellen Datenbankmodell wird die Abbildung eines realen Objekts als Entitätstyp bezeichnet und in Form einer Tabelle dargestellt. Die enthaltenen Ausprägungen werden Entitäten genannt, dabei handelt es sich um die einzelnen Datensätze. Die Eigenschaften des Entitätstyps werden durch Attribute beschrieben.

2.4.2 Relation und Tupel

In einem relationalen Datenbankmodell wird die Abbildung eines realen Objekts als Relation bezeichnet und entspricht der Darstellungsform einer Tabelle. Die enthaltenen Ausprägungen werden als Tupel bezeichnet, dabei handelt es sich um die einzelnen Datensätze. Die Eigenschaften der Relation werden durch Attribute beschrieben.

2.4.3 Attribut

Ein Attribut beschreibt sowohl im konzeptionellen, als auch im relationalen Datenbankmodell die Eigenschaften, d.h. das Merkmal einer Entität bzw. eines Tupels. Dabei handelt es sich im physischen Modell um die Tabellenspalten.

2.4.4 Attributwert

Bei einem Attributwert handelt es sich um einen Datenwert, der ein entsprechendes Attribut eines Tupels beschreibt.

2.4.5 NULL Wert

Enthält ein Attribut in einem Tupel einen NULL Wert, so ist kein Attributwert vorhanden, d.h. das Tupel enthält an dieser Stelle keine Information. Zu beachten ist, dass der NULL Wert nicht mit der Zahl null verwechselt werden darf, da null eine Information darstellt.

2.4.6 Domäne

Eine Domäne ist der Wertebereich eines Attributs, der definiert, welche Werte die Attributwerte annehmen können.

2.4.7 Gegenüberstellung der Grundbegriffe

In Tabelle 2.1 sind die wichtigsten Grundbegriffe zwischen dem konzeptionellen und relationalen Datenbankmodell gegenübergestellt.

ERM	relationale Datenbank	physisches Datenbankmodell
Entitätstyp	Relation	Tabellenname
Attribut	Attribut	Tabellenspalte
Entität	Tupel	Tabellenzeile
Entitätsmenge	Relationen	Menge von Zeilen
Entitätsschema	Relationenschema	Menge von Attributen

Tabelle 2.1: Zusammenhang der Datenmodelle

2.5 Datenbanktheorie

2.5.1 Entity Relationship Modell (ERM)

Ein ERM ist eine konzeptionelle Modellierungssprache, das versucht Zusammenhänge aus der realen Welt im Rahmen der Datenmodellierung abstrakt abzubilden. Es gehört zur Kategorie der semantischen Datenmodelle.
Im ERM werden die Benutzeranforderungen an die zu erstellende Datenbank analysiert und festgelegt. Das erstellte Modell dient anschließend als Grundlage zur Planung und Entwicklung der Datenbank. Für das ERM existieren verschiedene Notationen, die bekanntesten sind Chen, IDEF1X, Barker und Bachman.

2.5.1.1 Grafische Elemente

Ein kurzer Einblick in die allgemeine Darstellungsform der Chen Notation soll gegeben werden, da diese Notation im weiteren Verlauf Verwendung findet.

Entitätstyp

Jeder Entitätstyp wird in Form eines Rechtecks dargestellt. Die Objektbeschreibung befindet sich innerhalb der geometrischen Form und wird im Singular angegeben.

Abbildung 2.7: Beispiel eines Entitätstyps

Attribut und Primärschlüssel

Zu jedem Entitätstyp gehören Attribute, die in Form von Ellipsen über Linien mit dem Entitätstyp verbunden werden. Die Beschriftung erfolgt innerhalb der Ellipsen mit Substantiven im Singular. Der Primärschlüssel, der für die eindeutige Identifikation der Entitäten erforderlich ist, wird durch Unterstreichen hervorgehoben.

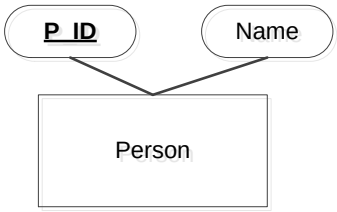


Abbildung 2.8: Beispiel eines Entitätstyps mit Attribut und Primärschlüssel

Beziehung

Eine Beziehung zwischen Entitätstypen wird über eine Raute ausgedrückt, deren Beschriftung aus einem Verb im Singular besteht.

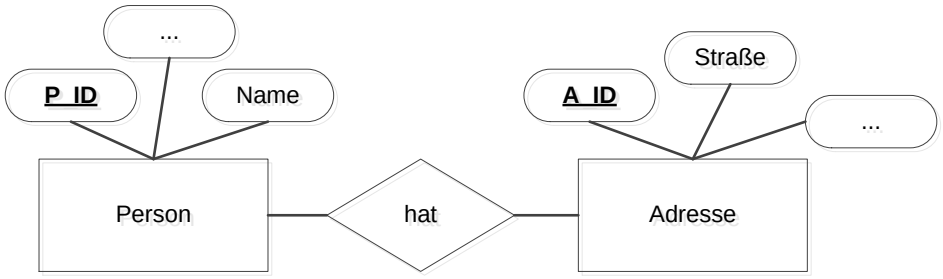


Abbildung 2.9: Beispiel von Entitätstypen in einer Beziehung

Kardinalität

Zu jeder Beziehung ist die Kardinalität anzugeben, welche auf die Verbindungslinien zwischen dem Entitätstyp und der Beziehung eingetragen wird.
Eine nähere Erläuterung zu den Kardinalitäten wird im weiteren Verlauf gegeben.

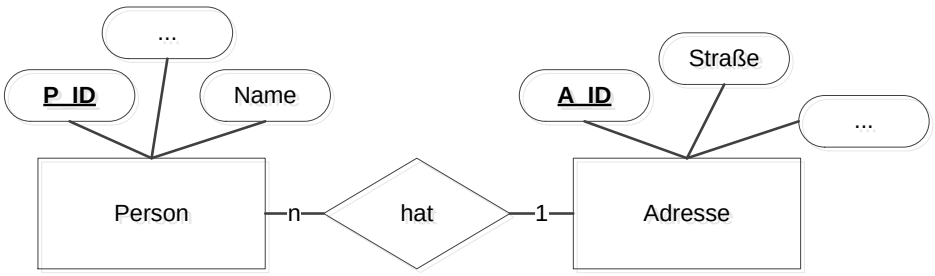


Abbildung 2.10: Beispiel einer Beziehung mit Kardinalität

Schwacher Entitätstyp

Als schwacher Entitätstyp werden alle Gegenstandstypen bezeichnet, die ohne einen übergeordneten Entitätstyp nicht alleine existieren können.

Als Beispiel seien zwei Entitätstypen gegeben: „Hotel“ und „Zimmer“.

Die Zimmer im gleichnamigen Entitätstyp können ohne das Gebäude „Hotel“ nicht existieren. Da jedes Zimmer über eine Zimmernummer verfügt, ist nur eine eindeutige Zuordnung innerhalb des Hotels gegeben. Das bedeutet beim Entitätstyp „Zimmer“ handelt es sich um einen schwachen Entitätstyp.

In der Chen Notation werden schwache Entitätstypen durch ein doppeltumrahmtes Rechteck dargestellt. Die Beziehung zum übergeordneten Entitätstyp wird durch eine doppeltumrahmte Raute beschrieben.

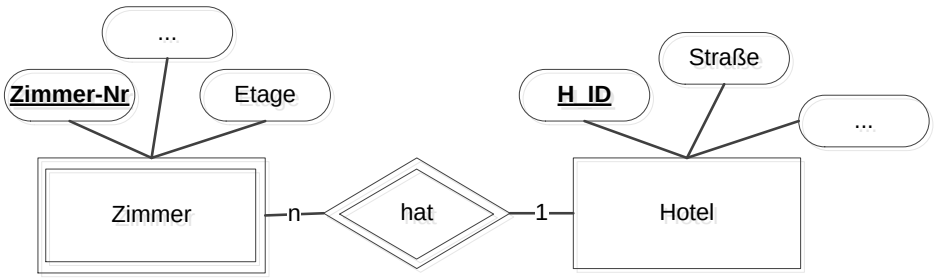


Abbildung 2.11: Beispiel eines schwachen Entitätstyps

2.5.1.2 Kardinalitäten

Die Kardinalitäten zwischen den Entitätstypen beschreiben, wie viele Entitäten eines Entitätstyps mit Entitäten eines anderen Entitätstyps in Beziehung stehen. Die Kardinalität wird durch zwei Angaben beschrieben und ist wechselseitig zu betrachten. Es existieren drei Arten, die im Folgenden anhand von Beispielen erläutert werden sollen. Aufgrund der Übersichtlichkeit sind die Attribute an den Entitätstypen vernachlässigt worden.

m:n - Beziehung

Mehrere Entitäten eines Entitätstyps beziehen sich auf mehreren Entitäten eines anderen Entitätstyps. Die Abbildung 2.12 beschreibt, dass ein Servicetechniker mehrere verschiedene Medizinsysteme wartet und ein Medizinsystem von mehreren Servicetechnikern gewartet werden kann.

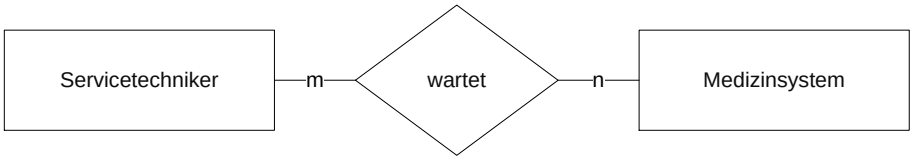


Abbildung 2.12: Datenmodell in m:n – Beziehung

1:n - Beziehung

Eine Entität des ersten Entitätstyps bezieht sich auf mehrere Entitäten des zweiten Entitätstyps. Das Datenmodell in Abbildung 2.13 zeigt, dass jeweils genau ein Regionalleiter mehrere Niederlassungen in einem Gebiet kontrolliert und jede Niederlassung von genau einem Regionalleiter überwacht wird.

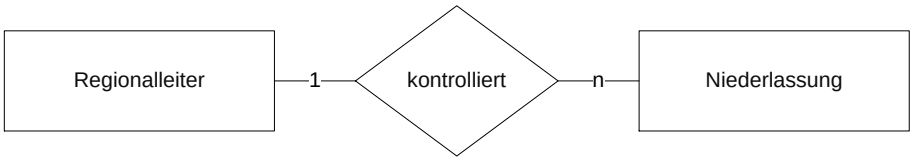


Abbildung 2.13: Datenmodell in 1:n – Beziehung

1:1 - Beziehung

Die Abbildung 2.14 beschreibt, dass ein Mensch genau eine Geburtsurkunde hat und exakt. eine Geburtsurkunde höchstens einem Menschen zugeordnet ist

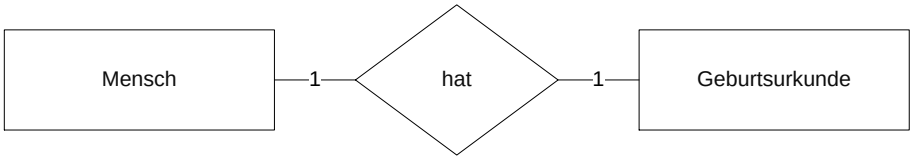


Abbildung 2.14: Datenmodell als 1:1 Beziehung

2.5.2 Normalisierung

Bei der Modellierung relationaler Datenbanken sind Redundanzen[7] von Tupel nicht zulässig, weil schwere Fehler bei Abfragen oder Speichervorgängen in Anwendungen auftreten können und keine Eindeutigkeit zur Identifizierung der Daten gegeben ist.
Datenbankmodelle müssen zuerst in eine Normalform gebracht werden, darunter ist zu verstehen, dass Informationen eines Sachverhalts in bestimmte Kategorien auf mehrere Relationen aufgeteilt werden, was auch als „One fact, One place“ Prinzip [6 S. 95] bezeichnet wird. Durch Anwendung bestimmter Regeln und schrittweises Zerlegen liegt eine Datenbank in einer Normalform vor. Bei der Zerlegung darf kein Datenverlust entstehen, sodass sich die Tupel wieder in ihre ursprüngliche Darstellung zusammensetzen lassen.

Es existieren mehrere Arten von Normalformen, an dieser Stelle sollen nur die bekanntesten erläutert werden.

[7] Datensätze in wiederholender Form

1. Normalform (1.NF)

Eine relationale Datenbank befindet sich in der 1. Normalform, wenn alle Attribute atomar, d.h. getrennt von anderen Eigenschaften vorliegen und ein Primärschlüssel für jedes Tupel vorhanden ist. Das folgende Beispiel in Abbildung 2.15 zeigt den Übergang einer nicht normalisierten Relation in die 1. Normalform.

PersNr	Name
100	Müller, Juliane, Bonn
101	Hansen, Hans, Hamburg
102	Meyer, Max, Köln

PersNr	Name	Vorname	Wohnort
100	Müller	Juliane	Bonn
101	Hansen	Hans	Hamburg
102	Meyer	Max	Köln

Abbildung 2.15: Nichtnormalisiert (l.), 1NF (r.)

2. Normalform (2.NF)

Eine Relation liegt in der 2. Normalform vor, wenn sie in der 1. Normalform ist und alle Nichtschlüsselattribute vom Primärschlüssel voll funktional abhängig sind.

Das folgende Beispiel in Abbildung 2.16 zeigt Mitarbeiter, die in einer Abteilung eine bestimmte Zeit gearbeitet haben.

PersNr	Name	AbtNr	AbtName	Arbeitszeit
100	Müller	SO5	Software	20
100	Müller	C20	Chemie	25
101	Meyer	SO5	Software	40

Abbildung 2.16: Relation in 1. NF

Die Attribute *PersNr* und *AbtNr* sind die Schlüsselkandidaten. Die folgende Überprüfung auf funktionale Abhängigkeiten zeigt, dass nur die Arbeitszeit voll funktional vom gesamten Schlüssel abhängig ist.

$$PersNr \rightarrow Name$$
$$AbtName \rightarrow AbtNr$$
$$PersNr, AbtNr \rightarrow Arbeitszeit$$

Um die Bedingung der 2. Normalform zu erfüllen ist eine Zerlegung in insgesamt drei einzelne Relationen notwendig.

PersNr	**Name**
100	Müller
101	Meyer

PersNr	**AbtNr**	**Arbeitszeit**
100	SO5	20
100	C20	25
101	SO5	40

AbtNr	**Abt Name**
SO6	Software
C20	Chemie

Abbildung 2.17: Relation erfüllt Bedingungen der 2.NF

3. Normalform (3.NF)

Eine Relation befindet sich in der 3. Normalform, wenn sie bereits in der 2. Normalform vorliegt und keine transitiven Abhängigkeiten zwischen einem Nichtschlüsselattribut und einem Schlüsselkandidaten existieren.

PersNr	**Name**	**Ort**	**AbtNr**	**AbtName**	**AbtChef**
100	Müller	Köln	SO5	Software	Becker
101	Meyer	Bonn	SO5	Software	Becker
102	Schmidt	Dortmund	C20	Chemie	Bayer

Abbildung 2.18: Relation in 2.NF

Das Beispiel in Abbildung 2.18 zeigt eine Relation, die in der 2. Normalform vorliegt. Die folgende Überprüfung soll Aufschluss geben, ob die Kriterien für die 3. Normalform eingehalten werden.

$$AbtNr \rightarrow AbtName$$
$$PersNr \rightarrow Name, Ort, AbtNr, AbtName, AbtChef$$
$$AbtNr \rightarrow AbtChef$$

An dieser Stelle ist bereits eine transitive Abhängigkeit des Nichtschlüsselattributs *AbtName* und den Schlüsselkandidaten zu erkennen.

$$PersNr \rightarrow AbtNr \rightarrow AbtName$$

Durch eine Auslagerung aller abteilungsrelevanten Informationen lässt sich die Relation in die 3. Normalform überführen.

PersNr	**Name**	**Ort**	**AbtNr**
100	Müller	Köln	SO5
101	Meyer	Bonn	SO5
102	Schmidt	Dortmund	C20

AbtNr	**AbtName**	**AbtChef**
SO5	Software	Becker
C20	Chemie	Bayer

Abbildung 2.19: Relation erfüllt 3.NF

Boyce-Codd-Normalform (BCNF)

Eine Relation ist in der BCNF, wenn sie in der 3. Normalformform vorliegt und jede Determinante ein Superschlüssel ist. Das folgende Beispiel in Abbildung 2.20 zeigt Personalnummern von Mitarbeitern, die an bestimmten Projekten arbeiten und genau einem Projektleiter unterstellt sind. Ein Projekt wird von mehreren Projektleitern durchgeführt.

PersNr	**Projekt**	**Projektleiter**
100	WP8.1	Friedrich
101	WP8.1	Gates
101	iOS	Tim
102	Wasserstoff	Seidler

Abbildung 2.20: Projektzuordnung in 3.NF

Die folgende Untersuchung soll Aufschluss geben, ob die Kriterien der BCNF verletzt werden. Hierzu muss der Schlüssel ermittelt werden.

$$PersNr, Projekt \rightarrow Projektleiter$$
$$PersNr \rightarrow Projekt$$
$$Projektleiter \rightarrow Projekt$$

Die Überprüfung zeigt, dass der Projektleiter vom zusammengesetzten Schlüssel aus den Attributen *PersNr* und *Projekt* funktional abhängig ist. Allerdings ist auch das Projekt vom Projektleiter funktional abhängig und verstößt gegen die BCNF, da das Attribut Projektleiter kein Schlüssel ist. Die Lösung ist die projektbezogene Informationen in eine neue Relation auszulagern.

PersNr	**Projektleiter**
100	Friedrich
101	Gates
101	Tim
102	Seidler

Projektleiter	**Projekt**
Friedrich	WP8.1
Gates	WP8.1
Tim	iOS
Seidler	Wasserstoff

Abbildung 2.21: Relation in BCNF

2.6 Datenbankwerkzeuge

2.6.1 SQL Developer

Der SQL Developer ist ein Entwicklungswerkzeug der Firma Oracle. Es besteht aus einer grafischen Bedienoberfläche, die das Design und die Bearbeitung von Oracle Datenbanken ermöglicht. Mit Hilfe von SQL Befehlen oder über ein Menü können alle Oracle Operationen ausgeführt werden, wie z.B. Datenmanipulation, Datentypänderungen, Erstellen oder Löschen von Datenbankobjekten und Attributwerten, sowie die Benutzerverwaltung und Zuordnung von Benutzerrollen. Die Relationen können auch in Dateiformate, wie Excel, CSV oder Text exportiert werden, um sie mit Hilfe anderer Programme weiterzuverarbeiten. Das in Java entwickelte Programm wird nicht auf dem Betriebssystem installiert, sondern muss nur entpackt werden. Die einzige Voraussetzung ist ein installiertes JDK[8]. Der SQL Developer ist mit dem JDBC[9] Treiber ausgestattet, der Verbindungen zu Datenbanken anderer Hersteller aufnehmen kann.

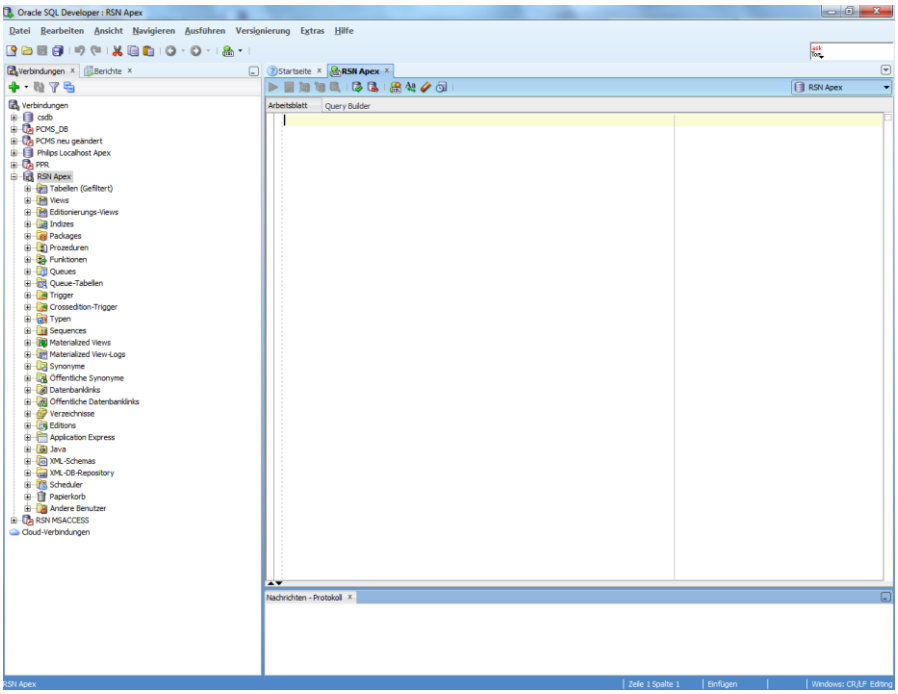


Abbildung 2.22: Arbeitsoberfläche des SQL Developer

[8] Abk. Java Development Kit: Enthält neben der Java Laufzeitumgebung weitere Entwicklerwerkzeuge, wie den Java Compiler.
[9] Abk. Java Database Connectivity

2.6.2 SQL Data Modeler

Der SQL Data Modeler ist ein von Oracle entwickeltes Werkzeug und basiert, wie der SQL Developer auf Java. Er ermöglicht das Erstellen von ER Modellen unterschiedlicher Notationen und unterstützt Reverse Engineering, d.h. aus einem SQL Code eines bestehenden Datenbankmodells lässt sich ein ER Modell generieren.

2.7 Microsoft Access

Microsoft Access (Kurzform: MS Access) ist ein RDBMS, welches Bestandteil der Bürosoftware Office Professional ist und vom Unternehmen Microsoft entwickelt und vertrieben wird. Es besitzt eine grafische Benutzeroberfläche und beinhaltet eine Entwicklungsumgebung mit verschiedenen Werkzeugen, die unter anderem zur Erstellung von Abfragen, Tabellen, Formularen und Makros dienen. Funktionen, die als Module bezeichnet werden, lassen sich mit Hilfe des integrierten VBA[10] Editors erstellen. MS Access bietet die Möglichkeit mit Datenquellen außerhalb der Entwicklungsumgebung Verbindungen herzustellen bzw. Daten als Text oder in tabellarischer Struktur zu importieren. Das RDBMS unterstützt SQL Befehle in eingeschränkter Form. MS Access Anwendungen und Datenbanken werden standardmäßig in einer Datei mit der Endung „accdb“ (in älteren Versionen als „mdb“) gespeichert. Integrierte Schnittstellen ermöglichen den Zugriff kompatibler Anwendungen von Drittanbietern auf MS Access Dateien. Eine mit MS Access erstellte Anwendung kann nicht unabhängig betrieben werden und setzt die Installation der Office Komponente voraus.

[10] Abk. der Programmiersprache Visual Basic

3 Oracle APEX

Der Software Konzern Oracle stellt mit seinem Produkt Oracle APEX (Oracle Application Express) ein lizenzfreies Werkzeug zur Verfügung, das webbasierte Datenbankanwendungsprogramme erstellen kann. Es ist nur mit RDBMS von Oracle kompatibel; in neueren Versionen, wie Oracle Database 11 g ist eine APEX Installation[11] bereits enthalten.
Oracle APEX gehört zur Kategorie der RAD-Tools[12] und ist die Nachfolgeversion von Oracle HTML-DB. Dem Entwickler wird ein Framework zur Verfügung gestellt, dabei handelt es sich um eine browserbasierte Entwicklungsumgebung mit grafischen Bedienelementen unterstützt von integrierten Assistenten mit denen die unterschiedlichsten Datenbankprozesse innerhalb einer Applikation realisiert werden können. Auf Basis einer Oracle Datenbank lassen sich mit grundlegenden Kenntnissen in den Bereichen Webdesign und Programmierung schnell effiziente und professionelle webbasierte Anwendungen erstellen, da wenig Quellcode erforderlich ist. Das Verhalten und die logische Struktur werden durch die Konfiguration von Parametern festgelegt. Die zugehörigen Metadaten sind in Systemtabellen innerhalb der Datenbank gespeichert. Lediglich SQL Anweisungen für Anwendungsdaten oder PL/SQL für definierte Programm Logik sind erforderlich.
APEX stellt Komponenten zur Verfügung mit denen sich nach einem Baukastenprinzip Formularmasken, Berichte oder grafische Darstellungsformen entwickeln lassen. Die Komponenten können mit HTML Code oder JavaScript erweitert werden. Allgemein ist die Entwicklungsumgebung mehrbenutzerfähig, ohne dass zusätzlich eine spezielle Software installiert werden muss. Anwendungen werden in einem virtuellen privaten Arbeitsbereich erstellt, der es mehreren Personen erlaubt gleichzeitig innerhalb einer APEX Installation zu arbeiten.

[11] Unter https://apex.oracle.com/i/index_de.html kann ein kostenloser Workspace angefordert werden, um die neusten APEX Versionen zu testen.

[12] Rapid Application Development Tools

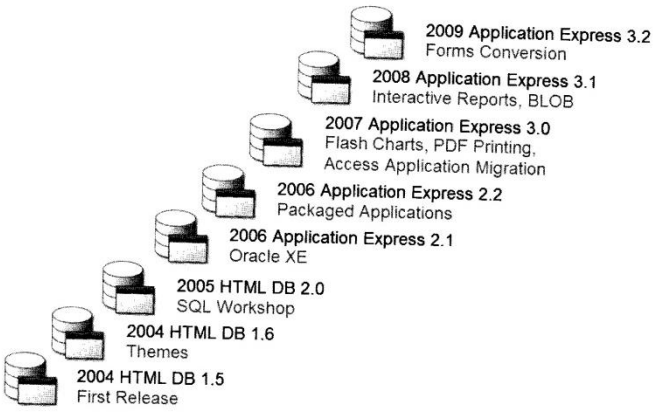


Abbildung 3.1: Entwicklungsverlauf von APEX bis zur Version 3.2 [7 S. 146]

3.1 Überblick zur Entwicklungsumgebung

3.1.1 Arbeitsbereich (Workspace)

Ein Arbeitsbereich ist ein virtueller privater Bereich auf einer Datenbank, der es mehreren Personen erlaubt darin zu arbeiten und Anwendungen zu erstellen. In jedem Arbeitsbereich können mehrere Applikationen vorhanden sein, alle Metadaten sind dort zusammenfasst. Datenbankobjekte befinden sich getrennt von der Applikation in einem Schema. Es können mehrere Schemata einem Arbeitsbereich zugeordnet sein.

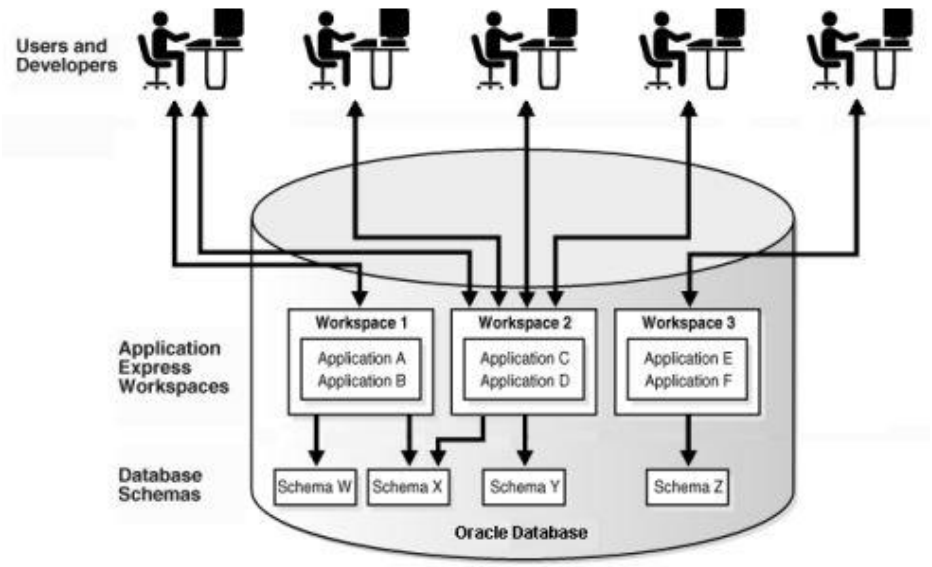


Abbildung 3.2: Mehrbenutzerzugriff auf verschiedene Workspace [8]

3.1.2 APEX Benutzerrollen

In einem Arbeitsbereich existieren vier unterschiedliche Benutzerrollen:

- APEX Administrator:

  Ist ein Superuser[13], der den gesamten APEX Service auf einem Oracle Server konfigurieren und verwalten kann.

- Workspace Administrator:

  Kann Verwaltungsaufgaben innerhalb eines Arbeitsbereichs ausführen, dazu gehören die Überwachung von Log Dateien und die Aktivitäten des Arbeitsbereichs, sowie das Anlegen und Verwalten von Benutzerkonten mit der Zuweisung von Administrator-, Entwickler- oder Endanwenderrechten.

- Entwickler (Developer):

  Anwender können Applikationen innerhalb eines zugewiesenen Arbeitsbereichs erstellen oder bearbeiten.

- Endanwender (End User):

  Kann nur erstellte Applikation ausführen und damit arbeiten, besitzt keine Möglichkeit eine Anwendung zu modifizieren.

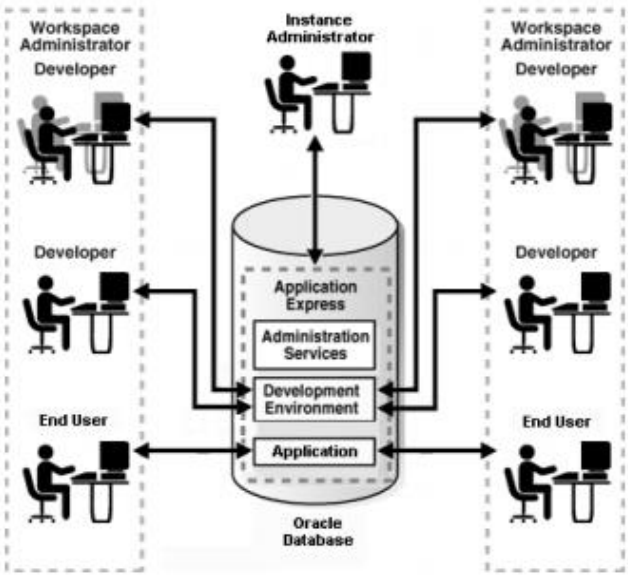


Abbildung 3.3: Benutzerrollenverteilung [8]

[13] Administrator mit Rechten der Systemverwaltung

3.2 Datenbankobjekt Werkzeuge

3.2.1 SQL Workshop

Der SQL Workshop enthält eine Sammlung von Werkzeugen zur Bearbeitung von Datenbankobjekten, die in diesem Abschnitt allgemein beschrieben werden.

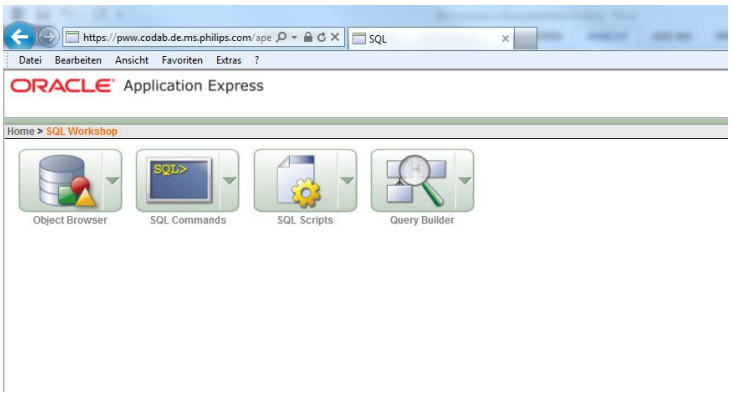


Abbildung 3.4: SQL Workshop Ansicht

Objekt Browser (Object Browser)

Im Objekt Browser lassen sich mit Unterstützung einer Menüführung:

- Relationen erstellen oder bearbeiten
- Sichten definieren
- Indizes optimieren
- Abfragen darstellen
- Trigger und Sequenzen programmieren
- Datenbankverknüpfungen konfigurieren
- Prozeduren und Funktionen erstellen

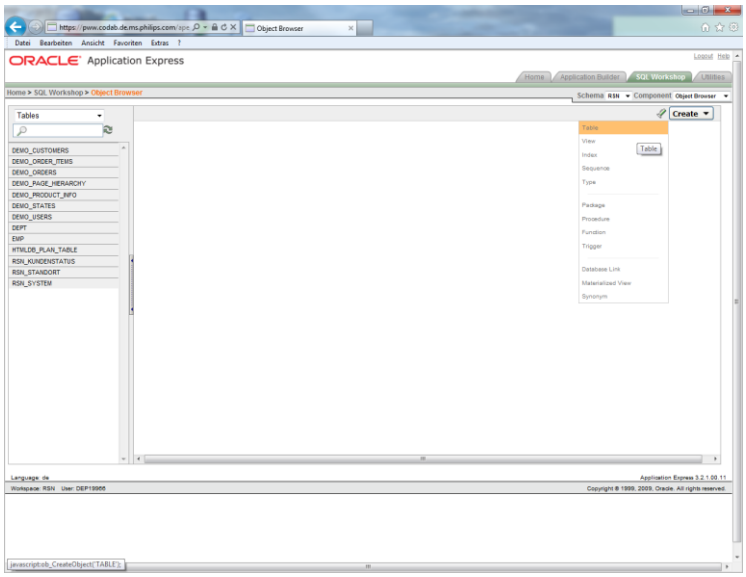


Abbildung 3.5: Oberfläche des Objekt Browsers

SQL Kommandos (SQL Commands)

Das Werkzeug SQL Kommandos entspricht einer Kommandozeilenanwendung und verarbeitet DML und DDL Anweisungen, die auf einem definierten Datenbankschema einzeilig ausgeführt werden können.

SQL Skripte (SQL Scripts)

SQL Skripte ermöglicht das Erstellen bzw. den Im- und Export von SQL Dateien, die über einen eingebauten Editor nachträglich bearbeitet werden können. Die Skripte lassen sich mit Hilfe des Werkzeugs ausführen. Ein Nachrichtenfenster informiert über den Ausführungsvorgang und zeigt bei nichterfolgreicher Ausführung die vorhandenen Fehler des SQL Skripts an.

Query Builder

Abfragen lassen sich über eine grafische Oberfläche erstellen, indem die entsprechenden Attribute, die zueinander in Beziehung stehen, ausgewählt werden. Automatisch wird durch den Query Builder die dazugehörige SQL Anweisung generiert. Das Abfrageergebnis kann in einem Ausgabefenster betrachtet werden.

3.2.2 Application Builder

Der Application Builder ist das Hauptwerkzeug der Entwicklungsumgebung, er dient zur Erstellung von Datenbankanwendungen innerhalb des Arbeitsbereichs. Ein integrierter Assistent unterstützt die Festlegung von Metadaten und erstellt auf Grundlage der gewählten Relationen eines Datenbankschemas eine klassische Client-Server Applikation, die die enthaltenen Daten über eine Benutzeroberfläche des Programms erfasst, bearbeitet und anzeigt.

Jeder Anwendung wird eine eindeutige Identifikationsnummer (Application ID) automatisch beim Erstellen zugewiesen, da jeder Arbeitsbereich mehrere Anwendungen enthalten kann. Export- und Importfunktionen bieten die Möglichkeit Applikationen auch in andere Arbeitsbereiche zu überführen.

Die Abbildung 3.6 zeigt zwei Anwendungen im Application Builder.

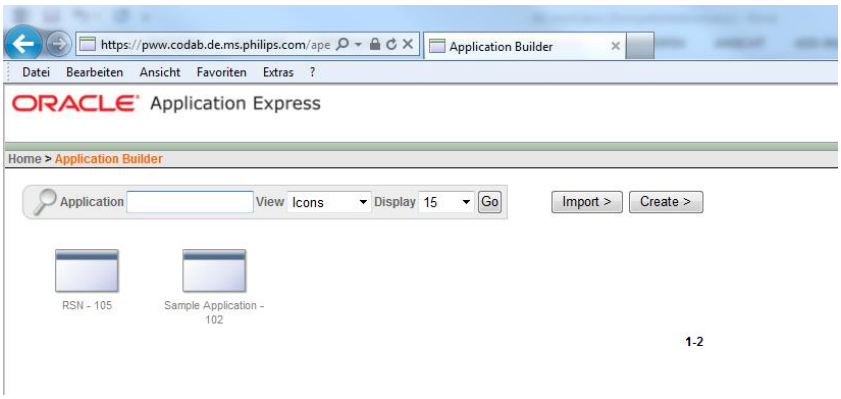


Abbildung 3.6: Ansicht im Application Builder

3.3 Allgemeiner Aufbau einer Applikation

Der folgende Abschnitt gibt einen ersten Einblick in den Aufbau einer APEX Applikation, um die im weiteren Verlauf angewandten Termini besser zu verstehen.

3.3.1 Seite (Page)

Jede webbasierte Anwendung ist vom Aufbau ähnlich mit einer Webseite zu vergleichen, die aus mehreren einzelnen Seiten besteht. Jede Seite verfügt über eine eindeutige Seitennummer (Page ID), die automatisch durch den Application Builder vergeben wird. Die jeweilige Seitennummer wird beim Ausführen einer erstellten Seite zusätzlich als Parameter in der Adresszeile des Webbrowsers angezeigt.

Die allgemeine Syntax einer URL Adresse für einen Seitenaufruf setzt sich zusammen aus: http://< Serveradresse>/apex/f?p=<App>:<Page>:<Session>
Die ausgewerteten Parameter sind in Tabelle 3.1 beschrieben.

Parameter	Beschreibung
App:	ID der Applikation
Page:	Aktuelle Seitennummer
Session:	APEX generierte Session ID

Tabelle 3.1: APEX Query String Parameter

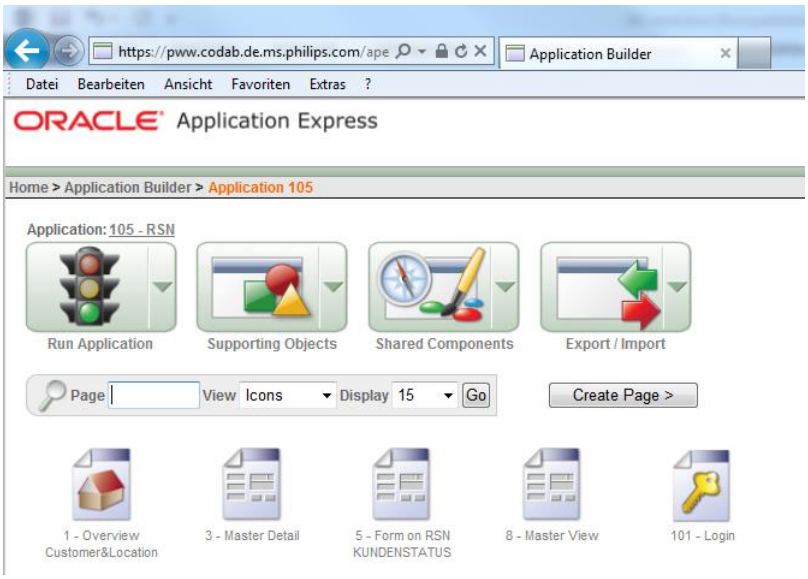


Abbildung 3.7: Aufbau einer Applikation

3.3.2 Regionen (Regions)

Eine Seite ist in mehrere Bereiche, den Regionen gegliedert, die bei der Erstellung bereits festgelegt werden müssen. Eine Region ist als Container für verschiedene Inhaltstypen zu verstehen. In der APEX Version 3.2 stehen folgende Regionstypen zur Verfügung, die in Tabelle 3.2 beschrieben sind:

Typ	Subtyp	Beschreibung
HTML	HTML	Container für Seitenelemente und zur HTML Inhaltsdarstellung der Regionsquelle.
	HTML Text (with shortcuts)	Erlaubt die Verwendung von HTML Shortcuts.
	HTML Text (escape special characters)	Sonderzeichen, die Bestandteil von HTML sind, werden ausgelassen.
Multi HTML		Ermöglicht das Anlegen mehrerer HTML Container.
Report		Ein klassischer oder interaktiver Bericht in tabellarischer Ansicht lässt sich aus einer SQL Anweisung generieren.
Form	Tabular	Ein Formular basierend auf einer Tabelle oder auf eine Sicht.
	Tabular Form	Formular in Tabellendarstellung.
	Procedure Arguments	Formular basierend auf eine Prozedur.
	Form with Report	Formular basierend auf Tabelle mit einem Bericht.
	SQL Query (fetch row)	Formular basierend auf SQL-Abfrage.
	Display Only on existing Items	Das Seitenelement zeigt enthaltenen Wert an, es kann nicht in das Element hineingeschrieben werden.
	Master Detail	Master-Detail-Formular zeigt zu jedem übergeordneten Datensatz, die darunterliegenden verknüpften Einträge.
	Form on Web service	Formular basierend auf Web Services.

	Form on Web service and report	Formular und Bericht basierend auf Web Service.
Chart		Daten werden mit Hilfe von SQL Anweisungen als Diagramm dargestellt.
PL/SQL Dynamic Content		Dynamische Inhalte von Packages können erstellt werden.
Tree		Darstellung einer Baumstruktur.
URL		Über eine URL können Inhalte eingebunden werden.
Calendar		Interaktiver Kalender
Help Text		Dient als Sammlung aller Hilfstexte der verschiedenen Seitenelemente.

Tabelle 3.2: Unterstützte Regionstypen

Alle Eigenschaften einer Region werden mit Hilfe einer Regionsdefinition (Region Definition) konfiguriert, darunter befinden sich die Darstellungsformen von Tabellen, die Anordnung innerhalb einer Seite, sowie die Datenquelle, d.h. welche Relation ausgelesen werden soll. Die folgende Tabelle 3.3 enthält die wichtigsten Definitionsbereiche, die in fast allen Regionstypen vorzufinden sind.

Definitionsbereich	Eigenschaft	Beschreibung
Identification	Page	Name der Seite
	Title	Überschrift der Region
	Type	Regionstyp
	Static ID	Ermöglicht die Vergabe einer statischen ID einer Region.
	Region Attributes	Regions Attribut
User Interface	Template	Darstellungsform der Region
	Sequence	Reihenfolgeposition innerhalb einer Seite.
	Display Point Page Template Body	Container Position innerhalb des Templates.
	Region HTML table cell attributes	HTML Formatierung der Region
Source	Region Source	Quelle der dargestellten Daten.

	Region Error Message	Fehlermeldung definieren
Conditional Display	ConditionType	Auswahl eines Bedingungstyps, d.h. Wahl einer bestimmten Regel.
	Expression 1	1. Ausdrucksparameter einer Bedingung.
	Expression 2	2. Ausdrucksparameter einer Bedingung.
Header and Footer	Region Header	Ermöglicht Eingaben von zusätzlichen HTML oder JavaScript Code in die Kopfzeile der Region.
	Region Footer	Ermöglicht Eingaben von zusätzlichen HTML oder JavaScript Code in die Fußzeile der Region.
Customization	Allow this region to be customized by end user	Definiert, ob der Endanwender die optische Darstellung einer Region verändern darf.
Comments		Kommentare für den Entwickler

Tabelle 3.3: Definitionsbereiche einer Region

3.3.3 Seitenelemente (Items)

Innerhalb einer HTML Region können Seitenelemente platziert werden.
Bei einigen Elementen ist vor der Verwendung auf die korrekte Definition des Datentyps zu achten. Die Entwicklungsumgebung stellt eine Auswahl von Seitenelementen bereit, die in Tabelle 3.4 beschrieben werden.

Typ	**Beschreibung**
Check Box	Kontrollkästchen zur Auswahl definierter Wertlisten
Date Picker	Ermöglicht die Auswahl des Datums über einen Kalender
Display Only	Dient nur zur Anzeige ausgelesener Werte
File Browse	Auf dem lokalen Dateisystem können Dateien über eine Schaltfläche hochgeladen werden

Hidden	Ausgeblendetes Seitenelement; kann auch im Hintergrund Werte übergeben/auslesen.
List Manager	Auswahl aus einer Werteliste.
Multiple Select	Mehrfachaus- und Abwahl von Einträgen einer Werteliste.
Password	Anmeldemaske mit Passworteingabefeld.
Popup List of Values	Pop-up Liste von der ein Wert ausgewählt werden kann
Radio	Auswahlliste, die genau eine Option zulässt
Select List	Eingabefeld mit Auswahlliste vordefinierter Werte
Shuffle	Auswahlsteuerelement zwischen zwei Listen
Text	Eingabe und Anzeige von Werten oder Strings
Text Area	Eingabe mehrzeiliger Texte
Stop and start table	Erzeugt im Tabellenlayout einen neuen Abschnitt

Tabelle 3.4: Unterstützte Items

3.3.4 Schaltfläche (Button)

Eine Schaltfläche kann beliebig einer Region zugewiesen und dort angeordnet werden. Beim Auslösen der Schaltfläche können bestimmte Ereignisse, wie Prozeduren oder Verzweigungen auf eine andere Seite hinterlegt werden.

3.3.5 Theme und Mustervorlage (Templates)

Ein Theme besteht aus HTML und JavaScript, es definiert das Erscheinungsbild von Seiten, Regionen, Seitenelementen oder Schaltflächen. Eine Anwendung kann jeweils nur aus einem Theme bestehen. Mustervorlagen legen das Layout innerhalb einer Seite fest, d.h. die Bereiche, wo Regionen, Seitenelemente und Schaltflächen positioniert werden.

3.3.6 Seitenwiedergabe (Page Rendering)

Die Seitenwiedergabe ist zuständig für das Erscheinungsbild und die Anordnung einzelner Seitenelemente. Sie teilt sich in sechs Abschnitte, jeder Abschnitt besitzt wiederum für jedes Element eine eigene Konfiguration:

- Seite (Page): Elementare Informationen zur HTML Region werden beschrieben.
- Regionen (Region): Definieren die einzelnen Containertypen und Bereiche in denen Inhalte dargestellt werden sollen.
- Schaltflächen (Buttons): Stoßen DML Anweisungen, Weiterleitungen auf eine andere Seite oder SQL/PL Parameter an.
- Seitenelemente (Items): Sind die einzelnen Felder des Webformulars.

- Berechnungen (Computations): Führt mathematische Berechnungen direkt beim Aufrufen der Webansicht aus.
- Prozesse(Processes): Geschäftslogik lässt sich hinterlegen, die bestimmte Arbeitsabläufe beim Ausführen der Seite durchführen soll.

Wird eine Seitenwiedergabe ausgeführt, so werden zunächst die Parameter der URL Adresse überprüft, hieraus werden Workspace, Anwendung, Seite und Session abgeleitet. Danach werden die Spracheinstellungen geprüft, das Datums- und Zahlenformat festgelegt und der Benutzer aus dem APEX Cookie abgeleitet und authentifiziert. Die Vorlage für die Seitendarstellung wird ermittelt und der aktuelle Sitzungszustand aller Variablen ausgelesen. Anschließend wird eine Zugriffskontrolle ausgeführt (Fine-Grained Access Control[14]), die sicherstellen soll, dass die Programmierschnittstelle (API) nur berechtigte Personenkreise, die Erlaubnis gibt auf Datenbankobjekte zuzugreifen.
Danach werden die Variablen an die URL übergeben und in den Sitzungszustand geladen. Der Prozess zum automatischen Abruf der Tupel wird angestoßen und Regionen oder Seitenelemente, deren Datenherkunft auf Attribute verweisen, werden mit den Werten gefüllt. In allen Formularmasken werden die `SELECT` Anweisungen ausgeführt. Die Ergebniswerte werden in einen temporären Speicher geschrieben und anschließend in das HTML Formular übernommen. Sind auf der Seite Berechnungen vorhanden, so werden diese ausgeführt, danach vorhandene Prozesse gestartet. Es folgt die Anzeige des Feldes, dabei werden Default Werte gesetzt, sofern kein Ergebnis existiert und überprüft, ob eine weitere Berechnung definiert ist. Das Ergebnis wird in den temporären Speicher geschrieben. Abschließend erfolgt eine implizite `COMMIT` Anweisung und der Ausgangspuffer wird an den Browser zurückgeliefert.

[14] Die Zugriffskontrolle wird auch als Virtual Private Database (VPD) bezeichnet [13]

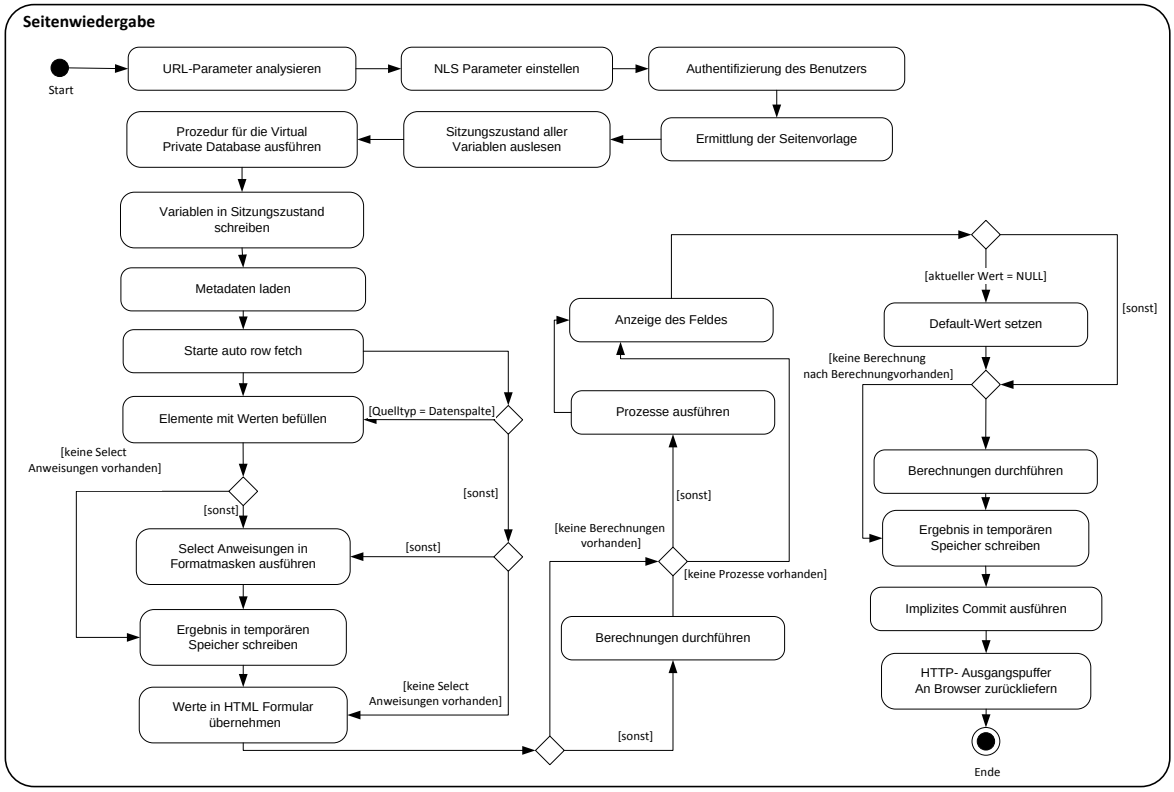


Abbildung 3.8: Aktivitätsdiagramm zur Seitenwiedergabe

3.3.7 Seitenverarbeitung (Page Processing)

Die Seitenverarbeitung führt Berechnungen, Verzweigungen, Prozesse oder Vergleiche aus und ist in vier Abschnitte unterteilt:

- Berechnungen (Computations): Führt mathematische Berechnungen mit den dargestellten Werten eines Webformulars durch und liefert ein Ergebnis.
- Validierungen (Validations): Unter bestimmten Bedingungen können Werte miteinander verglichen bzw. abgeglichen werden.
- Prozesse (Processes): Geschäftslogik lässt sich hinterlegen, die bestimmte Arbeitsabläufe nach Ausführen der Seite starten soll.
- Verzweigungen (Branches): Definiert wohin der Anwender nach Verarbeitung der Seite geleitet werden soll.

Bei der Seitenverarbeitung werden die Prozesse der Applikation und der aktuellen Seite nach einer bestimmten Reihenfolge abgearbeitet. Zuerst werden die Metadaten aus der Anwendung geladen, danach folgt die Sprach- und Datumsformateinstellung für die aktu-

elle Sitzung durch die NLS-Parameter. Der Sitzungszustand aller Variablen wird ausgelesen und der Prozess Fine-Grained Access Control ausgeführt.
Wie bei der vorherigen Seitenwiedergabe, erfolgt die Authentifizierung über ein APEX Cookie. Anschließend werden die Metadaten der Seite geladen und die Autorisierung der Seite überprüft. Die aktuellen Werte der Variablen des Sitzungszustands werden gesichert. Es werden mögliche Verzweigungen ausgeführt, die vor den Berechnungen ausgelöst werden sollen. Falls Berechnungen existieren, so werden diese anschließend durchgeführt. Als nächster Schritt wird überprüft, ob Verzweigungen vorliegen, die vor einer Validierung erfolgen sollen. Anschließend werden die Validierungen durchgeführt. Kommt es zu einem Fehler, so ist in der URL Adresse nur die *Angabe wwv_flow.accept* vorhanden, die eine APEX Fehlermeldung ausgibt. Ist die Validierung erfolgreich, so erfolgt eine Weiterleitung auf eine Zielseite und die URL Adresse ändert sich. Anschließend werden mögliche Verzweigungen ausgeführt, die noch vor bestimmten Prozessen arbeiten sollen. Danach folgen definierte Prozesse gefolgt von Verzweigungen, sodass APEX abschließend ein implizites `COMMIT` ausführt.

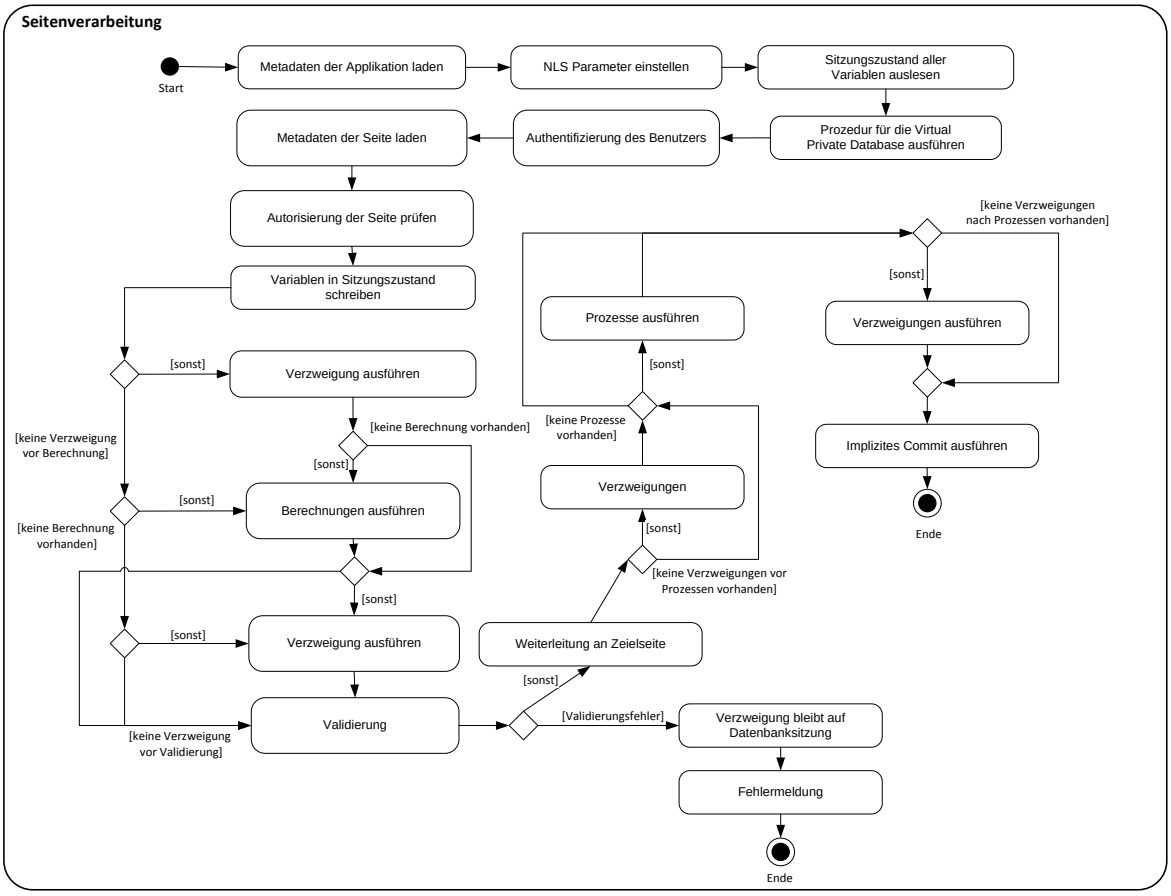


Abbildung 3.9: Aktivitätsdiagramm zur Seitenverarbeitung

3.3.8 Gemeinsam genutzte Komponenten (Shared Components)

Gemeinsam genutzte Komponenten stellen allen Bereichen einer Seite dieselben Komponenten zur Verfügung und definieren sich über eine einheitliche Layout Vorlage.

4 Analyse und Konzept

In diesem Kapitel werden zunächst die Anforderungen und Vorbedingungen der zu migrierenden Kundendatenbank und der zu realisierenden webbasierten Anwendung festgelegt. Anschließend wird eine Untersuchung des Systems im Ist-Zustand durchgeführt und verschiedene Konzepte erläutert, die eine geeignete Lösung der Anforderungen bieten.

4.1 Anforderungen

Verlustfreie Datenbankmigration
Die Anforderung mit der höchsten Priorität ist die Sicherstellung, dass bei der Migration der Kundendatenbank in ein Oracle Schema keine Verluste entstehen dürfen.

Einschränkung des Datenbankmodells
Die Anzahl der Relationen beschränkt sich auf dieselbe Anzahl der Relationen aus dem Ist-Zustand. Allgemein wird bei relationalen Datenbanken eine Aufteilung bis zur Boyce-Codd-Normalform angestrebt, was zu mehreren Teilrelationen führen kann.

Die Beschränkung ist weniger auf den Aspekt möglicher Performance Einbußen bezogen, sondern auf den Benutzerkreis und den Direktiven der IT Abteilung. Denn bei der Anwenderzielgruppe handelt es sich um Personen mit geringem Vorwissen über Datenbankstrukturen, die für die Pflege der Datenbank selbst verantwortlich sein werden, da in diesem Fall die firmeninterne IT Abteilung aufgrund Personalmangels keine Ressourcen zur Verfügung stellt.

Deshalb sind explizit einfache Strukturen, die nur bis zur 2. NF führen von der Benutzergruppe gefordert worden.

Filterfunktionen

Ausschlaggebend für eine Kundendatenbankanwendung sind Filterfunktionen, die es dem Anwender erlauben schnell alle notwendigen Informationen, die für das Kundengeschäft erforderlich sind, wiederzufinden oder um nach bestimmten Kriterien eine sortierte Darstellung zu erhalten.

Kontrollmechanismus

Beim Arbeiten mit Kundendaten sind Kontrollmechanismen sinnvoll, die nicht nur ausschließlich dem Datenbankadministrator vorbehalten sind, sondern auch dem Benutzerkreis Informationen zur Verfügung stellen, zu welchem Zeitpunkt welcher Benutzer an einem Kunden Änderungen ausgeführt hat. Eine Protokollierung des Datums und der Benutzerkennung ist hierzu notwendig.

Historie

Wie in jeder Anwendung, die auf Kundenbeziehungsmanagement ausgelegt ist, ist es erforderlich nach jedem Kundengespräch, die vereinbarten Aktionen festzuhalten.
Das bedeutet, dass für jeden Kunden eine Berichterstattung, eine Historie, die beliebig viele Einträge enthalten kann, benötigt wird.

Verlinkung auf lokales Dateisystem

Für Kunden existieren auf einem Netzlaufwerk Verzeichnisse in denen Dokumente, wie Verträge oder Datenschutzvereinbarungen hinterlegt sind. Aus diesem Grund besteht eine Anforderung, dass es über die Webapplikation dem Anwender ermöglicht werden soll mittels einer Verknüpfung direkt einen Kundenordner im lokalen Dateisystem zu öffnen.

Verlinkung auf webbasiertes Wartungs- und Analysewerkzeug

Im Tagesgeschäft müssen technische Störungen beim Kunden analysiert und behoben werden. Hierfür wird das unternehmenseigene webbasierte Wartungs- und Analysewerkzeug, das PRS Portal genutzt. Eine Verbindung zwischen der Kundendatenbankanwendung und dem PRS Portal tragen zu einer vereinfachten Arbeitsweise bei, um Problemstellungen noch schneller lösen zu können.

Erweiterbarkeit

Die Anwendung sollte so ausgelegt sein, dass sie auch zukünftig mit geringem Aufwand erweitert werden kann bzw. eine neuere Version auf die Grundstruktur der Anwendung schnell aufbauen kann

4.2 Voraussetzungen

Es wird vorausgesetzt, dass die Entwicklungsumgebung APEX in der Version 3.2 auf einem Server mit Oracle Lizenz bereits eingerichtet ist, ein Schema mit der Bezeichnung „RSN“ zur Verfügung steht und die Benutzerrolle „Entwickler“ vergeben wurde.
Die Administration des Servers und der APEX Verwaltung ist nicht Gegenstand dieser Bachelorthesis. Für das PRS Portal und den Kundenordner auf dem Netzlaufwerk müssen die entsprechenden Zugriffsrechte vorhanden sein. Der Kundenordner wird in der Voreinstellung mit dem Pfad *L:\Administration\~Kunden* in Windows eingebunden.

4.3 Ist-Zustand

Ausgangsbasis bildet die aktuell verwendete Kundendatenbank, die für die zu erstellende webbasierte Anwendung im weiteren Verlauf migriert werden muss. Hierfür ist es erforderlich zunächst einen Überblick über die optische Darstellung, die Funktionen des Datenverwaltungsprogramms und der Datenbankstruktur zu erhalten.

4.3.1 Kundendatenverwaltung

Bisher wird die Kundendatenverwaltung in MS Access über ein Hauptformular (siehe Abbildung 4.1), das aus mehreren Unterformularen besteht, ausgeführt. Jedes Unterformular ist mit einer Relation, der dahinterliegenden Datenbank verknüpft. Es ist nur eine Listenauswahl des Kundennamens zulässig und nicht möglich die Kundendatenverwaltung nach anderen Einträgen zu durchsuchen. Eine allgemeine Übersicht, die den Kundenbestand zeigt, ist nicht vorhanden. Der Anwender befindet sich direkt in einem Editiermodus; die Formularmaske ist nach der Master-Detail Beziehung aufgebaut. Der Detailbereich enthält Informationen über die Netzwerkparameter und Seriennummern der einzelnen Systeme, die beim Kunden angebunden sind.

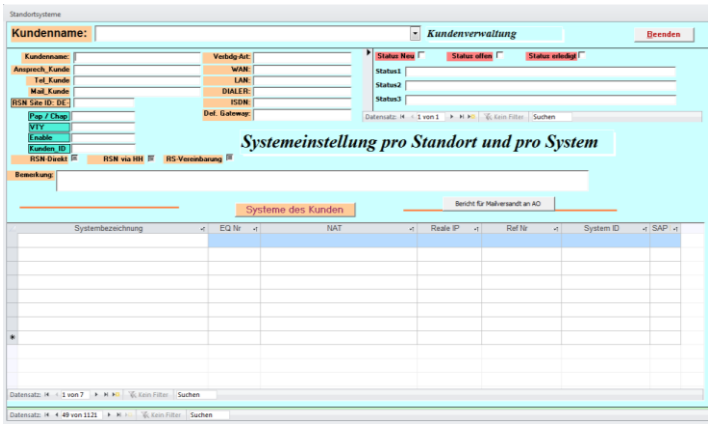


Abbildung 4.1: Ansicht der Kundenverwaltung in MS Access

4.3.2 Datenbankkonsistenz

Die Datenbank besteht aus drei Relationen mit den Bezeichnungen „Standort“, „System“ und „Kundenstatus“. Jedes der genannten Objekte verfügt über einen Primärschlüssel (ID), sodass sich die einzelnen Tupel eindeutig identifizieren lassen. Auffällig ist, dass mehr Attribute zur Verfügung stehen, als tatsächlich in dem Formular zur Kundendatenverwaltung verwendet werden. Jede Relation ist auf einen Sachverhalt abgegrenzt, dabei enthalten die Datenbankobjekte mit den Bezeichnungen „System“ und „Kundenstatus“ jeweils Fremdschlüssel, die sich auf die Relation „Standort“ beziehen, in der die Kundendaten und Zugangsparameter gespeichert sind. Allerdings liegt das Attribut zur Kundenbeschreibung in einer nicht atomaren Darstellung vor, da es mit einer Ortsangabe zusammengesetzt ist. Eine weitere Problematik zeigt sich in den Attributwerten, diese sind nicht immer vollständig oder fehlen ganz, was insbesondere die Angaben zur Kundennummer betrifft. Die Datentypen aller Relationen sind zum größten Teil Zeichenketten, da in vielen Fällen die Informationen aus Text bestehen und ganzzahlige Werte kaum vorkommen.
Die Relation „Standort“ enthält zudem für einige Kunden eine Art Berichterstattung und temporäre Anmerkungen, die genauer betrachtet einen Status beschreiben und an dieser Stelle in die dafür vorgesehene Relation „Status“ gehören. Daraus ist zu schließen, dass Richtlinien der Data Governance missachtet worden sind, d.h. die Pflege und Kontrolle der Stammdaten vernachlässigt worden ist.

4.3.3 Referentielle Integrität

Referentielle Integrität gewährleistet die Konsistenz und Integrität der Daten, d.h. Tupel mit Fremdschlüssel lassen sich nur anlegen, wenn in der referenzierten Relation der Wert des Fremdschlüssels eindeutig und einmalig vorhanden ist. Ist dies nicht der Fall, so lässt sich kein weiteres Tupel erstellen.

MS Access bietet in dieser Hinsicht die Möglichkeit vorhandene Beziehungen über ein Beziehungsfenster darzustellen. Das Ergebnis in Abbildung 4.2 gibt Aufschluss darüber, dass zwischen den drei Relationen „Standort“ (in der Abbildung als „Standort_2“ bezeichnet), „Status“ (in der Abbildung als „Status_Kunde“ bezeichnet) und „System“ (in der Abbildung als „Systeme_2“ bezeichnet) keine referentielle Integrität existiert.

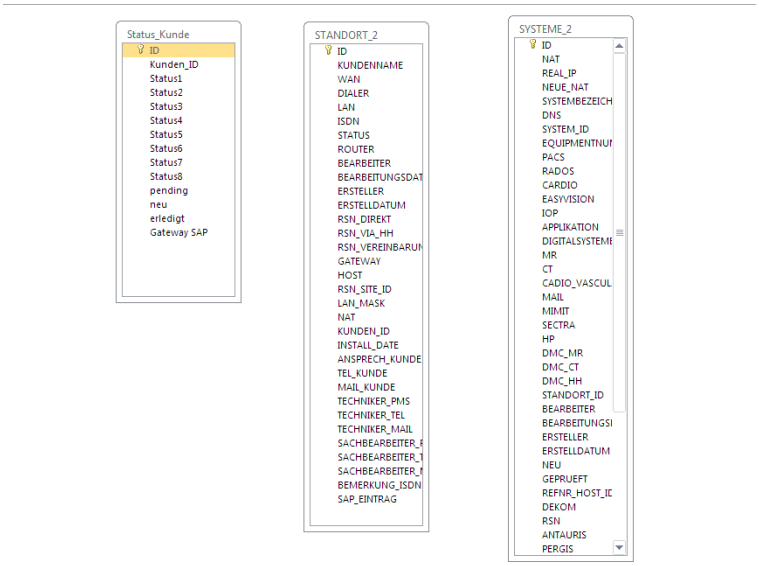


Abbildung 4.2: Beziehungen der Kundendatenbank

Eine Verknüpfung besteht nur über die Unterformulare in der Kundendatenverwaltung, welche die Fremdschlüssel mit der Hauptrelation referenzieren kann. Das Problem bei dieser Struktur ist, dass Anomalien auftreten können, d.h. wird ein Tupel im übergeordneten Datenbankobjekt entfernt, so folgt daraus, dass die verknüpften darunterliegenden Tupel nicht automatisch gelöscht werden. Verwaiste Einträge, die das Datenvolumen unnötig erhöhen und Speicherplatz belegen, sind das Resultat.

4.3.4 Zusammenfassung der Analyse

Die ersten Untersuchungen zeigen, dass es sich bei der Anwendung zur Kundendatenverwaltung inklusive der Datenbank um ein über die Jahre ständig modifiziertes System handelt, da einige Attribute anfangs noch genutzt, aber zu einem späteren Zeitpunkt für das Geschäftsfeld nicht mehr gebraucht wurden. Andererseits existieren Attribute, die keine Verwendung gefunden haben und somit keine Werte aufweisen.

Bei der Hauptrelation ist zu bemängeln, dass ein Verstoß gegen die Bedingung der 1. NF vorliegt, des Weiteren besitzt das Datenbankmodell keine referentielle Integrität und kann in den untergeordneten Relationen verwaiste Tupel enthalten, da keine Löschweitergabe existiert. Die fehlende Data Governance hat dazu beigetragen, dass Vermerke über Kundeninteraktionen in der falschen Relation abgespeichert wurden.

4.4 Migrationskonzept

Die erkannten Problemstellungen, die im vorherigen Abschnitt dargestellt worden sind, sollen nun durch geeignete Methoden gelöst werden.

4.4.1 Staging Area (Sammelplatz)

Staging[15] beschreibt ein Verfahren, das es ermöglicht Daten zentral in einem Bereich zu sammeln, der als Staging Area bezeichnet wird und von den Produktionsdaten isoliert ist (Sandbox). Der Bereich dient zur Aufbereitung der Daten, die aus verschiedensten Datenquellen gesammelt werden können. Dahinter steckt das Prinzip die Daten zunächst zu bereinigen, zu transformieren und anschließend aus der Staging Area in eine Zieldatenbank zu überführen. Nicht selten enthält der bezeichnete Sammelplatz mehr Relationen, als das spätere Zielsystem, das gilt insbesondere für ein Data Warehouse (Datenlager).

Ralph Kimball und Margy Ross beschreiben die Staging Area anhand eines abstrakten Beispiels mit einer Küche eines Restaurants, in der die rohen Speisen in ansehnliche Gerichte transformiert werden, bevor sie den Gästen serviert werden. [9]

Staging setzt sich aus den drei folgenden Prozessen zusammen, die nacheinander durchgeführt werden:

Extraktion
Aus verschiedenen Quelldaten, das schließt auch Dateien mit Textinhalten ein, werden die gewünschten Informationen, d.h. die Attribute selektiert, sodass sie in der Staging Area in neue Relationen abgelegt werden können.

Transformation
Die gesammelten Tupel werden bereinigt und die Struktur der Relationen an das Zielsystem angepasst.

Laden
Die aufbereiteten Daten werden abschließend in ein Zielschema geladen.

[15] eng.: Sammeln

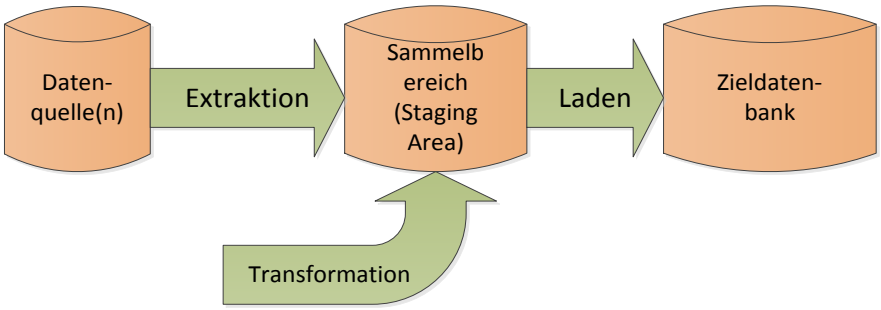


Abbildung 4.3: Staging mit ETL Prozess

Die genannten Prozesse werden zusammengefasst als ETL Prozesse bezeichnet und üblicherweise für ein Data Warehouse verwendet. Der Vorteil von Staging ist, dass aufgrund der Isolierung des Sammelbereichs, die Datenquellen nicht weiter belastet werden und dem Anwender ohne Einschränkungen weiter zugänglich sind. Ein weiterer Punkt ist die Wiederverwendbarkeit, denn sollten Änderungen von Datenbeständen oder der Struktur vorliegen, so lässt sich dieses mit Hilfe der Staging Area für das Zielsystem anpassen.

Das Staging Prinzip eignet sich für die zugrunde liegende Datenbank, da es mit Hilfe der Extraktion möglich wird die Kriterien der 1. NF zu erfüllen. Zu beachten ist aber, um den Anforderungen gerecht zu werden, dass die als Quelle definierten Relationen nicht zusammengefasst werden dürfen, sondern weiterhin als drei einzelne Datenbankobjekte in der Staging Area abgelegt werden müssen.

Der Sammelbereich wird innerhalb der MS Access Kundendatenbank angelegt werden, was keine Beeinträchtigung im Produktionssystem hervorrufen wird und die Anwender wie gewohnt weiter arbeiten können.

Der nächste Schritt, die Transformation, wird die Relationen von möglichen verwaisten Tupel bereinigen. Auf die Anpassung des Datenbankmodells, was Beziehungen betrifft, sollte an dieser Stelle verzichtet werden, da die Praxis gezeigt hat, dass bei Ladevorgängen zwischen MS Access nach Oracle die Beziehungen des Datenbankmodells nicht ordnungsgemäß übertragen werden und im Zielsystem erneut definiert werden müssen.

Abschließend können die Daten ins Oracle Schema geladen werden, was in Kapitel 4.4.4 detaillierter erläutert wird.

4.4.2 Muster Relationen (Dummy Tables)

Eine Webapplikation basiert auf einem relationalen Datenbankmodell (vgl. Kapitel 3.3.2), d.h. es ist zunächst erforderlich ein Muster ins Oracle Schema zu importieren. Beim Muster handelt es sich um die exakten Abbilder der Relationen, die in der Staging Area erzeugt worden sind. Allerdings werden nicht die Attributwerte, sondern nur die Struktur der Datenobjekte benötigt. Das Muster wird als Grundlage der zu realisierenden webbasierten Datenbankanwendung verwendet, um die Anwendung mit den geforderten Funktionen erstellen zu können.

Mit Hilfe des SQL Developer existieren verschiedene Möglichkeiten die Muster ins Oracle Schema zu überführen:

Migrationsassistent

Ein Migrationsassistent kann ein komfortables Hilfsmittel sein, wenn Datenbanken, ob lokal oder auf einem Server, in ein anderes RDBMS übertragen werden sollen. Hierfür werden als Vorbereitung alle Metadaten in ein Repositorium abgelegt und in Oracle konforme Metadaten konvertiert. Anschließend wird ein Skript generiert, um die Datenbank im Zielschema anzulegen (siehe Abbildung 4.4).

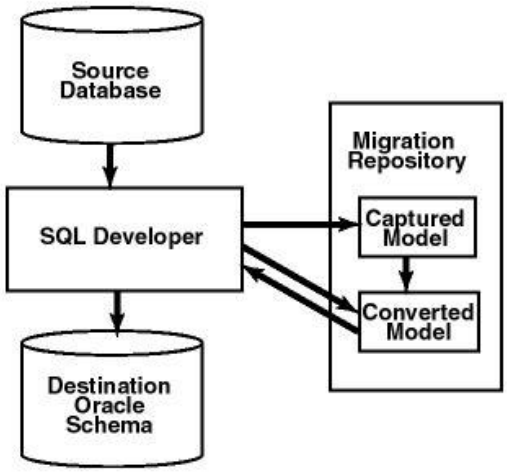


Abbildung 4.4: Migrationsarchitektur [10]

Ein wesentlicher Nachteil des Migrationsassistenten besteht aber darin, dass es nicht möglich ist einzelne Relationen für das Zielschema auszuwählen. Ein weiteres Problem sind die Benutzerrollen und Zugriffsrechte auf das verwendete Schema in dieser Arbeit, da der Workspace Administrator dem Entwickler nicht gestattet ein Repositorium zu erzeugen.

DDL Skript

Aus bestehenden Datenbankobjekten ist es möglich ein Skript zu generieren, das alle Informationen über die Metadaten und die Speicherorte enthält. Der Vorteil ist, dass eine Auswahl getroffen werden kann und ein Exportassistent die Möglichkeit bietet das Skript für alle gewählten Datenbankobjekte als eine Datei zu generieren. Außerdem kann das Skript jeder Zeit individuell angepasst werden. Das erstellte Skript wird anschließend auf dem Oracle Schema ausgeführt und erstellt die Muster aus der Staging Area der MS Access Kundendatenbank.

Direktkopie

Datenbankobjekte können auch direkt in ein Schema kopiert werden. Hierzu werden die benötigten Objekte einzeln oder gemeinsam ausgewählt und lassen sich ohne einen weiteren Arbeitsschritt in ein Zielschema kopieren.

Zusammenfassend bedeutet das hinsichtlich der drei erläuterten Methoden, dass die Direktkopie der Lösungsweg mit dem geringsten Aufwand ist die Muster Relationen zu migrieren.

4.4.3 Beziehungen

Nachdem die erwähnten Muster im Schema abgelegt sind, müssen die Beziehungen der Relationen im Datenbankmodell hergestellt werden. Die Analyse (vgl. Kapitel 4.3.3) hat ergeben, dass die Beziehungen zwischen den einzelnen Relationen über eine Verknüpfung auf Ebene der Formulare realisiert worden ist. Da bereits Fremdschlüssel in den Relationen „Status“ und „Systeme“ existieren, lassen sich die direkten Referenzen erkennen. Die beiden Fremdschlüssel zeigen auf den Primärschlüssel der Relation „Standort“.

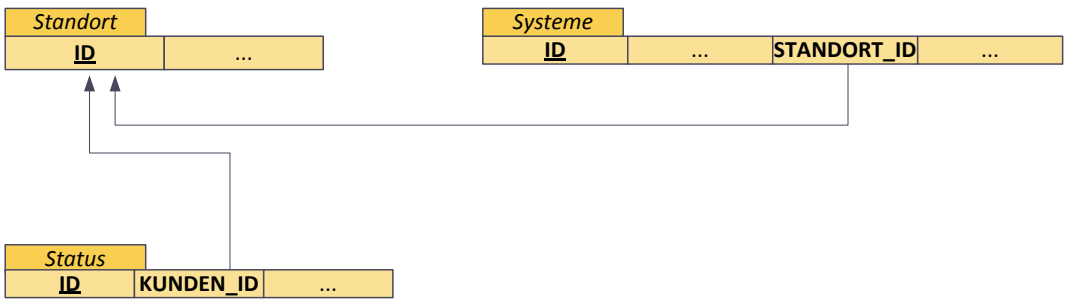


Abbildung 4.5: Referenz auf die Relation Standort

Auch die Kardinalitäten lassen sich anhand der durchgeführten Analyse (vgl. Kapitel 4.3.1) ermitteln, da ein Master-Detail Formular verwendet worden ist. Daraus ergeben sich die folgenden festgelegten wechselseitigen Beziehungen:

- Jeder Standort besitzt mehrere Systeme und mehrere Systeme sind genau einem Standort zugeordnet.
- Jeder Standort verfügt über einen Status und viele Status beziehen sich genau auf einen Standort.

Aus konzeptioneller Sicht betrachtet, bedeutet es, dass je zwei schwache Entitätstypen mit jeweils einer 1:n Kardinalität zur Entität „Standort“ in Beziehung stehen. Das zu erwartende ERM ist in Abbildung 4.6 dargestellt. Es zeigt ein vereinfachtes Modell in Chen Notation. Aufgrund der großen Anzahl von Attributen ist auf die Darstellung im ERM verzichtet worden.

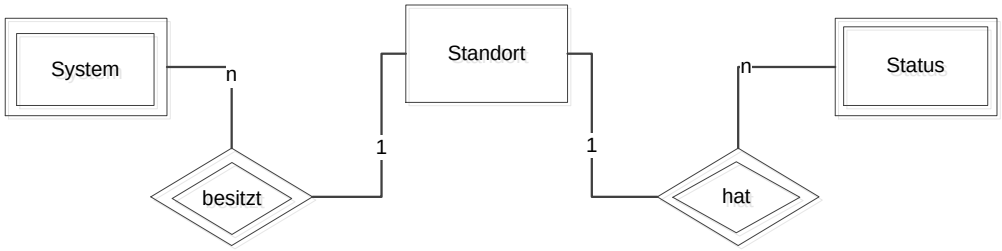


Abbildung 4.6: ERM in Chen Notation

Die schwachen Entitäten erfordern, dass das Datenbankmodell über eine Löschweitergabe verfügt, da die Entitäten in den Entitätstypen „System“ und „Status“ nicht weiter existieren können, wenn übergeordnete Entitäten aus dem Entitätstyp „Standort“ entfernt werden.

4.4.4 Daten laden

Das Laden der Kundendaten ins Oracle Schema wird erst vor Inbetriebnahme der Anwendung ausgeführt werden. Es wird zunächst erforderlich sein, die in der Staging Area gespeicherten Daten zu aktualisieren und anschließend mit Hilfe des SQL Developer in ein DDL Skript zu exportieren. Das Skript ist nachträglich auf die Relationen im Oracle Schema anzupassen, da sich die Bezeichnungen der Attribute während des Entwicklungsprozesses geändert haben können. Danach sind die Testdaten im Oracle Schema

aus den Mustern vollständig zu entfernen. Abschließend wird das DDL Skript ausgeführt und die Daten ins Schema geladen.

4.4.5 Gesamtkonzept der Datenmigration

Die Staging Area wird in der MS Access Kundendatenbank erzeugt. Der Extraktionsprozess erfolgt über Abfragen auf die Quelldaten. Gleichzeitig wird mit der Abfrage auf die Relation „Standort“ sichergestellt, dass die Relation in atomarer Darstellungsform erzeugt wird. Die Attributwerte vom Datentyp „Datum“ müssen ins deutsche Datumsformat zwangskonvertiert werden, damit diese ordnungsgemäß im Oracle Schema abgebildet werden können.

Anschließend ist eine Löschabfrage auf die Relationen „System“ und „Kundenstatus“ auszuführen, damit mögliche verwaiste Tupel entfernt werden.
Von jeder Relation ist eine Kopie der Struktur anzufertigen, welche mittels SQL Developer ins Zielschema übertragen wird. Die genannten Kopien ergeben nun das Muster.

Der nächste Schritt ist die Anpassung des Datenbankmodells, es wird zwischen den Muster Relationen die referentielle Integrität inklusive der Löschweitergabe erstellt.

Basierend auf dem Datenbankmodell wird anschließend die Anwendung entwickelt (siehe Kapitel 6.3). Zuletzt erfolgt das Laden der Kundendaten, was ein erneutes Ausführen des Extraktions- und Transformationsprozesses zur Folge hat, damit die neuesten Daten in bereinigter Form in ein DDL Skript exportiert werden können.
Das Skript wird zum Schluss ausgeführt, sodass alle Daten aus der Staging Area ins Schema geladen werden.

4.5 Überwachungsfunktionen

4.5.1 Kontrollfunktion (Audit)

Hinsichtlich den Anforderungen (vgl. Kapitel 4.1) ist eine Kontrollfunktion, die aus der Benutzerkennung und dem aktuellen Datum besteht, notwendig.
Das bedeutet, dass zu allen Tupel der Hauptrelation die Zugriffsinformationen gespeichert werden müssen.
Der aktuelle Benutzername wird aus einer hinterlegten Liste aller zugangsberechtigter Mitarbeiter ausgelesen, das aktuelle Datum aus dem System bezogen. Sobald eine Änderung an einem Tupel vorgenommen wird, erfolgt eine automatische Speicherung. Es sollen im Folgenden zwei mögliche Konzepte betrachtet werden:

Fortlaufende Listenprotokollierung

Jede Änderung an den Kundendaten wird kontinuierlich gespeichert. Als Speicherort dient eine untergeordnete Relation, die wie im dargestellten Beispiel in Tabelle 4.1 jeden Zugriff protokolliert.

Datum	Benutzerkennung
Datum 3	ABC 123
Datum 2	XYZ 456
Datum 1	ABC 123

Tabelle 4.1: Schematische Darstellung der fortlaufenden Protokollierung

Hierfür ist es erforderlich, dass für jeden neuen Protokolleintrag Fremdschlüssel generiert werden müssen, anschließend werden das Systemdatum und die Benutzerkennung ausgelesen und als Attributwerte übergeben. Abbildung 4.7 zeigt die Zugriffsprotokollierung in Form eines Aktivitätsdiagramms.

Der Vorteil einer fortlaufenden Protokollierung ist, dass alle Änderungszugriffe zu Auswertungszwecken exakt zurückverfolgt werden können. Ein Problem besteht aber, dass in diesem Fall vorausgesetzt wird, dass bereits ein übergeordnetes Tupel existiert. Dieses hat zur Folge, dass beim Erstellen eines neuen Kunden zunächst keine Protokollierung ausgeführt werden kann.
Seitens der Anwender besteht allerdings keine Anforderung anzuzeigen an welcher Stelle innerhalb der Kundendaten eine Änderung eingetreten ist, sodass eine fortlaufende Listenprotokollierung einen geringen Nutzen erbringt.

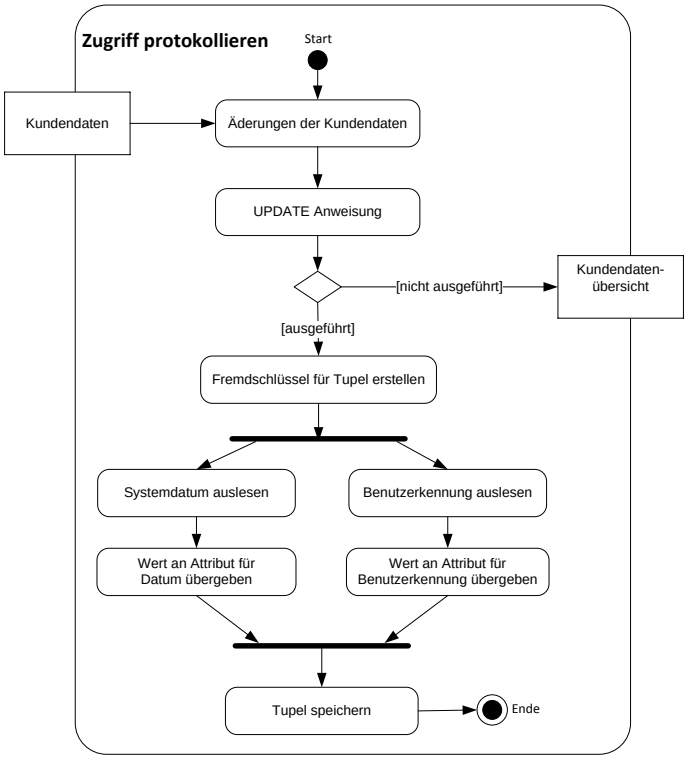


Abbildung 4.7: Aktivitätsdiagramm zur fortlaufenden Protokollierung

Kontinuierliches Überschreiben der Protokollwerte

Bei jeder Änderung eines Tupels in der Hauptrelation werden die Benutzerkennung und das aktuelle Datum im selben Datenbankobjekt gespeichert. Die Attributwerte werden bei jeder Zugriffsänderung überschrieben. Die Abbildung 4.8 zeigt den notwendigen Ablauf, der für dieses Konzept erforderlich ist.

Ein Vorteil dieser Lösung ist, dass nur der jeweils letzte Benutzerzugriff erhalten bleibt und hierfür bereits geeignete Attribute existieren, die wiederverwendet werden können (vgl. Kapitel 4.3.2). Auch beim Erstellen eines neuen Kunden ist eine sofortige Protokollierung möglich.

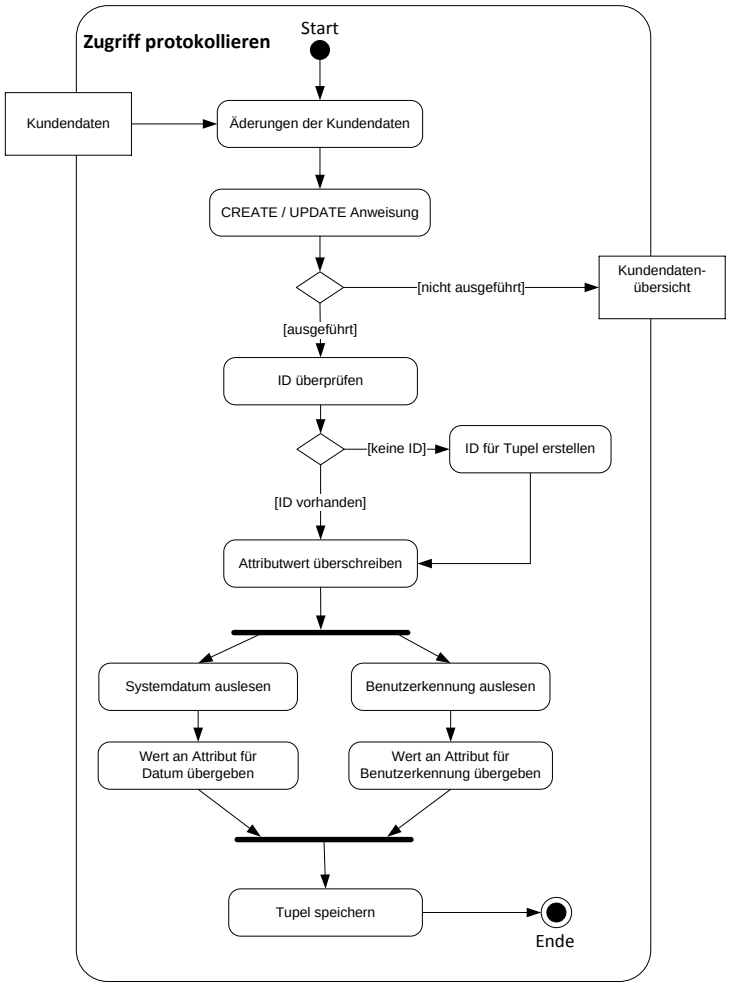


Abbildung 4.8: Aktivitätsdiagramm kontinuierliches Überschreiben der Protollwerte

4.5.2 Historie

Eine Historie enthält die gesamte Kommunikation zwischen dem Unternehmen und dem Kunden. Nach jeder Kundeninteraktion ist der zuständige Kundenbetreuer verpflichtet über den aktuellen Status und über die ausgeführten oder geplanten Tätigkeiten zu berichten. Die Berichte werden in Form einer Zusammenfassung dokumentiert, d.h. eine Historie setzt sich allgemein aus einer Beschreibung und einer Datumsangabe zusammen. Eine fortlaufende Listendarstellung ist hierfür erforderlich, die jeweils den letzten Eintrag an oberster Stelle anzeigen soll, damit der Anwender die Informationen schneller wieder finden kann (siehe Tabelle 4.2). Das Datum für jeden neuen Eintrag lässt sich, wie beim Konzept der Kontrollfunktionen (vgl. Kapitel 4.5.1) aus dem System entnehmen.

Datum	Beschreibung
Datum n	Log n
Datum 3	Log 3
Datum 2	Log 2
Datum 1	Log 1

Tabelle 4.2: Beispiel einer Historiendarstellung

Für die Speicherung der Historieneinträge ist die Relation „Kundenstatus“ geeignet, da dort einerseits schon Kundeninteraktionen in verschiedenen Attributen gespeichert sind und andererseits auch mehrere nicht genutzte Attribute existieren.

Aus der Analyse (vgl. Kapitel 4.3.2) ist bekannt, dass Teile der Kundenberichterstattung auch in der Hauptrelation zu finden sind. Es wird notwendig die Sachverhalte zwischen dem Kundenstandort und den Kundeninteraktionen strikt zu trennen, d.h. diese Attributwerte müssen in die untergeordnete Relation „Kundenstatus“ übertragen werden.

Ein weiterer Aspekt, der berücksichtigt werden muss ist, dass es in CRM[16] Anwendungen keine Möglichkeit geben darf Historieneinträge nachträglich zu bearbeiten. Der Grund ist die Sicherheit der lückenlosen Dokumentation. Eine Änderung älterer Einträge kann zu beabsichtigten Verfälschungen der Kundenberichterstattung führen.
Da es keine Korrekturmöglichkeit gibt, kann ein Anwender nur in einem weiteren Eintrag darauf hinweisen, an welcher Stelle ihm beim Berichten ein Fehler unterlaufen ist.

Zusammenfassung
Die Historie wird aus den Attributwerten der Relation „Kundenstatus“ dargestellt werden. Für die Speicherung des Datums und zukünftiger Einträge wird jeweils ein nicht genutztes Attribut verwendet werden. Die Attributwerte der Berichterstattung, die teilweise in der Hauptrelation existieren, werden ebenfalls in ein nichtgenutztes Attribut in der untergeordneten Relation „Kundenstatus“ gespeichert (siehe Abbildung 4.9). Dafür muss geprüft werden, ob ein Tupel mit dem passenden Fremdschlüssel bereits vorliegt, damit während des Kopierens die eindeutige Zuordnung erhalten bleibt. Existiert der Fremdschlüssel nicht, so wird er für das Tupel angelegt. Das nun überflüssige Attribut in der Relation „Standort“ wird danach vollständig entfernt.

[16] Customer-Relationship-Management: Kundenbeziehungsmanagement

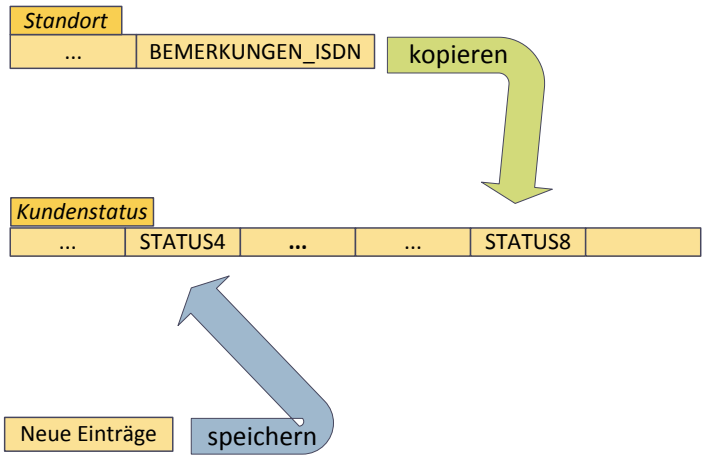


Abbildung 4.9: Schematisches Konzept

Abschließend sind alle Attributwerte der Relation „Kundenstatus“ verkettet und nach Datum absteigend sortiert darzustellen.

Die Historieneinträge sollen über ein Eingabeformular angelegt werden können, was im späteren Verlauf in Kapitel 6.3.4 näher erläutert wird.

4.6 Verlinkung auf externe Komponenten

Aus den Anforderungen (vgl. Kapitel 4.1) geht hervor, dass Verlinkungen auf ein Kundenlaufwerk und zum PRS Portal ermöglicht werden sollen. Hierfür müssen die beiden externen Komponenten getrennt voneinander analysiert werden, um anschließend eine Lösung zu erarbeiten.

4.6.1 Kundenlaufwerk

Auf einem eingebundenen Netzlaufwerk befindet sich ein Ordner mit kundenspezifischen Dateien, wie z.B. Verträge, Dokumente zu Datenschutzrichtlinien oder Checklisten.
Der Kundenordner wird über den Verzeichnispfad *L:\Administration\~Kunden\ erreicht.*

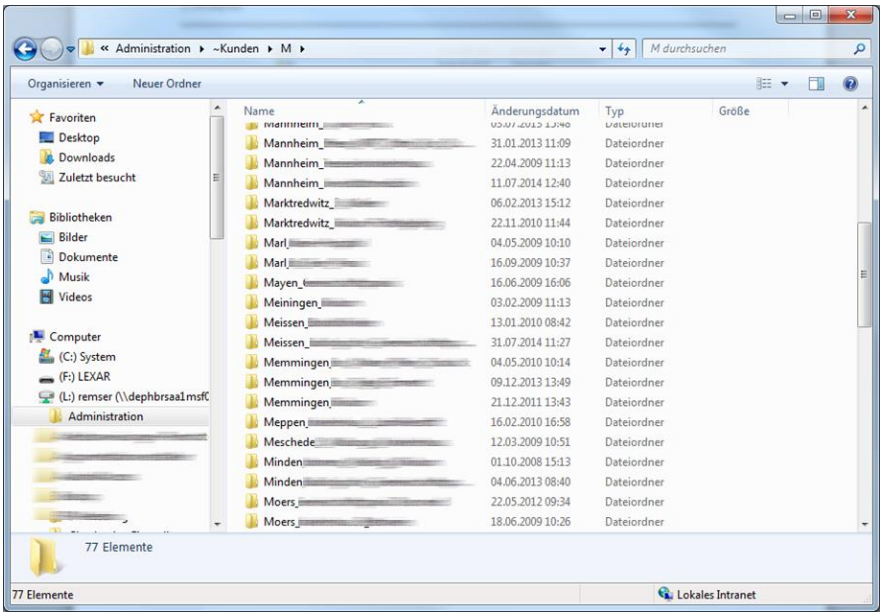


Abbildung 4.10: Kundenordner auf Netzlaufwerk

4.6.1.1 Verzeichnisstruktur

Vom Verzeichnis *Kunden* ausgehend sind in der darunterliegenden Ebene Ordner angelegt, die jeweils den ersten Buchstaben des Alphabets tragen. In diesen Verzeichnissen sind die entsprechenden Kundenstandorte zusammenhängend mit dem Standort in der Form „Ort_Kunde“ abgebildet, z.B. „München_Musterklinik“. In jedem Kundenverzeichnis sind die dazugehörigen Dokumente zu finden (siehe Verzeichnisschema in Abbildung 4.11).

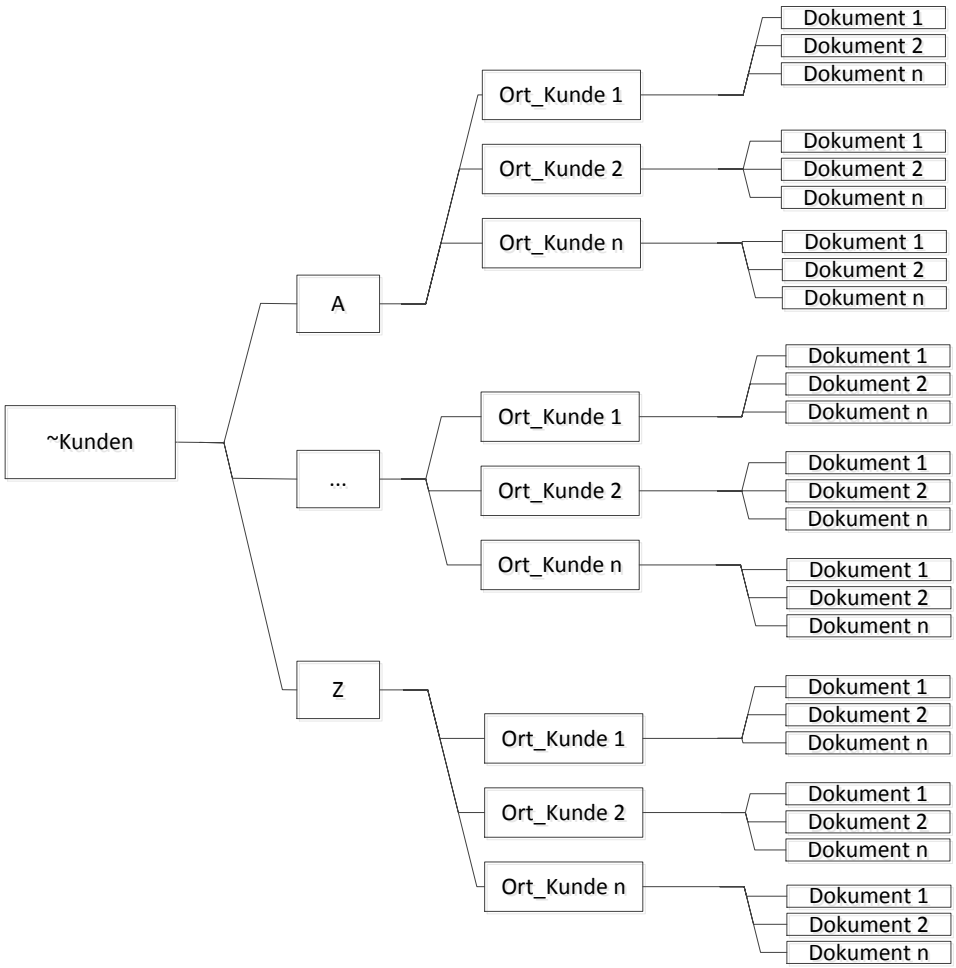


Abbildung 4.11: Schema des Kundenverzeichnisses

4.6.2 PRS Portal

Das PRS Portal ist ein webbasiertes Werkzeug, das für den Internet Explorer konzipiert wurde und von der Philips GmbH entwickelt worden ist. Es ermöglicht Wartungsarbeiten angebundener Medizinsysteme. Die Oberfläche des PRS Portal ist in drei Bereiche unterteilt (siehe Abbildung 4.12). Der Kopfbereich besteht aus einem Menü mit Werkzeugen und einem Suchfeld. Der linke Bereich der Benutzeroberfläche zeigt eine Baumstruktur über die bis auf unterster Ebene die installierten Systeme an den Kundenstandorten ausgewählt werden können. Nach Auswahl eines Systems werden im rechten Ansichtsbereich allgemeine Informationen über den Standort und die Konfiguration anzeigt. Der Anwender kann über die enthaltenen Registerreiter verschiedene Systemanalysen vornehmen und sich die Ergebnisse darstellen lassen.

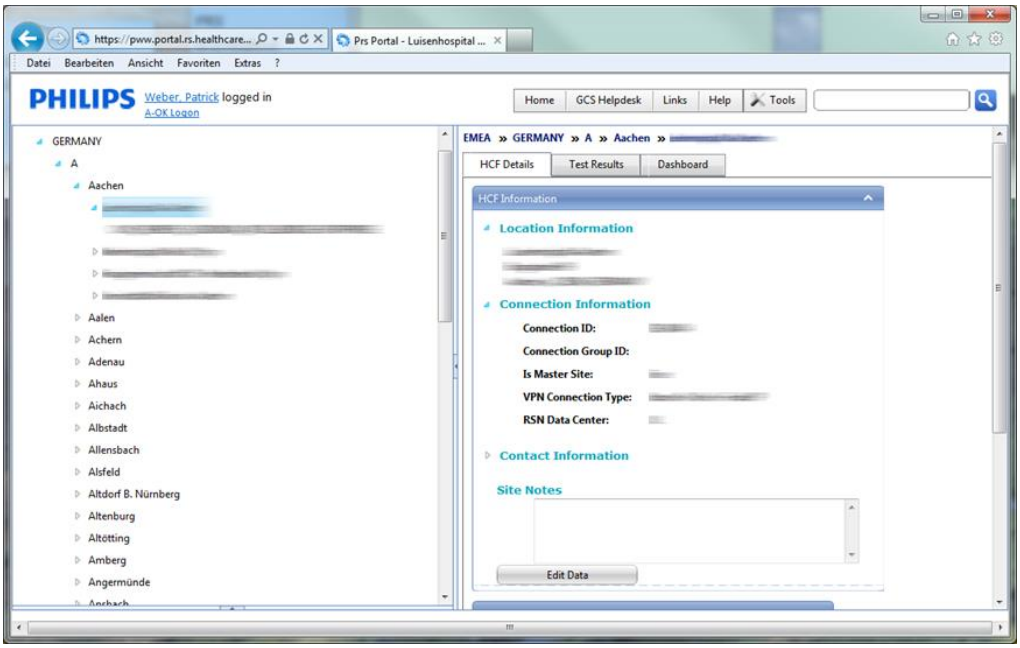


Abbildung 4.12: PRS Portal

4.6.2.1 Baumstruktur

Der linke Arbeitsbereich, die Baumstruktur, gliedert sich auf die folgenden Ebenen: Der erste Knoten enthält global alle Wirtschaftsräume in denen fernwartungsfähige Medizinsysteme vertrieben und installiert sind. In Abhängigkeit der Benutzerberechtigung sind für den Anwender nur bestimmte Regionen sichtbar. In diesem Fall ist es das Gebiet „EMEA“[17]. Innerhalb eines Wirtschaftsraums bilden einzelne Länder den zweiten Knoten, dabei ist die Darstellung erneut von der zugewiesenen Benutzerberechtigung abhängig.

In diesem Fall verfügt der Anwender in Deutschland auch nur über die Zugriffsrechte auf Deutschland. Jeweils das erste Zeichen des Alphabets bildet den dritten Knoten. Die darunterliegende Ebene enthält die Standorte, die der alphabetischen Kategorie zugeordnet sind. Der fünfte Knoten zeigt alle Kunden entsprechend des gewählten Standorts an. Auf der untersten Stufe bilden die einzelnen fernwartungsfähigen Systeme, des jeweiligen Kunden, die Blätter des vorherigen Knotens. In Abbildung 4.13 ist die Baumstruktur schematisch dargestellt. Das Beispiel veranschaulicht, wie ein Medizinsystem, das der „Musterklinik“ des Standorts „Hamburg“ zugeordnet sei, über den Baum wiederzufinden ist. Die als gepunktete Rechtecke gekennzeichneten Elemente dienen als Platzhalter für weitere Regionen, Standorte, Kunden und Systembeschreibungen, die im PRS Portal aufgelistet werden.

[17] Abk. Europe/Middle-East/Africa (Europa, Naher Osten und Afrika)

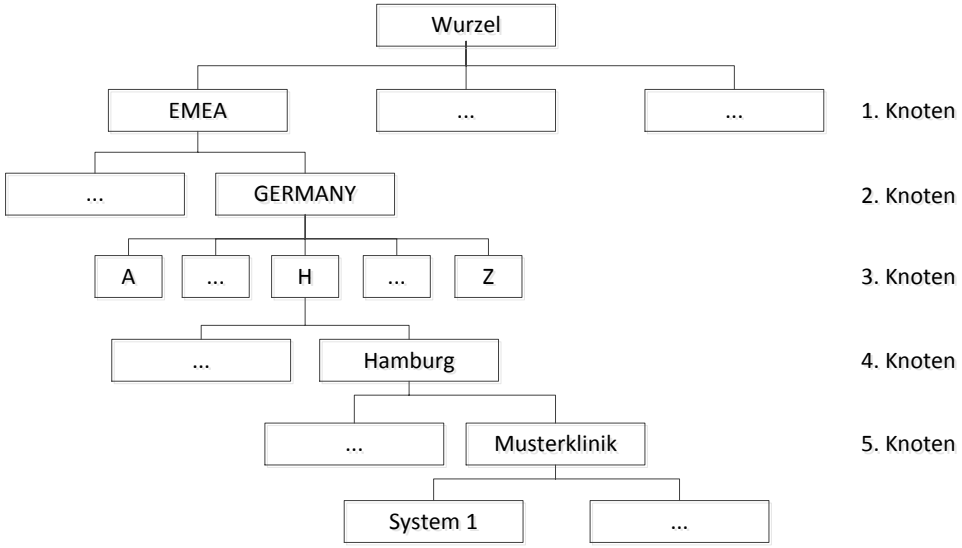


Abbildung 4.13 Schema der Baumstruktur im PRS Portal

Allgemein betrachtet ist das Aufteilungsprinzip bis zum dritten Knoten dem der Verzeichnisstruktur des Kundenlaufwerks ähnlich (vgl. Kapitel 4.6.1.1). Ab dem vierten Knoten werden die Standorte im PRS Portal vom Kunden separiert, sodass der Kundenname eine neue Ebene bildet. Als nächster Schritt muss überprüft werden, wie sich die einzelnen Knoten im PRS Portal erreichen lassen. Als Analysewerkzeug wird der Chrome Browser verwendet, da dieser über einen Entwicklermodus verfügt und mit Hilfe eines Konsolenfensters die Möglichkeit bietet einzelne Elemente oder Regionen einer webbasierten Browser Anwendung zu untersuchen.

4.6.2.2 URL Struktur

Die Untersuchung ergibt, dass die Knoten durch verknüpfte Parameterabfragen der URL Adresse des PRS Portal erreicht werden. Die Wurzel bildet die Standard URL Adresse in der Form: `https://<Intranetserver>/PrsPortal/default.aspx`

Anschließend werden die folgenden CGI[18] Parameter aus Tabelle 4.3 abgefragt. Je höher die Knotenzahl wird, desto mehr Parameter werden miteinander verknüpft.
Ab dem fünften Knoten besteht allerdings nicht mehr die vorherige kaskadierte Abfragelogik, sondern es werden Identifikationsnummern (ID) verwendet. Dabei entspricht die ID nicht einer Kundennummer, sondern einem generiertem Schlüssel. Nach dem gleichen Prinzip sind die Blätter des fünften Knotens, die Kundensysteme, zugeordnet.

[18] Common Gateway Interface

Parameter	Beschreibung	Knoten
SSR	Region	1
Country	Land	2
State	Ursprünglich für US-Bundesstaaten konzipiert, wird aber für Deutschland als Kategorisierung der Ortsangabe verwendet.	3
City	Standort	4
PrsCustomerID	zugeordnete ID für einen Kunden (entspricht nicht der realen Kundennummer)	5
PrsID	zugeordnete ID für Systeme	

Tabelle 4.3: PRS Portal URL Parameter

Die nachfolgenden Fallbeispiele demonstrieren die Parameterabfrage der URL Adresse, zur Vereinfachung wird die bereits erwähnte URL Adresse der Wurzel in der Form `https://<[Wurzel]>` dargestellt.

Fallbeispiel 1: Kunde mit dem Standort „Kiel“

- Zuerst wird der Parameter für die Region „EMEA“ aufgerufen, sodass der erste Knoten des Baums erreicht wird:
  `https://<[Wurzel]>?SSR=EMEA`
- Der nächste Schritt ist der Länderabgleich für „Deutschland“ und der Region „EMEA“:
  `https://<[Wurzel]>?Country=GERMANY&SSR=EMEA`
- Anschließend wird im dritten Knoten die Kategorie „K“ gewählt:
  `https://<[Wurzel]>?State=K&Country=GERMANY&SSR=EMEA`
- In der darunterliegenden Ebene wird abschließend der Standort gesucht:
  `https://<[Wurzel]>?City=Kiel&State=K&Country=GERMANY&SSR=EMEA`

Fallbeispiel 2 a: Ein Kunde wird aufgerufen

Jedem neuen Kunden ist ein Schlüsselwert zugewiesen, hier sei die ID=21054:

`https://<[Wurzel]>?PrsCustomerID=21054`

Fallbeispiel 2 b: Ein Medizinsystem wird aufgerufen

`https://<[Wurzel]>?PrsID=20986`

4.6.3 Lösungsansätze zur Verlinkung

Allgemein sind für die beiden Komponenten dynamische Parameter erforderlich, die sich in Abhängigkeit des gewählten Kunden automatisch anpassen können. Es stellt sich zunächst die Frage bis zu welchem Knoten eine Verknüpfung für die beiden analysierten Komponenten umgesetzt werden kann und ob der Aufwand dem Nutzen gerechtfertigt ist. Generell muss bedacht werden, dass die externen Komponenten nicht sehr häufig genutzt werden, sondern nur Arbeitsabläufe vereinfachen sollen.

Kundenordner

- **Verzeichnisstrukturanpassung**

  Die erste Möglichkeit ist die Anpassung der Verzeichnisstruktur mit dem Ziel die zusammengesetzte Beschreibung von Ortsangabe und Standort getrennt darzustellen, sodass die Ortsangabe das übergeordnete Verzeichnis zum Kundenstandort bildet (siehe Abbildung 4.14). Dieses ermöglicht durch eine Abfrage des Kundenamen im Tupel einen Zugriff auf das Netzlaufwerk.

  Der Nachteil ist der hohe Aufwand, der für die Trennung von Orts- und Kundenbezeichnung erforderlich ist und zusätzlich durch die strikte Firmenpolitik eingeschränkt wird, da jede strukturelle Veränderung, der für die Abteilung eingerichteten Verzeichnisse, eine Genehmigung des Vorgesetzten und des Verzeichniseigentümers erfordert. Der zweite Nachteil ist der Bezug auf den Kundennamen, da vorausgesetzt wird, dass die Kunden in der Datenbank und die Beschriftung der Verzeichnisse auf dem Netzlaufwerk aus der exakt gleichen Beschreibung bestehen müssen, d.h. der Name darf keine Abweichung enthalten, sonst wird der erzeugte Link ungültig. In der Praxis zeigt sich aber, dass diese Vorstellung kaum umsetzbar ist, da es immer vorkommen wird, dass entweder Sonderzeichen oder aufgrund von der Übersichtlichkeit gekürzte Bezeichnungen verwendet werden.

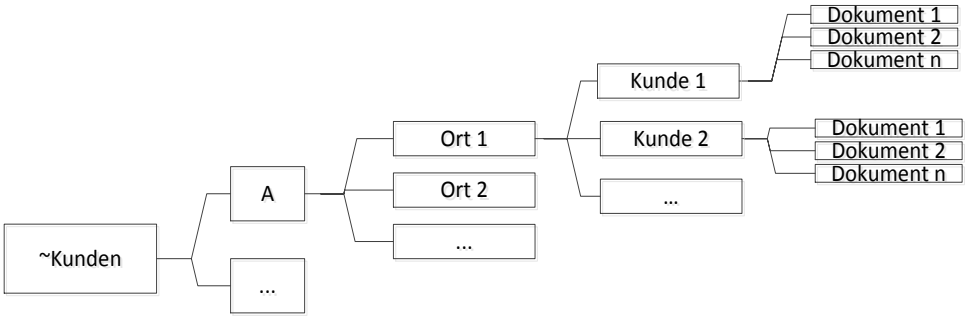


Abbildung 4.14: Schema zur Anpassung der Verzeichnisstruktur

- **Kundennummer zur Identifikation**

  Anstatt des Standorts und der Kundenbezeichnung wird allen Verzeichnissen nur die Kundennummer zugeordnet (siehe Abbildung 4.15). Der dynamische Link lässt sich durch Abfrage der Kundennummer aus dem Tupel erzeugen, das die elementaren Kundendaten enthält. Auf diese Weise wird ein direkter Zugriff auf alle Dokumente im Dateisystem möglich.

  Die Nachteile dieser Variante sind einerseits ähnlich des vorherigen Konzepts, da es sich um tiefgreifende Veränderungen in der Verzeichnisstruktur handelt, die ohne eine Genehmigung nicht durchgeführt werden dürfen. Die Arbeiten zur Umstrukturierung verursachen einen sehr großen Aufwand, außerdem ist aus der Analyse des Ist-Zustands (vgl. Kapitel 4.3.2) bekannt, dass nicht jedes Tupel über vollständige Attributwerte verfügt, was insbesondere die Kundennummer betrifft.

  Ein weiteres Problem ist, dass es für Benutzerkreise, die Zugriff auf die Verzeichnisse besitzen, nicht mehr möglich ist, ohne weitere Kenntnis der Kundennummer den gesuchten Kunden zu finden bzw. einen Überblick zu erhalten, ob schon ein Dokumentenverzeichnis für einen neuen Kunden angelegt worden ist.

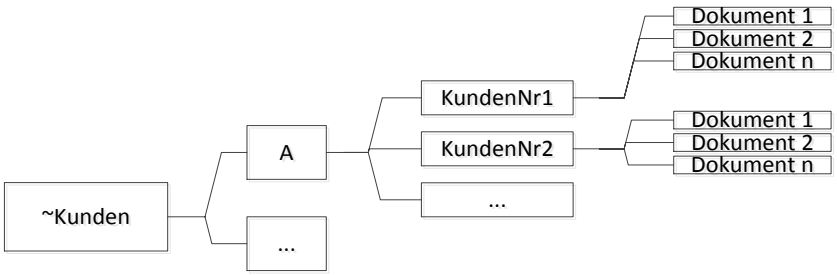


Abbildung 4.15 Schema mit Kundennummer

- **Zeichenkette des Standorts auslesen**

  Eine Verlinkung erfolgt nur in der ersten Stufe des Verzeichnisschemas (vgl. Abbildung 4.11), diese grenzt den Suchbereich des Standorts ein. Anschließend muss der Anwender selbstständig den Kunden auswählen.

  Die Abbildung 4.16 zeigt ein Beispiel, wie aus dem Standort „Hamburg“ das erste Zeichen der Standortangabe ausgelesen und an den vordefinierten Verzeichnispfad übergeben wird.

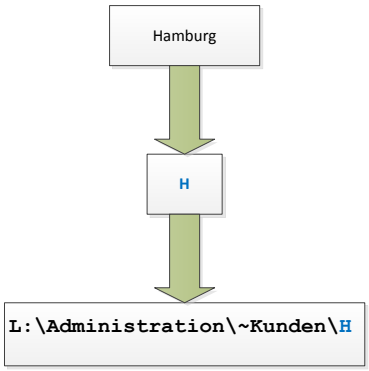


Abbildung 4.16: Teilstring auslesen

Der Vorteil dieser Methode ist, dass keine Änderungen der Verzeichnisstrukturen notwendig sind und firmenpolitische Entscheidungen entfallen.

Der einzige Nachteil ist, dass der Anwender noch einen Arbeitsschritt erledigen muss, um an die gesuchten Dokumente zu kommen, d.h. beim Bedienkomfort müssen Abstriche in Kauf genommen werden.

Die Tabelle 4.4 stellt die drei genannten Konzepte für das Kundenverzeichnis in Bezug des Aufwands und der bereits genannten geringen Nutzung gegenüber. Es zeigt sich, dass das dritte Konzept im Verhältnis zum geringen Nutzungsverhalten mit dem geringsten Aufwand verbunden ist.

Konzepte	Aufwand	Nutzungsverhalten
Verzeichnisstrukturanpassung	sehr hoch	wenig
Kundennummer zur Identifikation	sehr hoch	wenig
Zeichenkette des Standorts auslesen	klein	wenig

Tabelle 4.4: Konzeptvergleich

Das dritte Konzept ist in Absprache mit den Anwendern befürwortet worden und wird in der Realisierungsphase weiter erläutert werden.

PRS Portal

- **PrsCustomerID**

  Hinsichtlich der Analyse des PRS Portal ist bekannt, dass der Kundenname im fünften Knoten über einen Schlüsselwert (PrsCustomerID) identifiziert wird (vgl. Kapitel 4.6.2.2). Eine Attributerweiterung der Hauptrelation um die Bezeichnung „PrsCustomerID“, inklusive aller Attributwerte, ermöglicht eine direkte Referenz

zum PRS Portal. In Abbildung 4.17 ist das Prinzip dargestellt, wie durch das Auslesen des Attributs der Wert an die URL Abfrage übergeben wird.

Allerdings stößt dieses Konzept an seine Grenzen, denn neben dem hohen Aufwand der Zuordnung der Schlüsselwerte in die Datenbank, wird vorausgesetzt, dass alle Werte aus dem PRS Portal bekannt seien, was aus zwei Gründen nicht möglich ist:

1. Die PrsCustomerID wird automatisch generiert und richtet sich nicht nach einer Kundennummer. Weitere Unternehmensanwendungen besitzen Schnittstellen auf die Datenbank, die das PRS Portal ausliest. Über die Schnittstellen werden teils automatisiert neue Kunden angelegt, wenn die installierten Systeme M2M[19] unterstützen.
2. Das PRS Portal bezieht seine Daten aus einer globalen Firmendatenbank, deren Zugangsberechtigung nur bestimmten Personen vorbehalten ist, sodass keine Möglichkeit besteht einen Überblick aller erforderlichen Schlüsselwerte zu erhalten.

Das Konzept erweist sich als sehr wartungsintensiv, da die RSN Datenbank immer wieder validiert werden muss, nur damit die Werte korrekt an die URL übergeben werden können.

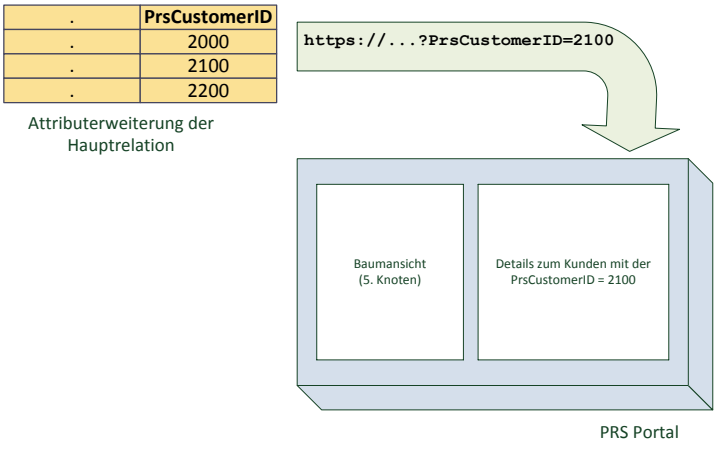


Abbildung 4.17: Link über PrsCustomerID erzeugen

- **Auslesen der Zeichenkette des Standorts**

  Die Standortbezeichnung wird auf zwei Arten gleichzeitig ausgelesen:

  Einmal als gesamte Zeichenkette und einmal nur das erste Zeichen. Die ausgelesenen Zeichen werden als Parameterangaben der URL Adresse übergeben. Dadurch ist es möglich den vierten Knoten der Baumstruktur (vgl. Abbildung 4.13)

[19] Machine-to-Machine beschreibt einen automatischen Informationsaustausch, z.B. einer Industrieanlage und einem Server. Dabei meldet sich die Maschine automatisch beim Server an oder ab.

anzuzeigen. Ist eine Ortsangabe in der RSN Datenbank fehlerhaft abgebildet, so wird der Anwender in die darüber liegende Hierarchiestufe, der dritte Knoten, geleitet. In Abbildung 4.18 ist ein Beispiel für den Standort „Hamburg“ dargestellt, die Ortsangabe wird als gesamte Zeichenkette abgefragt. Parallel wird nur das erste Zeichen für die Kategorisierung ermittelt. Beide Werte werden als dynamische Parameter übergeben, sodass eine URL Adresse zum PRS Portal erzeugt wird.

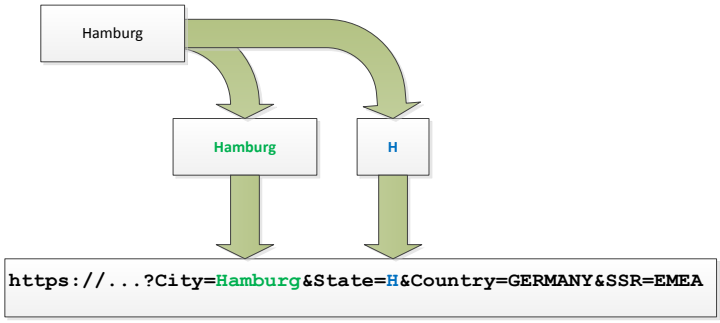


Abbildung 4.18: Übergabe des Zeichenkettenwerts an URL Parameter

Der Vorteil an dieser Lösungsvariante ist, dass keine Erweiterungen in der Datenbank erforderlich sind. Als Nachteil kann höchstens der Bedienkomfort aufgefasst werden, da der Anwender noch einen weiteren Mausklick tätigen muss.

4.7 Suchfunktion

Das Gespräch mit den Anwendern, bezüglich der Attribute, die durchsucht werden sollen oder ob eine allumfassende Suche gewünscht wird, hat ergeben, dass sich die Suche auf zwei Relationen beschränken wird. Neben der Hauptrelation soll das Datenbankobjekt, das den Kundenstatus enthält berücksichtigt werden, damit eine Möglichkeit gegeben wird die Historieneinträge zu durchsuchen.

APEX bietet die Möglichkeit über ein Plug-In des Regionstyps „interaktiver Bericht“ eine Toolbar mit Suchfeld zu verwenden, allerdings ist die Suche nur auf die Hauptrelation beschränkt und erfüllt nicht die gestellten Anforderungen.

Eine Lösung bietet eine HTML Region, die sich auf die Eigenschaft „Suche“ konfigurieren lässt, dort können Felder angelegt werden, die die Suchkriterien beschränken sollen.
Jedes Feld wird anschließend mit SQL Abfragen angepasst.

5 GUI Designkonzept

Nachdem im vorherigen Kapitel Konzepte erläutert worden sind, die sich auf die Datenbank und den geforderten Applikationsfunktionen bezogen haben, sollen in diesem Kapitel die erforderlichen Designkonzepte der angedachten Benutzeroberfläche (GUI[20]) analysiert werden, bevor die Realisierung in Kapitel 6 erfolgt.

5.1 Gebrauchstauglichkeit (Usability)

Usability ist eine vielseitige Bezeichnung, die mehrere Begriffe, wie Nutzerfreundlichkeit, Qualität und Effizienz vereint. Eine allgemeine Definition ist in der ISO-Norm 9241 beschrieben: „Usability bezeichnet das Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzerkontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und mit Zufriedenheit zu erreichen“ [11 S. 2]
Das bedeutet ein Design kann nur **Effektivität** vorweisen, wenn sich der Anwender leicht orientieren kann und die von ihm geforderten Informationen vorfindet. Auch muss der Nutzer in der Lage sein, die unterschiedlich dargestellten Elemente, die verschiedene Funktionen aufweisen, direkt zu erkennen. Die **Effizienz** ist gegeben, wenn der Anwender mit wenig Aufwand sein Ziel erreicht. So können z.B. visuelle Darstellungsmöglichkeiten einer Webanwendung bestimmte Bereiche abgrenzen, die dem Anwender die Orientierung erleichtern, damit dieser schneller zu seinem Ziel kommt. Effektivität und Effizienz müssen erfüllt sein, damit sich **Zufriedenheit** einstellt. Hierzu es erforderlich die Erwartungen des Benutzerkreises zu kennen, denn „Zufriedenheit entsteht dann, wenn die Erwartungen des Nutzers mindestens erfüllt oder besser noch übertroffen werden.“ [11 S. 3]

Die genannten Kriterien lassen sich als Matrix darstellen (siehe Tabelle 5.1) und zeigen, dass eine Webanwendung dreidimensional zu betrachten ist und in Bezug der dargestellten Inhalte, der visuellen Gestaltung und dem strukturellen Aufbau alle Kriterien zu prüfen sind.

[20] Graphical User Interface

	Inhalt	Design	Struktur
Effektivität	x	x	x
Effizienz	x	x	x
Zufriedenheit	x	x	x

Tabelle 5.1: Usability – Kriterien [11 S. 5]

5.2 Nutzererfahrung (User Experience Research)

Der Designprozess einer Anwendung erfordert das frühzeitige Einbinden der Anwender, um eine Designlösung zu erhalten, die auf die erforderlichen Bedürfnisse und Verhaltensstrukturen der Zielgruppe zugeschnitten ist. Der Vorgang wird als Nutzererfahrung bzw. UER (User Experience Research) bezeichnet und gliedert sich in folgende drei Phasen:

- **Untersuchungsphase:**
  Die Anwenderbedürfnisse werden gesammelt, Designkriterien festgelegt und mit Hilfe von Bezugssystemen bzw. existierenden Anwendungen das Anwenderverhalten diskutiert.
- **Prototypuntersuchung:**
  Erste skizzierte Prototypen werden mit den Anwendern diskutiert, um mit Hilfe des Feedbacks das Design zu überarbeiten. Ein rudimentäres Layout einer Anwendungsoberfläche, inklusive Schaltflächen, erleichtert es Einblicke in die gewünschten Verhaltensmuster der Nutzergruppe zu erhalten.
- **Feinabstimmung:**
  Die Anwender testen unter realen Bedingungen einen interaktiven Prototyp.
  Sollten einige Aspekte noch nicht den Präferenzen der Nutzergruppe entsprechen, so ist der Prototyp zu verfeinern.

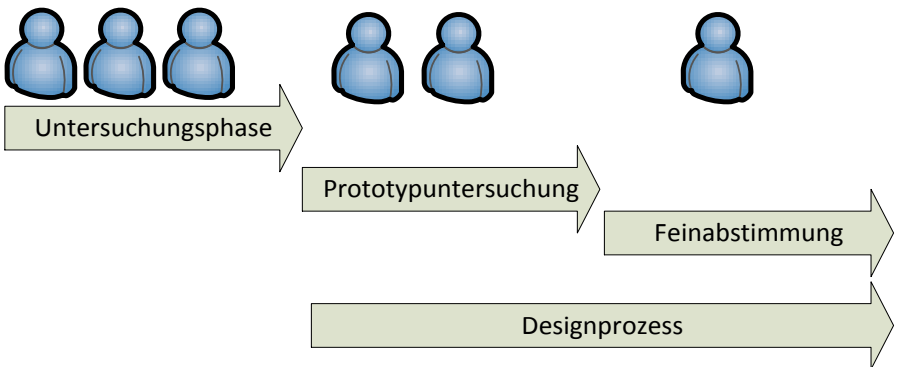


Abbildung 5.1: Drei Untersuchungsphasen, in Anlehnung an [11 S. 44]

5.3 Flaches Design (Flat Design)

Die Fokussierung auf den Inhalt einer Webanwendung erfordert ein minimalistisches Gestaltungsprinzip, das alle wichtigen und notwendigen Funktionen beinhaltet.
Ein flaches Design, das nach dem Grundgedanken: „Weniger ist mehr“ aufgebaut ist, kennzeichnet sich durch interaktive und dynamische Elemente. Es vereint schnelle Orientierung und intuitive Bedienung. Eines der bekanntesten Beispiele der minimalistischen Gestaltung ist die Startseite des US-amerikanischen Internetsuchdienstes Google; auf überflüssige Farben ist verzichtet worden, ein großes Suchfeld im Zentrum konzentriert sich auf die Hauptfunktion der Webanwendung.

5.4 Visuelle Wahrnehmungspsychologie

Das menschliche Auge ist ein visuelles System, das verschiedene Lichtreize in Nervenimpulse umwandelt und an das Gehirn zu Verarbeitung weiterleitet. Der Sehnerv eines Auges verläuft gekreuzt in die gegenüberliegende Gehirnhälfte (siehe Abbildung 5.2), sodass jeder Hirnbereich (Hemisphäre) unterschiedliche Aufgabenbereiche wahrnehmen muss. Die linke Hälfte ist spezialisiert auf analytisches und logisches Denken. Die rechte Hälfte vereint emotionales, bildhaftes und ganzheitliches Denken.

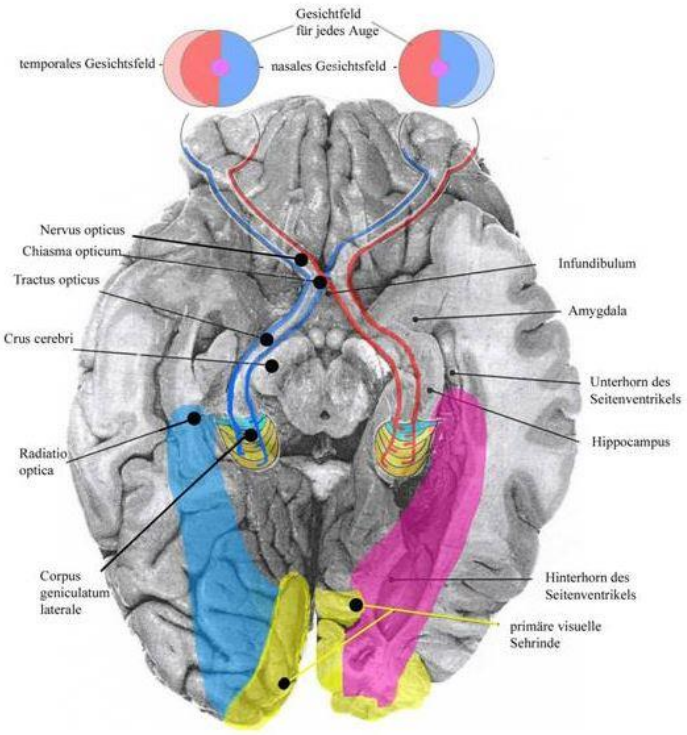


Abbildung 5.2: Visuelles System beim Menschen [12]

Das bedeutet hinsichtlich der Benutzeroberflächendarstellung einer Webanwendung, dass Informationen ganzheitlich besser verarbeitet werden können, wenn Elemente entsprechend des Blickfelds (siehe Abbildung 5.3) angeordnet werden, sodass die Informationen über das Auge schneller die entsprechende Gehirnhälfte ansprechen. Demnach sind z.B. Schaltflächen, Navigations- und Steuerelemente linksseitig auszurichten und Texte, außer Tabellen, besser rechtsseitig zu positionieren.

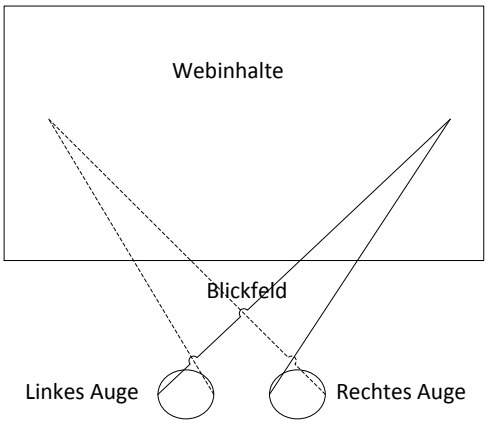


Abbildung 5.3: Blickfeldschema der Augen auf Webinhalte, angelehnt an [11 S. 34]

5.5 Prototypentwicklung

5.5.1 Low-fidelity-Prototyping

Low-fidelity-Prototyping ist Bestandteil der Untersuchungsphase der Nutzererfahrung (UER). Es beschreibt die Konzeptionsphase und skizziert den Aufbau und das Verhaltensmuster einer Anwendung. Dem Aufbau grafischer Darstellungsformen wird an dieser Stelle weniger Aufmerksamkeit geschenkt. Die Entwürfe werden in Papierform, auch Papier-Prototyping genannt, erstellt und mit den Anwendern diskutiert. Änderungen lassen sich auf diese Weise schnell durchführen.

Für den Prototyp der webbasierten Datenbankanwendung bedeutet es hinsichtlich der Anforderungen aus Kapitel 4.1, dass die Anwendung aus vier Seiten bestehen wird (siehe auch Entwürfe im Anhang):

- **Startansicht:**
  Die Startansicht bietet eine allgemeine Übersicht der Kunden und enthält Suchfelder. Das Suchergebnis wird in derselben Region angezeigt, die auch die Kundenübersicht enthält. Jeder Kunde lässt sich über eine kleine Schaltfläche bearbeiten, dadurch wechselt die Webanwendung in den Editiermodus. Wird der Kundenname mit dem Mauszeiger angeklickt, so wechselt die Anwendung in den Lesemodus. Das Anlegen neuer Kunden wird mittels einer Schaltfläche im oberen Bereich ermöglicht. Durch auslösen der Schaltfläche wechselt die Webanwendung in den Editiermodus.
- **Editiermodus:**
  Der Editiermodus setzt sich aus einem Master-Detail Formular zusammen. Der Detailbereich enthält drei Regionen, die neben den Kundensystemen, die Verknüpfungen zu den externen Komponenten und der Kundenhistorie enthalten.
  Wird über die Schaltfläche „Kunde anlegen“ aus der Startansicht in den Editiermodus gewechselt, so bleibt die Detailansicht ausgeblendet, da noch keine untergeordneten Tupel existieren. Oberhalb der Hauptansicht (Master) sind Schaltflächen angeordnet, die es einerseits ermöglichen zwischen den verschiedenen Kunden zu navigieren und andererseits die Möglichkeit bieten Änderungen zu speichern, den gesamten Kunden zu löschen oder über einen Abbruch den Modus wieder zu verlassen. Eine weitere Schaltfläche ermöglicht Einträge in die Historie zu schreiben, hierdurch wechselt die Webanwendung zum Eingabeformular.
- **Lesemodus:**
  Der Lesemodus besitzt die gleiche Struktur, wie der Editiermodus mit Ausnahme, der Region, die die Verknüpfungen zu den externen Komponenten enthält, da das Arbeiten mit externen Komponenten nur im Editiermodus sinnvoll erscheint, weil die Abläufe in der Historie festgehalten werden müssen.
- **Eingabeformular:**
  Mit dem Eingabeformular lassen sich Einträge in die Historie vornehmen.
  Es besteht aus den beiden Schaltflächen: „Abbrechen“ und „Neuer Eintrag“, sowie einer Region mit einem Textfeld. Ein Abbruch im Eingabeformular leitet den Anwender zurück in den Editiermodus zum aktuellen Kunden. Wird ein neues Tupel über die Schaltfläche „Neuer Eintrag“ angelegt, so wird der Anwender wieder zurück in den Editiermodus zum aktuellen Kunden geleitet.

Die Abbildung 5.4 zeigt ein Schema, das die jeweilige Seite in Abhängigkeit der ausgeführten Aktionen darstellt, in die die Webanwendung wechselt. Ausgangspunkt ist die

Startansicht. Ein Sonderfall besteht bei der Auswahl von einem Link, da die Webanwendung auf keine neue Seite wechseln wird, sondern über den Browser oder das Dateisystem ein Fenster öffnen soll.

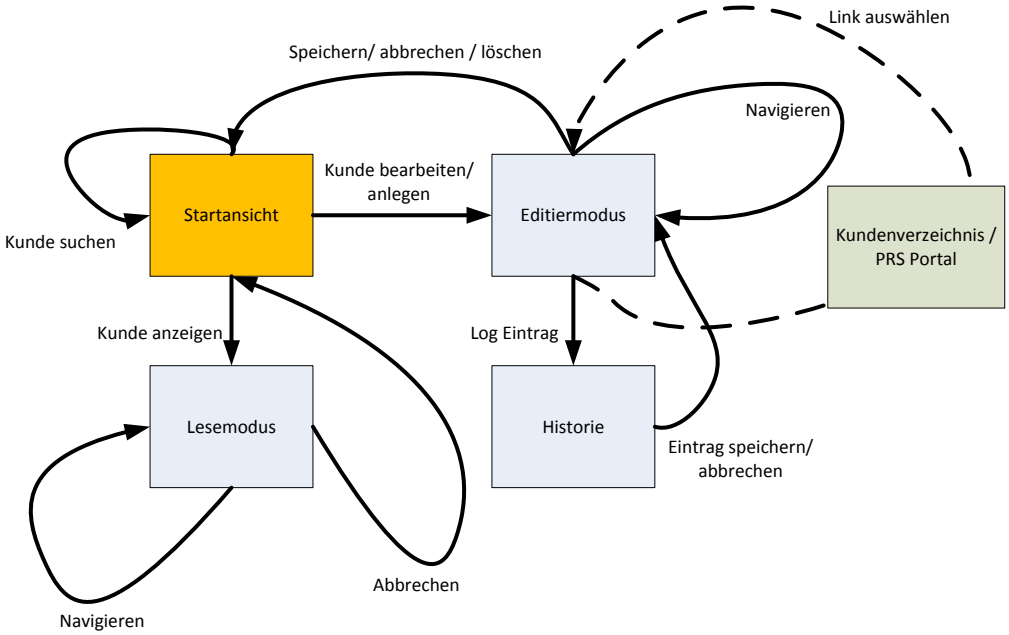


Abbildung 5.4: Schema der Seitenbeziehungen

5.5.2 Hi-fidelity-Prototyping

Im Gegensatz zum vorher erwähnten Verfahren, handelt es sich bei Hi-Fidelity-Prototyping um einen funktionsfähigen Web-Prototypen, der im Entwicklungsprozess weniger strukturelle Änderungen bis zum fertigen Produkt zulässt. Vorausgesetzt wird ein erfolgreiches Low-fidelity Prototyping, d.h. die ersten groben Strukturen des Seitenaufbaus und des Anwendungsverhaltens sind entworfen und von den Anwendern befürwortet worden. Der funktionsfähige Web-Prototyp entspricht weitestgehend dem fertigen Produkt und enthält detaillierte Funktionen. Es muss berücksichtigt werden, dass ein Prototyp fehlerbehaftet sein kann, was mit Hilfe von anschließenden Testdurchführungen überprüft werden muss.

Für den Prototyp dieser Arbeit sollen an dieser Stelle konkrete Lösungen beschrieben werden, wie sich die Entwürfe aus dem vorherigen Low-fidelity Prototyping in einen funktionsfähigen Web-Prototypen umsetzen lassen.

Startansicht

Die Startseite bzw. Startansicht lässt sich in zwei Regionen aufteilen: Eine Region vom Typ „HTML“ für die Suchfelder, sowie eine Region des Typs „interaktiver Bericht“, der die Informationen der Relation „Standort“ anzeigen wird. Aufgrund des Regionstyps „interaktiver Bericht“ wird gleichzeitig vom Application Builder das Master-Detail Formular angelegt werden, sodass die Seite für den Editiermodus nicht manuell erstellt wird. Die Schaltfläche auf der Startansicht, mit der neue Kunden angelegt werden, wird dadurch ebenfalls automatisch erzeugt. Damit von der Startansicht der Lesemodus aufgerufen werden kann, muss in der Spaltenkonfiguration des interaktiven Berichts mit Hilfe des Primärschlüsselwerts und dem Seitenelement der Zielseite eine zusätzliche Verknüpfung erstellt werden.

Editiermodus

Allgemein kann der Application Builder für Master-Detail Formulare nur eine Region mit der Kardinalität vom Typ 1:n erstellen, sodass die zweite Region derselben Kardinalität nachträglich anzulegen ist, damit neben den Systemen auch die Historie angezeigt wird.
Das Master-Detail Formular wird im Detailbereich aus drei Regionen bestehen:

- Systemübersicht:
  Die Region wird vom Typ „Report“ dargestellt. Die Konfiguration der tabellarischen Ansicht wird so eingestellt, dass ein direktes Schreiben in die Tabellenfelder ermöglicht wird. Das dynamische Anfügen oder Löschen neuer Zeilen lässt sich mit JavaScript realisieren.
- Verknüpfung auf externe Komponenten:
  Eine einfache HTML Region wird je einen Link zum Kundenverzeichnis und ins PRS Portal enthalten. Da die Webanwendung im Internet Explorer verwendet werden soll, wird durch Auswahl der Verknüpfung zum Kundenverzeichnis automatisch auf dem Anwender PC das Dateisystem geöffnet. Für das PRS Portal eignet sich ein Pop-Up Fenster, da der Internet Browser nicht die Webanwendung verlassen soll. Mit Hilfe von JavaScript kann das Pop-Up Fenster umgesetzt werden.
- Historie:
  Die Region, die die Historie enthalten wird, muss vom Typ „interaktiver richt“ sein, da der Regionstyp in der Voreinstellung über eine Suchleiste verfügt, die genutzt werden kann, um die Historieneinträge schneller zu durchsuchen.

Lesemodus

Die Seite entspricht der gleichen Struktur und den gleichen Regionstypen, wie die Seite für den Editiermodus. Allerdings sollen bis auf die Navigation die restlichen Schaltflächen nicht vorhanden sein, da es sich um eine Leseansicht handelt.

Eingabeformular

Für die Seite auf der das Eingabeformular für die Historie dargestellt wird, reicht eine Standard Region vom Typ „HTML“ aus.

6 Realisierung

Die in Kapitel 4 und 5 erarbeiteten Konzepte werden in diesem Abschnitt realisiert.
Die ersten beiden Unterkapitel beschreiben die Umsetzung der Datenbankmigration, die Anpassung des Datenbankmodells und die Erstellung notwendiger Funktionen für die Datenbank.

Resultierend daraus wird die Entwicklung der webbasierten Datenbankanwendung beschrieben. Sofern die Möglichkeit gegeben ist, werden die realisierten Funktionen und Anwendungskomponenten direkt getestet, um während des Entwicklungsprozesses Fehler ausschließen zu können.

6.1 ETL Prozesse

Die erforderlichen ETL Prozesse für das Staging werden innerhalb der MS Access Kundendatenbank angelegt. Da innerhalb von MS Access keine SQL Anweisungen als Skript, sondern nur einzeln ausgeführt werden können, müssen für die entsprechenden Relationen jeweils eine Abfrage erstellt werden.

6.1.1 Extraktion

Die Extraktion besteht aus genau drei Abfragen:

- **qryAPEX_RSN_STANDORT** wählt alle Attribute der Relation „Standort“ aus, erzeugt eine Relation mit der Bezeichnung „RSN_STANDORT“ und schreibt die Attributwerte in das Datenbankobjekt hinein (siehe Abbildung 6.1).

  Mit Hilfe von Funktionen wird der zusammengesetzte Kundenname, bestehend aus Ortsangabe und Kundenbezeichnung, der nur durch ein Komma getrennt wird, in zwei einzelne Attribute aufgeteilt. Alle Attributwerte des Attributs „Kundenname“ werden beginnend von der linken Seite mit den Funktionen `Left` und `Instr` überprüft. Wird das Komma in der Zeichenkette erkannt, so wird der linksseitige Teil zum neuen Attribut mit der Bezeichnung „Ort“ angelegt. Die rechte Seite wird als Attribut mit der Bezeichnung „Name“ erstellt. Die Funktionen `Right`, `Instr` und `Len` sind hierzu erforderlich, da rechtsseitig alle Zeichen bis zum

Komma ausgewählt werden. Die Funktion `Len` wertet die gesamte Länge der Zeichenkette aus, welche mit der Positionsangabe des Kommas durch `Instr` subtrahiert wird. Die Differenz ergibt den genauen Abschnitt auf der Zeichenkette, der von der rechten Seite ausgehend getrennt wird.

Die Funktion `Format` führt eine Zwangskonvertierung in das deutsche Datumsformat durch, damit beim späteren Laden der Daten ins Oracle Schema aufgrund unterschiedlicher Regionseinstellungen Komplikationen bezüglich der Datumsformate ausgeschlossen werden.

- **qryApex_RSN_SYSTEM** wählt alle Attribute samt Attributwerte aus der Relation „Systeme“ aus und erzeugt das neue Datenbankobjekt „RSN_SYSTEM“
- **qryApex_RSN_KUNDENSTATUS** kopiert die gesamte Struktur und alle Attributwerte aus der Relation „Kundenstatus“ und legt das Datenbankobjekt „RSN_KUNDENSTATUS“ an.

```
1  SELECT STANDORT.ID,
2     Left([KUNDENNAME],InStr([KUNDENNAME],",")-1) AS ORT,
3     Right([KUNDENNAME],Len([KUNDENNAME])-InStr([KUNDENNAME],","))
4     AS NAME,...,
5     Format([BEARBEITUNGSDATUM],"dd\.mm\.yyyy")
6     AS ["BEARBEITUNGSDATUM"],
7     INTO RSN_STANDORT
8     FROM STANDORT
9     WHERE (((STANDORT.KUNDENNAME) Like "*,*"))
10 ORDER BY STANDORT.ID;
```

Abbildung 6.1: Codeausschnitt qryAPEX_RSN_STANDORT

6.1.2 Transformation

Ein Teil der Transformation ist bereits durch den vorherigen Schritt erfolgt, als sichergestellt worden ist, dass alle Attribute atomar abgebildet werden. Eine Anpassung auf das Datenbankmodell erfolgt in Kapitel 6.1.4, da Beziehungen durch die Migration von MS Access nach Oracle verloren gehen.

Die Datenbereinigung wird auf die erstellten Relationen „RSN_SYSTEM“ und „RSN_KUNDENSTATUS“ ausgeführt, damit verwaiste Untereinträge erkannt und entfernt werden. Der Bereinigungsvorgang erfolgt erst zum Zeitpunkt der Inbetriebnahme, nachdem die Webapplikation erstellt und getestet worden ist, da die neuesten Kundendaten

mit Hilfe der erzeugten Abfragen des Extraktionsprozesses (vgl. Kapitel 6.1.1) aktualisiert werden müssen.
Die Löschabfragen bestehen aus den folgenden beiden Datenbankobjekten:

- **qryDel_RSN_SYSTEM** überprüft ob eine gültige Referenz zwischen dem Fremdschlüssel von „RSN_SYSTEM“ und der Relation „RSN_SATNDORT“ existiert (vgl. Abbildung 6.2). Falls die Referenz ungültig ist, wird das verwaiste Tupel aus der Relation für die Systembeschreibungen entfernt.

```
1  DELETE FROM RSN_SYSTEM
2  WHERE STANDORT_ID NOT IN
3  (SELECT ID FROM RSN_STANDORT );
```

Abbildung 6.2: SQL Code qryDel_RSN_SYSTEM

- **qryDel_RSN_KUNDENSTATUS** überprüft ob eine gültige Referenz zwischen dem Fremdschlüssel von „RSN_KUNDENSTATUS“ und der Relation „RSN_SATNDORT“ existiert (vgl. Abbildung 6.3). Falls die Referenz ungültig ist, wird das verwaiste Tupel aus der Relation für den Kundenstatus entfernt.

```
1  DELETE FROM RSN_KUNDENSTATUS
2  WHERE KUNDEN_ID NOT IN
3  (SELECT ID FROM RSN_STANDORT );
```

Abbildung 6.3: SQL Code qryDel_RSN_KUNDENSTATUS

6.1.3 Erzeugung der Muster Relationen

In der Staging Area wird von jeder Relation eine Kopie erzeugt, die nur die Struktur, aber keine Daten enthalten darf. Die Kopien bilden die Muster Relationen, die nach Oracle migriert werden sollen und tragen temporär die Bezeichnungen:

- „RSN_STANDORT1“
- „RSN_SYSTEM1“
- „RSN_KUNDENSTATUS1“

Mit Hilfe des SQL Developer sind die Verbindungen auf das Oracle Schema und die MS Access Kundendatenbank einzurichten. Im weiteren Verlauf dieser Arbeit wird das Oracle Schema nur noch als „RSN Schema“ bezeichnet werden. Anschließend werden die erstellten Muster aus MS Access direkt ins RSN Schema kopiert und die einzelnen Datenbankobjekte wieder in ihre ursprünglichen Bezeichnungen umbenannt. Diese Muster sol-

len im weiteren Verlauf nur noch als Relationen bezeichnet werden, da sie die Grundlage der webbasierten Anwendung sind.
Da das RSN Schema dem APEX Workspace zugeordnet ist, lassen sich die Relationen auch im Objekt Browser von APEX einsehen (siehe Abbildung 6.4).

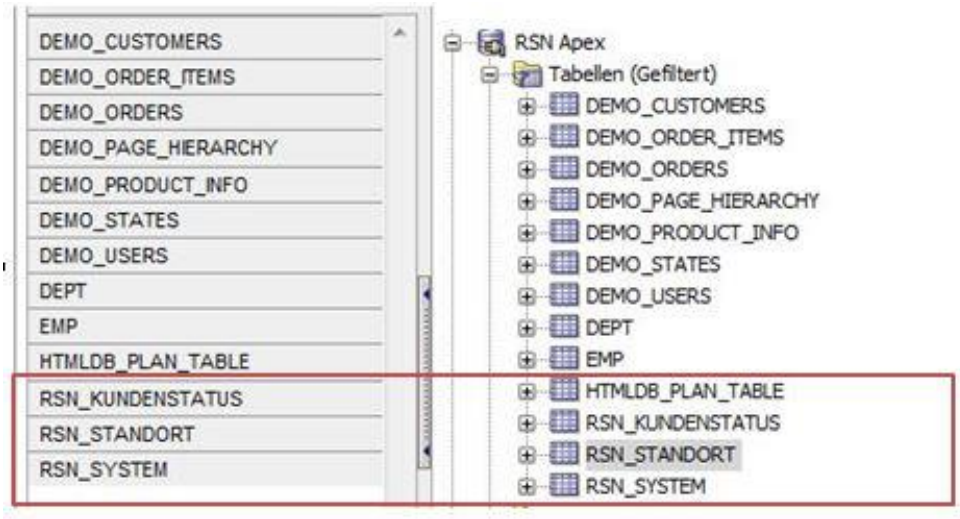


Abbildung 6.4 : Ansicht im APEX Objekt Browser (l.) und im SQL Developer (r.)

6.1.4 Gewährleistung referentieller Integrität

Die im RSN Schema befindlichen Relationen müssen an dieser Stelle miteinander in Beziehung gesetzt werden. Dabei sind die aus der Analyse ermittelten Kardinalitäten (vgl. Kapitel 4.3.3) zu berücksichtigen. Mit Hilfe der Anweisungen

`ADD CONSTRAINT` und

`REFERENCES`

werden die Beziehungen zwischen den Fremdschlüsseln der Relationen

- „RSN_SYSTEM“
- „RSN_KUNDENSTATUS“

zum Primärschlüssel von „RSN_STANDORT“ erstellt.

Damit beim Löschen eines übergeordneten Tupels auch das zu referenzierende Tupel entfernt wird, muss sichergestellt werden, dass eine Löschweitergabe existiert, d.h. kaskadiertes Löschen wird benötigt. Die Löschweitergabe wird durch die Anweisung

`ON DELETE CASCADE ENABLE` festgelegt.

Die Abbildung 6.5 zeigt die Anweisungen für die Relation „RSN_SYSTEM“.

```
1  ALTER TABLE RSN_SYSTEM
2  ADD CONSTRAINT RSN_SYSTEM_FK FOREIGN KEY (STANDORT_ID)
3  REFERENCES RSN_STANDORT (ID)
4  ON DELETE CASCADE ENABLE;
```

Abbildung 6.5: SQL Anweisung der Relation RSN_SYSTEM

Eine anschließende Überprüfung im APEX Objekt Browser zeigt ein schematisches Modell, das allerdings keine Kardinalitäten darstellen kann. Ein detailliertes ER Modell lässt sich mit Hilfe des SQL Data Modeler erstellen (siehe Anhang A2).

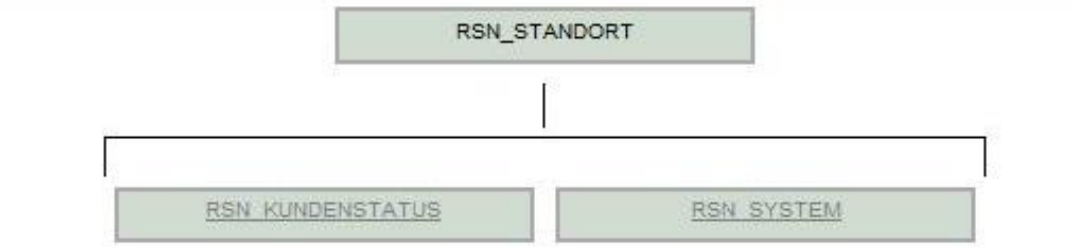


Abbildung 6.6: Beziehungsmodell

Der noch ausstehende Prozess des Ladens erfolgt in Kapitel 8.1.

6.2 Grundfunktionen

6.2.1 Autoinkrement

Datenanwendungsprogramme ermöglichen das Anlegen von Tupel, d.h. ein Benutzer sieht auf einer grafischen Oberfläche Formularfelder, tätigt die Eingaben und legt diese durch den Speichervorgang an. Allgemein benötigt aber jedes Tupel einen eindeutigen Primärschlüssel ohne den ein Anlegen nicht möglich ist. Allerdings ist es in der Regel unüblich, dass ein Anwender dem Primärschlüssel selbst einen Wert zuweisen kann, da dieser keine Kenntnis besitzt, welche Schlüsselwerte noch nicht verwendet worden sind und zur freien Auswahl stehen.

Im Gegensatz zu MS Access, das ein automatisches Hochzählen des Primärschlüssels ermöglicht, gibt es in einer Oracle Datenbank keinen voreingestellten Automatismus.

Das Beispiel in Abbildung 6.7 zeigt, wie mit Hilfe des SQL Developer das Anlegen eines Tupels ohne Primärschlüsselzuweisung eine Fehlermeldung hervorruft, da der Schlüsselwert einen NULL Wert enthalten darf.

Ein automatisiertes Generieren eindeutiger Schlüsselwerte lässt sich mit Hilfe von Trigger und Sequenzen realisieren.

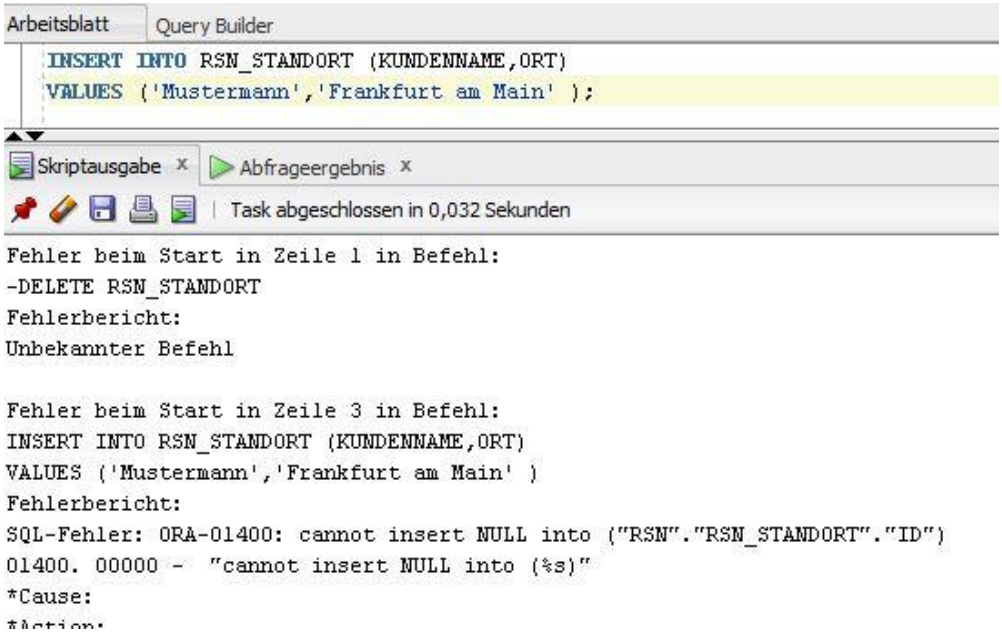


Abbildung 6.7: INSERT Beispiel ohne ID Wert

Für das vorliegende Datenbankmodell im RSN Schema werden zwei Sequenzen und drei Trigger erstellt. Die erste Sequenz wird gemeinsam vom Trigger, der die Kunden anlegt und vom Trigger, der die Systeme zuweist, genutzt. Die zweite Sequenz soll dem Trigger, der für die Einträge der Kundenhistorie zuständig ist, zur Verfügung stehen. Theoretisch kann auch eine Sequenz für alle Trigger verwendet werden, da aber die festgehaltenen Einträge in der Historie auf absehbare Zeit im Verhältnis zu den Kundensystemen zu nehmen werden, erscheint eine separate Sequenz für den Entwicklungsprozess als angemessen, da sich mögliche auftretende Fehler differenzierter analysieren lassen.

Sequenzen

- **RSN_SQ**

  Diese Sequenz dient zur Vergabe der Primärschlüsselwerte für die Kunden- und Systembeschreibung.

  Die Parameter MINVALUE und MAXVALUE definieren den Wertebereich den ein numerischer Wert annehmen kann und der dem Primärschlüssel (ID) übergeben wird. Die Schrittweite und der Startwert müssen definiert werden, normalerweise beträgt die Schrittweite = 1, was das Hochzählen um ‚+1' bedeutet. Der Startwert legt fest ab welchem Wert gezählt werden soll. Dieser ist unter Umständen zu einem späteren Zeitpunkt, beim Laden der Kundendaten, auf die ID des jeweils letzten Tupels anzupassen bzw. es muss ein Wert definiert werden von dem aus wei-

ter gezählt werden soll. Dabei dürfen die nachfolgenden numerischen Werte noch nicht verwendet worden sein.

Die Abbildung 6.8 zeigt den gesamten Aufbau der Sequenz. Ab der fünften Zeile sind die Anweisungen `NOCACHE`, `NOORDER` und `NOOCYCLE` vorhanden, die festlegen, dass keine Werte im Hauptspeicher für die Schlüsselwerte vorgehalten werden, die erzeugten Werte keine Reihenfolge besitzen in denen ein Wert angefragt wird und dass nach Erreichen des maximalen Grenzwertes die Integer Zahlen nicht zyklisch und erneut vergeben werden.

```
1  CREATE SEQUENCE "RSN"."RSN_SQ"
2  MINVALUE 0
3  MAXVALUE 99999999999999999999999999
4  INCREMENT BY 1 START WITH 2000
5  NOCACHE
6  NOORDER
7  NOCYCLE;
```

Abbildung 6.8: SQL Anweisung der Sequenz RSN_SQ

- **KUNDENSTATUS_SEQ**
  Diese Sequenz generiert für die Historie die ID Werte und gleicht syntaktisch der Sequenz „RSN_SQ“.

Trigger:

- **STANDORT_ID** wird durch die DML - Anweisungen `INSERT` oder `UPDATE` ausgelöst und prüft in der Relation, die sich auf den Kunden bezieht, für jede Zeile, ob eine ID bereits existiert oder der NULL Wert, d.h. kein Eintrag für das Tupel vorliegt. Existiert ein NULL Wert, so wird ein ID Wert aus der Sequenz „RSN_SQ“ erzeugt und in das Tupel gespeichert (vgl. Abbildung 6.9).

```
1   REFERENCING NEW AS NEW
2   OLD AS OLD
3   FOR EACH ROW
4   BEGIN
5   IF :NEW.ID IS NULL THEN
6   :NEW.ID := RSN_SQ.NEXTVAL;
7   END IF;
8   END;
```

Abbildung 6.9: Codeausschnitt RSN_STANDORT_ID

Ein folgender Test soll den erstellten Trigger auf seine Funktionalität überprüfen, hierfür werden drei `INSERT` Anweisungen ausgeführt, die Tupel mit bestimmten Attributwerten für die Relation „RSN_STANDORT“ anlegen soll. Das Ergebnis in Abbildung 6.10 bestätigt, dass die ID Werte automatisch angelegt worden sind und der Trigger ordnungsgemäß arbeitet.

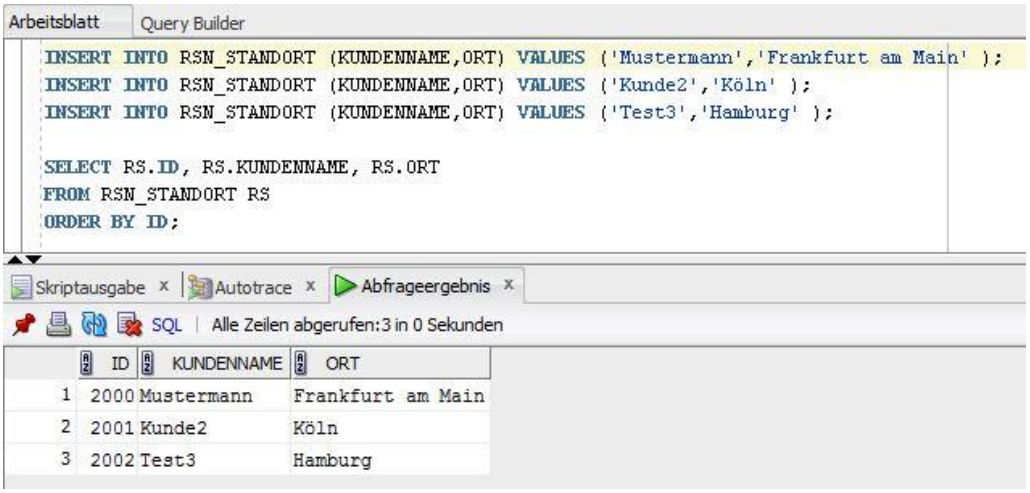


Abbildung 6.10: Test der automatischen ID Zuweisung

- **SYSTEM_ID** wird durch die DML - Anweisungen `INSERT` oder `UPDATE` ausgelöst, der Trigger ist syntaktisch mit dem vorherigen Trigger gleich, da er dieselbe Sequenz „RSN.SQ“ verwendet, um ID Werte für Tupel der Relation „RSN_STANDORT“ zu generieren.

  Der Funktionstest in Abbildung 6.11 liefert den Beweis, dass zum Erstellen eines Tupels keine manuelle ID Zuweisung erforderlich ist. Der Trigger wird, wie erwartet ausgelöst.

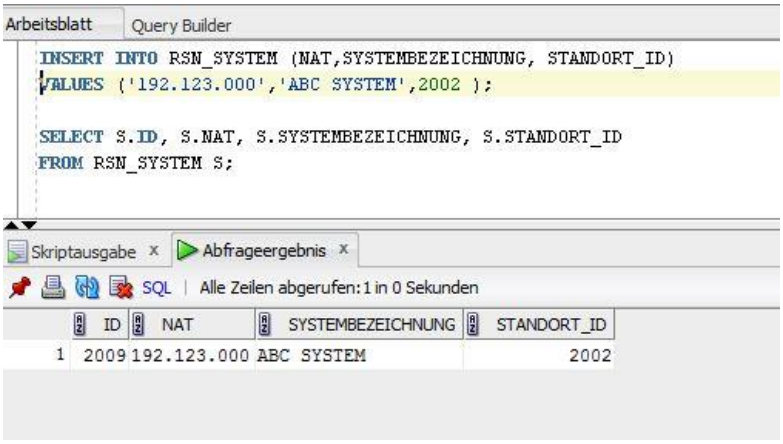


Abbildung 6.11: Test der ID Zuweisung in RSN_SYSTEM

- **KUNDENSTATUS_ID** ist der Relation „RSN_KUNDENSTATUS“ zugeordnet, damit ID Werte für die Tupel der Historie erstellt werden. Verwendet wird die Sequenz mit der Bezeichnung „KUNDENSTATUS_SEQ“.
  Die Prozedur unterscheidet sich von den vorher genannten (vgl. Abbildung 6.12), weil die Tupel einer Historie nicht mehr nachträglich verändert werden sollen, dadurch ist keine Prüfung der aktuellen ID notwendig. Jedes neue Tupel erhält eine fortlaufende ID, d.h. immer der nächste Wert der Sequenz wird an eine neue ID übergeben.

```
1  BEGIN
2  IF :NEW."ID" IS NULL THEN
3  SELECT "KUNDENSTATUS_SEQ".NEXTVAL INTO :NEW."ID" FROM DUAL;
4  END IF;
5  END;
```

Abbildung 6.12: Codeausschnitt von KUNDENSTATUS_ID

6.2.2 Protokollierung

Die in den Konzepten erarbeiteten Funktionen für Kontrollmechanismen bezüglich der Kundenänderungen und Historieneinträge (siehe. Kapitel 4.5) müssen mit Hilfe von Triggern realisiert werden. Hierfür werden zwei Trigger erstellt, die jeweils auf einer Relation operieren.

STANDORT_AUDIT

Der Trigger wird durch die DML Anweisungen `INSERT` bzw. `UPDATE` ausgelöst und schreibt in ein Attribut, das den Datentyp „ Datum“ enthält, den Zeitstempel hinein. Der erforderliche aktuelle Wert des Datums wird hierfür aus dem Systemdatum des Servers

ausgelesen. Parallel muss der aktuell angemeldete Nutzer der Anwendung ermittelt werden, der eine Datenbankoperation ausgeführt hat. Mit Hilfe der Anweisung `NVL(V('APP_USER'), 'USER')` wird die Benutzerkennung ausgelesen und in das ihr zu gewiesene Attribut übergeben (siehe Abbildung 6.13).

```
1   FOR EACH ROW
2   BEGIN
3    :NEW.BEARBEITUNGSDATUM := SYSDATE;
6      :NEW.ID := RSN_SQ.NEXTVAL;
7    :NEW.BEARBEITER := NVL(V('APP_USER'),'USER');
8   END;
```

Abbildung 6.13: Codeausschnitt von STANDORT_AUDIT

Ein Funktionstest ist bereits an dieser Stelle möglich, obwohl noch keine Applikation existiert, da die Anweisung `NVL(V('APP_USER') ), 'USER')` einerseits über die Variable `'APP_USER'` den Anwender der APEX Anwendung aufruft, andererseits über die Variable `'USER'`, den Datenbankbenutzer anzeigt, wenn der Trigger außerhalb einer APEX Anwendung ausgelöst wird. Das Beispiel in Abbildung 6.14 zeigt, dass zu jeder Änderung ein Systemdatum und eine Benutzerkennung automatisch angelegt werden.

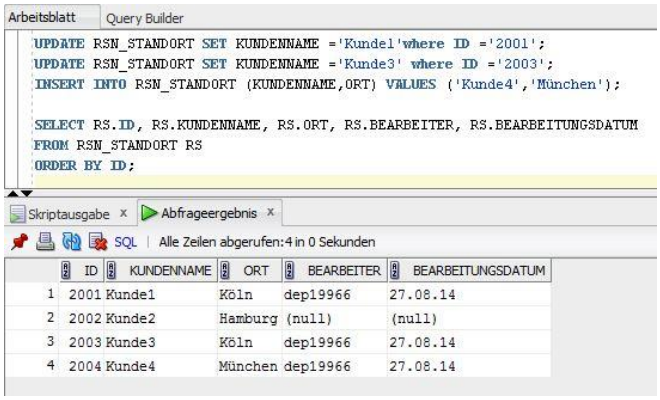


Abbildung 6.14: Funktionstest des Triggers STANDORT_AUDIT

KUNDENSTATUS_DATE

Der Trigger ist für den Speichervorgang des Datums, der Historieneinträge, notwendig. Er wird durch eine `INSERT` Anweisung ausgelöst. Der PL/SQL Anweisungsblock besteht nur aus einer Zuweisung des Systemdatums an das Attribut, welches das Datum speichern soll (vgl. Abbildung 6.15).

```
1   BEGIN
2     :NEW.DATUM := SYSDATE;
3   END;
```

Abbildung 6.15: Codeausschnitt von KUNDENSTATUS_DATE

Der folgende Funktionstest in Abbildung 6.16 zeigt, dass einerseits automatisch das Datum gespeichert wird und andererseits gleichzeig der Trigger für das Autoinkrement ausgeführt worden ist.

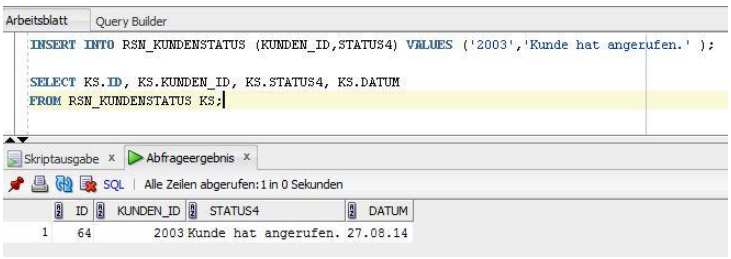


Abbildung 6.16: Trigger Test KUNDENSTATUS_DATE

6.3 Struktur der Webanwendung

6.3.1 Vorlage (Template)

Hinsichtlich der Designkonzepte des vorherigen Kapitels, wird es an dieser Stelle das Ziel sein einen funktionsfähigen Prototyp zu entwickeln.

Im Application Builder wird basierend auf einer gewählten Vorlage (siehe Abbildung 6.17) eine zunächst leere Anwendung erstellt, d.h. es wird eine leere Seite des Formats HTML angelegt. Gleichzeitig generiert der Application Builder eine Login Seite, damit später nur Nutzer auf die Anwendung zugreifen können, denen vom Workspace Administrator die Berechtigung erteilt wurde. Die gewählte Vorlage bietet einfache symmetrische Anordnungsmöglichkeiten für die verschiedenen Regionen, dabei soll festgelegt werden, dass alle Regionen die Position `Page Template Body (3)` verwenden sollen, weil sie dadurch linksseitig ausgerichtet sind.

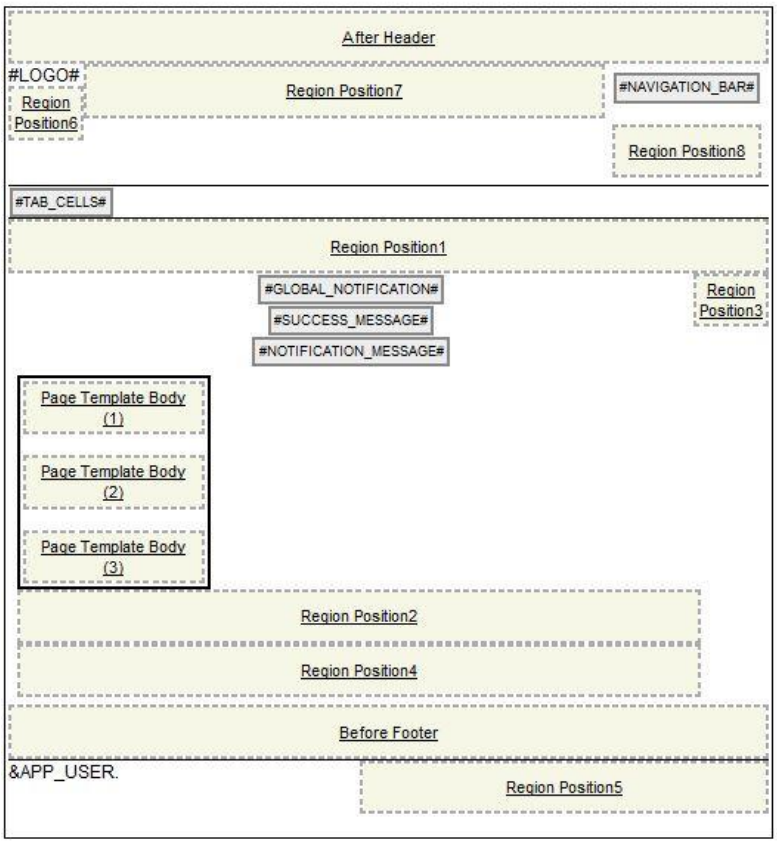


Abbildung 6.17: Template der Webapplikation

6.3.2 Kundenreport

Der Kundenreport soll eine Übersicht des gesamten Kundenbestands liefern und ist die Hauptseite der Anwendung. Das entspricht einer leeren HTML Seite, die vom Application Builder erzeugt worden ist.

Region „Kunden“

Die Übersicht ist eine Region des Typs „interaktiver Bericht“, der angelegt und mit der Bezeichnung „Kunden“ gespeichert wird. Der Bericht liefert auf Grundlage der gewählten Relation, die die Kundenbeschreibung enthält, eine tabellarische Übersicht und stellt ein Such- und Filter-Plug-In, sowie Export- und Analysewerkzeuge in einer darüber liegenden Toolbar bereit. Gleichzeitig wird mit der Wahl des interaktiven Berichts auch ein Master-Detail Formular auf einer neuen Seite angelegt (siehe auch Kapitel 6.3.3), was die Zuweisung einer untergeordneten Relation erforderlich macht. Der interaktive Bericht ist dadurch automatisch mit Editierschaltflächen für jedes Tupel ausgestattet, die als Bleistiftsymbol in der Übersicht dargestellt werden. Bei den Editierschaltflächen handelt es sich um Links, die sich auf den Primärschlüssel der Kundenrelation beziehen und bei Auswahl das entsprechende Tupel auf der Seite des Master-Detail Formulars anzeigt. Eine weitere Schaltfläche zum Erstellen neuer Tupel oberhalb des interaktiven Berichts ist durch den Regionstyp automatisch erstellt worden. Diese Schaltfläche leitet den Anwender zur Seite mit dem Master-Detail Formular weiter, wo das neue Tupel angelegt werden kann.

Datumsformat

Im interaktiven Bericht ist für die Datumsanzeige das Format: „DD.MM.YYYY“ festzulegen.

Region „Suchbereich“

Damit die Suchfunktion aus dem Konzept (vgl. Kapitel 4.7) umgesetzt werden kann, ist zunächst die Toolbar der Region „Kunden“ zu deaktivieren. Eine neue Region des Standard Typs „HTML“ wird mit der Bezeichnung „Suchbereich“ erstellt. Der Anzeigepunkt (Display Point), der die Reihenfolge der Regionen festlegt, wie diese angezeigt werden sollen, muss einen kleineren Wert als die Region „Kunden“ besitzen, um die Region über den interaktiven Bericht zu positionieren. Damit die Region die Suchergebnisse korrekt darstellen kann, ist in der Regionsdefinition im Bereich der Benutzerschnittstelle (User Interface) die Vorlage für einen einzeiligen Berichtsfilter zu wählen. Das Suchergebnis wird anschließend in der Region „Kunden“ die Tupel anzeigen, die mit dem Suchmuster übereinstimmen.

Abschließend sind für die Region die Suchfelder und eine Schaltfläche zu erstellen.

Damit der interaktive Bericht auf die Suchanfrage der einzelnen Felder reagieren kann, ist eine Anpassung des Quellcodes, d.h. die SQL Anweisung, in der Regionsdefinition notwendig. Der Code wird um die zweite Relation, die durchsucht werden soll, ergänzt und die Bindevariablen in der `where` Klausel hinzugefügt. Dabei soll berücksichtigt werden, dass die eingegebenen Zeichenketten nicht case-sensitive[21] sein sollen.

Link Text

Die Konfiguration eines „Link Text“ wird benötigt, damit vom interaktiven Bericht ein Kundenname ausgewählt werden kann, um in den Lesemodus, der in Kapitel 6.3.5 erläutert wird, zu wechseln.

Die eigentliche Verknüpfung erfolgt nicht über den im Bericht dargestellten Attributname, sondern über die ID des gewählten Tupels auf das Seitenelement der Zielseite, die ebenfalls die ID enthält. Mit Hilfe der Spaltenverknüpfung (Column Link) werden die erforderlichen Attribute (siehe Tabelle 6.1) übergeben.

Eigenschaft	Attribut	Beschreibung
Link Text	#KUNDENNAME#	Alle Attributwerte der Kundenbezeichnung werden als Link formatiert.
Target	Page in this Application	Eine Weiterleitung auf Seite innerhalb der Anwendung.
Page	8	Das Ziel ist die Seite mit der Nummer 8.
Item1	P8_ID	Das Ziel ist das Seitenelement, das die gleiche ID enthält.
Value	#ID#	Der Quellwert ist die ID.

Tabelle 6.1: Attribute der Spaltenverknüpfung

Ein erster Test der Seite „Kundenreport“ zeigt die zukünftige Startansicht der Webanwendung (siehe Abbildung 6.18).

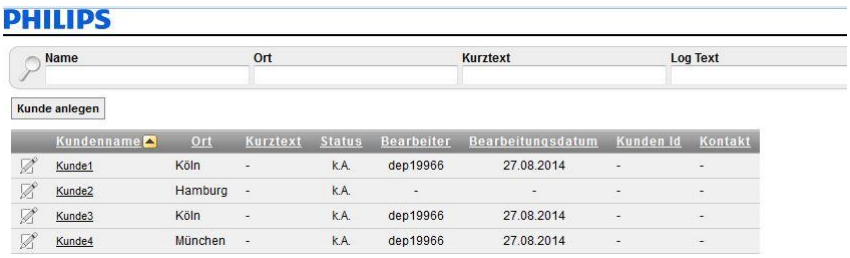


Abbildung 6.18: Ausschnitt der Kundenübersicht

[21] engl. groß- bzw. kleinschreibungsabhängig

6.3.3 Editiermodus

Die Seite „Editiermodus“ ist vom Application Builder automatisch durch die Erstellung des interaktiven Berichts generiert worden. Sie ist einerseits zur nachträglichen Bearbeitung, aber auch zum Erstellen von neuen Tupel, d.h. Kundendatensätzen, notwendig. Die Seite kann aus der Sicht des Anwenders nur über die Schaltflächen „Kunde anlegen“ und „Editieren“ von der Seite „Kundenreport“ erreicht werden. Die Seitenstruktur setzt sich aus einem einfachen Master-Detail Formular zusammen, das mit Navigationsschaltflächen ausgestattet ist und über drei weitere Knöpfe die Anweisungen CREATE, DELETE und CANCEL ausführen kann.

Systeme

Das Detailformular ist eine eigenständige Region mit der Bezeichnung „Systeme“, des Typs „Report“, die nur die zugeordneten Kundensysteme umfasst. Der Detailbereich ist automatisch mit folgenden Prozessen zur Seitenverarbeitung ausgestattet:

- ApplyMRU (Multi Row Update)
  Ermöglicht mehrere Tupel gleichzeitig zu aktualisieren
- ApplyMRD (Multi Row Delete)
  Ermöglicht mehrere Tupel gleichzeitig zu löschen

Diese dynamischen Prozesse können für jede Seite nur einmal erstellt werden. Die innere Struktur ist nicht sichtbar, sie befindet sich in einem sogenannten APEX Package.
Manuell lassen sich die beiden genannten Prozesse nur mit hohem Aufwand über eine Collection und eine Region des Typs „Report“ eingeschränkt nachbilden.

Damit „Systeme“ die angezeigten Tupel mit dem übergeordneten Formular (Master), korrekt anzeigen kann, existiert in der `where` - Klausel von „Systeme“ ein Equi-Join zwischen dem Fremdschlüssel mit der Bindevariable des Hauptformulars und dem Seitenelement „ID“ (vgl. Abbildung 6.19).

```
1  SELECT
2   "ID",
3  ...,
4  FROM "#OWNER#"."RSN_SYSTEM"
5  WHERE "STANDORT_ID" = :P3_ID
```

Abbildung 6.19: SQL Codeausschnitt der Region Systeme

Gelangt der Anwender über die Schaltfläche „Kunde anlegen“ auf die Seite „Editiermodus“, wird mit Hilfe einer Bedingung sichergestellt, dass die Region „Systeme“ solange nicht

sichtbar sein soll, bis der Kunde angelegt worden ist. In der Regionsdefinition sind die Anzeigeeinstellungen (vgl. Tabelle 6.2) zu definieren.

Regionsdefinition	Eigenschaft	Attribut
Conditional Display	Condition Type	Value of Item in Expression 1 is NOT null and the Item is NOT zero
	Expression 1	P3_ID

Tabelle 6.2: Ausschnitt der Regionsdefinition

Die Bedingungen sollen auch für die nachfolgenden Regionen gelten.

6.3.3.1 Bemerkungen

Das Detailformular wird um die Region „Bemerkungen“ erweitert, welche die Historie, d.h. alle gespeicherten Kundeninteraktionen abbildet. Als Regionstyp wird der „interaktive Bericht“ gewählt, da dieser über ein integriertes Suchfeld verfügt.
Benutzerdefinierte Suchfelder, wie auf der Seite „Kundenreport“ (vgl. Kapitel 6.3.2) werden nicht benötigt, da sich die Suche nur auf eine Relation beschränkt. Das Master-Detail Formular, das der Application Builder für den interaktiven Bericht erzeugt hat, findet keine Verwendung und ist zu entfernen.

Der Quellcode der Region wird anschließend angepasst, dabei werden die Attribute, die für die Historie relevante „Altinformationen“ enthalten mit einander verkettet und als eine Spalte in der Region „Bemerkungen“ dargestellt (vgl. Abbildung 6.20).
Durch Equi-Join zwischen dem Fremdschlüssel und der Bindevariablen wird sichergestellt, dass die korrekte Zuordnung der über- und untergeordneten Tupel erhalten bleibt.

```
1   SELECT KS.DATUM AS "DATUM",KS.STATUS8||' '||KS.STATUS1||'
2    '||KS.STATUS2||' '||KS.STATUS3||' '||KS.STATUS4 AS "LOG ÜBERSICHT"
3   FROM RSN_KUNDENSTATUS KS
4   WHERE KS.KUNDEN_ID = :P3_ID
5   ORDER BY ID;
```

Abbildung 6.20: SQL Abfrage in „Bemerkungen“

Ein anschließender Test zeigt, dass die Historie richtig dargestellt wird.

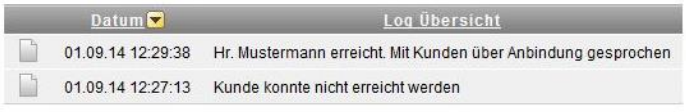


Abbildung 6.21: Ausschnitt der Historie

6.3.3.2 External Links

Die zweite Erweiterung des Detailformulars ist eine Region des Typs „HTML“, die die URL Adresse zum Kundenordner und zum PRS Portal enthält. Damit die dynamische Pfadangabe nach dem Konzept von Kapitel 4.6.3 generiert werden kann, ist ein Seitenelement erforderlich, welches das erste Zeichen der Standortbeschreibung enthalten soll. Das Seitenelement ist vom Typ „Hidden“ und somit für den Anwender auf der Benutzeroberfläche unsichtbar, es enthält eine Sub String - Anweisung, die die Zeichenkette an der ersten Stelle der Standortbeschreibung ausliest (vgl. Abbildung 6.22).

```
1  SELECT SUBSTR (ORT, 1,1)
2  FROM RSN_STANDORT
3  WHERE ID = :P3_ID;
```

Abbildung 6.22: Anweisung für Teil String im Item Source

In der HTML Region müssen anschließend die URL Adressen für die Hyperlinks zum Kundenordner und zum PRS Portal eingetragen und um den Platzhalter, das Seitenelement, welches das erste Zeichen der Standortbeschreibung enthält, erweitert werden.

```
1  <a href=file:///L:/Administration/~Kunden/&P3_SUBSTR.>
2  Öffne Kundenordner</a>
```

Abbildung 6.23: Verweis zum Kundenordner

Beim Verweis auf das PRS Portal ist zusätzlich das Seitenelement der Ortsbezeichnung in die URL Adresse mit aufzunehmen.
Mit Hilfe der JavaScript Methode *window.open()* wird das PRS Portal als eigenständiges Pop-up-Fenster geöffnet, wenn der Hyperlink ausgewählt wird (vgl. Abbildung 6.24).

```
1  <a href="https://pww.portal.rs.healthcare.philips.com/PrsPortal/
2  Default.aspx?City=&P3_ORT.&State=&P3_SUBSTR.&Country=GERMANY&
3  SSR=EMEA" target="_blank"
4  onclick="window.open(this.href,this.target,
5  'width=auto,height=auto');
6  return false;">PRS Portal</a>
```

Abbildung 6.24: Verweis zum PRS Portal

Ein durchgeführter Test mit einem Kunden für den Standort „Radebeul“ zeigt die korrekte Umsetzung der URL Adressen für beide Hyperlinks (siehe Abbildung 6.25 und Abbildung 6.26).

Abbildung 6.25: URL Adresse zum Kundenordner

Abbildung 6.26: URL Adresse zum PRS Portal

6.3.4 Log Formular

Damit der Anwender neue Einträge in die Historie tätigen kann, wird eine neue Seite mit der Bezeichnung „Log Formular“ vom Typ „Form Table“ erstellt. Sie bezieht sich auf die Relation, in der die Historieneinträge gespeichert sind. Das Formular besteht aus drei Seitenelementen, davon sind zwei ausgeblendet, dabei handelt es sich um den Primär- und Fremdschlüssel. Das dritte Seitenelement ist ein Textbereich, der seine Werte an ein Attribut übergibt, welches zukünftig alle neuen Historieneinträge speichern soll.

Schaltflächen

Weil in einer Historie keine Tupel entfernt werden dürfen, ist bei den Schaltflächen, die während der Seitenerstellung vom Application Builder erzeugt werden, festzulegen, dass nur `CREATE` und `CANCEL` Anweisungen erlaubt sind.

Verzweigung

Damit nach dem Speichervorgang oder durch einen Abbruch des Formulars die Seite zurück in den Editiermodus wechselt, ist die Verzweigung in der Seitenverarbeitung zu überprüfen, ob die richtige Seitenzahl angeben worden ist.
Die Seite „Log Formular“ soll für den Anwender nur aus dem Editiermodus aufgerufen werden können, dafür muss die gleichnamige Seite „Editiermodus“ um die Schaltfläche „Log“ erweitert werden und ein Verweis zur Seite „Log Formular“ definiert werden.
Ein Test von „Log Formular“ (siehe Abbildung 6.27) zeigt die erstellte Eingabemaske, in welche die Kundeninteraktionen einzutragen sind.

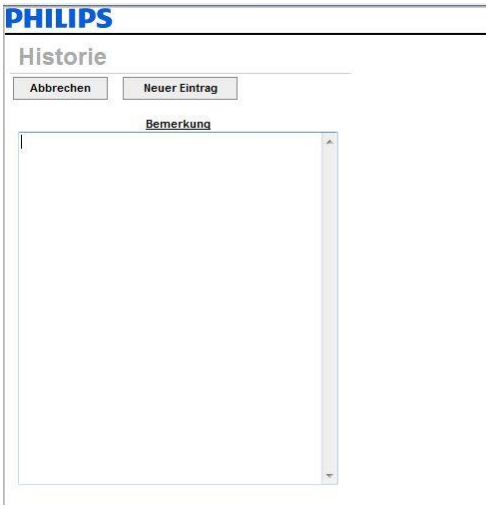


Abbildung 6.27 Eingabemaske für die Historie

6.3.5 Lesemodus

Nicht immer beabsichtigt ein Anwender einen Kunden zu bearbeiten, sondern es sollen nur Informationen betrachtet werden. Zwar lässt sich jeder Kunde über den Editiermodus öffnen und anschauen, jedoch sollten Lese- und Schreibzugriffe voneinander getrennt werden.

Hierfür wird eine Seite mit der Bezeichnung „Lesemodus“ erstellt, jedoch nicht auf die gleiche Art, wie die bisherigen, sondern aus einer Kopie der Seite „Editiermodus“. Dadurch bleibt das Layout der Regionen und Seitenelemente erhalten. Die Region „External Links“ ist zu entfernen, da sie nur im Editiermodus genutzt wird. Des Weiteren sind alle Schaltflächen, ausschließlich jener zur Navigation, zu entfernen. Für alle Seitenelemente ist das Anzeigeattribut auf „Display Only“ zu setzen, dadurch werden die Formularfelder zum Schreiben gesperrt.

6.3.6 Erweiterungen

Die Relation, welche die Standortbeschreibung enthält, ist auf Wunsch der Anwender um einige Attribute erweitert worden.

Die Attributwerte werden hierfür auf der Seite „Editiermodus“ als Checkbox dargestellt, welche statische Wertlisten enthalten, damit nur Integer Zahlen in der Relation gespeichert werden.

7 Testbetrieb

Nachdem im vorherigen Kapitel ein funktionsfähiger Prototyp erstellt worden ist, wird nach dem UER Konzept (vgl. Kapitel 5.2) ein Testbetrieb durchgeführt.

7.1 Nutzerakzeptanztest (User Acceptance Testing)

Ein Nutzerakzeptanztest bzw. UAT muss für den Prototyp durchgeführt werden, bevor die Anwendung in Betrieb gehen kann. Damit soll sichergestellt werden, dass eventuelle Fehler noch rechtzeitig entdeckt und behoben werden können.

Ein weiterer Punkt ist, dass die Anwender die Bedientauglichkeit bewerten sollen und überprüfen müssen, ob die Anwendung ihren Bedürfnissen gerecht wird. Bei einem negativen Feedback ist eine Feinabstimmung erforderlich und der UAT muss erneut ausgeführt werden.

7.2 Testskript

Ein Testskript beschreibt Schritt für Schritt die Anweisung und den Verlauf eines Testszenarios, das der Anwender ausführen soll. Dieser muss im Skript überprüfen, ob die zu erwartenden Ergebnisse eintreffen oder abweichen.
Folgende Szenarien sind durch die Anwender zu testen.

- Kunden anlegen
- Kunden sortieren
- Kundeneinträge suchen
- Leseansicht aufrufen
- Im Editiermodus arbeiten
- Zwischen Kundendatensätze im Editier- und Lesemodus navigieren
- Systeme hinzufügen
- Systeme löschen (eine Zeile)
- Systeme löschen (mehrere Zeilen)

- Einträge in die Kundenhistorie tätigen
- Kundenordner über einen Kunden aufrufen
- PRS-Portal über einen Kunden aufrufen
- Kunden löschen

Zum Testen werden zusätzlich in der Datenbank einige Tupel angelegt mit denen die Anwender im Prototyp arbeiten können.

7.3 Testergebnisse

Das Testszenario ist erfolgreich ausgeführt und durch die Anwender positiv bewertet worden. Die Testdaten lassen sich gezielt über die Such- und Sortierfilter wiederfinden.
Das gleichzeitige Arbeiten mehrerer Anwender ist ohne Verzögerungen möglich. Die Anwendung lässt sich intuitiv und einfach bedienen, alle Informationen sind für den Anwender gut sichtbar angeordnet. Eine erneute Feinabstimmung ist nicht erforderlich.

8 Inbetriebnahme der Webapplikation

In diesem Kapitel werden die letzten erforderlichen Schritte erläutert, damit der Übergang vom Prototyp zur fertigen Anwendung abgeschlossen wird und die Inbetriebnahme der Anwendung ausgeführt werden kann.

8.1 Kundendaten laden

Der Ladeprozess wird die Kundendaten aus der MS Access Datenbank ins RSN Schema in die dortige Datenbank laden. Hierfür müssen die erstellten Abfragen aus Kapitel 6.1.1 und Kapitel 6.1.2 erneut ausgeführt werden, damit die Relationen in der Staging Area aktualisierte Daten enthalten. Mit Hilfe des SQL Developer sind die transformierten Daten als DDL Skripte zu exportieren, die Attributbezeichnungen zu überprüfen und an das Zielschema anzupassen und abschließend im RSN Schema auszuführen.

8.2 Verifikation

8.2.1 Einordnung der Attributwerte

Die Feststellung, dass Kundeninteraktionen in den Relationen Kunden und Kundenstatus gespeichert worden sind (vgl. Kapitel 4.3.2), erfordert, dass die Attributwerte gemeinsam nur noch in einer Relation abgelegt werden sollen, d.h. in den Kundenstatus.
Hierfür ist eine Initialanweisung notwendig (siehe
Abbildung 8.1), die das Attribut aus der Relation „RSN_STANDORT“ selektiert und den Inhalt in ein ungenutztes Attribut der Relation „RSN_KUNDENSTATUS“ kopiert. Dabei wird geprüft, ob schon andere Tupel mit demselben Fremdschlüssel existieren, damit beim Kopiervorgang kein Datenverlust entsteht und die korrekte Zuordnung erhalten bleibt.

```
1   UPDATE RSN_KUNDENSTATUS
2   SET STATUS8 =
3    (SELECT BEMERKUNG_ISDN FROM RSN_STANDORT
4     WHERE RSN_STANDORT.ID=RSN_KUNDENSTATUS.KUNDEN_ID)
5   WHERE EXISTS (SELECT 1 FROM RSN_STANDORT
6   WHERE RSN_STANDORT.ID=RSN_KUNDENSTATUS.KUNDEN_ID);
```

Abbildung 8.1 SQL Code Initialanweisung

8.2.2 Ausblenden ungenutzter Attribute

Das Ausblenden von Attributen in einer Relation ist mit dem Löschen gleichzusetzen, da weder Anwender noch Anwendungen weiter darauf zugreifen können. Ein erneutes Einblenden ist nicht möglich.

Für die Datenbank im RSN Schema sollen alle Attribute, die keine Werte enthalten oder obsolet geworden sind, ausgeblendet werden. Durch die Anweisung `SET UNUSED` werden diese logisch gelöscht.

Der Vorteil gegenüber der `DROP` Anweisung ist, dass die gewählte Methode weniger Zeit und Ressourcen für den Löschvorgang benötigt und insbesondere die Attribute zunächst gesammelt werden können, um sie zu einem beliebigen Zeitpunkt mit der Anweisung `DROP UNUSED COLUMN` physikalisch zu löschen.

8.3 Inbetriebnahme

Die webbasierte Datenbankanwendung ist an dieser Stelle bereit für die Nutzung im Tagesgeschäft des RSN und löst die vorherige Kundendatenverwaltung in MS Access ab.

9 Zusammenfassung

Es sollen in diesem Kapitel noch einmal die Aufgabenstellungen kurz skizziert und die umgesetzten Ergebnisse der Arbeit zusammengefasst werden. Im Anschluss folgt ein Ausblick auf mögliche Erweiterungen, sowie ein Feedback über die Erfahrungen mit der APEX Entwicklungsumgebung.

9.1 Fazit

Das Ziel dieser Arbeit ist es gewesen eine bestehende Datenbank in ein Oracle Schema zu migrieren und auf dieser Basis mit Hilfe der APEX Entwicklungsumgebung ein webbasiertes Datenanwendungsprogramm zu realisieren. Im Verlauf der Entwicklung sind die folgenden Punkte behandelt worden:

- Verstehen der allgemeinen Funktionsprinzipien einer APEX Webanwendung
- Analyse und Diskussion des Ist-Zustands
- Entwurf von Methoden zu Migrationsvorbereitung der Datenbank nach Prinzipien der ETL Prozesse
- Anpassung des Datenbankmodells
- Konsolidierung des Datenbestands
- Entwurf von Protokollfunktionen
- Diskussion über die Möglichkeiten der Filterfunktion
- Diskussion über Verknüpfungsmöglichkeiten mit dem PRS Portal und dem Kundenordner auf einem Dateisystem
- Berücksichtigung der Wahrnehmungspsychologie und Designaspekte für die Benutzeroberfläche der Webanwendung
- Entwurf eines Prototyps und Überprüfung der Teilfunktionen
- Testphase unter Einbeziehung der Anwender
- Laden der Bestandsdaten ins RSN Schema
- Inbetriebnahme der Anwendung

Das Ziel der Arbeit ist erreicht worden, die Datenbank ist verlustfrei ins RSN Schema migriert worden. Das Datenbankmodell ist, wie gefordert in der 2.NF abgebildet, die Anzahl der Relationen ist gleich geblieben und die Beschreibungen der Kundeninteraktionen sind gemeinsam in einem Datenbankobjekt abgespeichert. Die referentielle Integrität zwischen den Relationen ist hergestellt, sodass keine Anomalien beim Anlegen oder Löschen entstehen. Alle Daten sind konsolidiert und bereinigt, nicht verwendete Attribute ohne Werte sind aus allen Relationen vollständig entfernt.
In der Webanwendung kann der Anwender mittels der Kundenübersicht die Einträge bearbeiten, anlegen oder nur lesen. Mit den Suchfeldern sind spezifische Suchen inklusive der Historieneinträge möglich.

Jederzeit ist es den Anwendern möglich anhand der zuletzt gespeicherten Anmeldeinformation zu erkennen, durch welchen Benutzer die Kundeneinträge verändert worden sind.

Ein Problem, was nicht gelöst werden konnte, ist die Möglichkeit die Anwender daran zu hindern erneut den gleichen Kunden anzulegen, da die Kundennummern allgemein unvollständig sind. Die Anwender besitzen keine Kenntnis über das Attribut, da es für ihre Arbeit keine Relevanz hat. Somit konnten die Kundennummern nicht als Merkmal mit der Einschränkung „UNIQUE“ verwendet werden.

Die Wahrscheinlichkeit, dass ein Kunde erneut angelegt wird, ist allerdings sehr gering, da der Arbeitsauflauf der Anwender vorschreibt zuerst zu prüfen, ob der entsprechende Kunde bereits existiert.

Dennoch sollte generell die Data Governance im Unternehmen kritisch überdacht werden.

9.2 Ausblick

Die realisierte Datenbankanwendung kann zukünftigen Anforderungen individuell angepasst oder erweitert werden. So ließe sich z.B. über die Einstellung „Database Link“ im Objekt Browser eine direkte Verknüpfung zu der Datenbank erstellen, welche vom PRS Portal verwendet wird, dadurch hätte der Anwender direkten Zugriff auf alle Informationen und wäre nicht von zwei einzelnen Anwendungen abhängig. Die Ausführung von „Database Link“ wird aber nur möglich, wenn seitens der hierfür zuständigen Datenbankadministratoren eine Zugriffserlaubnis vergeben wird, die im Einvernehmen mit den Datenschutzrichtlinien des Unternehmens sein müssen.

Die neuere APEX Version 4.2 unterstützt mittlerweile ein Framework für mobile Endgeräte, so ließe sich die Datenbankanwendung hierüber zukünftig auch als unternehmenseigene mobile App erweitern.

9.3 Erfahrungen mit APEX

APEX bietet vielseitige Möglichkeiten individuelle Webanwendungen zu erstellen und diese zusätzlich um JavaScript oder Standard HTML zu erweitern. Die verschiedenen Assistenten erleichtern die Erstellung der Anwendungsoberfläche. Allerdings erweist sich die Bearbeitungsoberfläche einer Seite, aufgrund vielseitiger Konfigurationsmöglichkeiten für nahezu alle Komponenten, als sehr unübersichtlich. Da Programmcode an verschiedenen Bereichen innerhalb der Konfiguration zur Seitenwiedergabe und Seitenverarbeitung eingetragen werden kann, ist es nicht immer ersichtlich, welche Syntax akzeptiert wird oder ob nur Codesegmente erforderlich sind. Zudem werden Änderungen von Variablen nicht automatisch übernommen, sodass in der gesamten Anwendung nachträglich Korrekturen erforderlich werden.

Allgemein ist zu sagen, dass mit APEX sehr gut anhand des prototyporientierten Konzepts sofort jeder realisierte Abschnitt eines Projekts getestet werden kann und die Aussage von Ralf Beckmann: „Die Geschwindigkeit, mit denen Sie eine solche Anwendung erstellen können, ist beachtlich und liegt deutlich unter der Zeit, die Sie mit herkömmlichen Programmieransätzen benötigen würden.“ [13] berechtigt ist.

Literaturverzeichnis

Bücher:

[1]. Steiner, René. *Grundkurs Relationale Datenbanken.* s.l. : Springer Vieweg, 2014. ISBN 978-3658-04286-8.

[5]. Stegemann, Gerhard. *Datenbanksysteme.* 1993. ISBN 3-528-04935-9.

[6]. Rolf Dippold, Andreas Meier, Walter Schnider, Klaus Schwinn. *Unternehmensweites Datenmanagement: Von der Datenbankadministration bis zum Informationsmanagement.* Braunschweig / Wiesbaden : Vieweg Verlag, 2005.

[7]. Dietmar Aust, Denes Kubicek, Jens-Christian Pokolm. *Oracle APEX und Oracle XE in der Praxis.* Heidelber, München, Landsberg, Frechen, Hamburg : mitp, 2010. Bd. 1. ISBN 978-3-8266-5549-4.

[9]. Ralph Kimball, Margy Ross. *The Data Warehouse Toolkit.* s.l. : Wiley Computer Publishing, 2002. S. 8. ISBN 9780471200246.

[11]. Markus Beier, Vittoria von Gizycki. *Usability Nutzerfreundliches Web-Design.* Berlin : Springer Verlag, 2002. ISBN 3-540-41914-4.

[13]. Beckmann, Ralf. *Oracle Applikation Express in der Praxis.* München : Hanser Verlag, 2013. S. 2. ISBN 978-3-446-43896-5.

Internet:

[2]. Power To Build. *ORACLE SEQUENCES.* [Online] [Zugriff am: 30. November 2014.] http://power2build.wordpress.com/2014/09/23/oracle-sequences.

[3]. Oracle. Oracle7 Server Concepts Manual. *Database Triggers.* [Online] [Zugriff am: 20.September 2014.] http://docs.oracle.com/cd/A57673_01/DOC/server/doc/SCN73/ch15.htm.

[4]. IT-Infothek. *IT-Systemkaufmann/-frau: Software-Technik.* [Online] [Zugriff am: 30. August 2014.] http://www.it-infothek.de/osz/swt3.html.

[8]. Oracle. Application Development. *2 Day + Developer's Guide.* [Online] [Zugriff am: 11. August 2014.] http://docs.oracle.com/cd/E11882_01/appdev.112/e11946/toc.htm.

[10]. Database Administration. *SQL Developer User's Guide.* [Online] [Zugriff am: 30. August 2014.] http://docs.oracle.com/cd/E11882_01/doc.112/e12152/toc.htm.

[12]. Karl R. Gegenfurtner, Sebastian Walter und Doris I. Braun. Visuelle Informationsverarbeitung im Gehirn. [Online] Justus-Liebig-Universität, Gießen. [Zugriff am: 6. August 2014.] http://www.allpsych.uni-giessen.de/karl/teach/aka.htm.

[14]. Oracle. Oracle Database. *Security Guide.* [Online] [Zugriff am: 4. Dezember 2014.] http://docs.oracle.com/cd/E11882_01/network.112/e36292/toc.htm.

Anhang

Anhang A1: Bildschirmfotos der webbasierten Datenbankanwendung

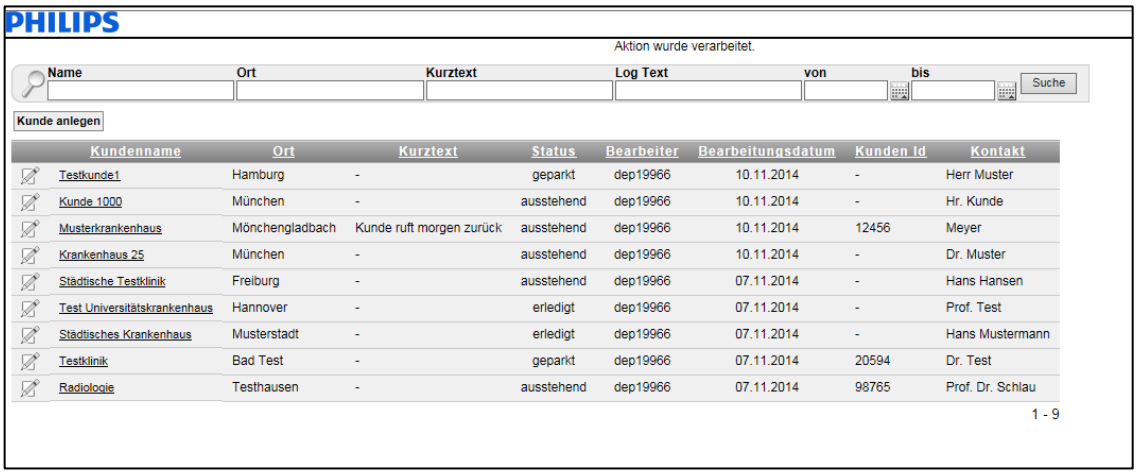
Anhang A2: Entity Relationship Modell

Anhang A3:

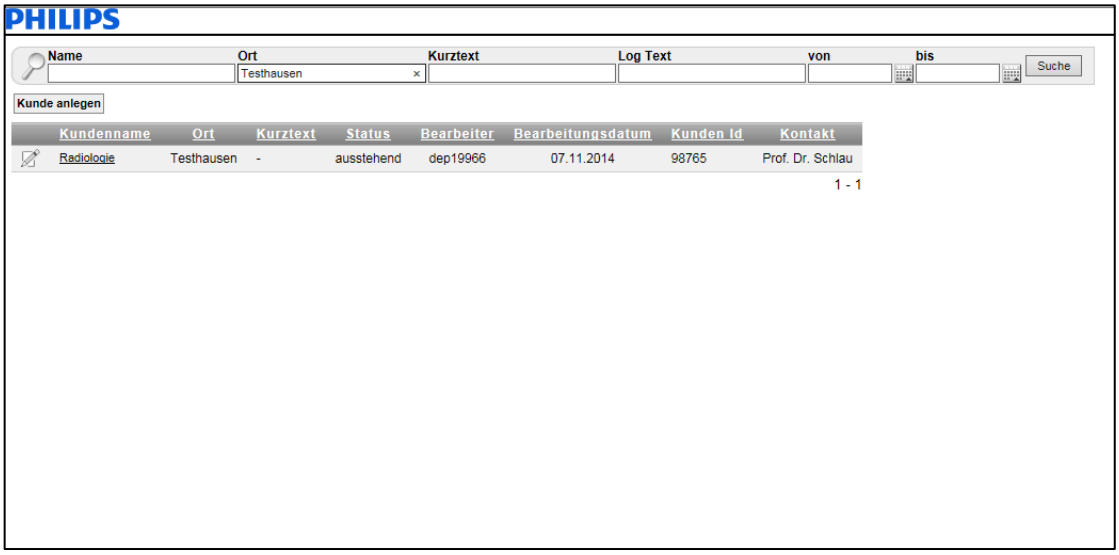
- Bachelorthesis
- Low-fidelity Prototypen
- Verwendete SQL Skripte für MS Access Kundendatenbank
- Datenbankanwendung als SQL Skript zur Installation in ein APEX Workspace
- SQL Script der Kundendatenbank bestehend aus Testdaten
- Seitenkonfigurationsübersicht

Der Anhang A3 dieser Bachelorthesis befindet sich auf einer CD-ROM und ist bei Herrn Prof. Dr.-Ing. Wilfried Wöhlke und Herrn Prof. Dr. Andreas Suhl einzusehen.

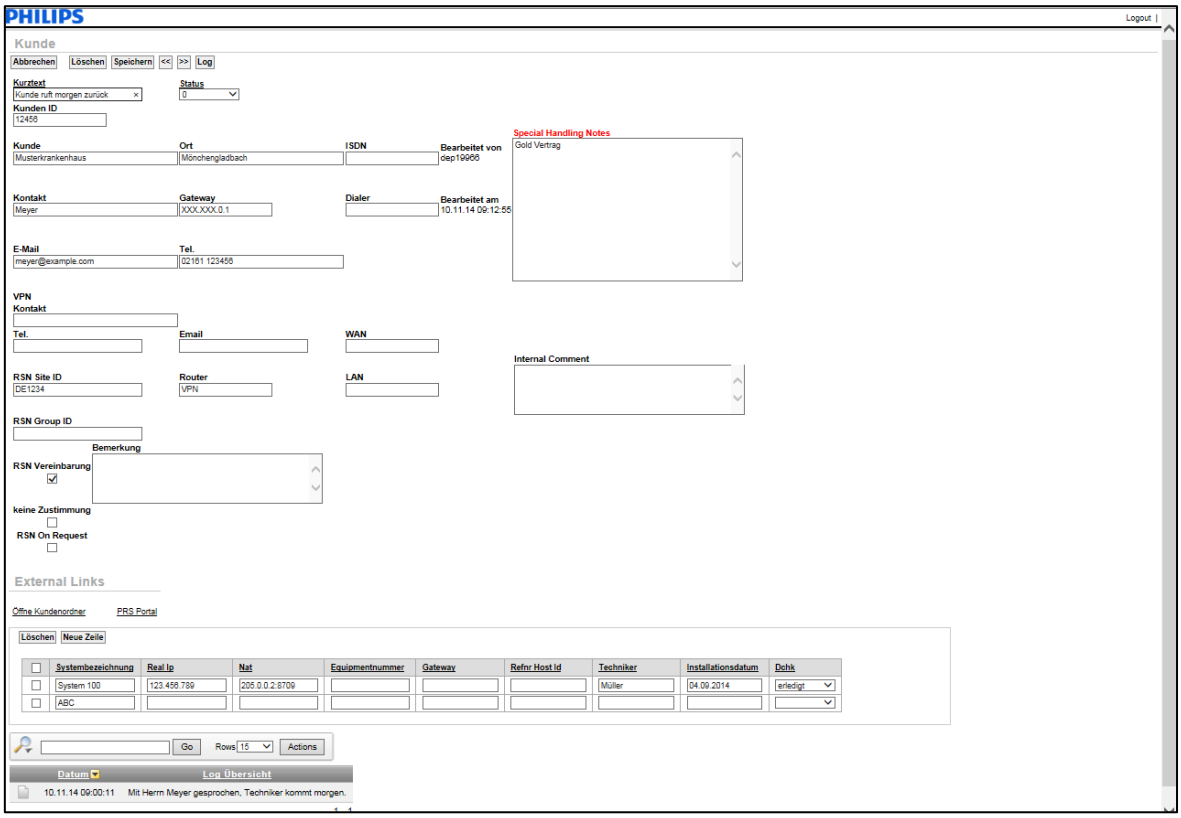
Anhang A1: Bildschirmfotos der webbasierten Datenbankanwendung



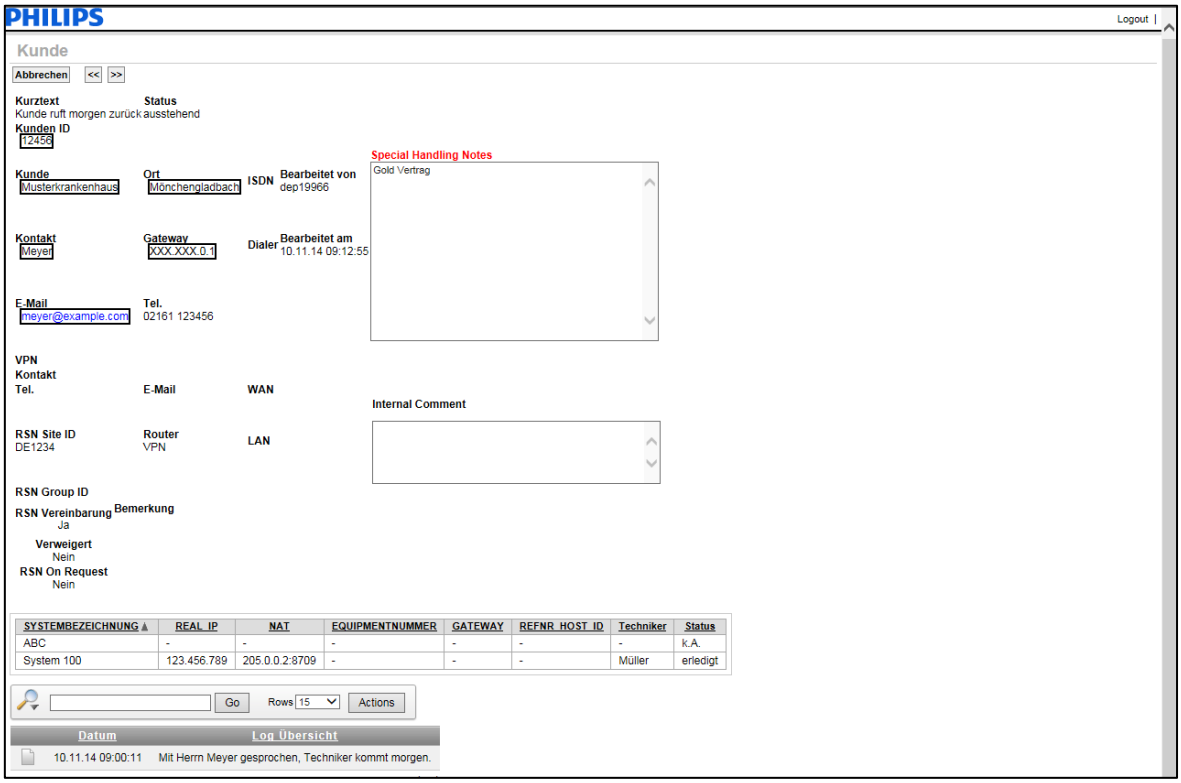
Allgemeine Startansicht mit einigen Testdatensätzen



Ergebnis einer Suche mit Hilfe des Suchfilters



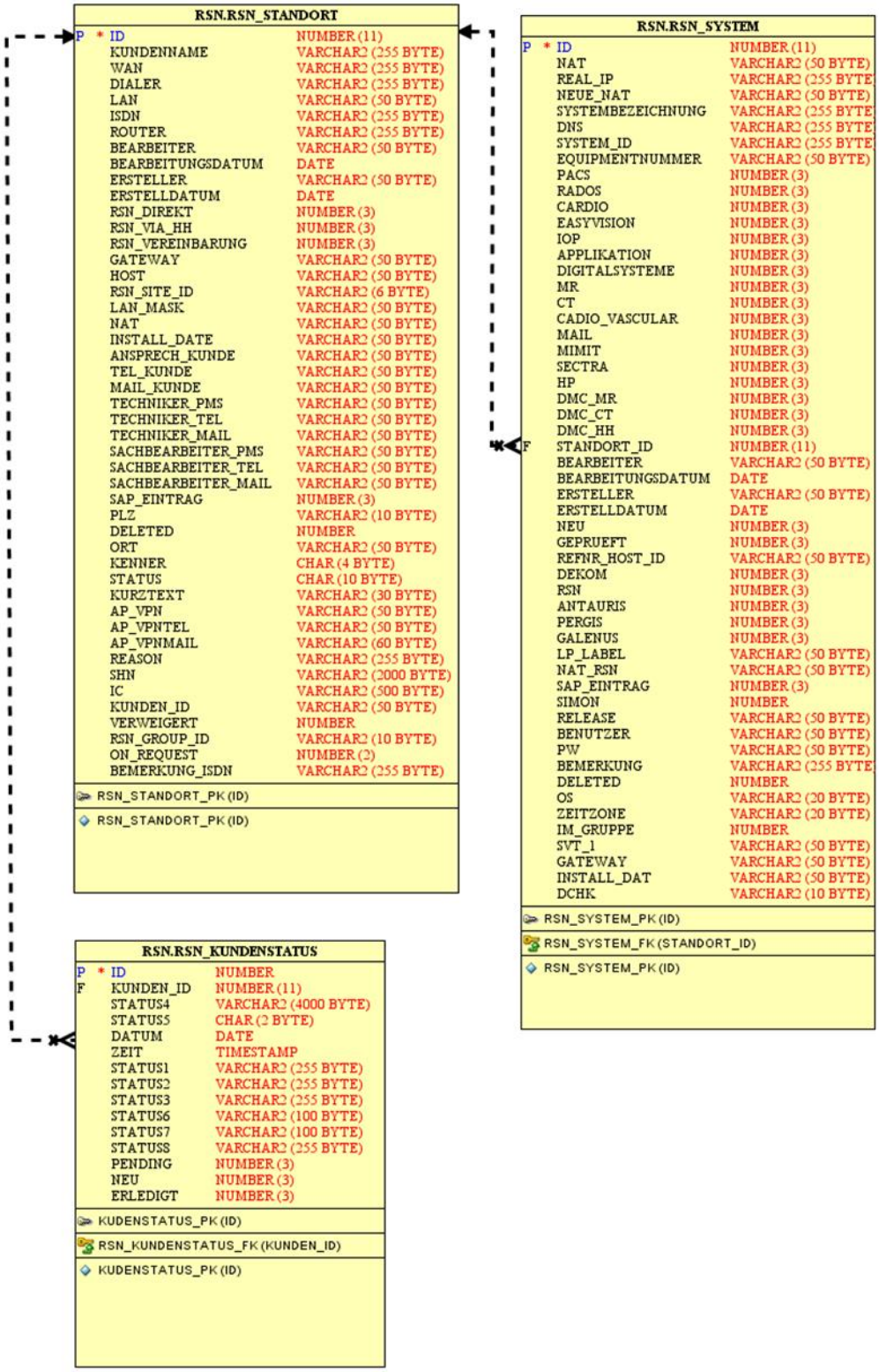
Gesamtansicht des Editiermodus



Gesamtansicht des Lesemodus

Formular für Historieneinträge wird über „Log“ Schaltfläche geöffnet

Anhang A2: Entity Relationship Modell



Kundendatenbank als ERM mit zwei 1:n Kardinalitäten

Glossar

API Application Programming Interface bezeichnet eine Schnittstelle, die anderen Programmen eine Anbindung zu einem System erlaubt.

CRM Customer-Relationship-Management steht für das Kundenbeziehungsmanagement, welches durch Kundenpflege und Interaktionen den Service auf den Kunden verbessern und abstimmen soll.

DDL Data Description Language dient zur Strukturbeschreibung einer Datenbank.

DML Data Manipulation Language dient für Datenbanken als Datenverarbeitungssprache.

NLS National Language Support stellt für Anwendungen die Spracheinstellungen und Datumsformate ein.

RAD Rapid Application Development ist ein Konzept zur Softwareentwicklung nach iterativen Prozessen.

RSN Remote Service Network ist der Fernwartungsservice für medizintechnische Geräte der Philips GmbH.

SQL Structured Query Language ist eine Datenbanksprache für relationale Datenbanken.

Versicherung über die Selbständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §16(5) APSO-TI-BM ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen habe ich unter Angabe der Quellen kenntlich gemacht.

Hamburg, 17.02.2015
Ort, Datum

Unterschrift