Projekt der Reiseveranstaltungssoftware

Projekterstellt von Ba & BE GmbH

Version 1.0   
Datum 14.09.2013

# Inhaltsverzeichnis

[Inhaltsverzeichnis 2](#_Toc367644033)

[Voruntersuchung 3](#_Toc367644034)

[Internetrecherche 3](#_Toc367644035)

[Brainstorming 3](#_Toc367644036)

[Vorbereitungen 4](#_Toc367644037)

[Einarbeiten in die bestehende Problematik 4](#_Toc367644038)

[Realisierungsmöglichkeiten feststellen 4](#_Toc367644039)

[Aufgabenliste erstellen 4](#_Toc367644040)

[Zeiterfassung 5](#_Toc367644041)

[Entwicklung 5](#_Toc367644042)

[Struktur Beschreibung 8](#_Toc367644043)

[Länder Tabelle 8](#_Toc367644044)

[Passagieren Tabelle 8](#_Toc367644045)

[Buchungen Tabelle 9](#_Toc367644046)

[Linien Tabelle 9](#_Toc367644047)

[Flugzeugtypen Tabelle 9](#_Toc367644048)

[Flughäfen Tabelle 10](#_Toc367644049)

[Fluggesellschaften Tabelle 10](#_Toc367644050)

[Hersteller Tabelle 10](#_Toc367644051)

[Flüge Tabelle 10](#_Toc367644052)

[Tests 12](#_Toc367644053)

[Abschluss 12](#_Toc367644054)

[Untersuchung der Aufgabe 12](#_Toc367644055)

[Datenstruktur 13](#_Toc367644056)

[ERM Model\* 13](#_Toc367644057)

[Welche Kriterien sind für das DB-Design in dieser Aufgabe wichtig 13](#_Toc367644058)

[Zeiterfassung 14](#_Toc367644059)

[Glossar 14](#_Toc367644060)

[Verwendete Software 15](#_Toc367644061)

# Voruntersuchung

**Analyse der Aufgabe**

Das Ziel ist eine Anwendung zu erstellen das die Daten von Excel Tabellen in der Datenbank einliest.

Hierbei ist zu beachten, dass jeder Kunde pro Buchung nur einen Ticket bestellen kann, dabei sind folgende Teilaufgaben zu unterscheiden:

* Analyse der vorhandenen Datenbestände.
* Entwurf einer relationalen Datenbank, die eine möglichst redundanzfreie und änderungsfreundliche Speicherung der Daten ermöglicht und gleichzeitig eine möglichst weitergehende Absicherung gegen Fehler beim Erfassen/Bearbeiten von Datensätzen bietet.
* Implementierung und Test der Datenbank.
* Entwicklung und Test einer Java- Anwendung, die die Ausgangsdaten in die Datenbank überträgt
* Die Dokumentationserstellung
* Die Zeiterfassung/Schätzung
* Erstellung von Glossary

## Internetrecherche

Für die Aufgabe wurde die Datenbank von Prof. Dr. Peter Kursawe Fachhochschule Ludwigshafen am Rhein - Hochschule für Wirtschaft gefunden und analysiert. Für das Erstellen des Datenbankkonzepts konnte ich die wichtigsten Punkte von der ERM abschauen.

Der Link zur Praktische Übungen mit Access (Thema: Reisebüro):  
<http://www.fh-lu.de/kursawe/db/reiseindex.htm>

## Brainstorming

Um die Aufgabe besser zu verstehen, habe ich Zeichnungen auf Papier skizziert und dabei Versucht die Aufgabeanforderungen besser zur Begreifen, dabei stellte ich mir die Fragen über die Anforderungen, die ich mit meine Kollege diskutiert habe und aufgeschrieben für die weitere Nutzung bei der Erstellung von Datenbank.

# Vorbereitungen

## Einarbeiten in die bestehende Problematik

Dadurch, dass die vorgegebene Excel-Datei zum Importieren in der Datenbank mit Hilfe von Java relative aufwändig zu lesen ist, wurde entschieden die Datei als CSV-Liste mit Hilfe von Excel abzuspeichern um später in Java einzulesen.

Es gab ein fehl Versuch Oracle Datenbank 11g zu installieren, dabei konnte ich nicht mich einloggen und korrekte Treiber finden.

## Realisierungsmöglichkeiten feststellen

Für das Speichern der Daten wurde für ein Relationsdatenbank MySql entschieden, weil es einfacher als Oracle Datenbank ist für die Aufgabe völlig ausreichend ist.

## Aufgabenliste erstellen

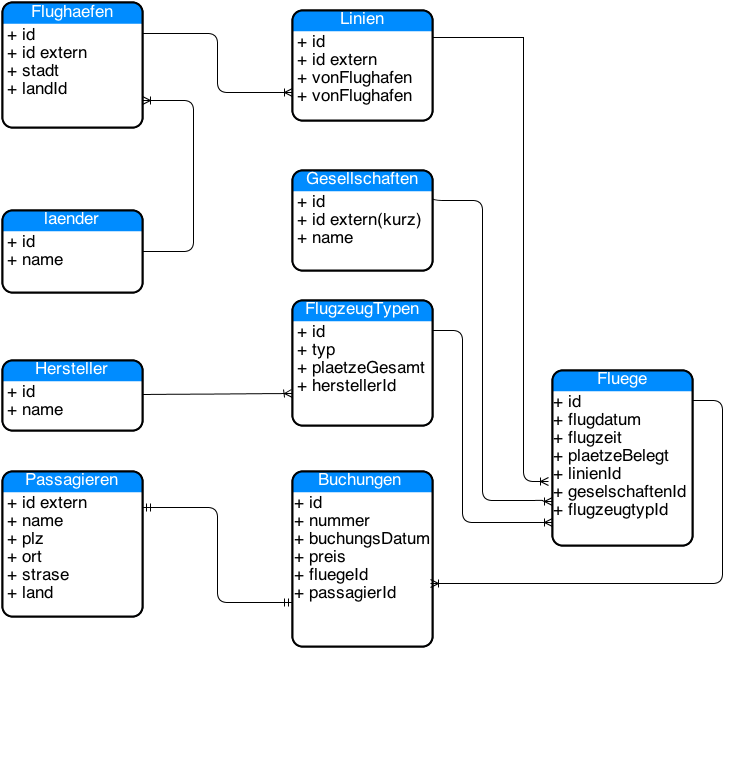
Für das Erstellen des Programms habe ich entsprechende Aufgaben gruppiert und angelegt die in der nachfolgende Zeiterfassung verwendet und abgebildet wurden.

## Zeiterfassung

|  |  |  |  |
| --- | --- | --- | --- |
| **Tätigkeiten** | **Soll-Zeit** | **Ist-Zeit** | **Differenz** |
| **Stunden** | | |
| **Voruntersuchung** |  |  |  |
| Analyse der Aufgabe | *2,00* | *2,00* | *0,00* |
| Internetrecherche | *1,00* | *0,50* | *0,50* |
| Brainstorming | *1,00* | *0,50* | *0,50* |
| *Zwischensumme* | *4,00* | *3,00* | *1,00* |
|  |  |  |  |
| **Vorbereitungen** |  |  |  |
| Einarbeiten in die bestehende Problematik | *4,00* | *6,00* | *-2,00* |
| Realisierungsmöglichkeiten feststellen | *1,00* | *2,00* | *-1,00* |
| Aufgabenliste erstellen | *3,00* | *2,00* | *1,00* |
| Zeiterfassung | *1,50* | *1,00* | *0,50* |
| *Zwischensumme* | *9,50* | *11,00* | *-1,50* |
|  |  |  |  |
| **Entwicklung** |  |  |  |
| Datenbank Design | *8,00* | *15,00* | *-7,00* |
| Datenbank Installation und Erstellung | *4,00* | *6,00* | *-2,00* |
| API Design und Erstellen von Klassen | *6,00* | *8,00* | *-2,00* |
| Businesslogik Umsetzung (Programmierung) | *10,00* | *15,00* | *-5,00* |
| *Zwischensumme* | *28,00* | *44,00* | *-16,00* |
|  |  |  |  |
| **Tests** |  |  |  |
| Datenintegrität Prüfungen | *4,00* | *5,00* | *-1,00* |
| Logiktests | *4,00* | *5,00* | *-1,00* |
| *Zwischensumme* | *8,00* | *10,00* | *-2,00* |
|  |  |  |  |
| **Abschluss** |  |  |  |
| Erfassung und Verbesserung des Dokumentation | *18,00* | *15,00* | *3,00* |
| Verbesserung des Quellkodes | *3,00* | *1,00* | *2,00* |
| *Zwischensumme* | *21,00* | *16,00* | *5,00* |
|  |  |  |  |
| ***Zeitaufwand gesamt:*** | ***70,50*** | ***84,00*** | ***-13,50*** |

# Entwicklung

**Datenbank Design**



*ERM Diagramm*

## Struktur Beschreibung

### Länder Tabelle

Dient zum Speichern von Ländern, wird gebraucht weil mehrere Flughäfen in einem Land existieren können. Spart Platz, keine Redundanz\*.

|  |  |  |  |
| --- | --- | --- | --- |
| **Spaltenname** | **Schlüssel** | **Datentypen** | **NULLS** |
| **Id** | primary key | number(10) |  |
| **name** |  | varchar2(30) | NOT NULL |

### Passagieren Tabelle

Dient zur Vergabe von Passagier Nummer und der Übergabe von Persönlichen Daten in die Buchungsliste

|  |  |  |  |
| --- | --- | --- | --- |
| **Spaltenname** | **Schlüssel** | **Datentypen** | **NULLS** |
| **externId** | primary key | number(10) |  |
| **anrede** |  | varchar2(100) |  |
| **name** |  | varchar2(80) | NOT NULL |
| **plz** |  | number(10) | NOT NULL |
| **ort** |  | varchar2(80) | NOT NULL |
| **strasse** |  | varchar2(80) | NOT NULL |
| **land** |  | varchar2(80) | NOT NULL |

### Buchungen Tabelle

Dient zur Speicherung von Reservierungen des Kunden und vergibt das Buchungsdatum für einen bestimmten Flug

|  |  |  |  |
| --- | --- | --- | --- |
| **Spaltenname** | **Schlüssel** | **Datentypen** | **NULLS** |
| **id** | primary key | number(10) |  |
| **id extern** |  | number(10) | NOT NULL |
| **preis** |  | number(8.2) |  |
| **buchungsDatum** |  | date | NOT NULL |
| **passagierId** | foreign key | number(10) | NOT NULL |
| **flugId** | foreign key | number(10) | NOT NULL |
| **flugzeugTyp** | foreign key | number(10) | NOT NULL |

### Linien Tabelle

Dient zur Bestimmung der Flugstrecke vom Punkt A zum Punkt B.

|  |  |  |  |
| --- | --- | --- | --- |
| **Spaltenname** | **Schlüssel** | **Datentypen** | **NULLS** |
| **id extern** | primary key | number(10) |  |
| **vonFlughafenId** | foreign key | number(10) | NOT NULL |
| **nachFlughafenId** | foreign key | number(10) | NOT NULL |
| **gesellschaftId** | foreign key | mumber(10) | NOT NULL |

### Flugzeugtypen Tabelle

Wird gebraucht um den Typ eines Flugzeugs zu speichern, das eine bestimmte Strecke fliegt. Auch der Hersteller und das Fassungsvermögen von einem Flugzeug sind hier gespeichert. Separate Führung von der Fluggesellschaften Tabelle ist deshalb notwendig weil sonst eine Gesellschaft nur einen Flieger haben könnte.

|  |  |  |  |
| --- | --- | --- | --- |
| **Spaltenname** | **Schlüssel** | **Datentypen** | **NULLS** |
| **id** | primary key | number(10) |  |
| **typ** |  | varchar2(30) | NOT NULL |
| **plaetzeGesamt** |  | number(5) | NOT NULL |
| **herstellerId** | foreign key | number(10) | NOT NULL |

### Flughäfen Tabelle

Dient zum Speichern der Flughäfen dar und zur Vergabe von unikalem Schlüssel für jeden Daten Satz.

|  |  |  |  |
| --- | --- | --- | --- |
| **Spaltenname** | **Schlüssel** | **Datentypen** | **NULLS** |
| **Id** | primary key | number(10) |  |
| **id extern** |  | Varchar2(8) | NOT NULL |
| **stadt** |  | varchar2(30) | NOT NULL |
| **landId** | foreign key | number(10) | NOT NULL |

### Fluggesellschaften Tabelle

Dient zum Speichern von unique Schlüsseln und Firmennamen. Redundanzfreie und änderungsfreundliche Speicherung ist damit gesichert.

|  |  |  |  |
| --- | --- | --- | --- |
| **Spaltenname** | **Schlüssel** | **Datentypen** | **NULLS** |
| **Id** | primary key | number(10) |  |
| **id extern** |  | varchar2(8) | NOT NULL |
| **name** |  | varchar2(100) | NOT NULL |

### Hersteller Tabelle

Diese Tabelle ermöglicht ein Benutzerfreundliche Bedingung der Datenbank. Deckt den Fall ab der häufig zum tage kommt eine Übernahme der Firma von einer anderen.

|  |  |  |  |
| --- | --- | --- | --- |
| **Spaltenname** | **Schlüssel** | **Datentypen** | **NULLS** |
| **Id** | primary key | number(10) |  |
| **herteller** |  | varchar2(100) | NOT NULL |

### Flüge Tabelle

Schließt in sich alle benötigten Ressourcen zusammen die für einen Erfolgreichen Flug von Punkt A zu Punkt B benötigt werden sowohl Abflugdatum, Flugdauer, Typ des Fliegers als auch die Gesellschaft die das Angebot anbietet.

|  |  |  |  |
| --- | --- | --- | --- |
| **Spaltenname** | **Shlüssel** | **Datentyp** | **NULLS** |
| **Id** | primary key | number(10) |  |
| **belegteplaetze** |  | number(10) | NOT NULL |
| **flugdatum** |  | date | NOT NULL |
| **flugdauer** |  | number(4.2) | NOT NULL |
| **linienId** | foreign key | number(10)) | NOT NULL |

Hinweis zur Mangel der Datenbankstruktur:

* Jeder Passagier darf ein Platz pro Buchung bestellen und nur für sich selbst.

**Datenbank Installation und Erstellung**Für das Abspeichern der Daten ist MySql Installation nötig gewesen. Danach sind entspechen der Datenbankdesigns sind die Tabellen mit Hilfe von SQL angelegt worden.

**API Design und Erstellen von Klassen**

**importAusgangsdaten(csv): boolean**Dient für einlesen der AusgangsDaten in die Daten Bank. Zurück lifert die methode boolean\* der besagt ob das einlesen der csv Datei erfolgreich war.

***importBuchungen(buchungenCsv): boolean***Wird gebraucht um die neu dazu gekomene buchungen aus csv.file einzulesen.

***buchungenFluege(datum, passagierId, prei,, flugId): buchungId***

Dient für einträge der Reservierungen. Um eine Buchung zu betätigen braucht das Entity- Buchungen eine Strecke und Name des Passagiers, deshalb ist es hier der Fremdschlüssel von FlugId und Passagier notig. Weiterhin enthält die Tabelle Fluege eine Referenz auf die Tabelle Linien die in sich eine ( von/nach flughafenId ) beinhaltet die weiterhin auf flughaefen und Adressen Verweist. Dadurch wird es klar *von* wo *nach* wo die Reise geht. Außerdem muss der Kunde wissen mit welchem Flugzeug er fliegt welche Gesellschaft Ihn Betreut wann der Flug Startet. Der Preis wird hier benötigt   
um festzuhalten mit welchem Betrag der Kunde gebucht hat. Bei der Abholung zahlt der Kunde den Gebuchten Preis. Somit ist auch die Möglichkeit gegeben die Preise zu variieren. Zurück gegeben wird die buchungId.

***speichernPassagieren(anrede, name, addresseId ): PassagierId***

Wird zum Speichern der Kunden benötig. Ein Kunde braucht einen Namen und eine Adresse, deshalb ist es hier ein Verweis auf die Tabelle Adressen im Bedarf. Zurück gibt die Tabelle eine identifikationsnummer.

***speichernAddressen(stadt, plz, strasse, landId): addressenId***

Dient zur Speicherung der Adressen zB. der neuen Flughäfen oder von Passagieren die als neue Kunden dazu kommen und deshalb braucht Diese Tabelle eine Referenz zur Entity-Laender.   
Land und Adresse werden in zwei Tabellen geteilt um nicht Redundant zu Arbeiten.

***listLaender(): laender* []**

Mit dieser Methode ermögliche ich das die Liste der Länder mit unikalen Schlüssel angefordert werden kann.

***speichernFluege(belegteplaetze, flugdatum, flugdauer, preis, linienId, flugzeugtypId, fluggeselschaftId): fluegeId***

Dient zur Speicherung einzelnen Flügen. Dafür braucht man einen Abflug Datum, Preis eine linienId, flugzeugtypId und fluggeselschaftId. Um fest zu stellen wie viele Plätze noch frei sind muss der Operator einen Vergleich mit der Tabelle Flugzeugtypen stellen. Mit der Referenz flugzeugtyp kann er das erfolgreich durchführen. Außer dem jeder Flug kann mit Verschiedenen Flugzeug typen geflogen werden und das spiegelt sich auf Flugdauer aus. Der Preis sagt aus, denn aktuellen Preis von Flug.

***speichernFlugzeugtypen(herstelle, typ, gesamtplaetze): flugzeugtypId***

Ist für Aktuellen Stand und für Erweiterung des Flug Parks notwendig wenn zB. ein neuer Fliegertyp eines Herstellers zum Einsatz kommt wird es in die Tabelle eingetragen.

***speichernFluggeselschaften(name): fluggeselschaftenId***

Für das Abspeichern der Daten ist MySql Installation nötig gewesen. Danach sind entspechen der Datenbankdesigns sind die Tabellen mit Hilfe von SQL angelegt worden.

**Businesslogik Umsetzung (Programmierung)**

Als erstes hab ich die Entities „EGOs“ festgehalten die für die Daten als Speicher Tabellen gelten werden. Bei mir sind es die neun Tabellen die oben in der Zeichnung aufgeführt sind.   
Danach hab ich das Problem als eine Hierarchie Beziehung angesehen die von ganz großem zu immer kleineren detail verweist also immer Detaillierter wird. Die Auslagerung von solchen Tabellen wie Länder und Herstelle spiegeln sich auf Benutzerfreundlichkeit ab. Das heißt ich kann Problemlos eine Firma umbenennen wenn dies über eine Übernahme geschieht und genauso eine Land Namensänderung durchführen. Als nächstes lege ich eine Flughaefen Tabelle an weil die erste Frage die auf mich zugeckommen ist *‚ Wie viel Flughäfen hat ein Land?* ’ was nichts anderes heißt das ein Land mehrere Flughäfen haben kann. Nach dieser Festlegung Erstelle ich eine Tabelle Flughaefen und übertrage das Primary key von Laender Entity in die Tabelle Flughaefen. Als nächstes schritt wird die Wahl sein zwischen der Tabellen Geselschaften und Linien. Die Überlegung ist die. Ein Flughafen hat mehrere Linien “Strecken“ im Angebot so wie auch kann mehrere Fluggesellschaften betreuen. Da aber es Ausdrücklich in der aufgaben Stellung gesagt wird das eine Gesellschaft kein Heimatflughafen besitz ist damit die Wahl eindeutig. Ich Erstelle eine Tabelle Linien und trage die Referenz der Flughaefen Tabelle in das Entity Linien ein. Hier bei ist zu beachten das eine Linie zwischen zwei Punkten statt findet was nichts anderes heist das es zwei unikale Schlüssel aus Entity Flughaefen in die Tabelle Linien einzutragen ist. Die können z.B. vonFlughafenId und bisFlughafenId heisen. In nächstem schritt des Designs hab ich mich wieder auf die Hierarchie bezogen. Weil die Tabelle Gesellschaften schon vorher den Bezug auf das Entity Flughaefen fest legen konnte aber in der Aufgabe dazu verneint wurde wird sie jetzt angelegt und damit könnte die Beziehung zu Tabelle Linie hergestellt werden. Aber nach langer Überlegung hab ich festgestellt das es logisch nicht korekt wäre weil es dan eine Redundanz geben würde. Es ist zwar richtig das eine Gesellschaft mehrere Linien haben kann aber dann wäre der Datensatz in der Tabelle Linie mehrvach gespeichert, der besagt häte das die gleiche Linie bei mehrere geselschaften im angeboten zu haben ist z.B. *LinienId: Berlin - München wird von Geselschaft Lufthansa angeboten* *weiter*  *LinieId: Berlin - München wird von Geselschaft AirBuss angeboten*. Wie sie sehen entseht hir eine unnötige Speicher Platz Verschwendung und deshalb wird die Tabelle Gesellschaften for erst keine Referenz vegabe tätigen. Hier mit hab ich festgestellt das es ein Logikfehler ist, eine Beziehung mit der Linien Tabelle von Entity Geselschaften festzulegen und bin auf der suche nach einem neuem Entity. Als nechstes in der   
Daten Hierarchie betrachte ich die Tabell Fluege als richtig und ausschlaggebend weil sie die nechste stuffe der Detallierung im meinem verständnis ist. Die nechsten Fragen die ich mir stelle *„ wie sieht es aus wenn ich die Referencen von Entities Geselschaften und Linien in die Tabelle Fluege eintrage “* und „ *welche Daten muss ein flug in seinem tupel ( Datensatz ) beinhalten ? “.* Also wenn ich das Primary key der Linien in die Flughaefen Tabelle übertrage, würde es heisen dass eine Strecke mit mehreren Flieger geflogen werden kann was eine richtige aussage ist. Genau so kann ein Flug von mehreren Geselschaften angeboten werden was auch richtig ist. Damit wäre die Redundanz die vor kurzem als ein Problem in der Tabelle Linien aufgetaucht ist beseitigt. Wichtige Frage zu diesem zeitpunkt ist.

„ *Was braucht das Entity Flüge noch für Atributte ?* und *Was Bezieht sich nur auf einen Flug* ?“. Als umunstritten dencke ich mir steht in der Tabelle Fluege ein flugdatum alle anderen alternatieven werden Redundantz nach sich ziehen. Mit nechsten schritt fechte ich die Flug dauer eines Fluges an. Also ein Flug hat eine bestimte Flugdauer, kontrolle *stimt es auch? Warum nicht in die Linien anzutragen es ist doch immer die gleiche Strecke?* Das Problem ist das die Flugdauer nicht nur von der Strecke abhängt sonder auch von Flugzeugtyp und der Geselschaft die das anbietet das wäre dan eine Vereigemeinlichung und das ist kaum zu gewerleisten. Im diesem sinne gielt die Aussage „ *Eine und die selbe strecke werden verschiedene Flugzeuge zur einer verschiedener zeit zurücklegen* „. Wie Sie sehen ist das nechste Entity von der Oben stehende Überlegung aus selbst bestiment. Die Tabelle Flugzeugtypen ist jetzt der nechste glied im dem „Daten Baum“. Mit diesem Schritt bewege ich das dass Entity in die „ Detaillierung Struktur “ einbezogen wird. *„Hir bei stelle ich mir for das es wie ein baumstam Aussieht welchen ich von oben nach unten zeichnen muss. - Wo bei Oben Globale datetn als Bläter zu sehen sind und unten die Wurzel als Detallierte Daten zur erscheinung treten. Die Tabellen Flugzeugtyp und Gesellschaft werden hier als ein Zweig der im nichts endet angesehen. -Betrachtung von unten nach oben- “.* Nach der Anlegung der Entity Flugzeugtypen übergebe ich ihr einen Parameter von der Ausgelagerten Entity Hersteller ( Benutzerfreuindlichkeit ) und vergebe einen unikat Schlüssel, welchen ich in die Tabelle Fluege oder in die Tabelle Buchungen eintragen werde da die aussage Besagt das der Typ des Fliegers mehrere Buchungen und der Flug mit mehreren Flugzeugtypen geflogen werden kann. Auf diese Frage werde ich nach volgenden äuserung eingehen. Das Entity Flugzeugtypen hat das Attribut plätzegesamt weil jeder Flugzeug sein eigenes Vassungs vermögen hat. Jetzt wichtige Frage wohin mit Priamary key ? „ *kann die flugzeugtypId im Entity Buchungen stehen? “* Wie ich festgestellt habkönnen ja, aber mit speicherplatz verlust (dazu muss gesgt werden dass in die Tabelle Buchungen eine Referenc auf das Objekt Fluge existieret ) weil ein Flug mehrere Buchungen haben kann und die Buchung nur einem Flug zugeordnet ist. Stellen wir uns nun vor das zwei Buchungen gemacht werden es ist egal welche Geselschaft das Flug anbietet und wohin der Flug geht. Der wichtige Punkt hier, ist das wenn ein Kunde mit einer maschinentyp von A zu B Fliegt und der andere mit gleichem typ entweder die gleiche strecke oder eine andere nutzt mus das flugzeugtyp zweimal in der Tabelle Buchungen in eine extra spalte angegeben werdern. Im anderem Design wenn das flugzeugtypId in der Tabelle Fluege steht und mit der Buchungen durch flugId Referenciert ist sihet es dann so aus. Dass mit der Referenc fluegeId das Row zu Fluege übergegeben wird und von dort nur einmal auf einen bestimmten Flugzeugtyp zeigt so mit Spare ich eine ganze Spalte in der Tabelle Buchungen un so mit auch Speicherplatz.  
Die nechste Überlegung die aufmich zugeschossen kamm ist. Wenn ein Flug mehrere Passagiere haben könte und ein Passagier nicht in mehreren Flügen gleichzeitig sein kann aber sehr wohl eine Buchung für verschiedene Flüge tätigen mag. Muss in die Tabelle Buchungen eine Referenz von Passagieren eingetragen werden. So mit habe ich die Beziehungs kette geschlossen. Letztere Frage die ich mir gestelt habe „ *kann eine Buchung mehreren Passagieren angehören?“* also im Regel fall wird eine Reservierung immer auf einen Nammen ausgestellt und deshalb ist die Antwort nein kann ich nicht.   
Jetzt noch ein paar kleinichkeiten.  
Die Verlagerung und Anknüpfung der Tabelle Passagieren könnte man weglassen und in der Tabelle Buchungen die Datenfelder zugähnlich machen. Es ist aber mehr, eine Estetisch geprägte Frage als Logische.

###### Wichtige Hintergrund Information

- Numerische Daten die keine Schlüssel sind- also Flugdauer, Preis und Sitzzahl müssen Rechen fähig sein.

- Datumswerte sollen auch als Datumswerte in die Datenbank geschrieben werden.

- Ihr Programm muss auch in der Lage sein, weitere Arbeitsblätter - zum Beispiel

Buchungslisten aus spätere Perioden - korrekt in die Bestehende Datenbank zu

übernehmen; d.h. es muss auch dann korrekt arbeiten, wenn bestimmte Mastersätze

bereit in der Datenbank vorhanden sind.

# Tests

Bei dem Erstellen des Programms sind viele Testvorgänge nötig gewesen um die Logik und Datenintegrität auf die Richtigkeit zu prüfen.

# Abschluss

## Untersuchung der Aufgabe

# Datenstruktur

# Welche Kriterien sind für das DB-Design in dieser Aufgabe wichtig

Was ist Ausschlaggebend für die Strukturierung der Datenbank.  
Als die wichtigsten punkte sind es Benutzerfreundlich zu gestalten und keine Redundanz zu zulassen.

Verwendet man um die Anbieter der Flugdienste zu speichern. Aus gelagert in die einzelne Tabelle um eine Redundanz zu sichern.

***listFluggeselschaften(): fluggeselschaften [ ]***

Diese Methode ermöglicht einen schnellen zu griff auf bestimmte Gesellschaften zB. ein Kunde hat bestimmte Bevorzugungen.

# Glossar

|  |  |
| --- | --- |
| DB | Datenbank- dient zur elektronischen Datenverwaltung |
| CSV | steht für *Comma-separated values* beschreibt den Aufbau einer Testdatei.  Verwendbar um komplizierten Format in ein String zu formatieren. |
| UML | **Unified Modeling Language**, ist eine graphische Modellieungssprache zur Spezifikation, Konstruktion und Dokumentation der Anwendung. |
| API | Programmierschnittstelle |
| unique | eindeutige Identifizierungsnummer. Eine Ziffern- oder Zeichenkombination, mit der sich Gegenstände, Personen oder allgemein Entitäten innerhalb einer bestimmten Kategorie eindeutig identifizieren lassen |
| Quelltext | Unter dem Begrief wird in der Informatik der für Menschen lesbare, in einer Programmiersprache geschriebene Text eines Computerprogrammes verstanden. |
| ER - Modell | dient dazu, im Rahmen der semantischen Datenmodellierung einen relevanten Ausschnitt der realen Welt zu beschreiben. |
| Redundanz | *„überlaufen, im Überfluss vorhanden sein* „ bezeichnet allgemein einen Zustand von Überschneidung oder Überfluss im Sinne von Überschüssigkeit |
| Primary Key | Dient dazu, die Tupel (Datensätze) einer Relation (Tabelle) eindeutig zu identifizieren |
| Tupel | Ist ein Datensatz noch ein „Row“ genant, ein Zeile |
| boolean | Ist ein typ, eine *Schaltvariable* kann nur einen von wenigen Werten annehmen: |
| Apache POI | ist Open-Source-Software, die Java-Schnitstellen zum Lesen und Schreiben von Dateien im Dateiformat von Microsoft Office wie z. B. Word und Excel bereitstellt |
|  |  |

## Verwendete Software

* Navicat Premium – Datenbank Verwaltungssoftware
* *NetBeans IDE* – Für die Erstellung der Software
* Oracle Java SDK – Ist die Programmiersprache der Wahl
* *Oracle MySql* – Datenbank zum Speichern von Importierten Daten
* *Microsoft Excel* - Tabellenbearbeitungsprogramm für das Erstellen von Zeiterfassung und Glossary
* *Microsoft Word* – Dokumentationssoftware
* *Microsoft Paint* – Bildbearbeitung Software