Bericht DBN Projekt: Lieferdienst von Gürdal Demir und Paul Voß

Inhaltsverzeichnis:

Jeder Gedankenprozess und jede Umsetzung vollständig und ausschließlich durch Paul Voß.

1.) Projektbeschreibung

Das Ziel dieses Projektes, war es ein Auswertetool für einen Lieferdienst zu entwerfen. Dieses Tool soll von dem Kunden (dem Lieferdienst) dazu verwendet werden eine wirtschaftliche aussagekräftige Analyse zu dem Betrieb bereitzustellen. Gleichzeitig soll eine Kundenfrequenzanalyse möglich sein.

Warum dieses Projekt?

Ich habe mich für dieses Projekt entschieden, da die Themen E-Commerce und Data Warehouses in unserer modernen Zeit immer mehr an Relevanz und Wichtigkeit gewinnen. Daher habe ich die Chance gesehen mich erstmalig mit diesen spannenden Themen beschäftigen zu können.

2.) Anforderungen an das System

Das System wird, ausgehend von der Aufgabenstellung, von drei zentralen Benutzergruppen verwendet. Dem Betreiber, welcher überwiegend die Wirtschaftlichkeit des Betriebes und eine Analyse der Arbeit seiner Mitarbeiter sehen möchte. Den Mitarbeitern, bei denen es sich generell um zwei verschiedene handelt. Eine Gruppe Mitarbeiter nimmt telefonisch Bestellungen auf und die andere gruppe liefert diese dem Kunden. Zu Letzt gibt es da noch die Buchhaltung, die natürlich auch Mitarbeiter des Betriebs sind, jedoch durch ihre Verantwortlichkeit für alles finanzielle einige „Sonderrechte“ erhalten.

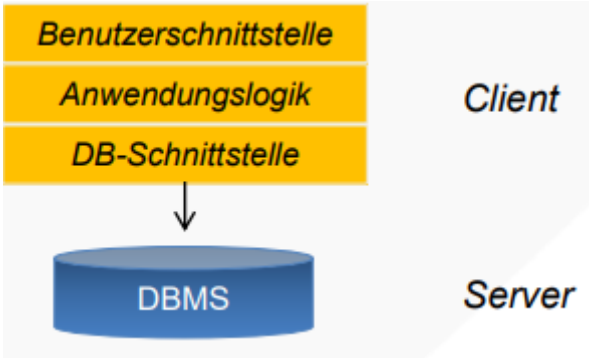
2.1) Was möchte wer sehen / eingeben?

Eingaben = …

	Betreiber	Buchhaltung	Mitarbeiter
Fahrten pro Schicht	X		
Umsatz / Gewinn (Zeitpunkt /-punkt)		X	
Fahrzeugbestand		X	
Gefahrene Touren			X
Km / Schicht (Auslastung)	X		X
Kasse der Fahrer			X
Fixkosten	X	X	
Kudenzahlen / MA	X		X
Neue MA / Fahrzeuge		X	
Kundenzahlen nach Zeitraum /-punkt	X		
Gesamtdaten nach Preisklasse	X	X	

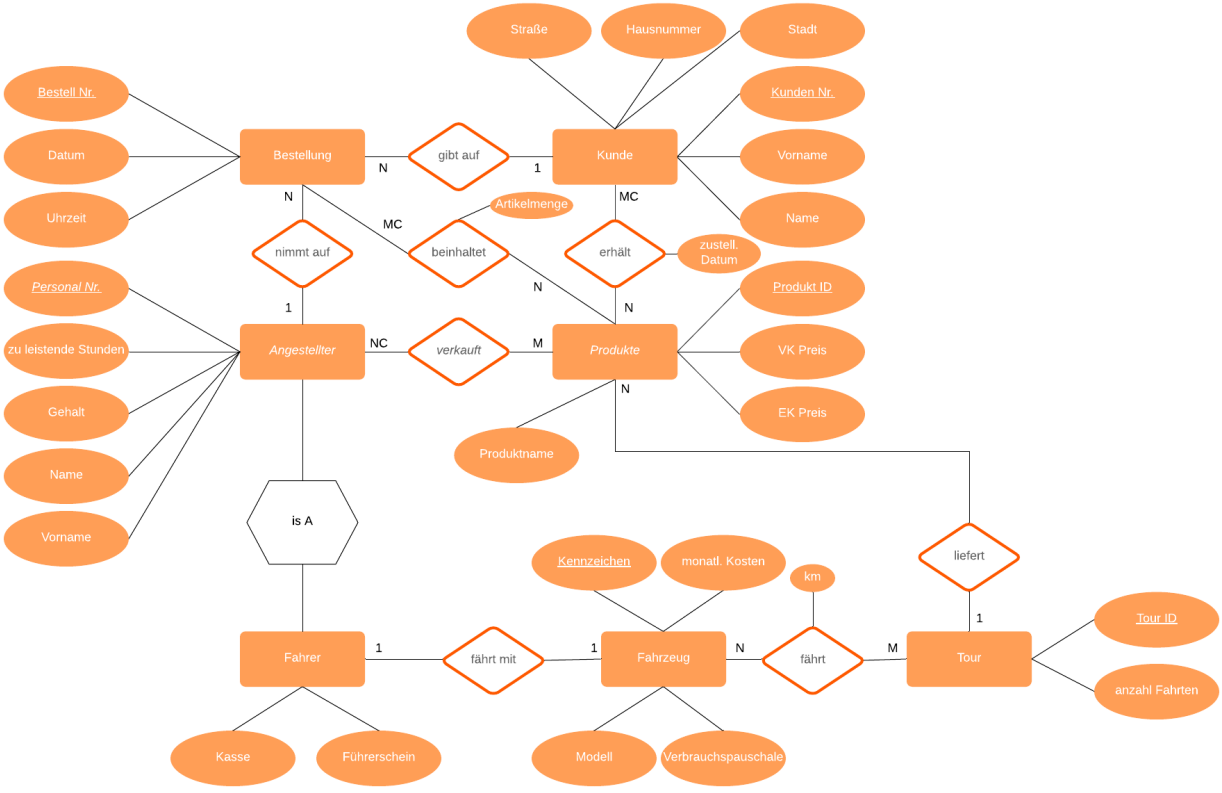
3.) Architektur

Grundsätzlich unterscheidet man zwischen der Zwei- und Drei-Schichten-Architektur. Jede dieser beiden Arten bietet abhängig von den Gegebenheiten und Anforderung des Projektes diverse Vor- und Nachteile. Für dieses Projekt habe ich mich für die Zwei-Schichten-Architektur entschieden. Im Detail bedeutet das, dass wir unsere Anwendungslogik auf den einzelnen Clients (in diesem Fall Computer der Mitarbeiter des Lieferdienstes) ablegen und diese nicht wie in der Drei-Schichten-Architektur auf einen separaten Server Auslagern. Diese Clients kommunizieren dann direkt mit dem Server, auf dem unser DBMS läuft, und können so die Datenbestände direkt abfragen und / oder verändern. Diese Architektur bietet den großen Vorteil, dass wir uns einen extra Server und dessen Konfiguration ersparen, dieser wird erst benötigt in sehr großen Unternehmen, in denen viele hunderte/tausende abfragen an einem Tag an den Server gestellt werden.



Die Zwei-Schichten-Architektur

4.) ER – Modell



Wie bereits angedeutet haben wir neben der Buchhaltung zwei Gruppen von Mitarbeitern, dies wird durch die Generalisierung abgebildet. An den Beziehungen „erhält“, „beinhaltet“ und „fährt“ befinden sich jeweils Attribute, die die Relation genauer beschreiben. Diese Attribute werden nachher in der „Beziehungstabelle“, welche entsteht da es sich jeweils um N zu M Beziehungen handelt, abgebildet. Bei der Erstellung des Modells wurde darauf geachtet, dass die Daten atomar vorliegen.

5.) Tabellenstruktur

	Tabelle
☐	angestellter
☐	bestellung
☐	bestellzuordnung
☐	fahrer
☐	fahrt
☐	kunde
☐	produkte
☐	tour
☐	verkäufe
☐	wareneinsatzkosten
☐	zustellung
	11 Tabellen

Gesamtübersicht der Tabellen

Personal_Nr.	zu leistende Stunden	Gehalt	Vorname	Name	Kundenzahlen
100	40.00	2400	Thomas	Mueller	*NULL*
101	20.00	435	Tom	Schmidt	*NULL*
102	25.00	1700	Jakob	Jakobi	8
103	20.00	900	Harry	Potter	*NULL*
110	15.00	180	Fabian	Zappel	*NULL*

Tabelle angestellter

Diese Tabelle ist lediglich eine Übersetzung aus dem ER – Modell. Die Kundenzahlen können nur für Telefonisten eingegeben werden.

Bestell_Nr.	Datum	Uhrzeit	Kunden_Nr.	Personal_Nr.
885	2019-01-22	12:34:39	503	103
886	2019-05-16	09:15:28	502	102
887	2019-12-23	23:54:59	501	101
888	2019-03-19	12:43:15	500	100

Tabelle bestellung

Die Tabelle Bestellung erhält neben den Attributen aus dem ER – Modell ebenfalls die Fremdschlüssel *„Kunden_Nr.“* und *„Personal_Nr.“,* um nachzuvollziehen welcher Kunde die Bestellung getätigt hat und welcher Mitarbeiter diese aufgenommen hat.

Bestell_Nr.	Produkt_ID	Artikelmenge
885	5556	4
885	5555	2
886	5558	1
886	5557	2
887	5555	100
887	5558	100
887	5557	100

Tabelle bestellzuordnung

Diese Tabelle resultiert aus der N zu MC Relation zwischen den Entitäten Produkte und Bestellung. Sie erhält *„Bestell_Nr.“* und *„Produkt_ID“* als Fremdschlüssel und das Beziehungsattribut *„Artikelmenge“*.

Personal_Nr.	Kasse	Führerschein	Modell	Verbrauchspauschale	monatl. Kosten	Kennzeichen
103	253	1	Sprinter	60	253	KI-E44
100	300	1	Polo	30	600	KI-VP111
101	57	1	Sprinter	60	253	KI-VP112
110	74	1	Micra	7	325	KI-VP113

Tabelle fahrer

Diese Tabelle dient der Identifizierung von ausliefernden Mitarbeitern. Das Feld Führerschein ist ein Boolean, welches in MySQL durch 0 oder 1 befüllt wird. Da es sich zwischen den Entitäten Fahrer und Fahrzeug um eine 1 zu 1 Beziehung handelt, finden sich in dieser Tabelle auch Informationen zu den Fahrzeugen, welche durch die bestimmten Mitarbeiter gefahren werden. Die Spalte Verbrauchspauschale gibt die Kosten pro Kilometer in Cent an.

Kennzeichen	Tour_ID	km
KI-VP111	4444	48
KI-E44	4445	18

Tabelle fahrt

Diese Tabelle beschreibt die N zu M Relation zwischen Fahrzeug und Tour. Des Weiteren enthält sie das Beziehungsattribut *„km“*.

Kunden_Nr.	Straße	Hausnummer	Stadt	Name	Vorname
500	Sesamstraße	24a	Kiel	Müller	Mark
501	Rosenstraße	182	Kiel	Klein	Petra
502	Projensdorfer Straße	123	Kiel	Thimm	Manuela
503	Langer Rehm	20e	Heikendorf	Mathiesen	Leon

Tabelle kunde

Diese Tabelle ist lediglich eine Übersetzung aus dem ER – Modell.

Produkt_ID	VK_Preis	EK_Preis	Produktname	Tour_ID
5555	10	6	Salami Pizza	4444
5556	15	10	Bohnen	4444
5557	19	12	Salsa	4445
5558	20	12	Chips	4446

Tabelle produkte

Die Tabelle enthält zusätzlich zu den normalen Attributen die Spalte „*Tour_ID*“, welches ein Fremdschlüssel aus der Tabelle tour ist und beschreibt in welcher Tour die Produkte ausgeliefert wurden.

Tour_ID	anzahl Fahrten
4444	4
4445	6
4446	2
4447	9

Tabelle tour

Diese Tabelle ist lediglich eine Übersetzung aus dem ER – Modell.

Personal_Nr.	Produkt_ID
101	5556
102	5558
101	5555
102	5557

Tabelle verkäufe

Diese Tabelle resultiert aus der NC zu M Relation zwischen Angestellter und Produkte. Sie enthält die „*Produkt_ID*“ des Verkauften Produktes und die „*Personal_Nr.“* des Mitarbeiters, der das Produkt verkauft hat.

Produkt_ID	Kunden_Nr.	zustell. Datum
5556	503	2019-01-24
5555	503	2019-01-24
5558	502	2019-05-18
5557	502	2019-05-18

Tabelle zustellung

Diese tabelle resultiert aus der N zu MC Relation zwischen Produkte und Kunde. Sie enthält die „*Produkt_ID“* des gelieferten Produktes, die „*Kunden_Nr.“* des belieferten Kunden und das Datum der Zustellung, welches ein Beziehungsattribut ist.

6.) Typische SQL – Abfragen

Die angegebenen Parameter sind beispielhaft und können natürlich variieren.

Kundenzahlen nach Zeitpunkt:

```
SELECT `Bestell_Nr.` from bestellung WHERE DATE(Datum) = '2019-01-22';
```

Umsatz an einem Tag:

```
SELECT SUM((VK_Preis - EK_Preis) * Artikelmenge) from bestellung join bestellzuordnung using(`Bestell_Nr.`) join produkte using(`Produkt_ID`) where DATE(Datum) = '2019-01-22';
```

Gewinn im Dezember:

```
SELECT SUM((VK_Preis - EK_Preis) * Artikelmenge) - Gehalt from bestellung join bestellzuordnung using(`Bestell_Nr.`) join produkte using(`Produkt_ID`) join angestellter where DATE(Datum) between '2019-12-01' and '2019-12-30';
```

Umsatz 2019:

```
SELECT SUM((VK_Preis - EK_Preis) * Artikelmenge) from bestellung join bestellzuordnung using(`Bestell_Nr.`) join produkte using(`Produkt_ID`) where DATE(Datum) between '2019-01-01' and '2019-12-30';
```

Fahrten der Mitarbeiter:

```
SELECT Vorname, Name, `anzahl Fahrten` from angestellter join fahrer using(`Personal_Nr.`) join fahrt using(`Kennzeichen`) join tour using(`Tour_ID`);
```

Fahrzeugbestand

```
SELECT `Modell`,`Kennzeichen`,`Verbrauchspauschale` from fahrer;
```

Mitarbeiter Infos

```
SELECT CONCAT(Vorname, " ", Name), Gehalt, `zu leistende Stunden` from angestellter;
```

CONCAT() Funktion verwendet um für den Anwender angenehmeren Output zu generieren.

Anlegen neuer Mitarbeiter:

```
INSERT INTO `angestellter`(`Personal_Nr.`, `zu leistende Stunden`, `Gehalt`, `Vorname`, `Name`, `Kundenzahlen`) VALUES (110,15,180,'Fabian','Zappel',NULL);
```

Anlegen neues Fahrzeug:

```
INSERT INTO `fahrer`(`Personal_Nr.`, `Kasse`, `Führerschein`, `Modell`, `Verbrauchspauschale`, `monatl. Kosten`, `Kennzeichen`) VALUES (110,74,1,'Micra',7,325,'KI-VP113');
```

Eingabe Fixkosten:

```
INSERT INTO `wareneinsatzkosten`(`Miete`, `Gehaelter`, `Zinsen`) VALUES (1400,4000,325);
```

Eingabe Kilometer einer Fahrt:

```
UPDATE `fahrt` SET `km`=48 WHERE `Kennzeichen`='KI-VP111';
```

Eingabe Kundenzahlen:

```
UPDATE `angestellter` SET `Kundenzahlen`=4 WHERE `Personal_Nr.`=102;
```

Gesamtdatenbestand, Verkaufspreise zwischen 5€ und 15€:

```
SELECT * FROM `produkte` WHERE `VK_Preis` between 5 and 15;
```

Info über Fahrten/km/Kasse:

```
SELECT Vorname, Name, km, `anzahl Fahrten`, Kasse from angestellter join fahrer
using(`Personal_Nr.`) join fahrt using(`Kennzeichen`) join tour using(`Tour_ID`);
```

Bei den JOIN Operationen benutze ich das Keyword using. Dieses hat den exakt selben Effekt wie die Verwendung mit ON, es ist lediglich eine verkürzende Schreibweise, die **nur** dann verwendetet werden kann, wenn die Spalten in den zwei Tabellen **identische** Namen haben.