GRUNDLAGEN DATENORGANISATION

INHALT ZUSAMENGETRAGEN VON: MICHAEL LINDNER & MANDY LUDAT
MICHAEL.LINDNER@MTH-IT-SERVICE.COM

INHALTSVERZEICHNIS

1. DATENORGANISATION

1.1 EINLEITUNG

Die logische Datenorganisation beschreibt Daten(Objekte) durch Eigenschaften und ihre Beziehungen.

Objekte sind zum Beispiel: Rechnungen, Stornierungen, Teilnehmer

Ziel der digitalen Datenorganisation ist es:

Analoge Prozesse und Daten

- zentral zu erfassen und darauf zu zugreifen
- schneller durchsuchbar zu machen
- mehreren Nutzern gleichzeitig zu Verfügung zu stellen
- räumlichen Platz zu sparen

1.1.1 ANALOGE DATENBANKEN

Zu Zeiten von analogen Datenbanken musste noch mehr Wert auf eine logische und physische Gliederung der Informationen gelegt werden, damit man die Daten im Anschluss schneller finden kann.

Abbildung 1 -Karteisystem

1.1.2 DIGITALE DATENBANKEN

Die Informationen die auf Papier existierten bzw. existieren werden für digitale Datenbanken aufbereitet. Diesen Vorgang nennt man Digitalisieren. Gründe für die Digitalisierung wurden in der Einleitung benannt.

1.1.3 VERGLEICH DER ANALOGER ZUR DIGITALEN DATENBANK

Im Folgenden werden die wichtigsten Gründe für den Einsatz einer Datenbank mal gegenübergestellt.

	Analog	**Digital**
Schnelle Suche	Durch manuelle Suchvorgänge (suche im Inhaltsverzeichnis)	Durch mathematische Algorithmen, Inhaltsverzeichnisse, Suchmatritzen Viel schneller gegenüber der menschlichen Suche.
Platzsparend	Nein, weil jede Information einen großen physischen Träger benötigt	Ja, weil viele Information geringen Platz beanspruchen(Festplatte)
Umwelteinfluss	Hoher Rohstoffeinsatz. Dateischränke aus Metall, Informationen auf Papier Energieaufwand für die Herstellung der Informationsträger Restauration alter Informationsträger	Mittlerer Rohstoffeinsatz, Energieaufwand für das Speichern und Benutzen der Daten

1.2 GRUNDLAGEN DER DATENORGANISATION

1.2.1 EINHEITEN DER DATENORGANISATION

Zeichen – 1Byte *Speicherverbrauch*

Das Zeichen ist die kleinste Einheit der Datenorganisation. Mit einem Zeichen können Buchstaben, Zahlen oder Sonderzeichen(Binärdaten) abgebildet werden.

Feld

Mehrere Zeichen werden zu einen Feld zusammengefasst. Für den Begriff Feld gibt es unterschiedliche Assoziationen: Attribut, Eigenschaft, Element, Field, Value…

In der objektorientierten Ansicht wird der Begriff „Feld" als „Eigenschaft" bezeichnet.

Datensatz

Ein Datensatz ist eine inhaltliche Zusammenfassung von Feldern (Adressdaten: Nachname, Vorname, Straße …)

Datensätze können auch als Zeilen, Recordsets, Tubel bezeichnet werden.

In der objektorientierten Ansicht wird der Begriff „Datensatz" als „Objekt" oder „Entität" bezeichnet.

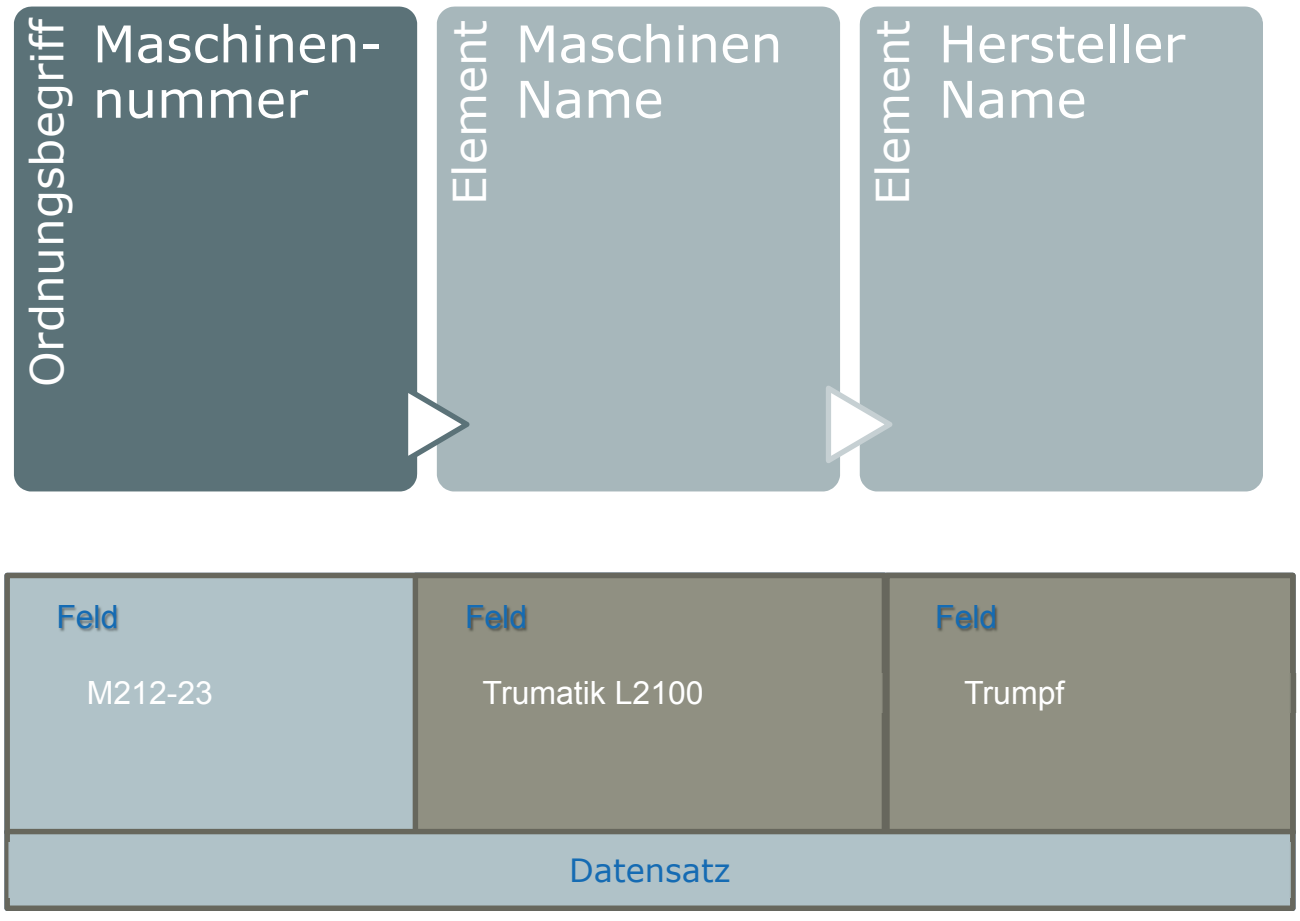
Datei

Eine Datei ist die inhaltliche Zusammenfassung mehrerer Datensätze gleicher Struktur. Eine Datei kann auch als Tabelle oder Container bezeichnet werden.

Datenbank

Eine Datenbank ist die inhaltliche Zusammenfassung mehrerer Dateien zu einem realen Thema (Buchhaltung, Qualitätssicherung, Intranet)

1.2.2 Aufbau eines Datensatz



1.2.7 KONSISTENZ VON DATENBANKEN

Die Konsistenz einer Datenbank beschreibt den funktionsfähigen Zustand einer Datenbank. Ist eine Datenbank Inkonsistent so steht fest dass die Daten nicht nach den vorgegebenen Regeln oder Planungen vorliegen.

1.2.8 DATENMODELLE

- Hierarchisches Datenbankmodell
- Netzwerkdatenbankmodell
- Relationales Datenbankmodell
- Objektrelationales Datenbankmodell
- Objektorientiertes Datenbankmodell
- Indexbasiertes Datenbankmodell

1.3 ENTWURF VON RELATIONALEN DATENBANKEN

1.3.1 ZIELSTELLUNG

Der Entwurf von relationalen Datenbanken verfolgt das Ziel einen Nachweis (Planungsdokument)über die Planung, den Inhalt, die Struktur, die Pflege , den Platzbedarf und die Auswahl der Datenbank zu erstellen.

1.3.2 KARDINALITÄTEN

Die Kardinalitäten (im Bereich Datenorganisation) stellen die Mengen der Datensätze(Objekte) die in **einer** Beziehung gegenüber.

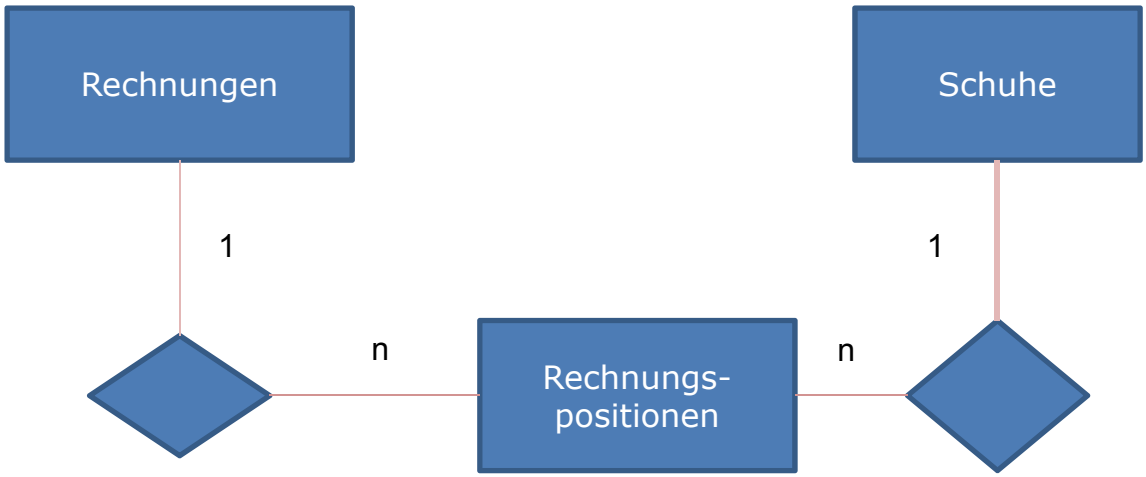
Folgende Beziehungsformen gibt es:

1:1 - ein Objekt auf der einen Seite steht genau mit einem Objekt auf der anderen Seite direkt in Beziehung (Ehe(Deutschland ☺) . Alles andere würde eine Inkonsistenz bedeuten.

1:n - ein Objekt steht mit mehreren anderen Objekten in Beziehung

n:m - mehrere Objekte stehen mit mehreren Objekten in Beziehung.

Diese Kardinalität wird bei der Überarbeitung des Entwurfes einer relationalen Datenbank aufgelöst. Eine Mehrfachbeziehung im echten Leben ist zwar möglich, kann aber durch das relationale Modell nicht abgebildet werden. Die Auflösung erfolgt durch Erstellung einer neuen Entität(Objekt) und zweier neuen „1:n" Beziehungen. Hierbei liegt die „n – Menge" am neuen Objekt an.



Beispiel der Auflösung einer m:m Beziehung

1.3.3 *Entity Relationship Modell*

Das Entity Relationship Modell (Objekt Beziehungs Modell) beschreibt mit welchen Darstellungsmitteln der Aufbau einer relationalen Datenbank beschrieben werden kann.

Das Produkt dieses Entwurfes ist ein ERD (Entity Relationship Diagramm)

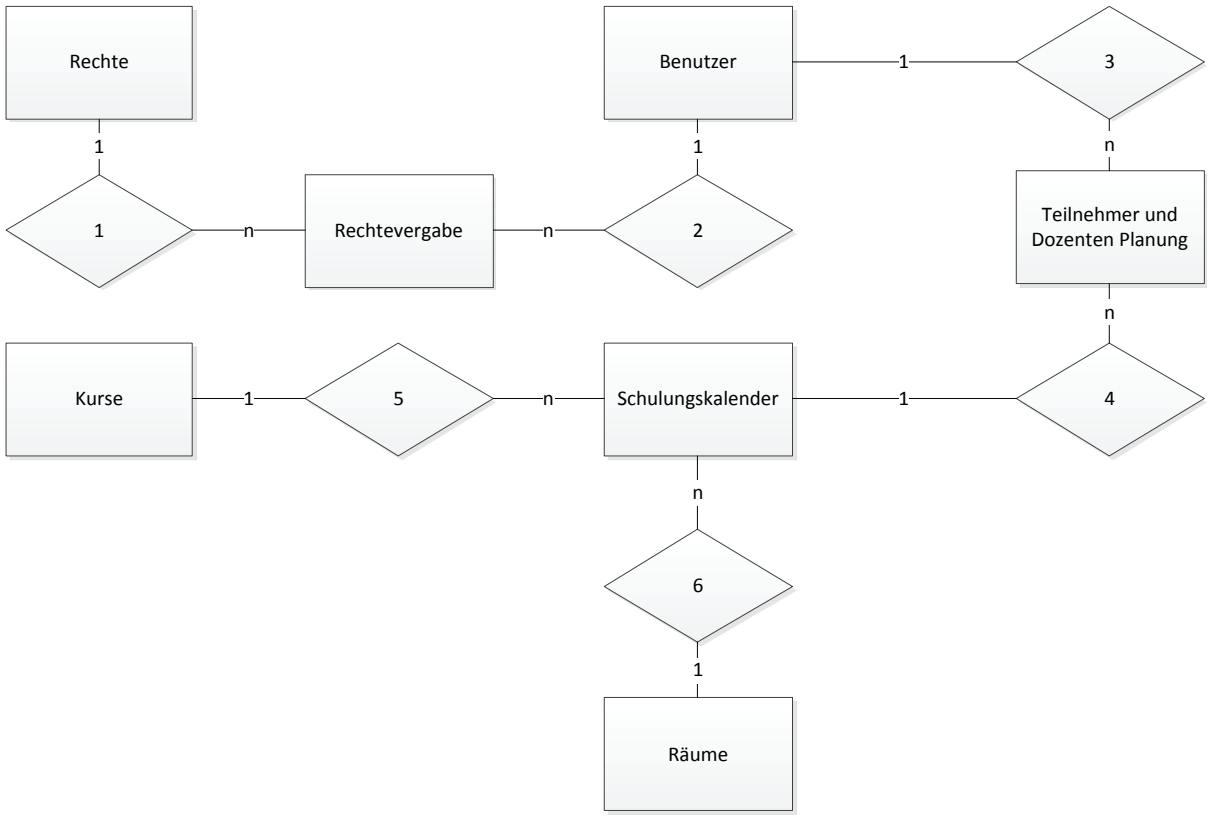
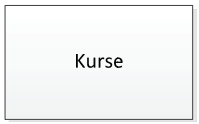


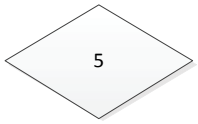
Abbildung 2 - Beispiel **eines ERD für eine Lehrgangsreservierung**

Darstellungsmittel:

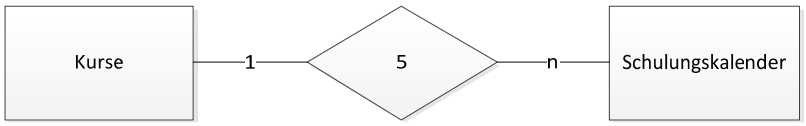
Objekt (Tabelle):



Beziehung (Relation):



Kardinalität (1:n):



1.3.4 REFERENTIELLE INTEGRITÄT

Die referenzielle Integrität ist die Durchsetzung der Beziehungsregeln. Ist die Integrität einer Beziehung aktiv können keine inkonsistente Datensätze angelegt werden die gegen die Kordialitäten verstoßen.

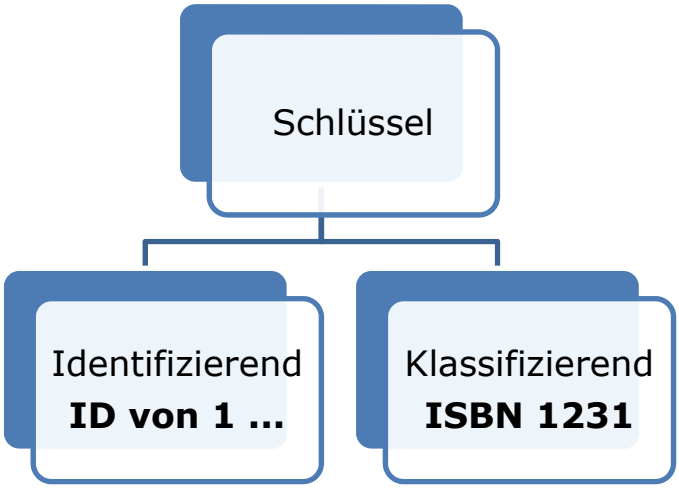
1.3.5 ANOMALIEN

Die referenzielle Integrität bestimmt zum Beispiel dass das Löschen eines Datensatzes in der Tabelle „Schuhartikel" nicht möglich ist, solange es Rechnungen zu diesem Objekt gibt.

Eine Anomalie (Abweichung) kann festlegen dass ein Datensatz in „Schuhartikel" trotz Regel gelöscht werden kann. Allerdings löscht es durch die möglich festgelegte Löschanomalie alle verbundenen Datensätze automatisch.

1.3.6 SCHLÜSSELTYPEN

Im Bereich der Datenorganisation werden Schlüssel als Ordnungsmittel benutzt um Daten eindeutig zu erkennen. Wir unterscheiden klassifizierende und identifizierende Schlüssel. Klassifizierende Schlüssel dienen dazu Aussagen über den Inhalt zu machen ohne alle Datenfelder zu sehen. Sie sind allerding nicht so schnell find bar wie identifizierende Schlüssel bei denen wir aber keine Aussage über den Inhalt machen können. Am meisten verbreitet ist der identifizierende Schlüssel.



1.3.7 SCHLÜSSELFUNKTIONEN

Primärschlüssel (PK- Primary Key)

Als Primärschlüssel werden alle Felder bezeichnet die einen Datensatz eineindeutig identifizieren. Einen Primärschlüssel ist einzigartig in seinem vorkommen.

Fremdschlüssel (FK- Foreign Key)

Ein Fremdschlüssel (Verweisschlüssel) dient zur Abbildung einer Beziehung zwischen zwei Objekten. Dabei verweist ein Fremdschlüssel auf Felder eines anderen Objektes in dem es eine Kopie des Schlüsselwertes aus der Primärschlüsseltabelle enthält.

Sekundärschlüssel

Bezeichnet man alle Teilschlüssel die einen Primär-oder Fremdschlüssel bilden.

1.3.8 NORMALISIERUNG

Normalisierung dient zur Redundanzverhinderung von Daten. Sekundär trägt die Normalisierung zur Verringerung des Speicherplatzbedarfes bei.

Mancchmal kann Normalisierung aber ein Zeit Problem bei Abfragen erzeugen. Dann kann auch eine Denormalisierung eine Lösung sein.

1. **Normalform:**

    - Eine Relation befindet sich in der ersten Normalform, wenn alle Attribute nur einfache Attributwerte aufweisen (Bezeichnung: atomar).

3. **Normalform:**

    - Erste Normalform muss vorliegen
    - Wenn alle Nichtschlüsselattribute voll von allen Schlüsselattributen abhängig sind

3. **Normalform**

    - Die zweite Normalform muss vorliegen
    - Nichtschlüsselattribute dürfen nicht mit andern Nichtschlüsselattributen in Beziehung stehen (transitive Abhängigkeit)

1.3.8.1 Beispiele von Tabellen

Ein Gebrauchtwagenhändler verkauft Autos sämtlicher Hersteller. Damit eine relationale Datenbank erstellt werden kann, fordern wir einige Tabellen mit zugehörigen Datensätzen ein. Es gilt diese nach den 3 Normalformen zu prüfen und gegebenenfalls zu ändern.

Folgende Listen wurden vom Kunden für die Ermittlung der Datenbank abgegeben

Verkäuferliste

Verkäufernummer	Name
V-2012-1	Renner, Thomas
V-2011-1	Tomschke, Sandra

Kundenliste

Kunden-nummer	Nach-name	Vor-name	Adr.-Nummer	Adresse	PLZ	ORT
K1001	Lindner	Michael	23	Grenzweg 18	01917	Kamenz
K1002	Hubert	Hans	12	Mühlweg 2	01900	Großröhrs-dorf
K1275	Müller	Uwe	23	Grenzweg 18	01917	Kamenz

Ausstattung

Fahrzeug-nummer	Farbe	Sport-lenkrad	Leder-sitze	Kraftstoff	Schaltung	Kilometer-stand
1334-2445-56556	blau	ja	Nein	Diesel	Automatik	10000
3435-5656-57675	rot	nein	ja	Benzin	Gang	39000

Rechnung

Rechnungs-nummer	Rechnungs-position	Fahrzeug-nummer	Verkäufer-nummer	Typ	Hersteller	Kunden-nummer
RN-562223	P-1	1334-2445-56556	V-2012-1	A6-Avant 2.8	Audi	K1192
RN-562223	P-2	5389-2346-78335	V-2012-1	Niva 4.0	Lada	K1047
RN-763511	P-1	3435-5656-57675	V-2011-1	Octavia	Skoda	K1148
RN-945624	P-1	1334-2445-56556	V-2013-5	A6-Avant 2.8	Audi	45271

Nach der Prüfung der 3 Normalformen bekommen wir folgende Tabellen als Ergebnis: (*=Primary Key)

Verkäuferliste

Verkäufernummer (*)	Nachname	Vorname
V-2012-1	Renner	Thomas
V-2011-1	Tomschke	Sandra

Kundenliste

Kundennummer (*)	Nachname	Vorname	Adr.-Nummer
K1001	Lindner	Michael	23
K1002	Hubert	Hans	12
K1275	Müller	Uwe	23

Adressliste

Adr.-Nummer (*)	Adresse	PLZ	ORT
23	Grenzweg 18	01917	Kamenz
12	Mühlweg 2	01900	Großröhrsdorf

Ausstattung

Fahrzeug-nummer (*)	Farbe	Sport-lenkrad	Leder-sitze	Kraft-stoff	Schaltung	Kilometer-stand
1334-2445-56556	blau	ja	Nein	Diesel	Automatik	10000
3435-5656-57675	rot	nein	ja	Benzin	Gang	39000

Rechnung

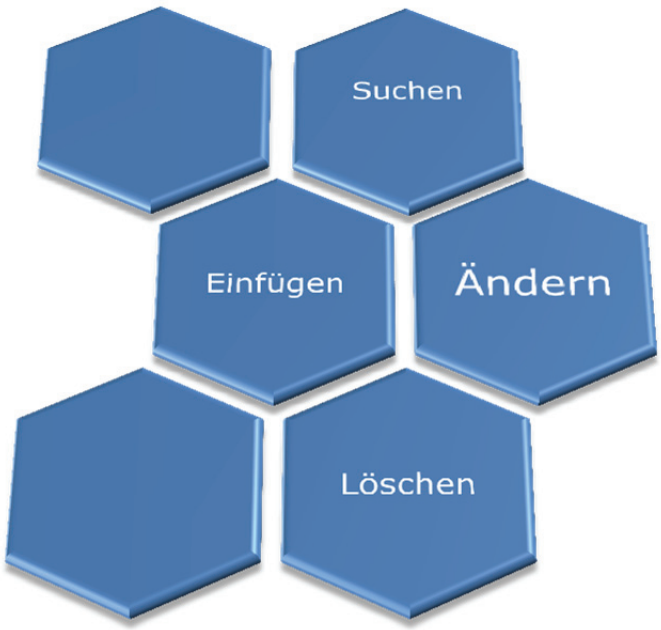
Rechnungs-nummer (*)	Rechnungs-position	Fahrzeugnummer	Verkäufer-nummer	Kundennummer
RN-562223	P-1	1334-2445-56556	V-2012-1	K1192
RN-562223	P-2	5389-2346-78335	V-2012-1	K1047
RN-763511	P-1	3435-5656-57675	V-2011-1	K1148
RN-945624	P-1	1334-2445-56556	V-2013-5	45271

Fahrzeugliste

Fahrzeugnummer (*)	Typ	Hersteller
1334-2445-56556	A6-Avant 2.8	Audi
3435-5656-57675	Niva 4.0	Lada

1.3.9 DATENFUNKTIONEN

Innerhalb der Datenorganisation werden folgende Funktionen berücksichtigt:



1.3.10 DATENEINSTUFUNG

Die Dateneinstufung in folgende Kategorien dient zur Einschätzung für den Sicherungsaufwand und Platzbedarf.

- **Stamm- und Bestandsdaten beschreiben Zustände**
- **Bewegungs- und Änderungsdaten beschreiben Ereignisse**

1.4 KONZEPTIONELLER ENTWURF

Ziel des konzeptionellen Entwurfes ist die Feststellung aller beteiligten Objekte und deren Beziehung. Als Darstellungsmittel kommt z.B.: das ERD in Frage.

1.4.1 BEISPIEL FÜR EIN ERD

Anhand der nach den 3 Normalformen erstellten Tabellen lassen sich nun die Objekte für das ERD im Beispiel des Autohändlers erstellen. Wichtige Informationen hierfür:

- Jedes Fahrzeug kann durch Ankauf mehrmals verkauft werden
- Wenn ein Fahrzeug angekauft wird, muss die aktuelle Ausstattung hinzugefügt und neu angelegt werden
- Sobald das Fahrzeug verkauft wird, wird die gespeicherte Ausstattung verworfen
- Auf einer Rechnung können mehrere Autos verkauft werden

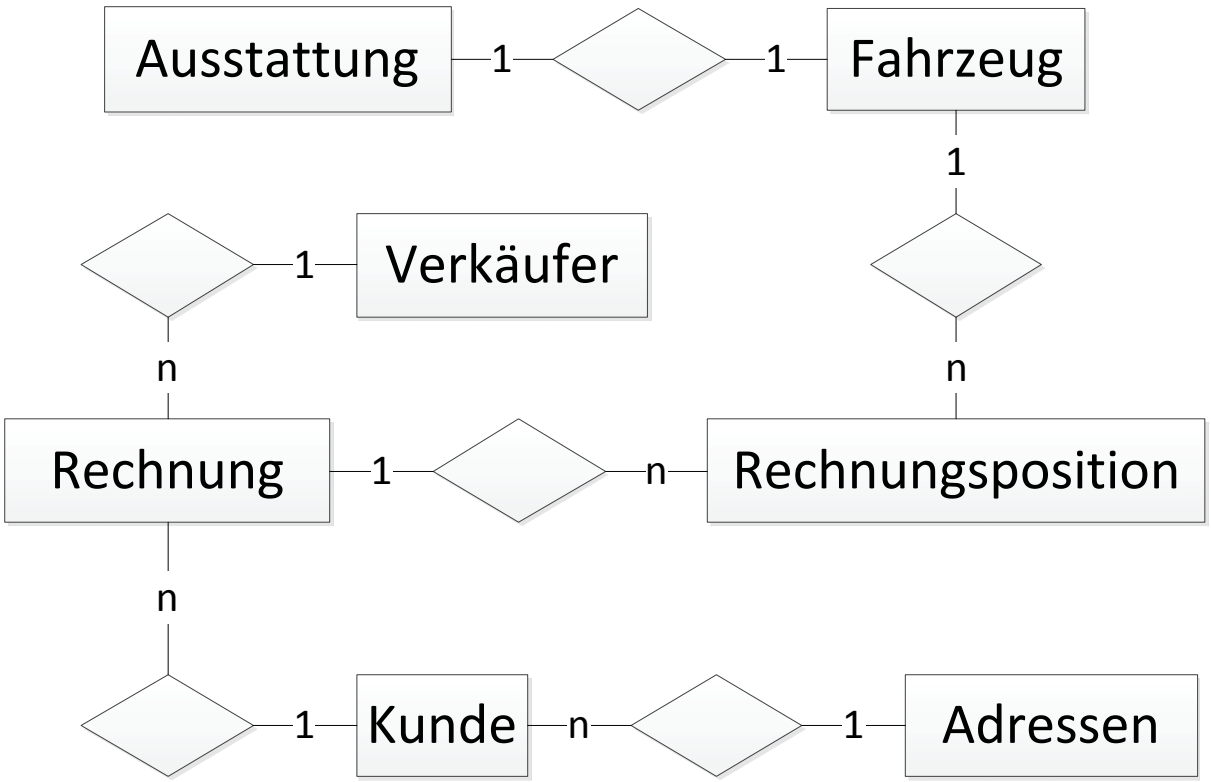


Abbildung 7 - Beispiel eines ERD für den Verkauf von Fahrzeugen von einem Gebrauchtwagenhändler

1.5 Logischer Entwurf

Der logische Entwurf legt die Felder und deren Datentypen (nummerisch, alphanummerisch, binär) fest. Die Datentypen dienen zur Umsetzung von:

- schnellen Zugriffen
- Speicherplatzersparnissen
- Bestimmung der Sortierreihenfolge

Nach der Festlegung der Datentypen werden die Dimensionen (Größen der Felder bestimmt um mit den analysierten Stückzahlen der Datensätze einen Platzbedarf feststellen zu können.

Berücksichtigen Sie den Faktor 2 für Archivdaten und noch einmal den Faktor 2 für Indexierung.

Jetzt können Sie Ihre Datenbank auswählen.

Sie unterscheiden hier zwischen „FlatFile" Datenbanken (Single User) oder „SOA"(Multi User) Datenbanken.

Kosten Nutzensind bei der Wahl der richtigen relationalen Datenbank nicht außer Acht zu lassen.

1.5.1 Beispiel eines Datenbankmodelldiagramms

Das folgende Datenbankmodelldiagramm bezieht sich auf das bereits erstellte ERD und auf die Tabellen des Autohändlers:

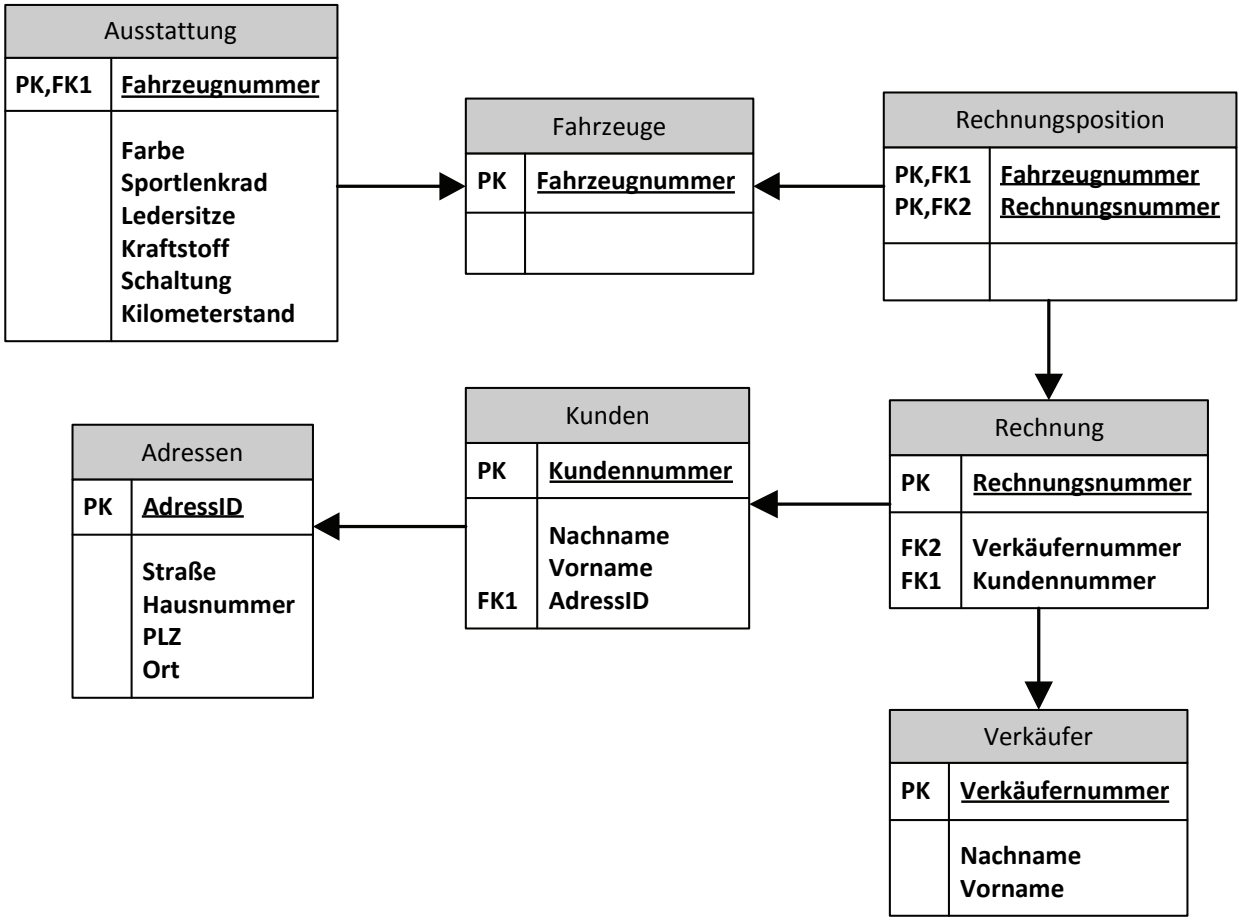


Abbildung 8 - Beispiel eines Datenbankmodelldiagramms für den Verkauf von Fahrzeugen von einem Gebrauchtwagenhändler

1.6 Physikalischer Entwurf

Haben Sie den Hersteller und die Datenbank gewählt z.B.:

- Microsoft SQL Server
- Oracle DB
- Oracle MYSQL
- Microsoft Access

Die im logischen Entwurf ermittelten Datentypen werden nun in den passenden Datentyp des Datenbankherstellers (Zahl in integer) umgewandelt und Dokumentiert.

1.7 Migrationsentwurf

Der Migrationsentwurf stellt den Weg der Installation einer Datenbank dar. Es muss mit dieser Anleitung der Startzustand einer Datenbank erreicht werden.

1.8 Wartungs- und Serviceentwurf

Je nach Kategorisierung der Daten wird hier ein Sicherungszeitplan erstellt. Es wird festgelegt Wann und Wie viel gesichert wird.

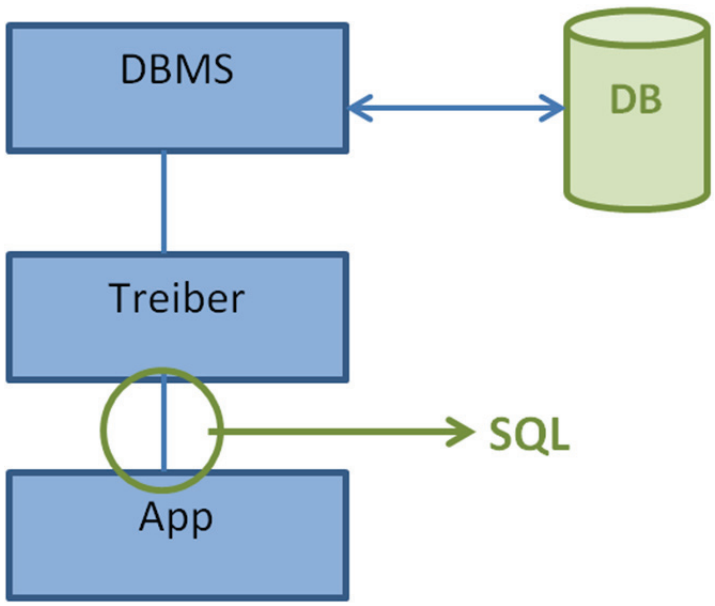
Eventuelle Hinweise zum Mehr Speicherbedarf müssen hier formuliert werden.

2. SQL-Abfragen

2.1 Was ist SQL?

SQL (=Structured Query Language) ist eine Datenbanksprache zur Definition von Datenstrukturen in relationalen Datenbanken sowie zum Bearbeiten (Einfügen, Verändern, Löschen) und Abfragen von darauf basierenden Datenbeständen.

Wo kommt SQL zum Einsatz?



2.2 Sprachkategorien

DDL – Data Definition Language

- *Strukturen anlegen, löschen, ändern*

  ↓

- *Datenbanken, Tabellen, Sichten, …*
- *Befehle: create, alter, drop*

DML – Data Manipulation Language

- *Hinzufügen, löschen und ändern der (Roh-)Daten*
- *Befehle: insert, delete, update*

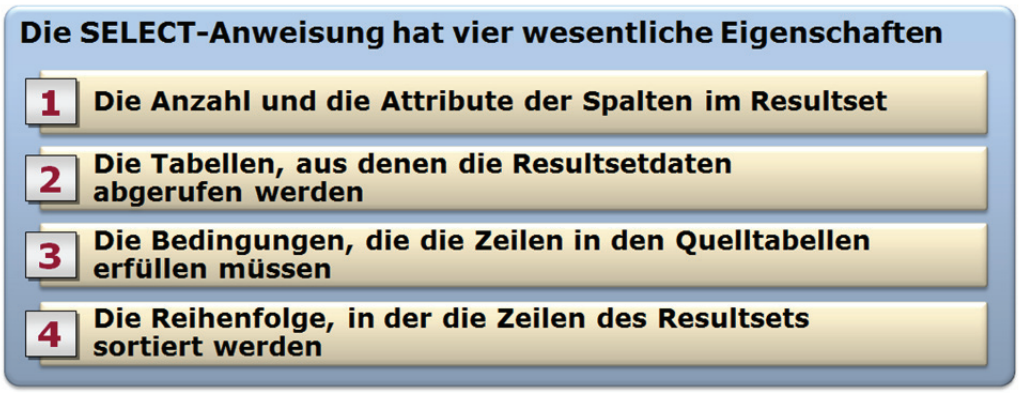
DQL – Data Query Language

- *Datenabfrage*
- *Holt Daten aus Datenbank*
- *Befehle: Select, Set*

2.3 Syntax

Grundsätzlicher Aufbau einer SQL-Abfrage

1- Was?	**Select**
2- Woher?	**from**
3- Bedingung	**where**
4- Sortierung	**order by**




```
SELECT ProductID, Name, ListPrice
FROM Production.Product
WHERE ListPrice > $40
ORDER BY ListPrice ASC
```

2.4 Beispiele für DDL

Beispiel ist unsere Datenbank „Autoverkauf"

- Datenbank anlegen:

  CREATE SCHEMA `auotverkauf` ;

- Tabelle anlegen namens „Kundenliste" mit den aus der Tabelle gegebenen Spaltennamen:

  CREATE TABLE `new_schema`.`Kundenliste` (
  `Kundennummer` INT NOT NULL ,
  `Nachname` VARCHAR(45) NOT NULL ,
  `Vorname` VARCHAR(45) NOT NULL ,
  `AdressID` INT NOT NULL ,
  PRIMARY KEY (`Kundennummer`));

- Tabellennamen „Kundenliste" in „Kunden" ändern:

  ALTER TABLE `new_schema`.`kundenliste` RENAME TO `new_schema`.`kunden` ;

- Datenbank löschen:

  drop schema `new_schema`;

2.5 Beispiele für DML

Beispiel ist unsere Datenbank „Autoverkauf"

- einen Datensatz zur Liste „Kunden" hinzufügen:

  INSERT INTO `autoverkauf`.`kunden` (`kundennummer`, `nachname`, `vorname`, `adr.-nummer `) VALUES ('1003', 'Kriegel', 'Julia', '10');

- einen Datensatz ändern (Vornamen ändern):

  UPDATE `autoverkauf`.`kunden` SET `vorname`='Annett' WHERE `kundennummer`='1003';

- einen Datensatz löschen:

  DELETE FROM `autoverkauf`.`kunden` WHERE `kundennummer`='1003';

2.6 Beispiele für DQL

Beispiel ist unsere Datenbank „Autoverkauf"

2.6.1 Einfache Abfragen mit Select

Anzeigen aller Datensätze der Tabelle „Kunden"

```
Select    autoverkauf.kunden.*
from      autoverkauf.kunden;
```

Anzeigen aller Datensätze der Spalten „Vorname" und „Nachname" der Tabelle „Kunden"

```
Select    autoverkauf.kunden.nachname,
          autoverkauf.kunden.nachname
from      autoverkauf.kunden;
```

Anzeigen aller Datensätze der Spalten „Nachname", jedoch soll die Spaltenbezeichnung als „Name" angezeigt werden

```
Select    autoverkauf.kunden.nachname as 'Name'
from      autoverkauf.kunden;
```

Anzeigen der Spalten „Nachname" und „Vorname" vereint in einer Spalte namens „Name"

```
use       autoverkauf;
Select    concat (nachname, '' , vorname)
          AS Gesamtname
from      kunden;
```

Anzeigen aller Spalten, jedoch nur die Datensätze, welche die Kundennummer „K1000" haben

```
use       autoverkauf;
Select    *
from      kunden
where     Kundennummer="K1000";
```

Anzeigen von den Spalten „Kundennummer“ und „Vorname“, nur Datensätze bei denen der Nachname mit „L“ anfängt, geordnet nach Vorname und Kundennummer

```
use         autoverkauf;
Select      kundennummer, vorname
from        kunden
where       nachname like 'L%'
order by    vorname, kundennummer
```

Verknüpfung von Bedingungen durch AND oder OR und Unterscheidung zwischen zwei Sortierformen ASC und DESC

```
use         autoverkauf;
Select      *
from        kunden
where       Nachname='Lindner' AND VORNAME='Michael'
order by    kundennummer asc, nachname desc;
```

2.6.2 KOMPLEXE ABFRAGEN MIT SELECT

Gruppierungsfunktionen

- COUNT → Datensätze zählen
- SUM → Summieren
- AVG → Mittelwert

Welches Auto wurde am meisten verkauft?

```
use         autoverkauf;
select      fahrzeugnummer, count(*) as AnzahlVerkäufe
from        rechnung
order by    AnzahlVerkäufe desc
group by    fahrzeugnummer;
```

Es sollen alle Rechnungen ausgegeben werden und der dazugehörige Datensatz aus der Tabelle Fahrzeuge

```
use         autoverkauf;

Select      rechnung.rechnungsnummer, rechnung.kundennummer,
            rechnung.fahrzeugnummer, fahrzeug.typ,
            fahrzeug.hersteller

from        rechnung inner join fahrzeug

on          rechnung.fahrzeugnummer =
            fahrzeug,Fahrzeugnummer;
```

LEFT JOIN & RIGHT JOIN

Um LEFT JOIN und RIGHT JOIN an einem Beispiel abzubilden, ändern wie die Tabellen unserer Datenbank „autoverkauf" wie folgt um:

Kundenliste (linke Tabelle)

Kundennummer (*)	Nachname	Vorname	Adr.-Nummer
K1001	Lindner	Michael	23
K1002	Hubert	Hans	12
K1275	Müller	Uwe	23

Adressliste (rechte Tabelle)

Adr.-Nummer (*)	Adresse	PLZ	ORT
23	Grenzweg 18	01917	Kamenz
12	Mühlweg 2	01900	Großröhrsdorf
5	Richtergasse 1	02994	Bernsdorf

Beim RIGHT JOIN werden definitiv ALLE Datensätze aus der rechten Tabelle angezeigt, und jeweils nur die Datensätze aus der linken Tabelle, die den Datensätzen der rechten zuordenbar sind. Andernfalls werden diese Felder mit Null-Werten gefüllt. Der LEFT JOIN arbeitet genau entgegen gesetzt. Bsp.:

```
use         autoverkauf;

Select      kunden.nachname, kunden.vorname, adresse.adresse,
            adresse.plz, adresse.ort

from        kunden LEFT JOIN adresse

on          kunden.adr.-nummer = adressen.adr.-nummer
```

Ergebnis:

Nachname	Vorname	Adresse	PLZ	Ort
Lindner	Michael	Grenzweg 18	01917	Kamenz
Hubert	Hans	Mühlweg 2	01900	Großröhrsdorf
Müller	Uwe	Grenzweg 18	01917	Kamenz

```
use        autoverkauf;
Select     kunden.nachname, kunden.vorname, adresse.adresse,
           adresse.plz, adresse.ort
from       kunden RIGHT JOIN adresse
on         kunden.adr.-nummer = adressen.adr.-nummer
```

Ergebnis:

Adresse	PLZ	ORT	Nachname	Vorname
Grenzweg 18	01917	Kamenz	Lindner	Michael
Grenzweg 18	01917	Kamenz	Müller	Uwe
Mühlweg 2	01900	Großröhrsdorf	Hubert	Hans
Richtergasse 1	02994	Bernsdorf	-	-

HAVING

HAVING benötigt man für Einschränkungen so, wie WHERE. Der Unterschied ist, dass man HAVING für Einschränkungen von Gruppierungen benötigt. Bsp.:

Zeige alle Fahrzeuge an, welche mehr als einmal verkauft wurden:

```
use        autoverkauf;
select     fahrzeugnummer, count(*) as AnzahlVerkäufe
from       rechnung
group by   fahrzeugnummer
having     AnzahlVerkäufe > 1;
```