# 5. Logische Datenanalyse

## 5.0 Einführung

Die logische Datenanalyse behandelt die Aspekte des Software Engineerings, die für die Anforderungsanalyse, die Spezifikation und das Design zu entwickelnder Datenbanksysteme (DBS) von Relevanz sind.

Das Software Engineering für Datenbanksysteme betrachtet, wie das Software Engineering für andere Softwaresysteme auch, ein Phasenmodell. Für die logische Datenanalyse reicht das folgende vereinfachte Phasenmodell aus:

- Phase 1: Anforderungsanalyse
- Phase 2: Spezifikation (Systemdefinition)
- Phase 3: Design
- Phase 4: Implementation
- Phase 5: Test
- Phase 6: Wartung

In Hinblick auf relationale Datenbanksysteme (RDBS) haben wir in Kap.2 der Vorlesung und im Praktikum im Wesentlichen mit der Phase 4 angefangen: Wir haben mit dem Anlegen von Tabellen (CREATE TABLE) begonnen. Das ist der Anfang der Implementation eines RDBS. Wir wollen hier die vorgelagerten Phasen (Phase 1 – Phase 3) diskutieren, um zu sehen, wie die Ergebnisse dieser Phasen zu einem

**vollständigen**, **erweiterbaren** und **dokumentierten**[1] Ansatz der Implementation eines DBS führen.

Im Software Engineering gibt es unterschiedliche Methodenansätze (MeAN): (a) Objektorientierte Analyse und Design (OOA/OOD), die zur Entwurfssprache UML (Unified Modelling Language) geführt hat, (b) Strukturierte Analyse und Design (StAD), die von einer Trennung von Informationsspeichern und Programmen / Funktionen ausgeht, (c) Rapid Prototyping (Scrum), die in schnellen Zykeln Implementation, Analyse und Design betreibt.

In nachfolgender Tabelle werden Arten von Phasenergebnissen bzw. Dokumentarten beschrieben, die für den Aufbau von DBS von Relevanz sind:

| **Nr.** | **Phase** | **Phasenergebnisse / Dokumentarten** | **MeAN / Zielsysteme** |
|---|---|---|---|
| **1** | **Anforderungsanalyse für ein DBS** | • Anwendungsfallbeschreibungen mit USE-CASE Diagrammen.<br>• Grobes Klassendiagramm | UML |
| | | • Anforderungskatalog | StAD |

[1] Eine strukturierte und transparente Dokumentation ist eine wichtige Voraussetzung für die **Qualitätssicherung** von Softwaresystemen.

| | | | |
|---|---|---|---|
| | | • Schnittstellenübersicht (IFX)<br>• Systemübersichtsdiagramm (IF0) | |
| | | • User Stories | Scrum |
| **2** | **Spezifikation eines DBS aus fachlicher Sicht** | • Verfeinertes Klassendiagramm mit Attributen<br>• Aktivitätsdiagramm / Sequenzdiagramm für wichtige Anwendungsfälle | UML |
| | | • Spezifikation von Informationsflüssen und Informationsspeichern in Data-Dictionary Notation | StAD |
| | | • Entity-Relationship Modell | (alle) |
| **3** | **DB Design** | • DBMS als Zielsystem auswählen (RDBMS, ORDBMS, OODBMS, NoSQL-DBMS, …). In Abhängigkeit vom Datenmodell des DBMS weitere Designaktivitäten ausführen: | (alle) |

| | | | |
|---|---|---|---|
| | | • Normalisierung (1NF, 2NF, 3NF) | RDBMS |
| | | • Abbildung des Klassendiagramms der persistenten Klassen auf abstrakte Datentypen (ADT) | ORDBMS OODBMS |
| **4** | **Implementation** | • CREATE TABLE – bzw. ALTER TABLE – Kommandos ausführen<br>• Z.B. JDBC-Programme schreiben | RDBMS |

## 5.1 Aktivitäten der strukturierten Analyse[2] in Phase 1

### a) Abgrenzung des zu entwickelnden Systems „nach außen“ (Schnittstellenübersicht / Kontextdiagramm)

Eine wichtige Aktivität in der Anforderungsanalyse ist die Beschreibung, wie das zu entwickelnde System mit **externen Schnittstellen** (Benutzern, Softwareagenten,

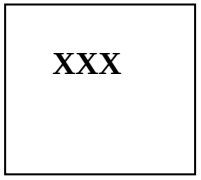
[2] Da die UML-bezogenen Aktivitäten in der Phase der Anforderungsanalyse in der Vorlesung Software Engineering ausführlich diskutiert werden (Anwendungsfalldiagramm, Klassendiagramm), wird hier der Aspekt der strukturierten Analyse etwas ausführlicher diskutiert.

benachbarten Programmsystemen, Dateischnittstellen usw.) kommunizieren soll. Diese Kommunikation findet über **Informationsflüsse** (Datenflüsse) statt.

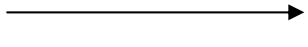
**Def.1:** Ein Diagramm, das die Kommunikation zwischen einem System und externen Schnittstellen mittels Informationsflüssen darstellt, heißt **Schnittstellenübersicht** oder Kontextdiagramm. In der Methodik der strukturierten Analyse wir der Diagrammtyp Schnittstellenübersicht als Diagrammtyp **IFX** bezeichnet[3]. (Das **X** steht für e**x**terne Schnittstelle („e**x**ternal interface")).

Eine Schnittstellenübersicht besteht aus den folgenden drei Symboltypen:

(1) Ein Symbol für eine externe Schnittstelle mit den Namen **XXX**:

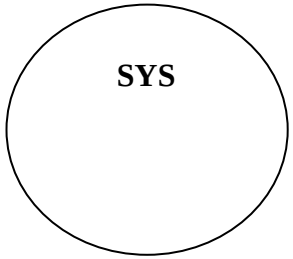



(2) Ein Symbol für einen Informationsfluss mit dem Namen **IFNAME**:




[3] Wird von „Datenflüssen" anstelle von „Informationsflüssen" gesprochen, bezeichnet man die Schnittstellenübersicht auch mit dem Symbol **DFX**.

(3) Ein Symbol für das gesamte System **SYS**:



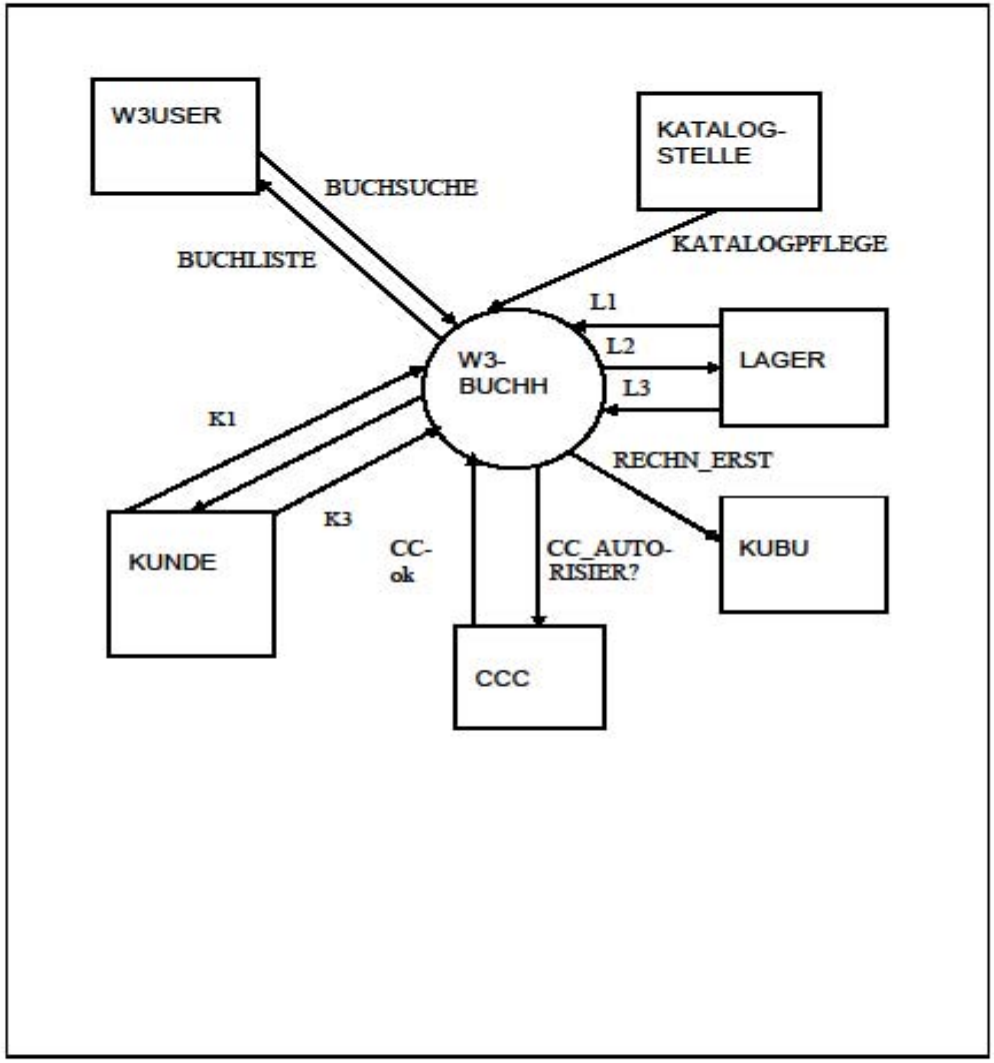

**BSP.1:** Nachfolgend ist ein **Anforderungskatalog** für ein WWW-Buchhandlungs-system W3-BUCHH gegeben:

1. Die Waren einer WWW-Versandbuchhandlung können für alle WWW-User (PUBLIC) über HTML-Seiten angezeigt werden.
2. WWW-User, die Versandkaufhauskunden werden möchten, können sich per HTML-Formular unter Angabe einer gültigen EMAIL-und Versandadresse registrieren lassen.
3. Bestellungen werden über HTML-Formular ausgeführt. Hierbei gibt der Kunde eine gültige Kreditkartennummer ein. Die Kreditkartennummer wird verschlüsselt mit einem zertifizierten Verfahren übertragen.
4. Der Kunde erhält für eine Bestellung eine Auftragsbestätigung per EMAIL.

5. Auf dem Postweg erhält der Kunde seine bestellten Waren. Für die Versandabwicklung ist das Lager zuständig. Damit das Lager den Versand ausführt, bekommt es den für die Bestellung gültigen Lieferschein. Der Lieferschein wird der Ware beigepackt.
6. Ist der Lieferschein erstellt, bekommt die Kundenbuchhaltung (externes System) eine Kopie des Lieferscheines (= erstellte Rechnung). Sie veranlasst dann die Rechnungszahlung via CC-Lastschrift und leistet die Zahlungsverfolgung.

Aus diesem Anforderungskatalog kann folgende Schnittstellenübersicht (**IFX**) abgeleitet werden:

**Schnittstellenübersicht:**



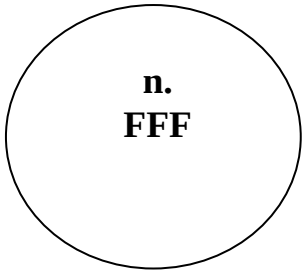
**b) Zerlegung des zu entwickelnden Systems in logische Hauptfunktionen (Hauptkomponenten)**

In Hinblick auf die Ergebnisse der Anforderungsanalyse besteht die Notwendigkeit, das geplante System in Komponenten zu zerlegen. Im Fall der objektorientierten Analyse, die UML nutzt, ist man hier in der Entscheidungsphase, in der man auf Grundlage der beschriebenen Anwendungsfälle ein grobes Klassendiagramm entwirft. Die darin enthaltenen **Klassen** bilden dann die **Hauptkomponenten** des Systems.

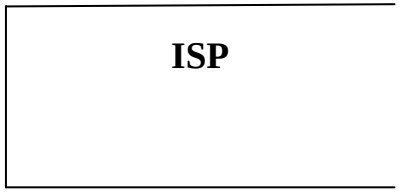
In der strukturierten Analyse verfolgt man einen strengen TOP-Down-Ansatz. Hier entstehen die **logischen Hauptfunktionen** als Zerlegung des Systemsymbols im IFX-Diagramm. Die logischen Hauptfunktionen kommunizieren entweder synchron zueinander mit Informationsflüssen oder asynchron über **Informationsspeicher**. Die Informationsspeicher werden in späteren Phasen, je nach Anforderung durch Dateien oder Datenbanksegmente implementiert. Weiterhin können Hauptfunktionen durch zeitlich abhängige Kontrollflüsse aktiviert werden.

Die Gesamtheit des in logischen Hauptfunktionen zerlegten Systems mit ihren externen Schnittstellen und Informationsspeichern wird durch ein Informationsflussdiagramm **IF0 Systemübersicht** dargestellt. Das **Systemübersichtsdiagramm IF0** enthält ergänzend zum Diagrammtyp IFX auch noch die folgenden Symbole:

(1) Ein Symbol für eine logische Hauptfunktion mit Nummer **n** ($1 \leq n \leq N$; N sollte eine kleine Zahl sein (N=10, N=12, …)) und Name **FFF**:



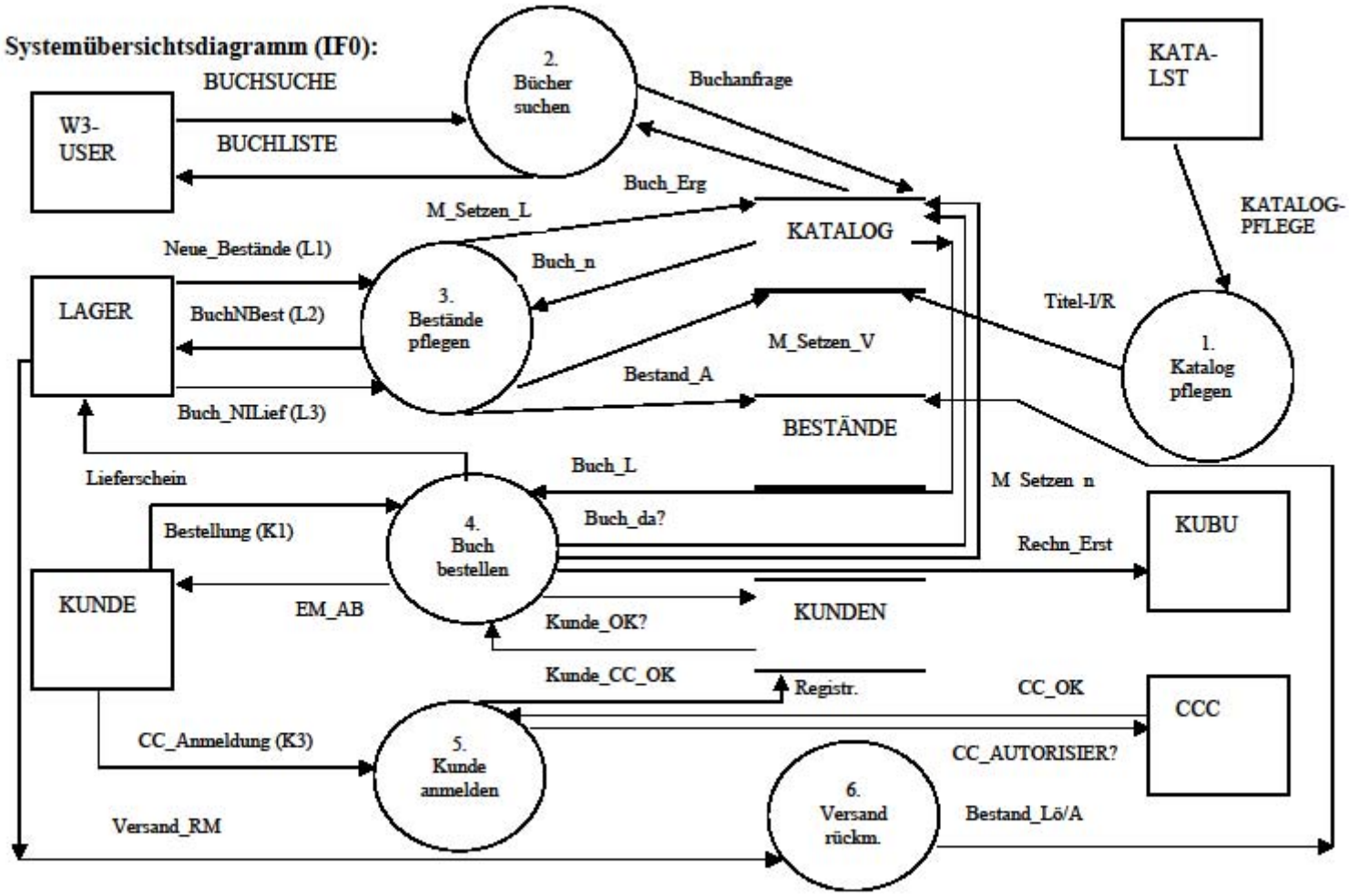

(2) Ein Symbol für einen Informationsspeicher **ISP**:




Nachfolgend ist als Beispiel ein **Systemübersichtsdiagramm IF0** für das System W3-BUCH gegeben:

1

**Systemübersichtsdiagramm (IF0):**

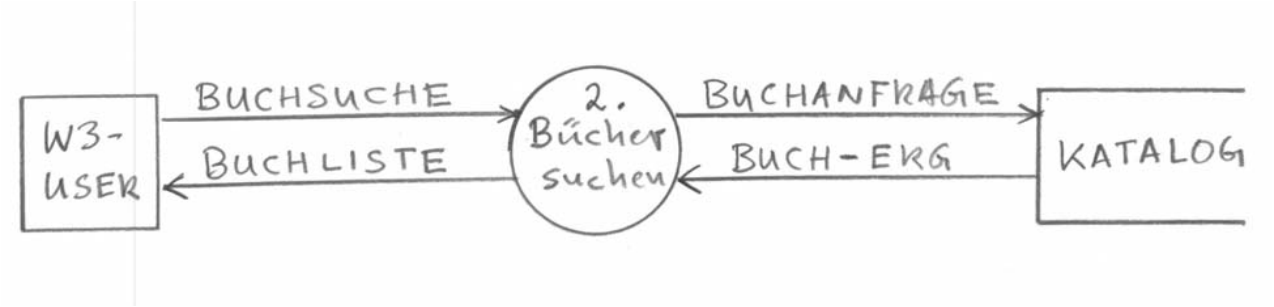


## 5.2. Spezifikation von Informationsflüssen (IFL) und Informationsspeichern (ISP) in Data-Dictionary-Notation

In der Phase 2 (Systemdefinition / Spezifikation eines DBS aus fachlicher Sicht) geht es darum, die Informationsflüsse (IFL) und die Informationsspeicher (ISP) aus fachlicher Sicht vollständig zu beschreiben. Das heißt insbesondere, den Aufbau der IFL und ISP aus Informationselementen vollständig anzugeben.

Informationsspeicher sind Kandidaten für später einzurichtende DB-Segmente. Informationselemente (IE) eines ISP sind Kandidaten für Attribute entsprechender DB-Segmente. Informationselemente (IE) sind die atomaren Bestandteile der IFL und ISP.

**BSP.2**: Für folgenden Ausschnitt aus dem **IF0-Diagramm** des zu entwickelnden DBS für das System W3-BUCHH (s.o.), der auf die Hauptfunktion „**2. Bücher suchen**“ bezogen ist, soll der beteiligte ISP KATALOG und die daran angreifenden IFL spezifiziert werden:



In diesem Ausschnitt sind der ISP KATALOG und die IFL BUCHSUCHE, BUCHANFRAGE, BUCH-ERG (Ergebnis der Kataloganfrage) und BUCHLISTE zu spezifizieren.

Die Spezifikation eines IFL bzw. eines ISP besteht aus folgenden Einzelaufgaben:
(a.1) Alle benötigten IE werden aus fachlicher Sicht für den IFL bzw. für den ISP definiert. Jeder IE wird identifiziert mit einem systemweit eindeutigen IE-Namen. Die Menge der **IE-Namen** bildet somit ein **kontrolliertes Vokabular**[4]. Diese IE-Definitionen werden in eine Data-Dictionary der Phase 2 eingetragen.

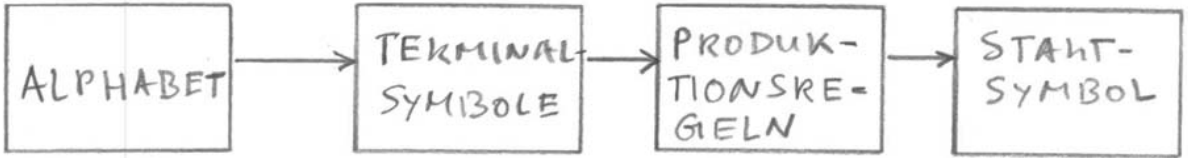
(a.2) Im Data-Dictionary prüfen: Liegt ein IE bereits dort mit gleicher Bedeutung aber anderem Namen vor?

(a.3) Einen IFL bzw. einen ISP in ihrem syntaktischen Aufbau aus IE zusammensetzen. Für jeden IFL bzw. ISP wird eine **Produktionsregel** angegeben. Die Produktionsregeln der Data Dictionary-Notation (DDN) haben einen BNF-ähnlichen Aufbau (BNF: Backus-Naur-Form). Im Unterschied zur BNF, die in einem Bottom-Up-Verfahren erstellt wird, wird eine DDN-Beschreibung eines IFL bzw. eines ISP in einem Top-Down-Verfahren erzeugt:

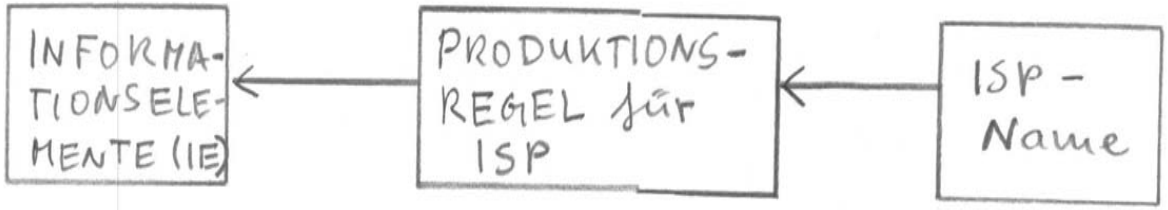
---

[4] Ein kontrolliertes Vokabular ist eine Sammlung von Begriffen, wobei jeder Begriff nur eine Bedeutung hat. [Unterschied zum Vokabular einer natürlichen Sprache: Hier hat jedes bedeutungstragende Wort häufig mehrere Bedeutungen.]

(1) Bottom-Up-Verfahren einer BNF, das von der Festlegung eines Alphabets über die Definition einer Menge von Terminalsymbolen und einer Folge von Produktionsregeln für Nichtterminalsymbole zur Definition des Startsymbols der Grammatik verläuft:




(2) Top-Down-Verfahren der Spezifikation eines IFL bzw. eines ISP in DDN, das von der Festlegung eines eindeutigen ISP- bzw. IFL-Namens über die Produktionsregel für einen ISP bzw. IFL, in der dessen Aufbau aus Informationselementen bestimmt wird, bis hin zur Definition aller beteiligten Informationselemente (IE) führt. Diese Definition erfolgt aus fachlicher Sicht. Hierbei werden noch keine Datentypen der IE in Bezug auf das Zielsystem festgelegt:



**Allgemeiner Aufbau einer DDN-Produktionsregel:**

( I ) Produktionsregel für ein IE: **IE_NAME := IE_Bedeutung**
(IE_Bedeutung: Freitext, der aus fachlicher Sicht eindeutig definiert ist)

( II ) Produktionsregel für einen IFL bzw. einen ISP:

**IFIS_NAME := rsPROD(IE1,..,IEn)**

Hierbei ist **rsPROD( )** ist die rechte Seite der Produktionsregel, in der Informationselemente **IE1,..., IEn** (Informationselemente) durch folgende Operatoren verknüpft werden können:

(1) Sequenz: + ( in BNF: Kein Zeichen)
(2) Alternative: [...|...] (in BNF: | )
(3) Wiederholung: {...} ( in BNF: { } )
(3a) Wiederholung mit Vielfachheiten: a{...}b oder {...}a : b (wie in der EBNF)
(mindestens a mal, höchstens b mal ( $a, b \in \mathbb{N} \cup \{0\}$ ) )
(4) Option: (...) (in BNF: [...] )

**BSP.3:** Spezifikation des IFL BUCHANFRAGE:
(I) Benötigte IE:

- BE := Benutzereingabe, die aus einer Zeichenkette mit Wildcard Zeichen * am Ende bestehen darf (z.B. "UML*" , "Elektro*")
- KAT := Kategorie der bibliographischen Angabe (z.B. "Titel", "Verfasser", "Erscheinungsort", "Jahr", "Schlagwort") a.2)

(II) IFL BUCHANFRAGE:

- BUCHANFRAGE:= (KAT) + BE

**Anm.1:** Keine Angabe von KAT <=> über alle Kategorien wird gesucht.
**Anm.2:** Variante: KAT = "ALLE" <=> über alle Kategorien wird gesucht.

**BSP.4:** Spezifikation des IFL BUCH-ERG:
(I) Benötigte IE:

- MELD1 := "Kein Treffer".
- TITEL := Hauptsachtitel eines Buches.
- VERF := Folge der Verfasser eines Buches.
- EORT := Folge der Erscheinungsorte eines Buches.
- EJAHR := Erscheinungsjahr des Buches
- SWK := Schlagwortkette zum Buch.
- ISBN := ISBN-Nr. des Buches.
- ANZ := Anzahl der gefundenen Titel.

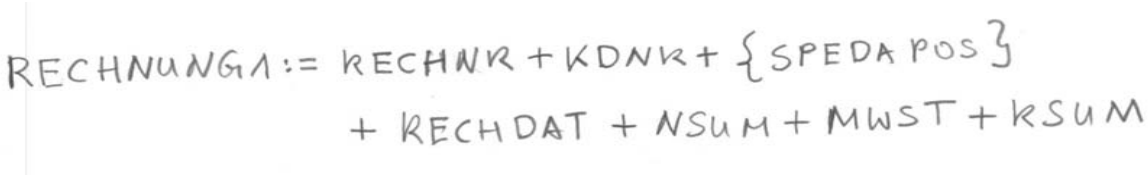
(II) IFL BUCH-ERG:

- BUCH-ERG:= ANZ + [MELD1 | { TITEL + VERF + EORT + EJAHR + (SWK) + ISBN } ]

**Anm.3:** Informationsflüsse bzw. Informationsspeicher können auch durch Klassendiagramme beschrieben werden. Der ISP KATALOG kann z.B. durch eine

Klasse KATALOGEINTRAG, die die Katalogisierung eines Buches beschreibt dargestellt werden. Auch hier werden die Attribute ohne Datentypen des Zielsystems angegeben:

**BSP.5:** Spezifikation eines IFL RECHNUNG1, der über eine Kundennummer KDNR auf Kunden-IE referenziert und der Positionen enthält, die durch einen IFL SPEDAPOS definiert sind (SPEDAPOS := POSNR + ARTBEZ + PREIS + AANZ + PWERT):




**Vollständigkeitsanalyse für Informationsspeicher (ISP)**

In einen ISP können mehrere IFL eingehen, d.h. der ISP ist die Senke dieser IFL, und von einem ISP können mehrere IFL ausgehen, d.h. der ISP ist die Quelle dieser IFL. Die ISP werden als passive Elemente des Systems angesehen. D.h. alle IE der ausgehenden IFL müssen durch IE der eingehenden IFL erzeugt werden können. Ein ISP kann daher durch eine Eingabe- / Ausgabe-Matrix (**E/A-Matrix**) beschrieben werden. Die **Zeilen** dieser Matrix sind durch die auf den ISP zugreifenden IFL bestimmt und die **Spalten** der E/A-Matrix sind durch die im ISP vorhandenen Informationselemente (IE) bestimmt.

Eine E/A-Matrix **EAM(ISP)** kann für einen ISP, der **K** Informationselemente IE enthält und auf den **n** IFL lesend bzw. schreibend zugreifen, folgendermaßen formal beschrieben werden:

$$EAM(ISP) = \begin{pmatrix} m\mathbf{11} & m\mathbf{12} & \dots & m\mathbf{1K} \\ m\mathbf{21} & m\mathbf{22} & \dots & m\mathbf{2K} \\ \dots & \dots & \dots & \dots \\ m\mathbf{n1} & m\mathbf{n2} & \dots & m\mathbf{nK} \end{pmatrix}$$

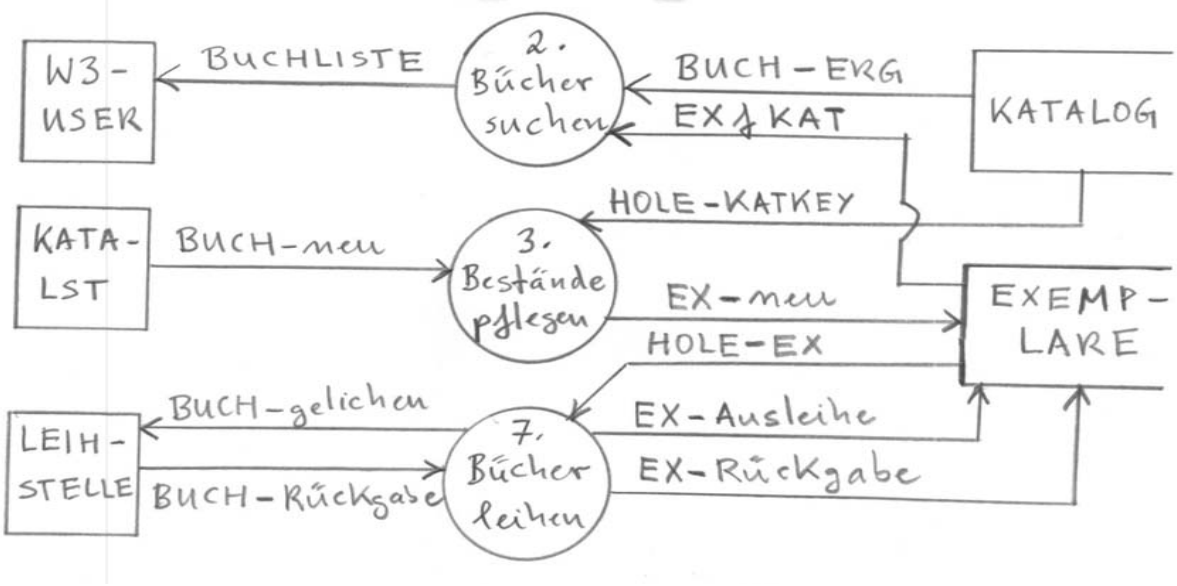
Die Koeffizienten $m_{ij}$ der Matrix EAM(ISP) können für alle i ∈ {1, ..., n} und j ∈ {1,..., K} folgende Werte annehmen:

$m_{ij}$ = **I** ⇔ der IFL **i** greift **einfügend** auf das IE **j** zu.
$m_{ij}$ = **U** ⇔ der IFL **i** greift **überschreibend** auf das IE **j** zu.
$m_{ij}$ = **D** ⇔ der IFL **i** greift **löschend** auf das IE **j** zu.
$m_{ij}$ = **A** ⇔ der IFL **i** greift **lesend** auf das IE **j** zu.
$m_{ij}$ = **_** ⇔ der IFL **i** greift **nicht** auf das IE **j** zu.

**Vollständigkeitsregel für einen ISP**: Wenn für alle Informationselemente IE**j** des ISP mit **j** ∈ {1,..., K} es einen IFL **i** mit **i** ∈ {1, ..., n} gibt, so dass $m_{ij}$ = **I** ist, dann ist der ISP vollständig.

**BSP.6:** Das System W3-BUCHH (s. BSP.2) soll um die Leihfunktionalität, wie sie bei Bibliotheken benötigt wird, ergänzt werden. D.h. neben den Katalogeinträgen sollen die ausleihbaren Bücher in einem ISP EXEMPLARE verwaltet werden. Für den ISP EXEMPLARE soll eine E/A-Matrix aufgestellt werden. Die Verbindung einer Menge gleicher Buchexemplare mit ihrem zugehörigen Katalogeintrag soll über ein IE

KATKEY (Katalogschlüssel) hergestellt werden[5]. In nachfolgender Abbildung ist ein Auszug aus dem erweiterten IF0-Diagramm des Systems W3-BUCHH dargestellt:




(Abb.1: IF0-Auszug für ISP EXEMPLARE)

---

[5] Im späteren DB-Design ist das IE KATKEY das PRIK-Attribut im DB-Segment KATALOG und ein FKEY-Attribut im DB-Segment EXEMPLARE. Das Klassendiagramm KATALOGEINTRAG ist um das Attribut KATKEY zu ergänzen (vgl. Anm.3).

**(I) IE des ISP EXEMPLARE:**

- SIGNATUR:=Zeichenfolge der Form BBBBNNNNNZZ, mit der jedes Exemplar eindeutig gekennzeichnet ist (vgl. Klebeschildeintrag auf dem Buchrücken eines Bibliothekexemplars).
- STATUS := Kennzeichen, das den Verleih-Zustand des Exemplars kennzeichnet. (z.B. 1 = ausleihbar, 2 = vorbestellt, 3 = ausgeliehen, ...).
- RDAT := Rückgabedatum, wann das Exemplar wieder an die Bibliothek zurückzugeben ist.

**(II) Produktionsregeln in DDN für die IFL, die an den ISP EXEMPLARE angreifen:**

- EXfKAT := KATKEY+SIGNATUR+STATUS+(RDAT)
- EX-neu := KATKEY+SIGNATUR+STATUS[6]
- HOLE-EX := KATKEY+SIGNATUR+STATUS+(RDAT)
- EX-Ausleihe := KATKEY+SIGNATUR+STATUS[7]+RDAT
- EX-Rückgabe := KATKEY+SIGNATUR+STATUS[8]+RDAT[9]

---

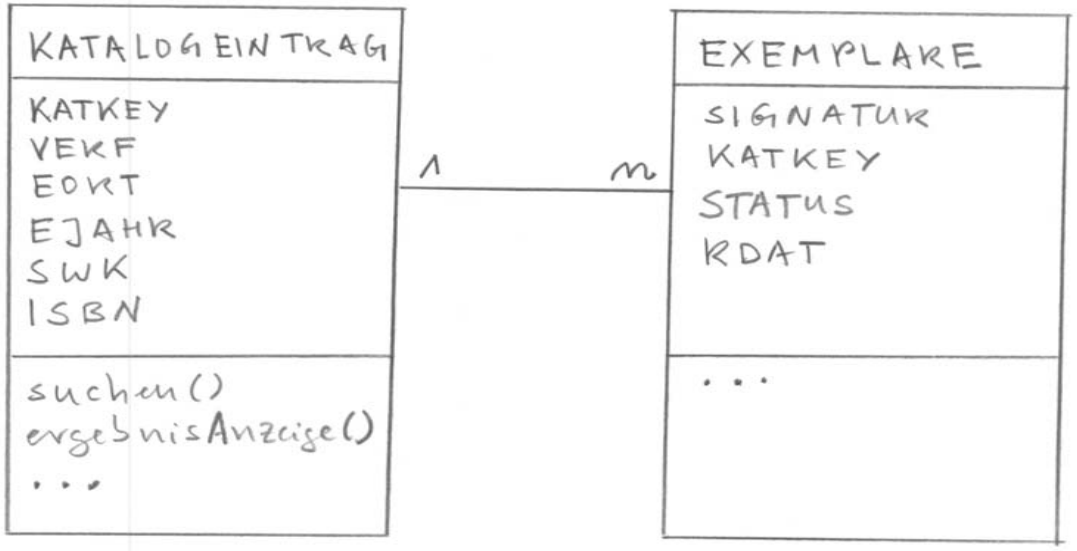
[6] Hier gilt: STATUS = 1.
[7] Hier gilt: STATUS = 2.
[8] Hier gilt: STATUS = 1.
[9] Hier gilt: RDAT = NULL.

Die E/A-Matrix für den ISP EXEMPLARE hat folgenden Aufbau:

| **IFL** | | **IE** | | | |
|---|---|---|---|---|---|
| Nr. | Bezeichnung | KATKEY | SIGNATUR | STATUS | RDAT |
| 1 | EXfKAT | A | A | A | A |
| 2 | EX-neu | I | I | I | _ |
| 3 | HOLE-EX | A | A | A | A |
| 4 | EX-Ausleihe | A | A | U | I |
| 5 | EX-Rückgabe | A | A | U | D |

Der ISP EXEMPLARE ist **vollständig**.

**Anm.4**: (Klassendiagramm 2. Fassung gegenüber Anm.3) Ordnet man die IE des ISP KATALOG und des ISP EXEMPLARE fachlich nach gleichartigen Objektmengen, so hat man a) die Menge der Katalogeinträge und b) die Menge der Exemplare. Damit dann kann man folgendes Klassendiagramm aufstellen, in dem **1** Objekt der Klasse KATALOGEINTRAG ist mit **n** Objekten der Klasse EXEMPLARE verbunden ist. Diese (**1:n)**-Beziehung wird im folgenden Klassendiagramm als **Assoziation** dargestellt.




(Abb.2: Klassendiagramm mit Klassen KATALOGEINTRAG und EXEMPLARE)

### 5.3. Spezifikation von Informationsspeichern mit Entity-Relationship-Diagrammen (ERD)

Man ordnet die IE eines ISP nach fachlich **gleichartigen** Objektmengen. Diese Objektmengen heißen **Entitätenmengen**. Ihre Elemente heißen Entitäten (entities). Entitätenmengen enthalten Objekte mit gleichen **Attributen**. Die **IE**, die in einer

Entitätenmenge zugeordnet sind, sind die Attribute dieser Entitätenmenge. Zwischen Entitätenmengen können **Beziehungen** bestehen, die mathematisch als **Relationen** modelliert werden. Jede Beziehung wird durch einen Beziehungstyp (relationship) beschrieben. Der Verbund mehrerer Entitätenmengen mit ihren zugehörigen Relationen heißt **Entity-Relationship-Modell (ERM)**.

**Geg.:** Zwei Entitätenmengen **E1** und **E2** . Eine **Relation R** ist Teilmenge des **kartesischen Produkts E1 x E2:** $R \subset E1 \; x \; E2$ . Man kann schreiben:

$R=\{(a\ ,b\ ) \in E1 \; x \; E2 \mid$ *eine Regel r(a ,b) ist wahr* $\}$

**BSP.7**: Man betrachte folgende Entitätenmenge E:

E = { x | x ist Bahnhof im VRS}

Als Teilmenge im kartesischen Produkt E x E ist die Menge der Bahnhofspaare definiert, die durch eine S-Bahnstrecke verbunden sind:

R = {(*a , b) ∈ E x E| a und b sind durch eine S−Bahnstrecke verbunden*}

Weitere Beispiele zu Relationen:

**BSP.8**: Man betrachte die beiden Mengen A und B:

A = {3, 5, 7, 11} und B = {3, 5, 7, 11, ..., 31, 33, 35}

Relation **R2**: (*a , b) ∈ A x B* ist in **R2** enthalten <=> *a∈A ist* Teiler von *b∈B.*

**Beschreibung von R2**: a) **durch Angabe einer Regel** (hier: eine prädikatenlogische Regel): R2 = { (*a , b) ∈ A x B* | ∃*a∈A b* ≡ 0(*mod a)* }

b) **in aufzählender Form**: [ entspricht Menge der Einträge in einer Tabelle T, wenn R2 durch T implementiert würde]: R2 = {(3,3), (3,9), (3,15), (3,21), (3,27), (3,33), (5,5), (5,15), (5,25), (5,35), (7,7), (7,21), (7,35), (11,11), (11,33)}

**BSP.9**: A = { Studenten eines Studienganges } , B = { Vorlesungen in einem Studiengang }. Relation **R3**: R3 = {(st, v) ∈ A x B | *st* ∈ *A* hört *v* ∈ *B*} .

**BSP.10**: A = $R_{0+}$ = {$x \in R \mid x \geq 0$}, B = R
Relation **R4**: R4 = {(x,y) ∈ A x B | $x = y^2$ } = { (x,y) ∈ A x B | $y = \sqrt{x} \vee y = -\sqrt{x}$}

Zur Beschreibung einer Beziehung (Relation) zwischen Entitätenmengen A und B sind drei Aspekte von Bedeutung:
(1) der **Name** der **Relation R**.
(2) das **Quantitätenpaar q** der Relation R**: q = (q1, q2)**, d.h. wievielen *a* ∈ *A* sind wieviele *b* ∈ *B* zugeordnet? Für ein beliebiges aber festes *a* ∈ *A* wird gezählt: Wieviele (a,b) gibt es in R ?
(3) die **Folge AF der Attribute** der Relation. Die Attributfolge von R kann leer sein (vgl. BSP.10). Die Attributfolge kann mehrere Attribute beinhalten (z.B. STRECKENLAENGE, FAHRZEIT, SBAHNNAME in BSP.7).

**Def.2**: Sind A und B Entitätenmengen und ist $R \subseteq A \times B$ eine Relation, dann heißt der Tripel **RP = (R, q, AF)** der **Beziehnungstyp** von R. (Relationship := (engl.) Bezie-

hungstyp). Hierbei ist R der Relationsname, q das Quantitätenpaar von R und AF die Attributfolge von R.

**Anm.5**: Quantitätenpaare haben die Form (q1,q2) mit q1, q2 ∈ {1,x,c,n,m}. Hierbei ist x eine konstante natürliche Zahl und für c gilt: c ∈ {0,1}. Weiterhin können additive und multiplikative Ausdrücke für q1 und q2 gebildet werden: Z.B. q2 = 1 + c oder q1 = c * n sind zulässige Ausdrücke. Folgende Paare sind häufig verwendete Quantitätenpaare (q1, q2) von Relationen:

**(1, 1)**: **1** $a \in A$ gehört zu **1** $b \in B$.
**(1, n)**: **1** $a \in A$ gehört zu **n** $b \in B$, wobei $n \geq 1$ ist.
**(1, c)**: **1** $a \in A$ gehört zu **höchstens 1** $b \in B$.
**(n, m)**: **n** $a \in A$ gehören zu **m** $b \in B$[10].

**BSP.11:** Beziehungstypen der Relationen aus BSP.7 bis BSP.10:

| BSP: | R | (q1,q2) | AF |
|---|---|---|---|
| 7 | ist verbunden durch eine S- | **(1, c*n) :** Ein Bahnhof **a** ist verbunden durch eine S- | STRECKENLAENGE, FAHRZEIT, |

[10] (n, m)-Relationen sind, wie wir in Kapitel 5.4 sehen werden, nicht zureichend spezifiziert. D.h. dem Beziehungstyp (R, q, AF) fehlt die Richtungsangabe, die durch das geordnete Paar (a, b) der mathematischen Relationsbeschreibung gegeben ist.

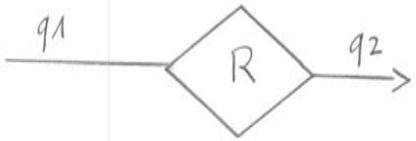
| | | | |
|---|---|---|---|
| | Bahn-Strecke | Bahn-Strecke mit **c*n** Bahnhöfen **b**[11]. | SBAHNNAME |
| 8 | ist Teiler von | **(1,n) :** Jedes a ∈ A ist Teiler von **n** b ∈ B. | FaktorX (wenn a Teiler von b ist, dann lässt b sich darstellen als: b = x * a) |
| 9 | hört | **(1,n) :** Jeder Student **st** hört **n** Vorlesungen **v**. | SEM (Semester, in dem **v** gehört wird) |
| 10 | hat als Wurzel | **(1, 1 + c):** x = 0 hat nur eine Wurzel y = 0. Jedes x > 0 hat zwei Wurzeln y = √x und y = -√x | <keine Attribute> |

**Aufbau eines Entity-Relationship-Diagramms (ERD):**
Ein ERD enthält folgende Symbolarten:

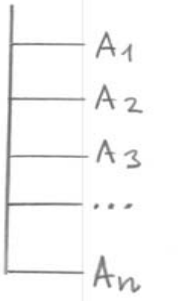
[11] Liegt der Bahnhof **a** an einer S-Bahn-Strecke, dann ist er verbunden mit **n** Bahnhöfen **b**. Liegt **a** an keiner S-Bahn-Strecke, dann ist **a** mit keinem Bahnhof **b** über eine S-Bahn-Strecke verbunden.

(1) Symbol für Entitätenmengen A:


(2) Symbol für Beziehungstypen mit Relationsname R[12] und Quantitätenpaar (q1,q2):

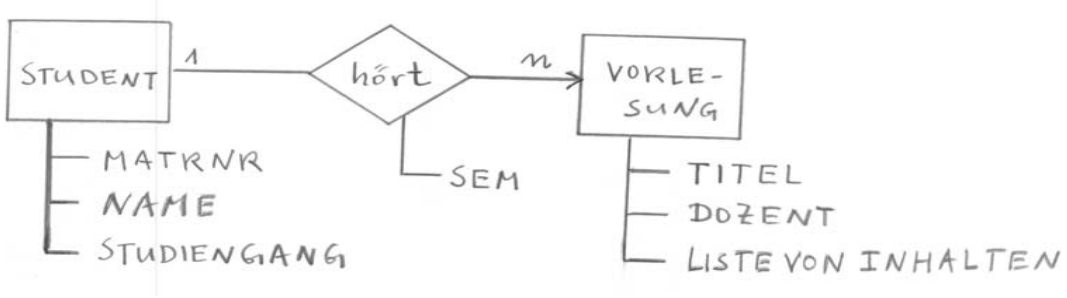



(3) Symbole für Attribute A1, ..., An die einer Entitätenmenge A oder einem Beziehungstyp R zugeordnet sind:

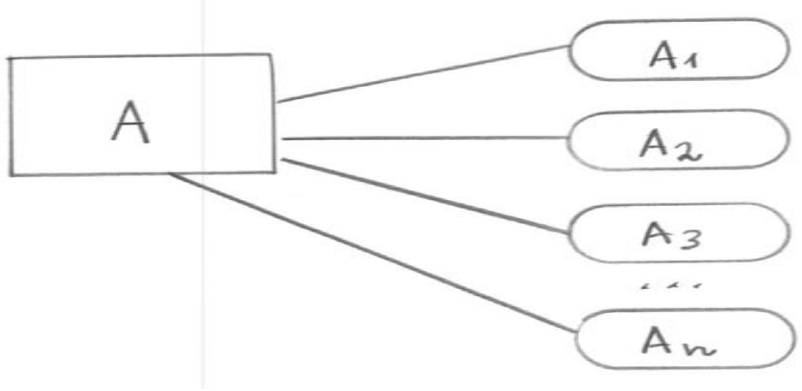



[12] Der Standard ist eine Relation R mit fester Leserichtung (gerichtete Relation).

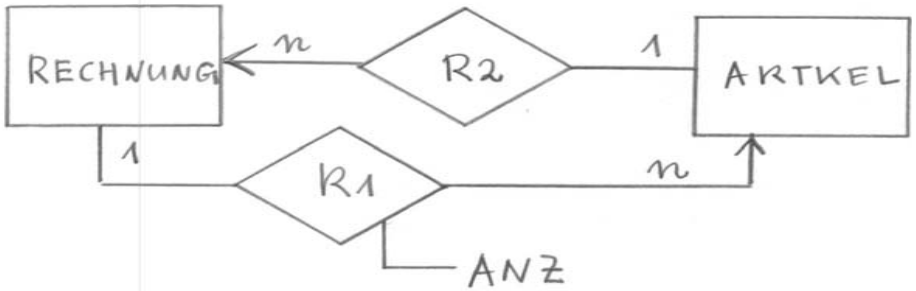
BSP.12: ERD zu BSP.9:




**Anm.6**: Attributnotation: Neben der Busnotation (s.o. (3)) gibt es für Attribute im ERD auch eine Blasennotation. Die Busnotation wird wegen ihrer Kompaktheit bevorzugt.
Blasennotation:



**BSP.13**: ERD zur Beziehung zwischen Handelsrechnungen (Entitätenmenge RECHNUNG) und verkauften Artikeln (Entitätenmenge ARTIKEL): Hierbei sind zwei Relationen zu beachten: a) R1:= „enthält“: **eine** Rechnung enthält **n** Artikel. Diese Relation hat das Attribut ANZ (Anzahl der verkauften Stücke eines Artikels).
b) R2:= „kommt vor in“: **ein** Artikel kommt vor in **n** Rechnungen.




**Anm.7**: Die Entity-Relationship-Analyse, die zu einem ERD führt, ist ein Verfahren der **semantischen Datenmodellierung**. Die semantische Datenmodellierung ist ein Verfahren zur Überprüfung, ob die Datenstruktur, die der Informatiker entwirft, mit den Bedeutungen übereinstimmt, die aus der Sicht des Anwenders in dem Weltausschnitt des geplanten Systems bestehen. Ein **Weltausschnitt** ist die Menge der Objekte, die der Anwender als Gegenstände des geplanten Systems identifiziert.

**5.4. Datenbank-Design** <wird fortgesetzt>