

Inhaltsverzeichnis

1	Datenbanken	1
1.1	Datenbankmodelle und -modellierung	4
1.1.1	Relationale und nicht-relationale Datenbanken	4
1.1.2	ERD (Entity-Relationship-Modell)	4
1.1.3	NoSQL	4
1.1.4	Welche Arten von NoSQL-Datenbanken gibt es?	5
1.2	Normalisierung	6
1.2.1	Anomalien	7
1.3	SQL	7
1.3.1	Projektion vs. Selektion	8
1.3.2	DDL, DML & DCL	9
1.3.3	CRUD	9
1.3.4	Subqueries	10
1.3.5	Aggregatfunktionen	10
1.3.6	Schnitt-, Vereinigungs- und Differenzmenge	10
1.3.7	SQL Injection	10
2	Qualitätssicherung	10
2.1	Prüfverfahren (Parität, Redundanz)	10
2.2	PDCA-Zyklus, KVP	10
2.3	Incident Management	10
2.4	Service Level Agreement (SLA), Servicelevel 1-3	10
2.5	Testen	10
2.5.1	Klassifizierung von Testverfahren	10
2.6	Versionsverwaltung	11
3	IT-Sicherheit	12
4	Datenschutz	12
5	Netzwerktechnik	13
6	Softwareentwicklung	14
6.1	Algorithmen	15
6.1.1	Flussdiagramm	15
6.1.2	Struktogramm (Nassi-Shneiderman-Diagramm)	16
6.2	Schnittstellen, APIs, Datenaustausch	17
6.3	Objektorientierung	17
6.4	Programmiersprachen	18
6.5	UML Diagramme	19
6.5.1	Klassendiagramm	19
6.5.2	Use-Case-Diagramm (Anwendungsfalldiagramm)	20

6.5.3 Zustandsdiagramm	21
6.5.4 Aktivitätsdiagramm	22
6.6 Softwarearchitektur	23
6.7 Softwareergonomie	23
6.8 Software Engineering	23
6.9 Design Patterns	23
6.10 Softwarequalität	23
6.11 Webentwicklung	23

Quellen	24
----------------	-----------

1 Datenbanken

IHK Belegsatz [6]

Syntax	Beschreibung
<i>Tabelle</i>	
CREATE TABLE Tabellennamen(Spaltenname <DATENTYP>, Primärschlüssel, Fremdschlüssel)	Erzeugt eine neue leere Tabelle mit der beschriebenen Struktur
ALTER TABLE Tabellennamen ADD COLUMN Spaltenname Datentyp DROP COLUMN Spaltenname Datentyp ADD FOREIGN KEY (Spaltenname) REFERENCES Tabellennamen(Primärschlüssel)	Änderungen an einer Tabelle: Hinzufügen einer Spalte Entfernen einer Spalte Definiert eine Spalte als Fremdschlüssel
CHARACTER	Textdatentyp
DECIMAL	Numerischer Datentyp (Festkommazahl)
DOUBLE	Numerischer Datentyp (Doppelte Präzision)
INTEGER	Numerischer Datentyp (Ganzzahl)
DATE	Datum (Format DD.MM.YYYY)
PRIMARY KEY (Spaltenname)	Erstellung eines Primärschlüssels
FOREIGN KEY (Spaltenname) REFERENCES Tabellennamen(Primärschlüsselspaltenname)	Erstellung einer Fremdschlüssel-Beziehung
DROP TABLE Tabellennamen	Löscht eine Tabelle
<i>Befehle, Klauseln, Attribute</i>	
SELECT * Spaltenname1 [, Spaltenname2, ...]	Wählt die Spalten einer oder mehrerer Tabellen, deren Inhalte in die Liste aufgenommen werden sollen; alle Spalten (*) oder die namentlich aufgeführten
FROM	Name der Tabelle oder Namen der Tabellen, aus denen die Daten der Ausgabe stammen sollen
SELECT ... FROM ... (SELECT ... FROM ... WHERE ...) AS tbl WHERE ...	Unterabfrage (subquery), die in eine äußere Abfrage eingebettet ist. Das Ergebnis der Unterabfrage wird wie eine Tabelle - hier mit Namen tbl behandelt.

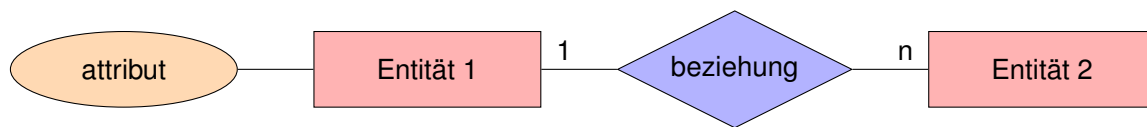
<i>Aggregatfunktionen</i>	
AVG (Spaltenname)	Ermittelt das arithmetische Mittel aller Werte im angegebenen Feld
COUNT (Spaltenname *)	Ermittelt die Anzahl der Datensätze mit Nicht-NULL-Werten im angegebenen Feld oder alle Datensätze der Tabelle (dann mit Operator *)
SUM (Spaltenname Formel)	Ermittelt die Summe aller Werte im angegebenen Feld oder der Formelerggebnisse
MIN (Spaltenname Formel)	Ermittelt den kleinsten aller Werte im angegebenen Feld
MAX (Spaltenname Formel)	Ermittelt den größten aller Werte im angegebenen Feld
<i>Funktionen</i>	
LEFT (Zeichenkette, Anzahlzeichen)	Liefert <i>Anzahlzeichen</i> der Zeichenkette von links.
RIGHT (Zeichenkette, Anzahlzeichen)	Liefert <i>Anzahlzeichen</i> der Zeichenkette von rechts.
CURRENT	Liefert das aktuelle Datum mit der aktuellen Uhrzeit
CONVERT (time,[DatumZeit])	Liefert die Uhrzeit aus einer DatumZeit-Angabe
DATE (Wert)	Wandelt einen Wert in ein Datum um
DAY (Datum)	Liefert den Tag des Monats aus dem angegebenen Datum
MONTH (Datum)	Liefert den Monat aus dem angegebenen Datum
TODAY	Liefert das aktuelle Datum
WEEKDAY (Datum)	Liefert den Tag der Woche aus dem angegebenen Datum
YEAR (Datum)	Liefert das Jahr aus dem angegebenen Datum
DATEADD (Datumsteil, Intervall, Datum)	Fügt einem Datum ein Intervall (ausgedrückt in den unter Datumsteil angegebenen Einheiten) hinzu
DATEDIFF (Datumsteil, Anfangsdatum, Enddatum) Datumsteile: DAY, MONTH, YEAR	Liefert Enddatum-Startdatum (ausgedrückt in den unter Datumsteil angegebenen Einheiten)
<i>Operatoren</i>	
AND	Logisches UND
LIKE	Überprüfung von Text auf Gleichheit wenn Platzhalter ('regular expressions') eingesetzt werden.
NOT	Logische Negation
OR	Logisches ODER
IS NULL	Überprüfung auf NULL
=	Test auf Gleichheit
>, >=, <, <=, <>	Test auf Ungleichheit
*	Multiplikation
/	Division
+	Addition, positives Vorzeichen
-	Subtraktion, negatives Vorzeichen

1.1 Datenbankmodelle und -modellierung

1.1.1 Relationale und nicht-relationale Datenbanken

Eine nicht relationale Datenbank ist eine Datenbank, die nicht das tabellarische Schema mit Zeilen und Spalten verwendet, das in den meisten herkömmlichen Datenbanksystemen zum Einsatz kommt. Nicht relationale Daten verwenden stattdessen ein Speichermodell, das für die spezifischen Anforderungen des gespeicherten Datentyps optimiert ist. So können die Daten beispielsweise als einfache Schlüssel-Wert-Paare, als JSON-Dokumente oder als Diagramm mit Edges und Scheitelpunkten gespeichert werden. [9]

1.1.2 ERD (Entity-Relationship-Modell)



Kardinalitäten:

Entitäten besitzen unterschiedliche Kardinalitäten, also die Anzahl zuordenbarer Objekte einer anderen Entität. Es gibt die Ausprägungen 1:1 (eins zu eins), 1:n (eins zu mehreren) und n:m (mehrere zu mehreren). [4]

1.1.3 NoSQL

NoSQL-Datenbanken wurden speziell für bestimmte Datenmodelle entwickelt und speichern Daten in flexiblen Schemas, die sich leicht für moderne Anwendungen skalieren lassen. NoSQL-Datenbanken sind für ihre einfache Entwicklung, Funktionalität und Skalierbarkeit weithin bekannt. [1]

Flexibilität NoSQL-Datenbanken bieten in der Regel flexible Schemata, die eine schnellere und iterativere Entwicklung ermöglichen. Das flexible Datenmodell macht NoSQL-Datenbanken ideal für halbstrukturierte und unstrukturierte Daten.

Skalierbarkeit NoSQL-Datenbanken sind in der Regel so konzipiert, dass sie durch die Verwendung von verteilten Hardware-Clustern skaliert werden können, im Gegensatz zu einer Skalierung durch das Hinzufügen teurer und robuster Server. Einige Cloud-Anbieter übernehmen diese Vorgänge im Hintergrund als vollständig verwaltete Dienstleistung.

Hohe Leistung NoSQL-Datenbanken sind für bestimmte Datenmodelle und Zugriffsmuster optimiert. Diese ermöglichen eine höhere Leistung, als wenn Sie versuchen würden, ähnliche Funktionen mit relationalen Datenbanken zu erreichen.

Hochfunktionell NoSQL-Datenbanken bieten hochfunktionelle APIs und Datentypen, die speziell für ihre jeweiligen Datenmodelle entwickelt wurden.

1.1.4 Welche Arten von NoSQL-Datenbanken gibt es?

Schlüsselwertdatenbanken Eine Schlüsselwertdatenbank speichert Daten als eine Sammlung von Schlüsselwertpaaren, in denen ein Schlüssel als eindeutiger Identifikator dient. Schlüssel und Werte können alles sein, von einfachen Objekten bis hin zu komplexen zusammengesetzten Objekten. Anwendungsfälle wie Gaming, Werbung und IoT eignen sich besonders gut für das Schlüssel-Werte-Datenspeicherungsmodell.

Dokumentdatenbanken Dokumentdatenbanken verfügen über das gleiche Dokumentmodellformat, das Entwickler in ihrem Anwendungscode verwenden. Sie speichern Daten als JSON-Objekte, die flexibel, halbstrukturiert und hierarchisch aufgebaut sind. Aufgrund des flexiblen, semi-strukturierten und hierarchischen Aufbaus der Dokumente und Dokumentdatenbanken können diese entsprechend den Anforderungen der Anwendungen weiterentwickelt werden. Das Dokumentdatenbankmodell eignet sich gut für Kataloge, Benutzerprofile und Content-Management-Systeme, bei denen jedes Dokument einzigartig ist und sich im Laufe der Zeit weiterentwickelt.

Graphdatenbanken Der Zweck einer Graphdatenbank besteht darin, das Entwickeln und Ausführen von Anwendungen zu vereinfachen, die mit hochgradig verbundenen Datensätzen arbeiten. Sie verwenden Knoten zur Speicherung von Dateneinheiten und Edges zur Speicherung von Beziehungen zwischen Einheiten. Ein Edge hat immer einen Startknoten, einen Endknoten, einen Typ und eine Richtung. Er kann Eltern-Kind-Beziehungen, Aktionen, Besitzverhältnisse und Ähnliches beschreiben. Die Anzahl und Art der Beziehungen in einem Knoten ist nicht beschränkt. Sie können eine Graphdatenbank verwenden, um Anwendungen zu erstellen und auszuführen, die mit stark verbundenen Datensätzen arbeiten. Typische Anwendungsfälle für eine Graphdatenbank sind Social Networking, Empfehlungsmodule, Betrugserkennung und Wissensdiagramme.

In-Memory-Datenbanken Während andere nicht-relationale Datenbanken Daten auf Festplatten oder SSDs speichern, sind In-Memory-Datenspeicher so konzipiert, dass kein Zugriff auf Festplatten erforderlich ist. Sie eignen sich ideal für Anwendungen, die Reaktionszeiten im Mikrosekundenbereich erfordern oder große Verkehrsspitzen aufweisen. Sie können sie in Gaming- und Ad-Tech-Anwendungen für Features wie Bestenlisten, Sitzungsspeicher und Echtzeitanalysen verwenden.

Suchmaschinendatenbank Eine Suchmaschinendatenbank ist eine Art **nichtrelationaler Datenbank**, die sich der Suche nach Dateninhalten widmet, z. B. nach Anwendungsausgabeprotokollen, die von Entwicklern zur Problembehandlung verwendet werden. Sie verwendet Indizes, um ähnliche Merkmale unter den Daten zu kategorisieren und die Suchfunktion zu vereinfachen. Suchmaschinendatenbanken sind für die **Sortierung unstrukturierter Daten wie Images und Videos** optimiert.

1.2 Normalisierung

Quelle: DatabaseCamp [2]

1. Normalform Eine Relation liegt in der ersten Normalform vor, wenn alle Attributwerte **atomar** vorliegen.

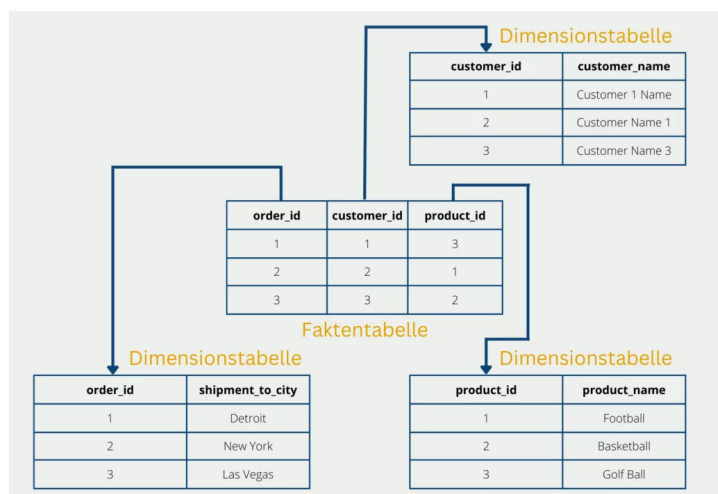
Das bedeutet, dass **jedes Datenfeld lediglich einen Wert enthalten darf**. Außerdem sollte sichergestellt sein, dass jede Spalte nur Werte desselben Datentyps (Numerisch, Text, etc.) enthält. Folgende Beispiele müssten entsprechend verändert werden, damit eine Datenbank in der 1. Normalform vorhanden ist:

- Adresse: "Hauptstraße 1, 12345 Berlin"
 - Straße: "Hauptstraße"
 - Hausnummer: "1"
 - PLZ: "12345"
 - Ort: "Berlin"
- Rechnungsbetrag: "128,45 €"
 - Betrag: "128,45"
 - Währung: "€"

2. Normalform Eine Relation liegt in der zweiten Normalform vor, wenn sie in der ersten Normalform vorliegt und alle **Nichtschlüsselattribute voll funktional vom gesamten Primärschlüssel abhängig** sind.

Der Primärschlüssel bezeichnet ein Attribut, das zur eindeutigen Identifikation einer Datenbankzeile verwendet werden kann. Dazu zählen beispielsweise die Rechnungsnummer zur Identifikation einer Rechnung oder die Ausweisnummer zur Identifikation einer Person.

Konkret bedeutet dies in der Anwendung, dass **alle Merkmale ausgelagert werden müssen, die nicht ausschließlich vom Primärschlüssel abhängig sind**. In der Praxis führt dies dann oft zu einem sogenannten Sternschema.



3. Normalform Eine Relation liegt in der dritten Normalform vor, wenn sie in der ersten und zweiten Normalform vorliegt und **keine transitiven Abhängigkeiten** bestehen.

Eine transitive Abhängigkeit liegt vor, wenn ein Attribut, welches kein Primärschlüssel ist, nicht nur von diesem abhängt, sondern auch von anderen Attributen.

Wenn wir in unserem Beispiel eine Tabelle haben, in der die Rechnungsnummer, die Produktnummer und der Preis gegeben ist, haben wir höchstwahrscheinlich eine transitive Abhängigkeit. **Der Preis des Produktes hängt nämlich nicht wirklich von der Rechnungsnummer ab, sondern vielmehr von der Produktnummer,** da für jedes Produkt ein fester Preis definiert ist.

Diese Abhängigkeit kann man auflösen, indem man die Produkte in eine neue Tabelle auslagert und somit das Attribut Preis aus der ursprünglichen Tabelle rausfällt.

1.2.1 Anomalien

Quelle: DatabaseCamp [3]

Anomalien in Datenbanken treten bei einer **nicht existierenden oder fehlerhaften Normalisierung** auf. Es existieren drei Arten von Datenbank-Anomalien, die Einfüge-Anomalie, die Änderungs-Anomalie und die Löschanomalie.

In der Datenbankentwicklung ist die Dritte Normalform oft ausreichend, um die perfekte Balance aus Redundanz, Performance und Flexibilität für eine Datenbank zu gewährleisten. Sie eliminiert auch die meisten Anomalien in einer Datenbank, aber nicht alle.

Einfügeanomalie Bei einem fehlerhaften oder inkorrekten Datenbankdesign kann es bei der Einfüge-Anomalie passieren, dass Daten gar nicht in die Datenbank übernommen werden, wenn zum Beispiel der Primärschlüssel keinen Wert erhalten hat, oder eine **unvollständigen Eingabe von Daten** zu Inkonsistenzen führt.

Änderungsanomalie Bei der Änderungs-Anomalie, auch Update-Anomalie genannt, werden **gleiche Attribute eines Datensatzes in einer Transaktion nicht automatisch geändert**. So entsteht eine Inkonsistenz der Daten.

Löschanomalie Bei einer Löschanomalie kann es passieren, dass ein Benutzer einer Datenbank aktiv Informationen löschen will und damit indirekt, aufgrund des fehlerhaften Datenbankdesigns, **andere zusammenhängende Informationen parallel mitlöscht**.

1.3 SQL

Alle SQL-Komponenten für Abfragen siehe *1 Datenbanken auf Seite 1*.

1.3.1 Projektion vs. Selektion

Quelle: Tino Hempel [12]

Selektion Bei der Selektion werden Zeilen aus einer Tabelle ausgewählt, die bestimmten Eigenschaften genügen.

Aus der Tabelle Schüler sollen alle Zeilen selektiert werden, in denen der Name Müller steht. Die Selektion hat also die Form: $S_{Name = 'Müller'}(Schüler)$

Schüler

<u>SNr</u>	Vorname	Name
4711	Paul	Müller
0815	Erich	Schmidt
7472	Sven	Lehmann
1234	Olaf	Müller
2313	Jürgen	Paulsen

$S_{Name = 'Müller'}(Schüler)$

<u>SNr</u>	Vorname	Name
12	Paul	Müller
308	Olaf	Müller

Projektion Bei der Projektion werden Spalten aus einer Tabelle ausgewählt, die bestimmten Eigenschaften genügen.

Aus der Tabelle Schüler sollen alle Spalten mit dem Attribut 'Name' projiziert werden. Die Projektion hat also die Form: $P_{Name}(Schüler)$

Schüler

<u>SNr</u>	Vorname	Name
4711	Paul	Müller
0815	Erich	Schmidt
7472	Sven	Lehmann
1234	Olaf	Müller
2313	Jürgen	Paulsen

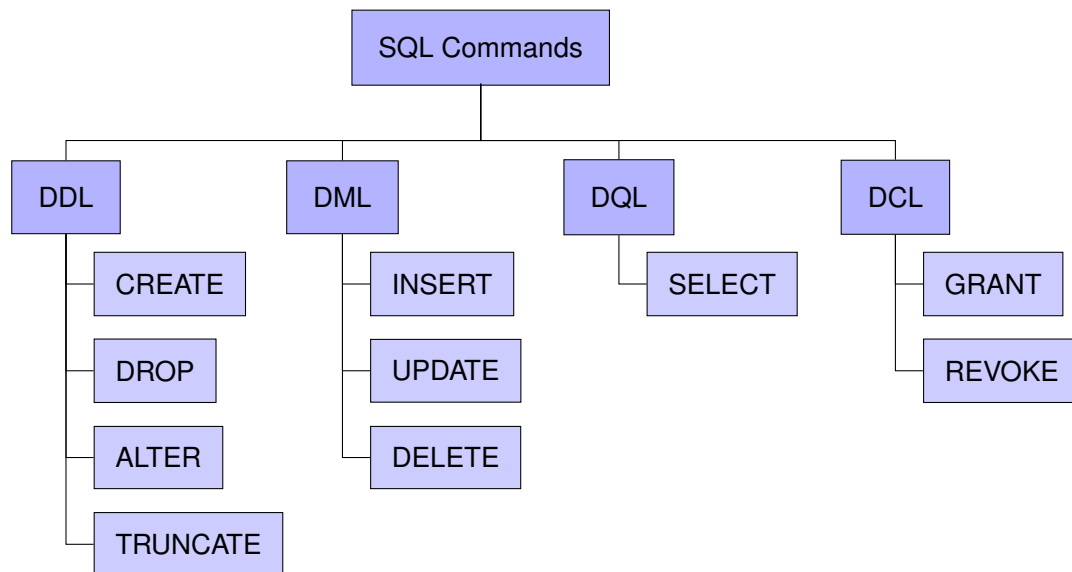
$P_{Name}(Schüler)$

Name
Müller
Schmidt
Lehmann
Müller
Paulsen

1.3.2 DDL, DML & DCL

Quelle: GeeksforGeeks [5]

1. DDL - Data Definition Language
2. DML - Data Manipulation Language
3. DQL - Data Query Language
4. DCL - Data Control Language



1.3.3 CRUD

Quelle: sqlshack.com [11]

C refers to CREATE aka add, insert. In this operation, it is expected to insert a new record using the SQL insert statement. SQL uses **INSERT INTO** statement to create new records within the table.

```
1 INSERT INTO <tablename> (column1 ,column2 ,...)
2 VALUES (value1 ,value2 ,...) , ( value1 ,value2 ,...) , ( value1 ,value2 ,...) ...
```

```
1 INSERT INTO dbo.Demo
2 (id , name)
3 VALUES
4 (2, 'Jayaram' ),
5 (3, 'Pravitha' );
```

R refers to SELECT (data retrieval) operation. The word 'read' retrieves data or record-set from a listed table(s). SQL uses the SELECT command to retrieve the data.

```
1 SELECT * FROM <TableName>;
```

U refers to Update operation. Using the Update keyword, SQL brings a change to an existing record(s) of the table. When performing an update, you'll need to define the target table and the columns that need to update along with the associated values, and you may also need to know which rows need to be updated. In general, you want to limit the number of rows in order to avoid lock escalation and concurrency issues.

```
1 UPDATE <TableName>
2 SET Column1=Value1 , Column2=Value2 , ...
3 WHERE <Expression>
```

D refers to removing a record from a table. SQL uses the SQL DELETE command to delete the record(s) from the table.

```
1 DELETE FROM <TableName>
2 WHERE <Expression>
```

1.3.4 Subqueries

1.3.5 Aggregatfunktionen

1.3.6 Schnitt-, Vereinigungs- und Differenzmenge

1.3.7 SQL Injection

2 Qualitätssicherung

2.1 Prüfverfahren (Parität, Redundanz)

2.2 PDCA-Zyklus, KVP

2.3 Incident Management

2.4 Service Level Agreement (SLA), Servicelevel 1-3

2.5 Testen

2.5.1 Klassifizierung von Testverfahren

- Wer testet?

- Mensch (manuell) vs. Maschine (automatisch)
- Entwickler vs. Benutzer
- Was wird getestet?
 - Komponente (Unit-Test/Funktionstest/Klassentest) vs. Integration vs. System (End-to-End)
 - Testpyramide
- Wie wird getestet?
 - Bottom-Up vs. Top-Down
 - statisch (Kompilierzeit) vs. dynamisch (Laufzeit)
 - ohne Kenntnis des Codes (Blackbox) vs. mit Kenntnis des Codes (Whitebox)
 - explorativ
 - Schreibtischtest/Review
- Wann wird getestet?
 - Vor vs. nach der Entwicklung
 - Abnahmetest
- Warum wird getestet?
 - Regressionstest
 - Lasttest/Belastungstest
 - Smoketest

2.6 Versionsverwaltung

- Werkzeuge zur Versionsverwaltung einsetzen
- Eigenschaften eines Versionsverwaltungssystems beschreiben
 - SVN, CVS, TFS mit Source Safe, Git
 - VCS vs. DVCS
- Nutzen und Anwenden einschlägiger Systeme, z.B. Git
- Funktionen, z.B. Commit, Revert, Branch, Merge, Cherry-Pick,
- Pull/Push, Rebase
- Übliche Workflows im Team, z.B. Pull/Merge Requests

3 IT-Sicherheit

- Datensicherheit (Authentifizierung, Autorisierung, Verschlüsselung)
- Bedrohungsszenarien erkennen und Schadenspotenziale unter Berücksichtigung wirtschaftlicher und technischer Kriterien einschätzen
- Für jede Anwendung, die verwendeten IT-Systeme und die verarbeiteten Informationen gilt: Betrachtung zu erwartender Schäden, die bei einer Beeinträchtigung von Vertraulichkeit, Integrität oder Verfügbarkeit entstehen könnten
- geeignete Gegenmaßnahmen, z.B. USV-Anlagen, Klimageräte, Firewalls
- Einteilung in die drei Schutzbedarfskategorien „normal“, „hoch“ und „sehr hoch“ (analog IT-Grundschutz des BSI)
- Begriffe kennen/erläutern
 - Hacker (White Hat, Black Hat), Cracker, Script-Kiddies
 - Spam, Phishing, Sniffing, Spoofing, Man-in-the-Middle
 - SQL-Injection, XSS, CSRF, Session Hijacking, DoS, DDoS
 - Viren, Würmer, Trojaner, Hoax, Dialer (veraltet), Keylogger, Botnetze, Spyware, Adware, Ransomware, Scareware
 - Backdoor, Exploit, 0-Day-Exploit, Rootkit
 - Verbreitung von Viren/Würmer/Trojaner erläutern

4 Datenschutz

- Datenschutzgesetze – national und auf EU-Ebene, z.B. Datenschutzgrundverordnung (DSGVO), BDSG
- Grundsätze des Datenschutzes (Art. 5)
 - Rechtmäßigkeit/Gesetzmässigkeit (Erfordernis der gesetzlichen Grundlage)
 - Transparenz gegenüber den betroffenen Personen Zweckbindung
 - Datenminimierung/Verhältnismässigkeit (Datensparsamkeit und Datenvermeidung)
 - Richtigkeit
 - Speicherbegrenzung
 - Integrität und Vertraulichkeit

- Rechenschaftspflicht
- Informationssicherheit
- Betroffenenrechte
 - Recht auf Information
 - Recht auf Auskunft
 - Recht auf Berichtigung
 - Recht auf Löschung
 - Recht auf Einschränkung der Bearbeitung
 - Recht auf Widerspruch
 - Recht auf Datenübertragbarkeit
- Persönlichkeitsrechte
 - Recht auf informationelle Selbstbestimmung
 - Recht am eigenen Bild
 - Recht am geschriebenen/gesprochenen Wort
 - Recht auf Schutz vor Imitation der Persönlichkeit
 - Recht auf Schutz der Intim-, Privat- und Geheimsphäre

5 Netzwerktechnik

- Adressierung
 - IPv4/IPv6, MAC, ARP
- Routing, Switching
- DNS, DHCP
- TCP/UDP
- HTTPS, TLS/SSL, IPsec
 - Hash, Signatur, Zertifikat, Certificate Authority
- Verschlüsselung (pre-shared key, RADIUS ...)
- LAN/WAN/MAN/GAN
- Strukturierte Verkabelung

- primäre/sekundäre/tertiäre Verkabelung
 - Kabeltypen (Twisted Pair, LWL)
- VLAN
- Sicherheitskonzepte und -risiken: WEP, WPA
- Netzwerktopologien
- Netzwerkplan
- VPN
 - Funktionsweise und Vorteile von VPN beschreiben
 - VPN-Modelle
 - Tunneling
- Serverarten: Mailserver, Webserver, Groupware, Datenbanken, Proxy
- Sicherstellung des Betriebs
 - Elektrotechnisch (USV)
 - Hardwaretechnisch (Redundanzen), RAID
 - Softwaretechnisch (Back-ups. . .)
- Firewall
- Portsecurity, Port-Forwarding

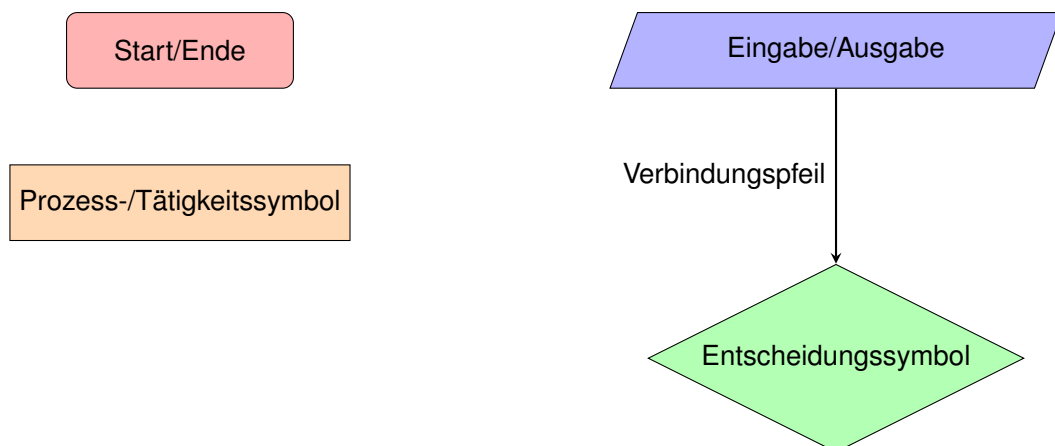
6 Softwareentwicklung

- Allgemeines Fehlerhandling in Programmen
 - Exceptions, Return/Exit Codes
 - Unterschied syntaktische/semantische Fehler
- Rechnerarchitektur: CPU, BUS, Speicher und deren Adressierung
- Lizenzen unterscheiden
 - Open Source, proprietär
- Informationspflichten zu Produkten, Namens- und Markenrecht, Urheber- und Nutzungsrecht, Persönlichkeitsrecht, unlauterer Wettbewerb

6.1 Algorithmen

- Abbildung der Kontrollstrukturen mittels Struktogramm, PAP oder Pseudocode als didaktisches Hilfsmittel
- grundlegende Algorithmen kennen, eigene Algorithmen auch programmiersprachenfrei formulieren und zur Lösung von Problemen, z.B. in einem IT-System bzw. einer Softwareanwendung einsetzen
- Entwickeln und Darstellen von Programmlogiken unabhängig von der Programmiersprache, z.B. mithilfe von Struktogrammen nach Nassi-Shneidermann sowie Strukturdiagrammen und Verhaltensdiagrammen aus der UML
- Rekursion: Funktionsweise, Vor-/Nachteile
- Algorithmen implementieren/durchspielen
 - Mittelwert
 - doppelte Einträge in einem Array finden/löschen
 - Dateibäume rekursiv kopieren
 - (Zinses-)Zinsberechnung
 - Planen eines regelmäßigen Backups
 - Ablauf einer Benutzerauthentifizierung an einer Website
 - Abbuchen von einem Konto
 - Lineare Suche
 - Binäre Suche
 - Bubble Sort

6.1.1 Flussdiagramm



6.1.2 Struktogramm (Nassi-Shneiderman-Diagramm)

Quelle: Lehrerfortbildung-bw.de [8]

Anweisung

gehe 10er-Schritt

Sequenz

gehe 10er-Schritt

schalte Stift ein

sage 'Hallo!'

Schleife mit Bedingung

wiederhole bis Rand berührt

ändere x um 10

Schleife mit Zähler

wiederhole 10 mal

gehe 4er-Schritt

drehe dich nach rechts
um 5 Grad

Endlosschleife

wiederhole fortlaufend

gehe 8er-Schritt

pralle vom Rand ab

Verzweigung mit
Alternative

wird Ball
berührt?

ja

nein

stoppe alles

6.2 Schnittstellen, APIs, Datenaustausch

- Datenaustauschformate: CSV, XML, JSON
- XML
 - Wohlgeformtheit, Validität
 - DTD, Schema, RelaxNG, Schematron
 - XSLT, XSL-FO
- JSON
 - Syntax, Vor-/Nachteile, Einsatzgebiete
- REST
 - Adressierbarkeit, Zustandslosigkeit, einheitliche Schnittstelle (uniform interface), Ressource vs. Repräsentation
- Webservices
 - SOAP

6.3 Objektorientierung

- Prinzipien der OOP
 - Begriffe der OOP erläutern: Attribut, Nachricht/Methodenaufruf, Persistenz, Schnittstelle/API/Interface, Polymorphie, Vererbung
 - Bestandteile von Klassen
 - Unterschied Klasse/Objekt
 - Unterschied Klasse/Interface
 - Erklärung Klassenbibliothek vs. Framework
 - Klassenbeziehungen: Assoziation, Aggregation, Komposition, Spezialisierung, Generalisierung
- Unterschied statische/nicht-statische Methoden und Attribute
- Datenstrukturen (Baum, Array)
- funktionale Aspekte in modernen Sprachen: Lambda-Ausdrücke, Functional Interfaces, Map/Filter/Reduce, deklarativ vs. imperativ

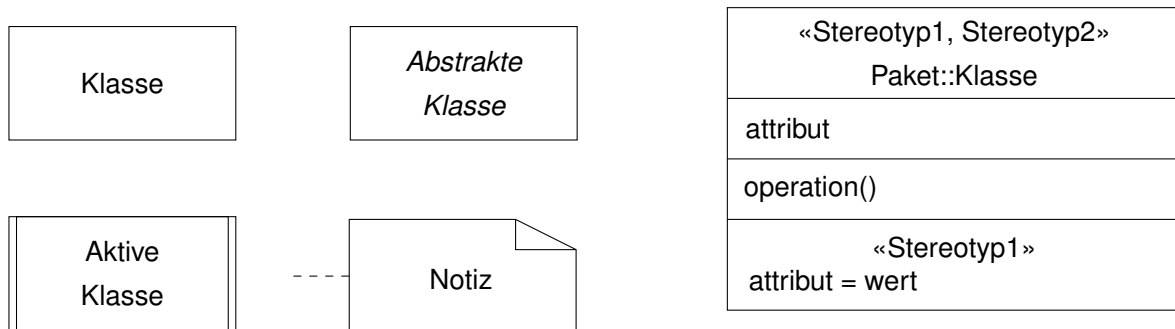
6.4 Programmiersprachen

- Programmierparadigmen: unstrukturiert, strukturiert, prozedural, funktional, objektorientiert, logisch
- imperativ vs. deklarativ
- synchrone vs. asynchrone Programmierung
- Herausforderungen paralleler Programmierung
- Eigenschaften funktionaler Programmierung
 - Pattern Matching

6.5 UML Diagramme

Quelle: Oose.de - Notationsübersicht UML [10] & IHK Belegsatz

6.5.1 Klassendiagramm



Sichtbarkeit:

- Öffentlich (+)
- Privat (-)
- Geschützt (#)
- Paket (~)
- Abgeleitet (/)
- Statisch (unterstrichen)

Syntax für Attribute:

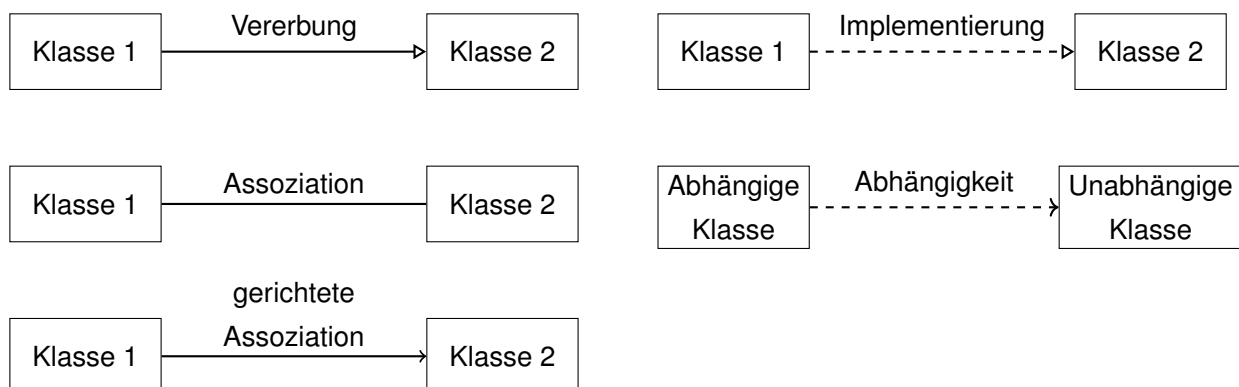
Sichtbarkeit Attributname: Typ {Eigenschaften}

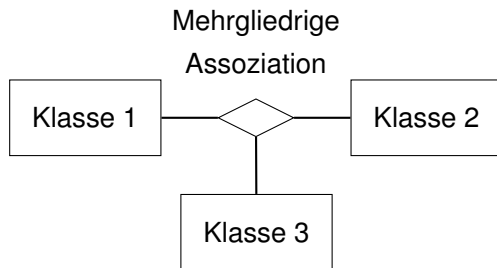
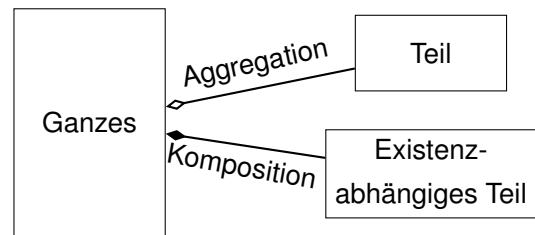
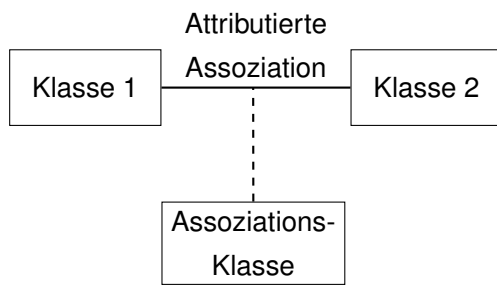
Syntax für Methoden:

Sichtbarkeit Methodenname(paraparameter1: Typ, ...):
Rückgabewert {Eigenschaften}

Eigenschaften:

{static,final, ...}

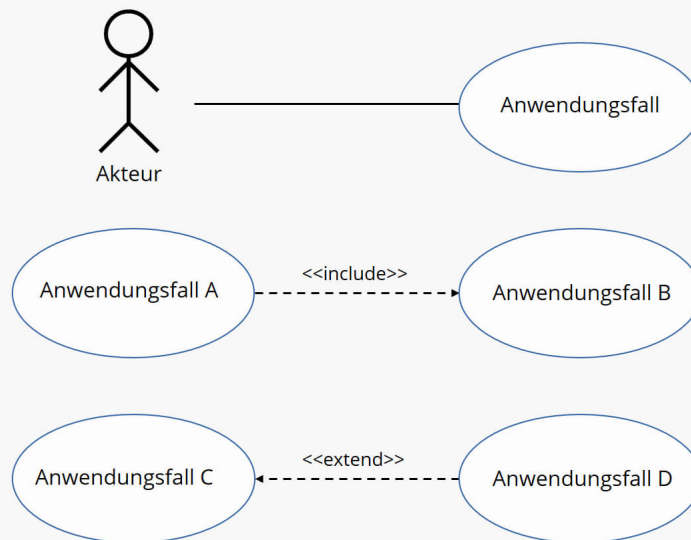




6.5.2 Use-Case-Diagramm (Anwendungsfalldiagramm)

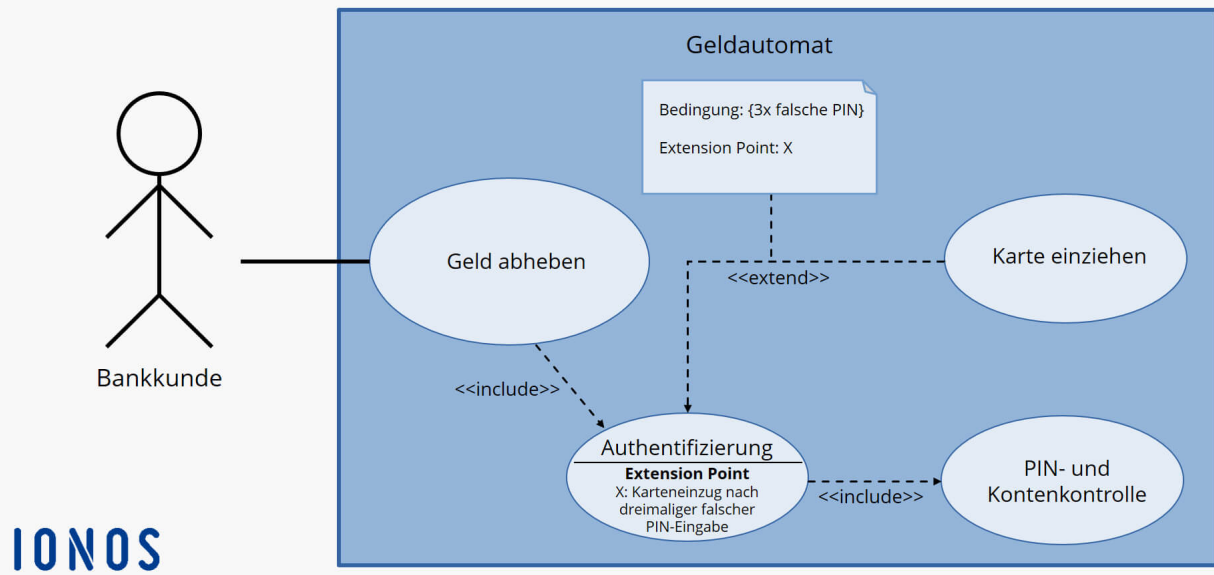
Quelle: Ionos [7]

Assoziationen im UML-Anwendungsfalldiagramm

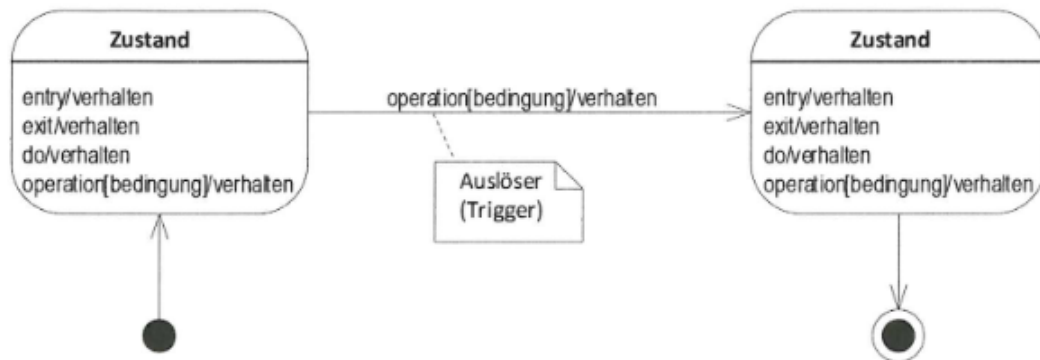


IONOS

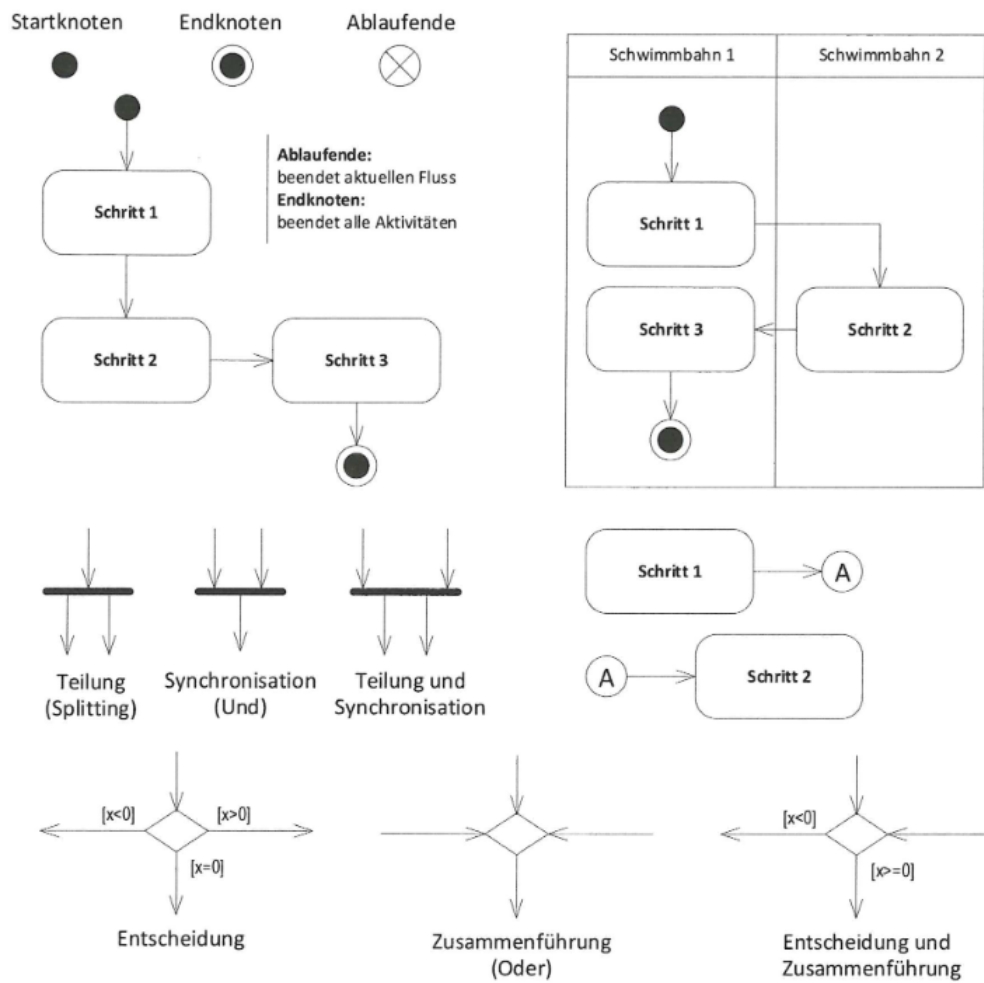
UML-Anwendungsfalldiagramm am Beispiel „Geld abheben“



6.5.3 Zustandsdiagramm



6.5.4 Aktivitätsdiagramm



6.6 Softwarearchitektur

- Bottom-Up- und Top-Down-Verfahren bei der Modellierung erläutern
- Funktion/Vorteile der Modularisierung von Programmen
- Softwarearchitektur
 - 3-Schichten-Modell
 - 7-Schichten-Modell
 - Model View Controller (MVC)
 - Model View Presenter (MVP)
 - Model-View-ViewModel (MVVM)
 - REST

6.7 Softwareergonomie

- Mock-up
- Usability vs. User-Experience
- Entwurf der Bildschirmausgabemasken (Softwareergonomie, Barrierefreiheit)
- Barrierefreiheit bzw. Inklusives Design

6.8 Software Engineering

6.9 Design Patterns

6.10 Softwarequalität

6.11 Webentwicklung

Quellen

- [1] Amazon. Was ist NoSQL? <https://aws.amazon.com/de/nosql/>, 2024.
- [2] Database camp. Normalisierung. <https://databasecamp.de/daten/normalisierung>, May 2023.
- [3] Database camp. Anomalien in Datenbanken. <https://www.datenbanken-verstehen.de/datenmodellierung/normalisierung/anomalien-datenbank/>, 2024.
- [4] Datenbanken-verstehen.de. Entity Relationship Diagram. <https://www.datenbanken-verstehen.de/lexikon/entity-relationship-diagram/>, 2024.
- [5] GeeksforGeeks. SQL | DDL, DQL, DML, DCL and TCL Commands. <https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/>, October 2023.
- [6] Industrie und Handelskammer. Belegsatz. <https://cloud.agb.schule/apps/files/?dir=/Klassen/FIA101/Tauschen/3.%20Jahr/02%20-%20Pr%C3%BCfungsvorbereitung/03%20-%20Sommer%202023&fileid=10240654>, February 2024.
- [7] Ionos. Das Use-Case-Diagramm (Anwendungsfalldiagramm) in UML. <https://www.ionos.de/digitalguide/websites/web-entwicklung/anwendungsfalldiagramm/>, March 2020.
- [8] Lehrerfortbildung-bw.de. Struktogramme. https://lehrerfortbildung-bw.de/u_matnatech/informatik/gym/bp2016/fb1/2_algorithmen/1_hintergrund/2_hintergrund/6_struktogramm/, April 2024.
- [9] Microsoft, Zoiner Tejada. Nicht relationale Daten und NoSQL. <https://learn.microsoft.com/de-de/azure/architecture/data-guide/big-data/non-relational-data>, 2024.
- [10] oose.de. Notationsübersicht UML 2.5. <https://www.oose.de/wp-content/uploads/2012/05/UML-Notations%C3%BCbersicht-2.5.pdf>, 2013.
- [11] Prashanth Jayaram, sqlshack.com. CRUD operations in SQL Server. <https://www.sqlshack.com/crud-operations-in-sql-server/>, December 2018.
- [12] Tino Hempel. Selektion, Projektion und Join in PROLOG. https://www.tinohempel.de/info/info/datenbanken_prolog/abfragen_II.htm, 2006.