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 . . . 5

1.1.4 Welche Arten von NoSQL-Datenbanken gibt es? . . . 5

1.2 Normalisierung . . . 6

1.2.1 Anomalien . . . 7

1.3 SQL . . . 8

1.3.1 Projektion vs. Selektion . . . 8

1.3.2 DDL, DML & DCL . . . 9

1.3.3 CRUD . . . 10

1.3.4 Subqueries . . . 10

1.3.5 Aggregatfunktionen . . . 10

1.3.6 Mengenoperationen (Schnitt-, Vereinigungs- und Differenzmenge) . . . 11

1.3.7 SQL Injection . . . 12

2 Qualitätssicherung **13**

2.1 PDCA-Zyklus . . . 13

2.2 Incident Management . . . 15

2.3 Service Level Agreement (SLA), Servicelevel 1-3 . . . 15

2.4 Testen . . . 16

2.4.1 Klassifizierung von Testverfahren . . . 16

2.5 Versionsverwaltung . . . 17

3 IT-Sicherheit **18**

3.1 Datensicherheit . . . 18

3.1.1 Vertraulichkeit, Integrität, Verfügbarkeit (C.I.A Prinzip) . . . 18

3.1.2 USV (Unterbrechungsfreie Stromversorgung) . . . 19

3.1.3 Firewall . . . 19

3.1.4 Schutzbedarfskategorien . . . 20

3.1.5 Begriffe zu Hacking . . . 20

3.1.6 Hacker . . . 20

3.1.7 Cracker . . . 21

3.1.8 Spam . . . 21

3.1.9 Phishing . . . 21

3.1.10 Spoofing . . . 21

3.1.11 Man-in-the-Middle . . . 21

3.1.12 SQL-Injection . . . 22

3.1.13 Session Hijacking . . . 22

3.1.14 DoS, DDoS . . . 22

3.1.15 Viren . . . 22
3.1.16 Würmer . . . 22
3.1.17 Trojaner . . . 22
3.1.18 Sniffer, Spyware und Keylogger . . . 23
3.1.19 Botnetze . . . 23
3.1.20 Adware . . . 23
3.1.21 Scareware . . . 23
3.1.22 Kryptotrojaner & Ransomware . . . 23
3.1.23 Backdoor . . . 24
3.1.24 Exploit . . . 24
3.1.25 Rootkit . . . 24
3.1.26 Verbreitung von Viren/Würmern/Trojanern . . . 24

4 Datenschutz **24**

5 Netzwerktechnik **26**

6 Softwareentwicklung **27**
6.1 Algorithmen . . . 27
6.1.1 Flussdiagramm . . . 28
6.1.2 Struktogramm (Nassi-Shneiderman-Diagramm) . . . 29
6.2 Schnittstellen, APIs, Datenaustausch . . . 30
6.3 Objektorientierung . . . 30
6.4 Programmiersprachen . . . 31
6.5 UML Diagramme . . . 32
6.5.1 Klassendiagramm . . . 32
6.5.2 Use-Case-Diagramm (Anwendungsfalldiagramm) . . . 33
6.5.3 Zustandsdiagramm . . . 34
6.5.4 Aktivitätsdiagramm . . . 35
6.6 Softwarearchitektur . . . 36
6.7 Softwareergonomie . . . 36
6.8 Software Engineering . . . 36
6.9 Design Patterns . . . 37
6.10 Softwarequalität . . . 37
6.11 Webentwicklung . . . 38

7 Sonstiges **38**
7.1 Umrechung von Datengrößen . . . 38
7.2 Berechnung von Bildgrößen . . . 39

Quellen **40**

1 Datenbanken

IHK Belegsatz [15]

Syntax	Beschreibung
Tabelle	
CREATE TABLE Tabellenname(Spaltenname <DATENTYP>, Primärschlüssel, Fremdschlüssel)	Erzeugt eine neue leere Tabelle mit der beschriebenen Struktur
ALTER TABLE Tabellenname **ADD COLUMN** Spaltenname Datentyp **DROP COLUMN** Spaltenname Datentyp **ADD FOREIGN KEY**(Spaltenname) **REFERENCES** Tabellenname(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** Tabellenname(Primärschlüsselspaltenname)	Erstellung einer Fremdschlüssel-Beziehung
DROP TABLE Tabellenname	Löscht eine Tabelle
Befehle, Klauseln, Attribute	
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.

SELECT **DISTINCT**	Eliminiert Redundanzen, die in einer Tabelle auftreten können. Werte werden jeweils nur einmal angezeigt.
JOIN / **INNER JOIN**	Liefert nur die Datensätze zweier Tabellen, die gleiche Datenwerte enthalten
LEFT JOIN / **LEFT OUTER JOIN**	Liefert von der erstgenannten (linken) Tabelle alle Datensätze und von der zweiten Tabelle jene, deren Datenwerte mit denen der ersten Tabelle übereinstimmen
RIGHT JOIN / **RIGHT OUTER JOIN**	Liefert von der zweiten (rechten) Tabelle alle Datensätze und von der ersten Tabelle jene, deren Datenwerte mit denen der zweiten Tabelle übereinstimmen
WHERE	Bedingung, nach der Datensätze ausgewählt werden sollen
WHERE EXISTS (subquery) **WHERE NOT EXISTS** (subquery)	Die Bedingung EXISTS prüft, ob die Suchbedingung einer Unterabfrage mindestens eine Zeile zurückliefert. NO EXISTS negiert die Bedingung.
WHERE ... IN (subquery) **WHERE NOT ... IN** (subquery)	Der Wert des Datenfelds ist in der ausgewählten Menge vorhanden. Der Wert des Datenfelds ist in der ausgewählten Menge nicht vorhanden.
GROUP BY Spaltenname1 [, Spaltenname2, ...]	Gruppierung (Aggregation) nach Inhalt des genannten Feldes
ORDER BY Spaltenname1 [, Spaltenname2, ...] **ASC** | **DESC**	Sortierung nach Inhalt des genannten Feldes oder der genannten Felder ASC: aufsteigend; DESC: absteigend
Datenmanipulation	
DELETE FROM Tabellenname	Löschen von Datensätzen in der genannten Tabelle
UPDATE Tabellenname **SET**	Aktualisiert Daten in Feldern einer Tabelle
INSERT INTO Tabellenname [(spalte1, spalte2, ...)] **VALUES** (Wert Spalte 1 [, Wert Spalte 2, ...]) oder **SELECT ... FROM ... WHERE**	Fügt Datensätze in die genannte Tabelle ein, die entweder mit festen Werten belegt oder Ergebnis eines SELECT-Befehls sind
Berechtigungen kontrollieren	
CREATE Benutzer | Rolle **IDENTIFIED BY** 'Passwort'	Erzeugt einen neuen Benutzer oder eine neue Rolle mit einem Passwort

GRANT Recht | Rolle **ON** *.* | Datenbank.* | Datenbank.Objekt **TO** Benutzer | Rolle [WITH GRANT OPTION]	Weist einem Benutzer oder einer Rolle ein Recht auf ein bestimmtes Datenbank-Objekt Weist einem Benutzer eine Rolle zu
REVOKE Recht | Rolle **ON** *.* | Datenbank.* | Datenbank.Objekt **FROM** Benutzer | Rolle	Entzieht einem Benutzer oder einer Rolle ein Recht auf ein bestimmtes Datenbank-Objekt Entzieht einem Benutzer eine Rolle
Aggregatfunktionen	
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 Formelergebnisse
MIN (Spaltenname | Formel)	Ermittelt den kleinsten aller Werte im angegebenen Feld
MAX (Spaltenname | Formel)	Ermittelt den größten aller Werte im angegebenen Feld
Funktionen	
LEFT (Zeichenkette, Anzahlzeichen)	Liefert *Anzahlzeichen* der Zeichenkette von links.
RIGHT (Zeichenkette, Anzahlzeichen)	Liefert *Anzahlzeichen* 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 (ausgefrückt 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)

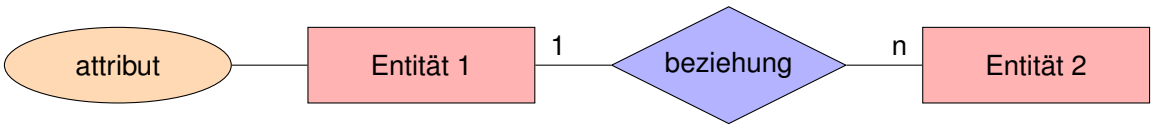
Operatoren	
AND	Logisches UND
LIKE	Überprüfung von Text auf Gleichheit wenn Platzhalter (’regular expressions’) eingesetzt werden.
NOT	Logische Negation
OR	Logisches ODER
IS NULL	Überprüfung auf NULL
=	Test auf Gleichheit
>, >=, <, <=, <>	Test auf Ungleichheit
*	Multiplikation
/	Division
+	Addition, positives Vorzeichen
-	Subtraktion, negatives Vorzeigen

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. [19]

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). [8]

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. [2]

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 [6]

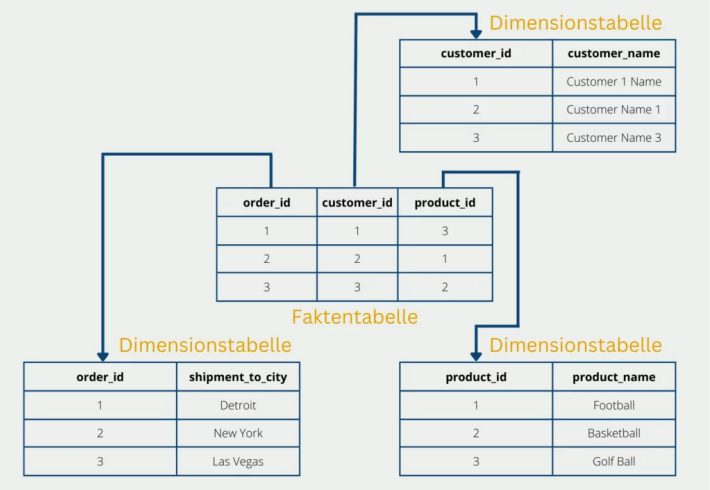
1. Normalform Eine Relation liegt in der ersten Normalform vor, wenn alle Attributwerte atomar vorliegen.

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 [7]

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 Lösch-Anomalie.

In der Datenbanktentwicklung 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 [24]

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 MMüllerßteht. Die Selektion hat also die Form: $S_{Name} =_{'Mueller'} (Schueler)$

Schüler

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

$S_{Name} =_{'Mueller'} (Schueler)$

SNr	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: $P_{Name}(Schueler)$

Schüler

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

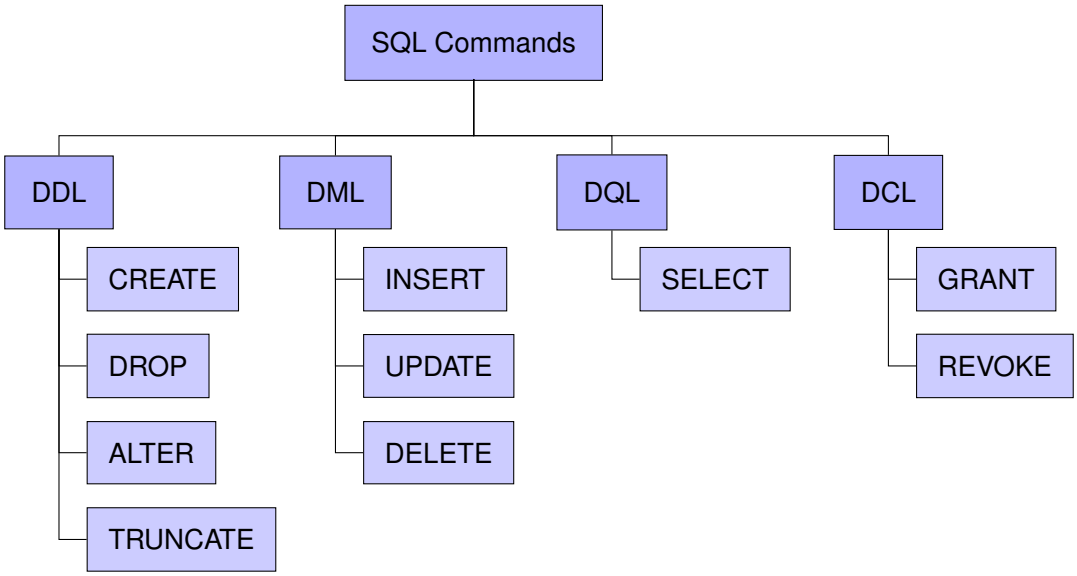
$P_{Name}(Schueler)$

Name
Müller
Schmidt
Lehmann
Müller
Paulsen

1.3.2 DDL, DML & DCL

Quelle: GeeksforGeeks [12]

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 [21]

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.

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

```
INSERT INTO dbo.Demo
(id, name)
VALUES
(2, 'Jayaram'),
(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.

```
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.

```
UPDATE <TableName>
SET Column1=Value1, Column2=Value2,...
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.

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

1.3.4 Subqueries

Fehlt

1.3.5 Aggregatfunktionen

Fehlt

1.3.6 Mengenoperationen (Schnitt-, Vereinigungs- und Differenzmenge)

Quelle: Glossar hs-augsburg [14]

Mengenoperatoren verbinden zwei Abfragen zu einem Resultat.

Beispieltabelle:

Tabelle student

matrikel_nr	name	vorlesung
911574	Meier	Java
676676	Schulz	Datenbanken

Tabelle lehrender

matrikel_nr	name	vorlesung
878999	Kowa	Datenbanken
665544	Müller	XML

UNION bildet die Vereinigung zweier Relationen indem Zeilen der ersten Menge oder des ersten Operanden mit allen Zeilen der zweiten Menge zusammengefasst werden. Zeilen, die in der Ergebnismenge zweimal vorkommen, werden zu einer einzigen Zeile zusammengefasst. Die Datentypen der Spalten müssen kompatibel sein, d.h. es muss entweder ein impliziter Cast (z.B. int auf double) möglich sein, oder wenn dies nicht möglich ist, muß ein expliziter Cast erfolgen. - dies bezieht sich auch auf die Anordnung der Spalten in der Abfrage.

```
1 SELECT name  FROM student
2 UNION
3 SELECT name FROM lehrender
```

Ergebnis:

name
Meier
Schulz
Kowa
Müller

UNION ALL vereinigt alle Zeilen der ersten Menge oder des ersten Operanden mit allen Zeilen der zweiten Menge. Im Unterschied zu UNION werden auch die Duplikate ausgegeben.

INTERSECT überprüft die Zeilen der beiden Eingangsmengen und gibt nur jene Zeilen aus, die in beiden Eingangsmengen vorkommen. Die Durchschnittsmenge wird aus den zwei Relationen gebildete. Auch hier werden vor dem Erstellen der Ergebnismenge die redundanten Zeilen ausgeschaltet.

```
1 SELECT vorlesung FROM student
2 INTERSECT
3 SELECT vorlesung FROM lehrender
```

Ergebnis:

vorlesung
Datenbanken

MINUS gibt die Zeilen aus, die in der ersten Menge, NICHT aber in der zweiten Menge enthalten sind. Zeilen, die in der ersten Menge zweimal vorkommen, werden auf Redundanz überprüft und komprimiert, bevor der Vergleich mit der zweiten Menge beginnt.

```
1 SELECT vorlesung FROM student
2 MINUS
3 SELECT vorlesung FROM lehrender
```

Ergebnis:

vorlesung
Java

1.3.7 SQL Injection

Quelle: kaspersky [17]

Eine SQL-Injection, manchmal abgekürzt als SQLi, ist eine Art von Sicherheitslücke, bei der ein Angreifer einen Teil des SQL-Codes verwendet, um eine Datenbank zu manipulieren und Zugriff auf potenziell wertvolle Informationen zu erhalten. Dies ist eine der häufigsten und bedrohlichsten Angriffsarten, da sie potenziell gegen jede Webanwendung oder Webseite eingesetzt werden kann, die eine SQL-basierte Datenbank verwendet (was bei den meisten der Fall ist).

Arten der SQL-Injection

In-band SQLi Diese Art von SQLi-Angriff ist für Angreifer sehr einfach, da sie denselben Kommunikationskanal verwenden, um Angriffe zu starten und Ergebnisse zu sammeln. Diese Art von SQLi-Angriff hat zwei Untervarianten:

- **Fehlerbasierte SQLi:** Die Datenbank erzeugt aufgrund der Aktionen des Angreifers eine Fehlermeldung. Anhand der von diesen Fehlermeldungen generierten Daten sammelt der Angreifer Informationen über die Datenbankinfrastruktur.
- **Union-basierte SQLi:** Der Angreifer verwendet den UNION-SQL-Operator, um die gewünschten Daten zu erhalten, indem er mehrere Select-Anweisungen in einer einzigen HTTP-Antwort zusammenfasst.

Inferentielle SQLi (auch bekannt als Blind SQL Injection) Bei dieser Art von SQLi nutzen Angreifer die Antwort- und Verhaltensmuster des Servers nach dem Senden von Daten-Nutzdaten, um mehr über seine Struktur zu erfahren. Die Daten werden nicht von der Datenbank der Webseite an den Angreifer übertragen, so dass der Angreifer die Informationen über den In-Band-Angriff nicht sieht (daher der Begriff 'blinde SQLi'). Inferentielle SQLi kann in zwei Untertypen unterteilt werden:

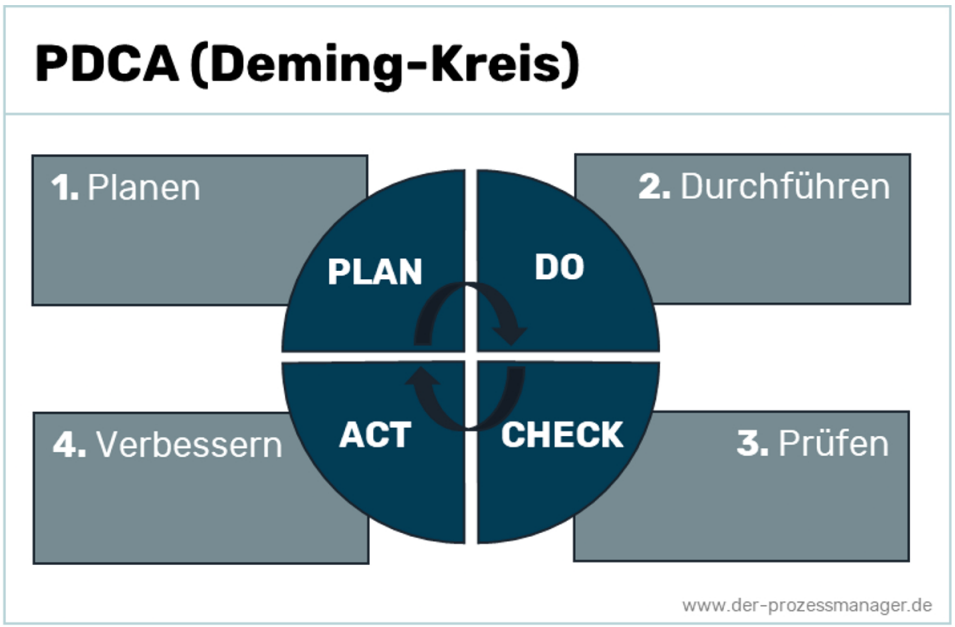
- **Zeitbasierte SQLi:** Angreifer senden eine SQL-Abfrage an die Datenbank und lassen die Datenbank einige Sekunden warten, bevor sie die Abfrage als wahr oder falsch beantwortet.
- **Boolean SQLi:** Angreifer senden eine SQL-Abfrage an die Datenbank und lassen die Anwendung daraufhin entweder ein wahres oder ein falsches Ergebnis erzeugen.

2 Qualitätssicherung

2.1 PDCA-Zyklus

Quelle: der-prozessmanager.de [9]

Der PDCA-Zyklus (auch Deming-Kreis, Deming-Zyklus oder PDCA Kreislauf) bezeichnet ein grundlegendes Konzept im kontinuierlichen Verbesserungsprozess. Es dient der Weiterentwicklung von Produkten und Dienstleistungen sowie bei der Fehler-Ursache-Analyse. Der PDCA-Kreis besteht aus den vier sich wiederholenden Phasen: Plan-Do-Check-Act (dt. Planen – Umsetzen – Überprüfen – Handeln).



Vorteile des PDCA-Kreises

Der wesentliche Vorteil der PDCA-Methode ist wohl die einfache Anwendbarkeit. Das Vorgehen hinsichtlich der spezifischen Aufgaben und Problemstellungen kann nahezu uneingeschränkt angepasst werden. Mit den Schritten Plan, Do, Check und Act bleibt dennoch ein solides Gerüst bestehen.

Weitere Vorteile:

- Benötigt wenig Anleitung auf Grund des einfachen Aufbaus
- Die kreisförmige Konzeption ermöglicht ständige Verbesserung
- Durch den iterativen Ansatz lässt der PDCA-Zyklus Kontrolle und Analyse zu

Nachteile des PDCA-Kreises

Der große Vorteil des Demingkreises ist gleichzeitig auch ein wesentlicher Nachteil: Schnelle Problemlösungen lassen sich mit Hilfe des PDCA-Zyklus nicht umsetzen.

Nachteile auf einen Blick:

- Unklare Definition der einzelnen Schritte kann zu falschem Einsatz führen
- Verbesserungen im Unternehmen müssen langfristig gedacht sein
- Eher ein reaktiver Ansatz, statt proaktiv

2.2 Incident Management

Quelle: it-processmaps.com [22]

Incident Management verwaltet alle Incidents über ihren gesamten Lebenszyklus. Das primäre Ziel dieses ITIL-Prozesses besteht darin, einen IT Service für den Anwender so schnell wie möglich wieder herzustellen.

ITIL unterscheidet zwischen Incidents (Service-Unterbrechungen) und Service Requests (d. h. Anfragen von Anwendern, die keine Service-Unterbrechungen betreffen, wie z.B. das Zurücksetzen eines Passworts). Service-Unterbrechungen werden vom Incident-Management-Prozess behandelt, während Service-Anfragen vom Request Fulfilment bearbeitet werden.

Der Incident-Management-Prozess kann auf verschiedenen Wegen angestoßen werden: Ein Anwender, Kunde oder Supplier kann eine Störung melden, technisches Personal kann einen (drohenden oder tatsächlichen) Ausfall feststellen, oder ein Incident kann automatisch von einem Event-Monitoring-System ausgelöst werden.

Alle Incidents sollten in Incident Records festgehalten werden, so dass ihr Status verfolgt und ihr vollständiger Verlauf dokumentiert werden kann. Die initiale Kategorisierung und Priorisierung der Incidents ist ein wichtiger Schritt zur Bestimmung, wie mit dem Incident verfahren wird und wieviel Zeit für dessen Lösung verfügbar ist.

Falls möglich, sollten Incidents mit anderen Incidents, Problems und Known Errors verknüpft werden.

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

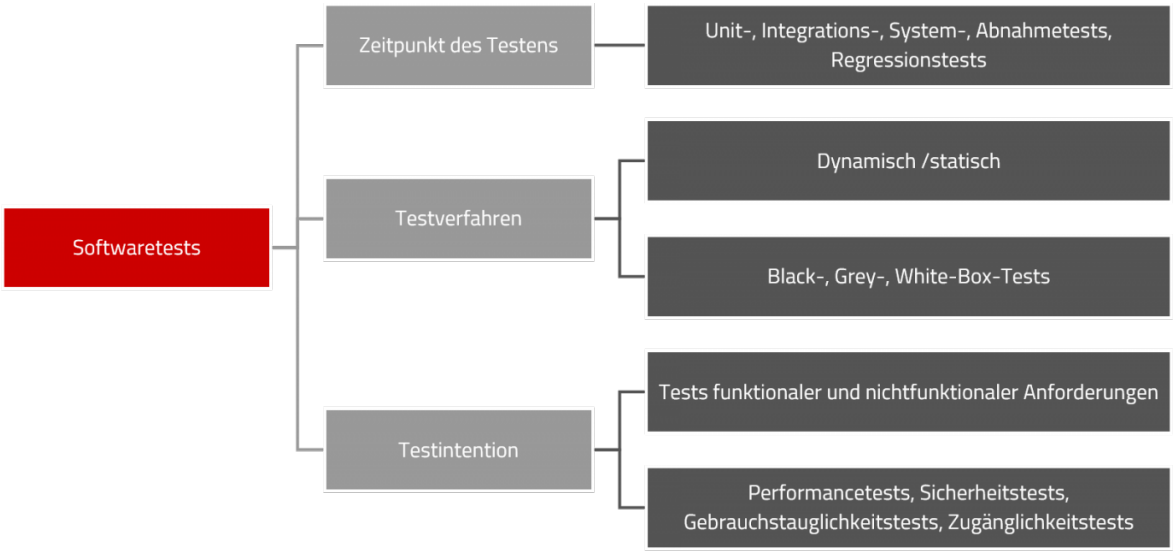
Quelle: Amazon [1]

Ein Service Level Agreement (SLA) ist ein Vertrag mit einem Outsourcing- und Technologieanbieter, in dem das Serviceniveau festgelegt ist, das ein Anbieter dem Kunden zu liefern verspricht. Sie gibt Aufschluss über Kennzahlen wie Betriebszeit, Lieferzeit, Reaktionszeit und Lösungszeit. In einem SLA ist auch festgelegt, was zu tun ist, wenn die Anforderungen nicht erfüllt werden, z. B. zusätzliche Unterstützung oder Preisnachlässe. SLAs werden in der Regel zwischen einem Kunden und einem Service-Anbieter vereinbart, obwohl auch Geschäftseinheiten innerhalb desselben Unternehmens untereinander SLAs abschließen können.

2.4 Testen

2.4.1 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



Quelle: redbots [23]

2.5 Versionsverwaltung

Quelle: Atlassian [3]

git commands

git add	Verschiebt Änderungen aus dem Arbeitsverzeichnis in die Staging-Umgebung. Auf diese Weise kannst du einen Snapshot vorbereiten, bevor du an den offiziellen Verlauf committest.
git branch	Dieser Befehl ist dein Allzwecktool zur Branch-Administration. Damit kannst du isolierte Entwicklungsumgebungen innerhalb eines einzigen Repositorys erstellen.
git checkout	Neben dem Auschecken alter Commits und alter Dateiüberarbeitungen kannst du mit 'git checkout' auch zwischen bestehenden Branches navigieren. In Kombination mit den grundlegenden Git-Befehlen kann dadurch in einer bestimmten Entwicklungslinie gearbeitet werden.
git clone	Erstellt eine Kopie eines bestehenden Git-Repositorys. Klonen ist für Entwickler die gängigste Art, eine Arbeitskopie eines zentralen Repositorys zu erhalten.
git commit	Committet den Snapshot aus der Staging-Umgebung in den Projektverlauf. Zusammen mit 'git add' bildet er den grundlegenden Workflow für alle Git-Benutzer.
git fetch	Mit 'git fetch' wird ein Branch von einem anderen Repository zusammen mit allen zugehörigen Commits und Dateien heruntergeladen. Dabei wird jedoch nichts in dein lokales Repository integriert. Auf diese Weise hast du die Möglichkeit, Änderungen vor dem Merge in dein Projekt noch zu überprüfen.
git init	Initialisiert ein neues Git-Repository. Wenn du für ein Projekt eine Versionskontrolle einrichten möchtest, ist dies der erste Befehl, den du kennen musst.
git log	Damit kannst du ältere Überarbeitungen eines Projekts ansehen. Der Befehl bietet mehrere Formatierungsoptionen zur Anzeige committeter Snapshots.
git merge	Eine leistungsstarke Option zur Integration von Änderungen von voneinander abweichenden Branches. Nach dem Forken des Projektverlaufs mit 'git branch', kann diese mit 'git merge' wieder zusammengeführt werden.
git pull	Pulls sind die automatisierte Version von git fetch. Dabei wird ein Branch von einem Remote-Repository heruntergeladen und dann direkt in den aktuellen Branch gemergt. Dies ist das Git-Äquivalent von svn update.

git push	'git push' ist das Gegenteil von 'git fetch' (mit ein paar Einschränkungen). Du kannst mit diesem Befehl einen lokalen Branch in ein anderes Repository verschieben, was eine bequeme Methode zur Veröffentlichung von Beiträgen ist. Dies ist wie 'svn commit', aber hierbei wird eine Reihe von Commits statt eines einzigen Changesets gesendet.
git rebase	Mit Rebasing kannst du Branches verschieben, um unnötige Merge-Commits zu vermeiden. Der daraus resultierende lineare Verlauf ist oft leichter zu verstehen und zu durchsuchen.
git revert	Macht einen committeten Snapshot rückgängig. Wenn du einen fehlerhaften Commit entdeckst, kannst du ihn mit 'git revert' sicher und einfach von der Codebasis entfernen.
git status	Gibt den Status des Arbeitsverzeichnisses und den Status des Snapshots in der Staging-Umgebung zurück. Diesen Befehl solltest du zusammen mit 'git add' und 'git commit' ausführen, um genau zu sehen, was im nächsten Snapshot enthalten sein wird.

3 IT-Sicherheit

3.1 Datensicherheit

3.1.1 Vertraulichkeit, Integrität, Verfügbarkeit (C.I.A Prinzip)

Quelle: Enginsight [10]

Vertraulichkeit Ein System liefert Vertraulichkeit, wenn niemand unautorisiert Informationen gewinnen kann.

Integrität Ein System gewährleistet die Integrität, wenn es nicht möglich ist, zu schützende Daten unautorisiert und unbemerkt zu verändern.

Verfügbarkeit Ein System gewährt Verfügbarkeit, wenn authentifizierte und autorisierte Subjekte in der Wahrnehmung ihrer Berechtigungen nicht unautorisiert beeinträchtigt werden können.

3.1.2 USV (Unterbrechungsfreie Stromversorgung)

Quelle: hagel-it [13]

Was ist eine USV?

Viele Geräte, wie Server und Router müssen hoch verfügbar sein, um z.B. das Internet am Laufen zu halten. Nicht nur Hard- und Software Probleme, sondern auch die Stromversorgung bildet eine Sicherheitslücke. Daher werden meist sensible IT-Systeme mit einer USV (unterbrechungsfreie Stromversorgung) ausgestattet. Denn im Falle eines Netzausfalls möchte vermieden werden, dass beispielsweise dem Server plötzlich der Strom fehlt und ausgeschaltet wird. Viel mehr möchte man erreichen, dass der Server in diesem Fall sauber und ordentlich herunterfährt.

3.1.3 Firewall

Quelle: cisco [5]

Eine Firewall ist eine Netzwerksicherheitsvorrichtung, die eingehenden und ausgehenden Netzwerkverkehr überwacht und auf Grundlage einer Reihe von definierten Sicherheitsregeln entscheidet, ob bestimmter Datenverkehr zugelassen oder blockiert wird.

Firewalls bilden bereits seit über 25 Jahren die erste Verteidigungslinie beim Schutz von Netzwerken. Sie fungieren als Barriere zwischen geschützten und kontrollierten Bereichen des internen, vertrauenswürdigen Netzwerks und nicht vertrauenswürdigen, äußeren Netzwerken wie dem Internet.

Eine Firewall kann Hardware, Software, Software-as-a-Service (SaaS), eine Public Cloud oder eine Private Cloud (virtuell) sein.

3.1.4 Schutzbedarfskategorien

Quelle: BSI [4]

Beispiel Für das Beispielunternehmen, die RECPLAST GmbH, wurde bezüglich der Schadensszenarien „finanzielle Auswirkungen“ und „Beeinträchtigung der Aufgabenerfüllung“ folgendes festgelegt:

Normaler Schutzbedarf:

- „Der mögliche finanzielle Schaden ist kleiner als 50.000 Euro.“
- „Die Abläufe bei RECPLAST werden allenfalls unerheblich beeinträchtigt. Ausfallzeiten von mehr als 24 Stunden können hingenommen werden.“

Hoher Schutzbedarf:

- „Der mögliche finanzielle Schaden liegt zwischen 50.000 und 500.000 Euro.“
- „Die Abläufe bei RECPLAST werden erheblich beeinträchtigt. Ausfallzeiten dürfen maximal 24 Stunden betragen.“

Sehr hoher Schutzbedarf:

- „Der mögliche finanzielle Schaden liegt über 500.000 Euro.“
- „Die Abläufe bei RECPLAST werden so stark beeinträchtigt, dass Ausfallzeiten, die über zwei Stunden hinausgehen, nicht toleriert werden können.“

3.1.5 Begriffe zu Hacking

Quelle: Buch zu Hacking von mitp [11]

3.1.6 Hacker

Scriptkiddies Sie haben wenig Grundwissen und versuchen, mithilfe von Tools in fremde Systeme einzudringen. Dabei sind diese Tools meist sehr einfach über eine Oberfläche zu bedienen. Die Motivation ist meistens Spaß und die Absichten sind oft krimineller Natur. Oftmals möchten Scriptkiddies mit ihren Aktionen Unruhe stiften. Die Angriffe sind meist ohne System und Strategie. Viele Hacker starten ihre Karriere als Scriptkiddie, nutzen die Tools zunächst mit wenig Erfahrung, lernen aus dem Probieren, entwickeln sich weiter und finden dadurch einen Einstieg in die Szene.

Black Hats Diese Gattung Hacker beschreibt am ehesten die Hacker, die man aus den Medien kennt. Hier redet man von Hackern mit bösen Absichten. Sie haben sehr gute Kenntnisse und greifen bewusst und strukturiert Unternehmen, Organisationen oder Einzelpersonen an, um diesen Schaden zuzufügen. Die Ziele der Black Hats sind vielfältig und reichen vom einfachen Zerstören von Daten bis hin zum Diebstahl von wertvollen Informationen, wie Kontodaten oder Unternehmensgeheimnissen. In manchen Fälle reicht es den Black Hats auch, wenn sie erfolgreich die Server ihres Opfers lahmlegen und damit Sabotage verüben.

White Hats Einen *White Hat Hacker* nennt man oft auch einen *Ethical Hacker*. Er nutzt das Wissen und die Tools eines Hackers, um zu verstehen, wie Black Hats bei ihren Angriffen vorgehen. Im Gegensatz zum Black Hat will der White Hat jedoch die betreffenden Systeme letztlich vor Angriffen besser schützen und testet daher die Schwachstellen aktiv aus. Damit hat ein White Hat Hacker grundsätzlich keine bösen Absichten, im Gegenteil, er unterstützt die Security-Verantwortlichen der jeweiligen Organisation. White Hat Hacker oder Ethical Hacker versuchen im Anschluss an ihre Hacking-Tätigkeit herauszufinden, welche Sicherheitslücken es gibt und geben eine Anleitung dazu, diese möglichst effizient zu schließen.

3.1.7 Cracker

Fehlt

3.1.8 Spam

Fehlt

3.1.9 Phishing

Fehlt

3.1.10 Spoofing

Fehlt

3.1.11 Man-in-the-Middle

Fehlt

3.1.12 SQL-Injection

1.3.7 SQL Injection auf Seite 12

3.1.13 Session Hijacking

Fehlt

3.1.14 DoS, DDoS

Fehlt

3.1.15 Viren

Der oder das Computervirus (beides ist mittlerweile statthaft) ist klassischerweise ein Programm, das sich selbst verbreitet, indem es sich in ein anderes programm, den Wirt, einschleust. Im Gegensatz zu Trojanern oder Würmern benötigen Viren also einen Wirt und können sich nicht selbstständig fortpflanzen.

In dem Moment, in dem das Wirtsprogramm ausgeführt wird, kann auch der Virus aktiv werden und seine Payload ausführen. Zur Replikation führt der Virus immer einen Prozess aus, der es ihm ermöglicht, sich an einen neuen Wirt zu hängen, um sich weiterzuverbreiten.

3.1.16 Würmer

Würmer können sich - im Gegensatz zu Viren - selbstständig und ohne Wirt vermehren. Sie verbreiten sich über Netzwerke oder Wechselmedien, wie USB-Sticks. Dadurch können sie sich höchst effektiv replizieren und verbreiten.

Ebenso wie Trojaner und Viren können auch Würmer einen Schadcode ausführen.

3.1.17 Trojaner

Die Trojaner werden als harmloses Programm getrarnt, enthalten aber Schadcode, der im Hintergrund ohne Wissen des Anwenders ausgeführt wird. Trojaner sind die flexibelste und die am weitesten verbreitete Form von Malware. Im Grunde genommen ist ein Trojaner nur eine Tarnung für eine beliebige weitere Malware-Funktion wie z.B. Installation von:

- Backdoors
- Keylogger
- Ransomware
- Sniffer
- Bots für ein Botnet
- und so weiter

3.1.18 Sniffer, Spyware und Keylogger

Zahlreiche Schadprogramme versuchen, den Anwender auszuspionieren, und senden die Daten anschließend an vordefinierte Zielsysteme im Internet. Die Spionagetätigkeiten können diverse Aspekte und Komponenten umfassen. Angefangen von Daten wie Browser-Verlauf und anderen Zugriffsverläufen (*Most Recently Used*, MRU) über das Mitschneiden des Netzwerk-Traffics oder Aktivieren von Mikrofon und Webcam bis hin zum Protokollieren jedes einzelnen Tastenanschlags mittels Keylogger kann ein Angreifer umfassende Daten über das Opfer sammeln.

3.1.19 Botnetze

Ein *Botnet* besteht aus zahlreichen *Bots*, also einem Stückchen Software, das sich im Opfer-System eingenistet hat und bereit ist, vom Command & Control-Server aus dem Internet Befehle zu empfangen. Bots könenn diverse Funktionen erfüllen - angefangen vom Spamversand über Spyware-Funktionen bis hin zu gezielten und konzentrierten *Distributed-Denial-of-Service-Angriffen*.

3.1.20 Adware

Fehlt

3.1.21 Scareware

Eine perfide Art, Malware auf ein Opfer-System zu bringen, ist die *Scareware*. Hierbei täuscht ein Popup einer Website oder eine als kostenloses Antiviren-Programm verteilte Software dem Benutzer den Fund zahlreicher, gefährlicher Vireninfektionen vor, die über ein zu installierendes, ggf. kostenpflichtiges Programm beseitigt werden können.

3.1.22 Kryptotrojaner & Ransomware

Kryptografie soll die Sicherheit erhöhen. Dank ausgeklügelter, komplexer mathematischer Verfahren und Algorithmen gelingt dies in der Regel sehr effektiv und für die »bösen Jungs« ist es teilweise sehr schwierig bis unmöglich, kryptografisch gesicherte Daten und Datenströme zu knacken bzw. zu entschlüsseln. Umso perfider wird es, wenn diese Technologien gegen uns verwendet werden. So geschehen bei den sogenannten *Kryptotrojanern* bzw. der gefürchteten *Ransomware*.

Hinter Kryptotrojanern steckt sehr viel kriminelle Energie, denn die Angreifer wollen in fast allen Fällen Geld ergaunern. Die Schädlinge verschlüsseln persönliche Daten auf dem System des Opfers so, dass sie für den Eigentümer unbrauchbar sind. Es gibt auch Varianten, die den Zugriff auf das komplette System blockieren. Erst gegen Bezahlung eines Lösegeldes (engl. *Ransom*) werden die Dateien entschlüsselt und damit wieder verfügbar gemacht. Nachdem das Opfer über eine anonyme Zahlungsmethode, wie zum

Beispiel die Kryptowährung *Bitcoin*, das Lösegeld bezahlt hat, ist natürlich nicht gewährleistet, dass die Daten tatsächlich wieder freigegeben werden!

3.1.23 Backdoor

Eine »Backdoor«, oder deutsch: »Hintertür«, bezeichnet ganz allgemein einen Zugang zu einem System unter Umgehung der Zugriffsschutzmaßnahmen. Der Zweck einer Backdoor besteht in der Regel darin, einem nicht authentisierten und autorisierten Benutzer einen jederzeit verfügbaren Zugriff zum Zielsystem bereitzustellen. Dies impliziert, dass dieser Zugang den autorisierten Benutzern meistens nicht bekannt ist bzw. sein soll.

3.1.24 Exploit

Ein *Exploit* ist ein Prozess, mit dem Sie eine Schwachstelle ausnutzen können. Doch was bedeutet »ausnutzen« eigentlich? Das hängt ganz von der Schwachstelle und vom Exploit ab. In unserem Fall geht es darum, eine *Remote-Shell* bereitzustellen - egal ob Bind- oder Reverse-Shell. Diese Shell ist die *Payload* des Exploits. Durch die Schwachstelle wird z.B. die Ausführung beliebiger Kommandos möglich. Der Exploit setzt dies technisch um. Die Payload stellt dann genau diese Befehle dar, mit denen der Angreifer Zugriff auf das System erlangt.

3.1.25 Rootkit

Eine sehr effektive Möglichkeit, Malware und unerwünschte Prozesse zu verstecken und zu tarnen, besteht darin, sie für den Anwender und das System unsichtbar zu machen. Der Begriff »Rootkit« setzt sich aus dem Linux/UNIX-Administator *root* und dem Wort »Kit« zusammen und bedeutet wörtlich übersetzt ungefähr »Administrator-Werkzeugkasten«. Damit wird zum Ausdruck gebracht, dass es sich eigentlich um eine Toolsammlung handelt.

3.1.26 Verbreitung von Viren/Würmern/Trojanern

Fehlt

4 Datenschutz

Fehlt

- 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

Fehlt

- 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

Fehlt

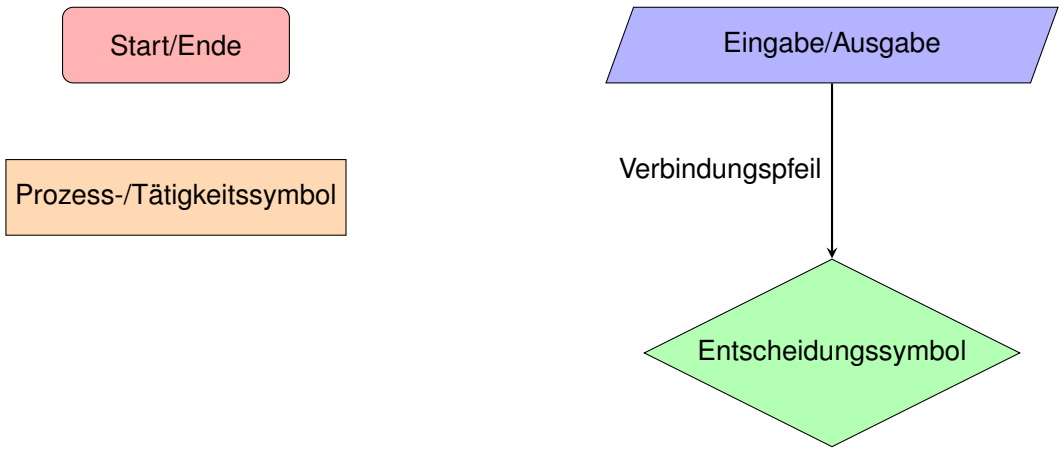
- Allgemeines Fehlerhandling in Programmen
  - Exceptions, Return/Exit Codes
  - Unterschied syntaktische/semantische Fehler
- Rechnerarchitektur: CPU, BUS, Speicher und deren Adressierung
- Lizenzen unterscheiden
  - Open Source, proprietär
- Informationspflichten zu Produkten, Namens- und Markenrecht,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

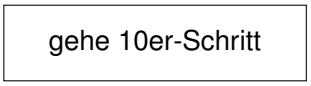
6.1.1 Flussdiagramm



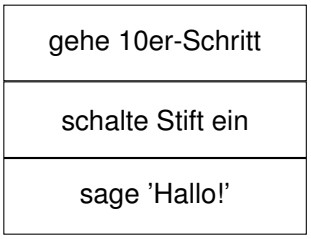
6.1.2 Struktogramm (Nassi-Shneiderman-Diagramm)

Quelle: Lehrerfortbildung-bw.de [18]

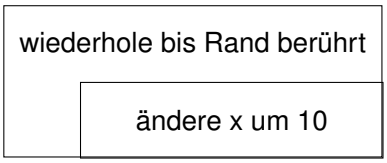
Anweisung



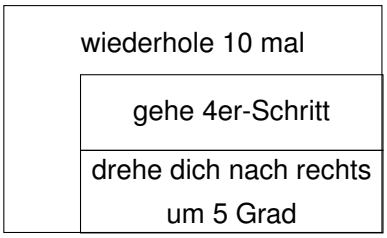
Sequenz



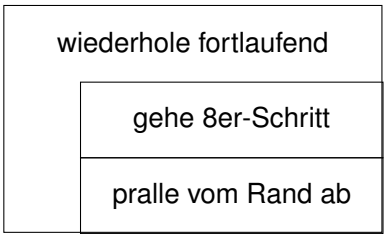
Schleife mit Bedingung



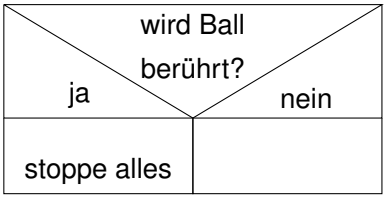
Schleife mit Zähler



Endlosschleife



Verzweigung mit Alternative



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

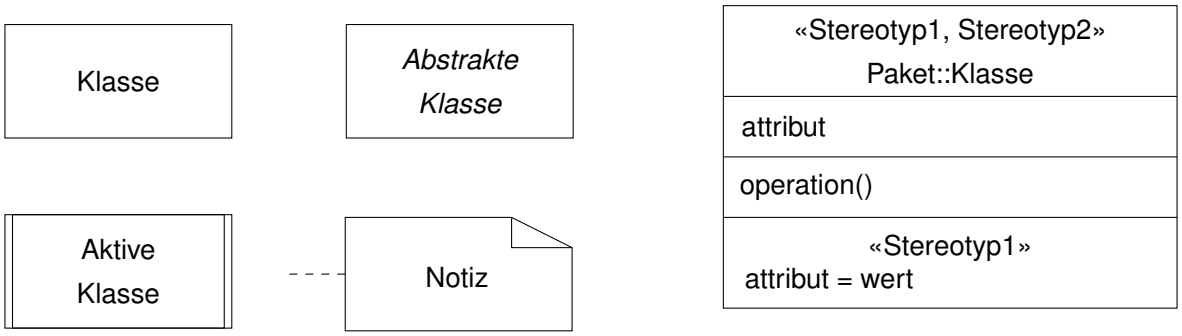
6.4 Programmiersprachen

- 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 [20] & IHK Belegsatz

6.5.1 Klassendiagramm



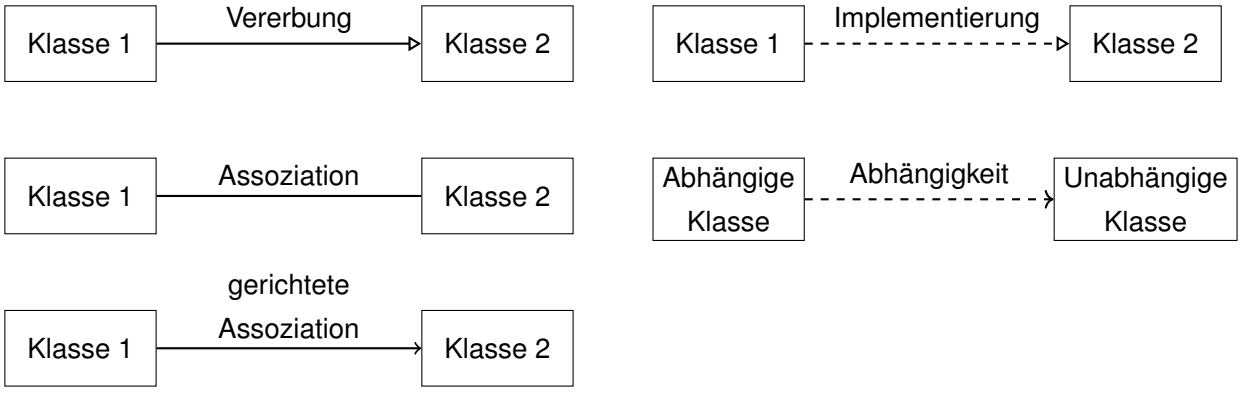
Sichtbarkeit:

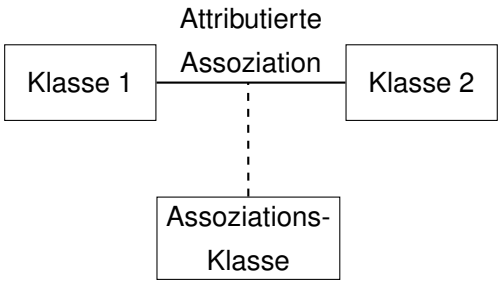
- Öffentlich (+)
- Privat (-)
- Geschützt (#)
- Paket (~)
- Abgeleitet (/)
- Statisch (unterstrichen)

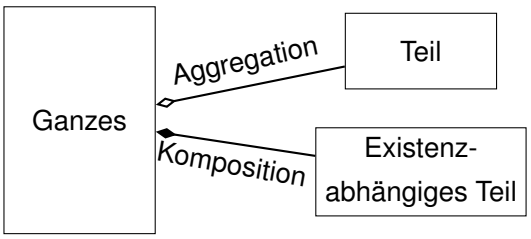
Syntax für Attribute:
Sichtbarkeit Attributname: Typ {Eigenschaften}

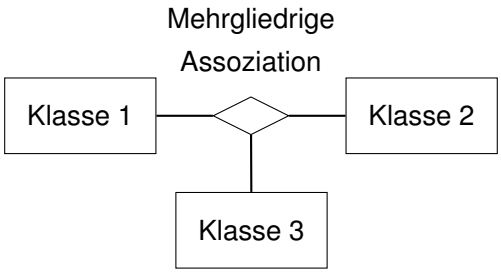
Syntax für Methoden:
Sichtbarkeit Methodenname(parpameter1: Typ, ...): Rückgabewert {Eigenschaften}

Eigenschaften:
{static,final, ...}



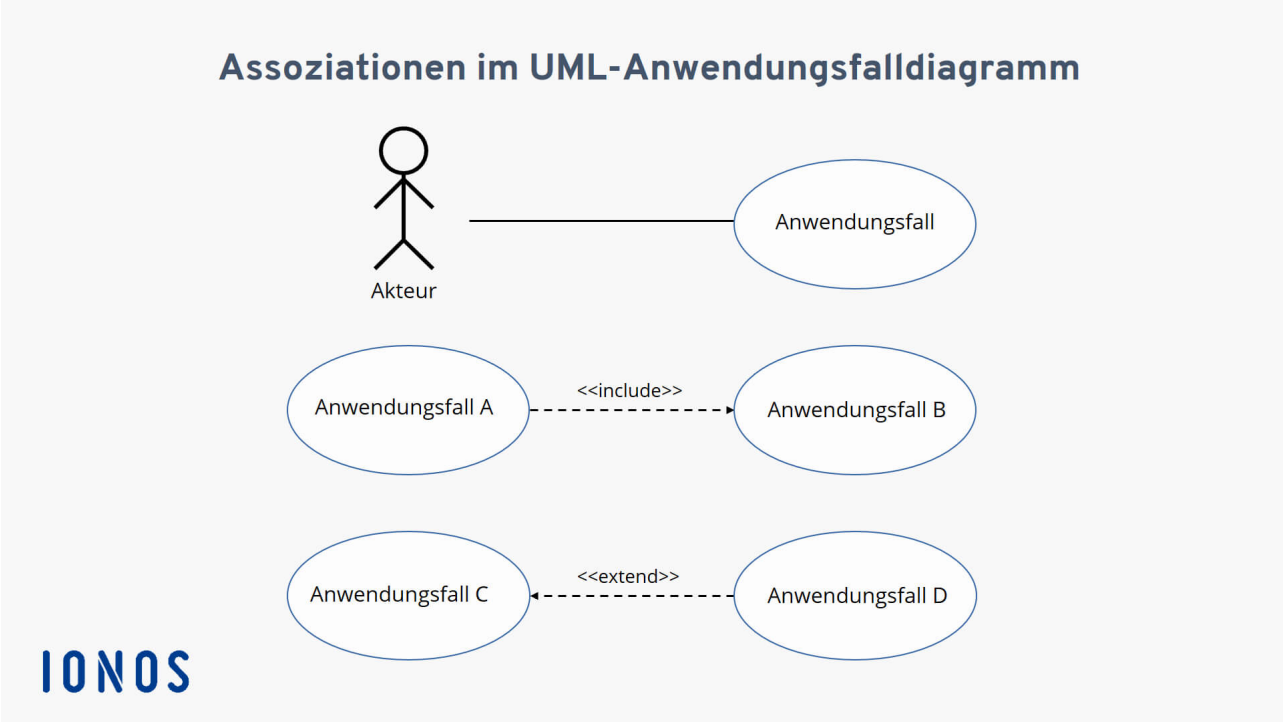


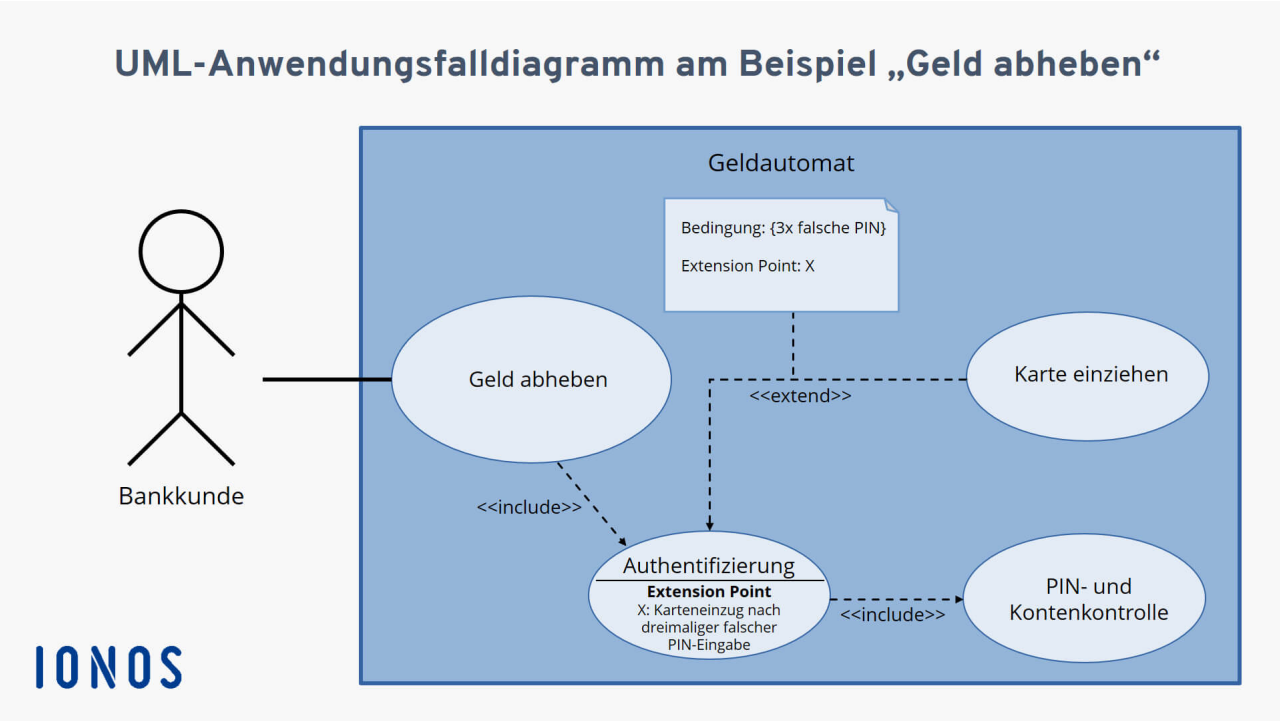




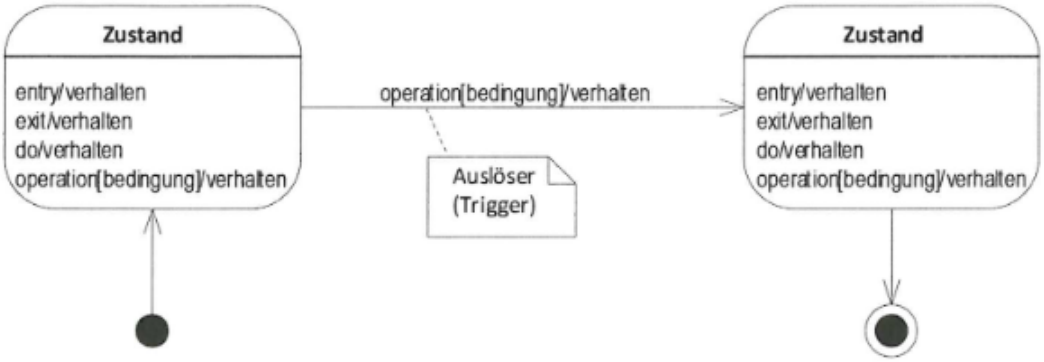
6.5.2 Use-Case-Diagramm (Anwendungsfalldiagramm)

Quelle: Ionos [16]

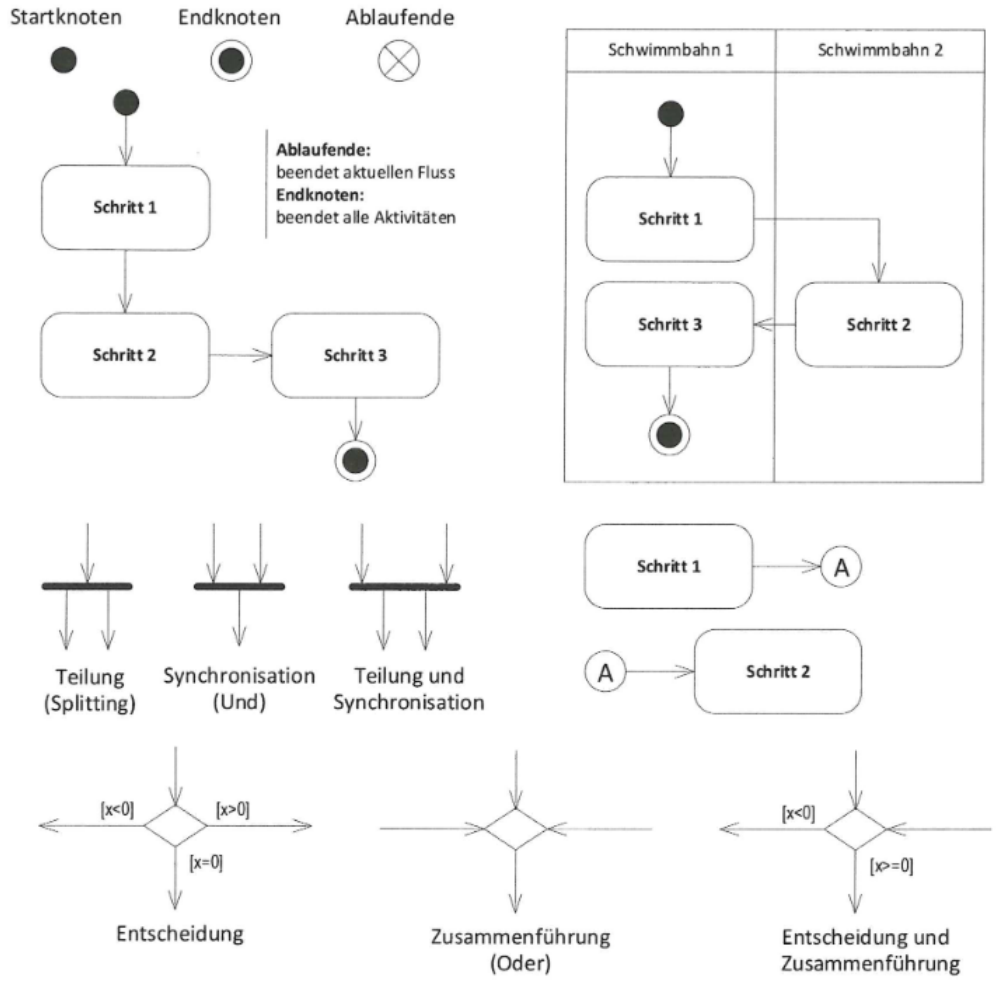




6.5.3 Zustandsdiagramm



6.5.4 Aktivitätsdiagramm



Fehlt

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

- Entwicklungsprozesse wie das Wasserfallmodell
- Iterative Modelle, z.B. Spiralmodell, V-Modell (XT)
- Agile Modelle: Scrum, Extreme Programming, Kanban
- Top-Down-Entwurf vs. Bottom-Up-Entwurf

6.9 Design Patterns

- Design Patterns kennen/erklären/implementieren
  - Singleton
  - Observer
  - Factory
  - Strategy
  - Decorator
  - MVC

6.10 Softwarequalität

- Software-Qualitätsmerkmale nach ISO 9126 nennen und erläutern
  - Funktionalität: Angemessenheit, Interoperabilität, Ordnungsmäßigkeit, Richtigkeit, Sicherheit
  - Änderbarkeit: Analysierbarkeit, Modifizierbarkeit, Testbarkeit, Stabilität
  - Übertragbarkeit: Anpassbarkeit, Austauschbarkeit, Installierbarkeit, Koexistenz
  - Effizienz: Verbrauchsverhalten, Zeitverhalten
  - Zuverlässigkeit: Fehlertoleranz, Reife, Wiederherstellbarkeit
  - Benutzbarkeit: Attraktivität, Bedienbarkeit, Erlernbarkeit, Verständlichkeit
- Software-Qualitätsmerkmale nach ISO 25010 nennen und erläutern
  - Functional Suitability: Functional Completeness, Functional Correctness, Functional Appropriateness
  - Performance Efficiency: Time Behaviour, Resource Utilization, Capacity
  - Compatibility: Co-existence, Interoperability
  - Usability: Appropriateness Recognizability, Learability, Operability, User Error Protection, User Interface Aesthetics, Accessibility
  - Reliability: Maturity, Availability, Fault Tolerance, Recoverability
  - Security: Confidentiality, Integrity, Non-repudiation, Authenticity, Accountability
  - Maintainability: Modularity, Reusability, Analysability, Modifiability, Testability
  - Portability: Adaptability, Installability, Replaceability
- Maßnahmen zur Qualitätssicherung

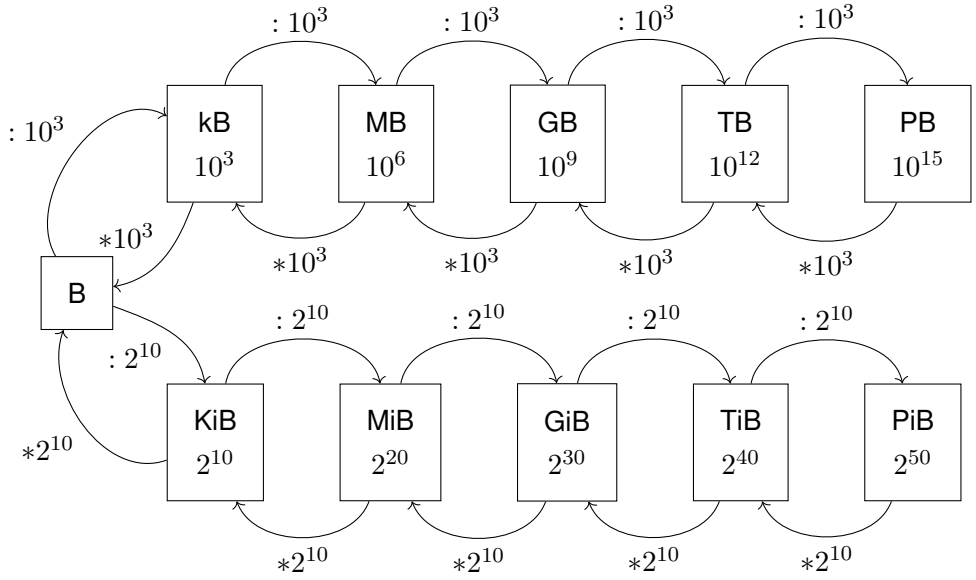
- Continuous Integration/Delivery/Deployment

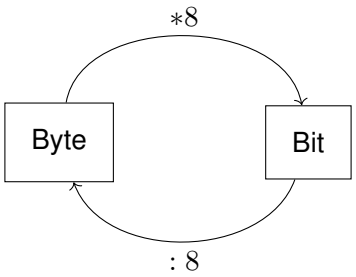
6.11 Webentwicklung

- Web 2.0
  - Social Networks, Wikis, Blogs, Twitter, Forum, Podcast
- Web 3.0
- Angriffsmöglichkeiten gegen Anwendungen abgrenzen
  - SQL-Injection, Session Hijacking, DoS, DDoS

7 Sonstiges

7.1 Umrechung von Datengrößen





7.2 Berechnung von Bildgrößen

Farbtiefe, RGB, Pixel, DPI, etc.

Quellen

[1] Amazon. Was ist SLA (Service Level Agreement)? https://aws.amazon.com/de/what-is/service-level-agreement/, 2023.

[2] Amazon. Was ist NoSQL? https://aws.amazon.com/de/nosql/, 2024.

[3] Atlassian.com. Git kennenlernen: Git-Befehle. https://www.atlassian.com/de/git/glossary#commands, 2024.

[4] BSI. Schutzbedarfskategorien. https://www.bsi.bund.de/dok/10990084, 2024.

[5] cisco. Was ist eine Firewall? https://www.cisco.com/c/de_de/products/security/firewalls/what-is-a-firewall.html, 2024.

[6] Database camp. Normalisierung. https://databasecamp.de/daten/normalisierung, May 2023.

[7] Database camp. Anomalien in Datenbanken. https://www.datenbanken-verstehen.de/datenmodellierung/normalisierung/anomalien-datenbank/, 2024.

[8] Datenbanken-verstehen.de. Entity Relationship Diagram. https://www.datenbanken-verstehen.de/lexikon/entity-relationship-diagram/, 2024.

[9] der-prozessmanager, Michael Durst, Sascha Hertkorn, Christopher Eischer, Nico Schweisser. Was ist ein PDCA-Zyklus? Plan-Do-Check-Act einfach erklärt. https://der-prozessmanager.de/aktuell/wissensdatenbank/pdca-zyklus, 2021.

[10] Enginsight, Max Tarantik. Vertraulichkeit, Integrität, Verfügbarkeit – Einfach erklärt! https://enginsight.com/de/blog/vertraulichkeit-integritaet-verfuegbarkeit-einfach-erklaert/, January 2022.

[11] Eric Amberg, Daniel Schmid. *Hacking; Der umfassende Praxis-Guide Inkl. Prüfungsvorbereitung zum CEHv11*. mitp, Frechen, 2. auflage edition, 2022.

[12] GeeksforGeeks. SQL | DDL, DQL, DML, DCL and TCL Commands. https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/, October 2023.

[13] hagel-it, Sebastian Hansen. WAS IST EINE USV ? https://www.hagel-it.de/it-service/was-ist-eine-usv.html, 2024.

[14] hs-augsburg, Kowa. Mengenoperatoren in SQL. https://glossar.hs-augsburg.de/Mengenoperatoren_in_SQL, July 2012.

[15] 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.

[16] Ionos. Das Use-Case-Diagramm (Anwendungsfalldiagramm) in UML. https://www.ionos.de/digitalguide/websites/web-entwicklung/anwendungsfalldiagramm/, March 2020.

[17] kaspersky. Was ist SQL-Injection? Definition und Erläuterung. https://www.kaspersky.de/resource-center/definitions/sql-injection, 2024.

[18] Lehrerfortbildung-bw.de. Struktogramme. https://lehrerfortbildung-bw.de/u_matnatech/informatik/gym/bp2016/fb1/2_algorithmen/1_hintergrund/2_hintergrund/6_struktogramm/, April 2024.

[19] Microsoft, Zoiner Tejada. Nicht relationale Daten und NoSQL. https://learn.microsoft.com/de-de/azure/architecture/data-guide/big-data/non-relational-data, 2024.

[20] oose.de. Notationsübersicht UML 2.5. https://www.oose.de/wp-content/uploads/2012/05/UML-Notations%C3%BCbersicht-2.5.pdf, 2013.

[21] Prashanth Jayaram, sqlshack.com. CRUD operations in SQL Server. https://www.sqlshack.com/crud-operations-in-sql-server/, December 2018.

[22] processmaps.com, Stefan Kempter. Incident Management. https://wiki.de.it-processmaps.com/index.php/Incident_Management, 2024.

[23] redbots.de, Thomas Klein, Elena Semenova. Tests in der Softwareentwicklung: Ein Klassifizierungsansatz. https://www.redbots.de/blog/software-tests-klassifizierung/, 2021.

[24] Tino Hempel. Selektion, Projektion und Join in PROLOG. https://www.tinohempel.de/info/info/datenbanken_prolog/abfragen_II.htm, 2006.