HAW HAMBURG

BACHELORARBEIT

Performance – Optimierung von Datenbanken

vorgelegt am 26. März 2022

Daniel Freire Mendes

Erstprüferin: Prof. Dr. Stefan Sarstedt
Zweitprüfer: Prof. Dr. Olaf Zukunft

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Department Informatik
Berliner Tor 7
20099 Hamburg

Zusammenfassung

Der Arbeit beginnt mit einer kurzen Beschreibung ihrer zentralen Inhalte, in der die Thematik und die wesentlichen Resultate skizziert werden. Diese Beschreibung muss sowohl in deutscher als auch in englischer Sprache vorliegen und sollte eine Länge von etwa 150 bis 250 Wörtern haben. Beide Versionen zusammen sollten nicht mehr als eine Seite umfassen. Die Zusammenfassung dient u. a. der inhaltlichen Verortung im Bibliothekskatalog.

Abstract

The thesis begins with a brief summary of its main contents, outlining the subject matter and the essential findings. This summary must be provided in German and in English and should range from 150 to 250 words in length. Both versions combined should not comprise more than one page. Among other things, the abstract is used for library classification.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Überblick

1.1 Einführung in Benchmarks

Benchmarks dienen dazu, praktisch und effektiv zu untersuchen, wie sich ein System unter Last verhält. Die wichtigste Erkenntnis, die man aus Benchmarks gewinnen kann, sind die Probleme und Fehler, die man systematisch dokumentieren und nach Priorität abarbeiten sollte. Das Ziel von Benchmarks ist die Reduzierung und Bewertung von unerwünschtem Verhalten sowie die Analyse, wie sich das System derzeit und unter simulierten, zukünftigen, anspruchsvolleren Bedingungen verhalten könnte.

Es gibt zwei verschiedene Techniken für Benchmarks. Die erste zielt darauf ab, die Applikation als Ganzes zu testen (full-stack). Dabei wird nicht nur die Datenbank getestet, sondern die gesamte Applikation, einschließlich des Webservers, des Netzwerks und des Applikationscodes. Der Ansatz dahinter ist, dass ein Nutzer genauso lange auf eine Abfrage warten muss, wie das gesamte System benötigt. Daher sollte diese Wartezeit so gering wie möglich sein. Es kann dabei vorkommen, dass MySQL nicht immer das Bottleneck ist.[1]

Full-Stack-Benchmarks haben jedoch auch Nachteile. Sie sind schwieriger zu erstellen und insbesondere schwieriger korrekt einzurichten. Wenn man lediglich verschiedene Schemas und Abfragen in MySQL auf ihre Performance testen möchte, gibt es sogenannte Single-Component-Benchmarks. Diese analysieren ein spezifisches Problem in der Applikation und sind deutlich einfacher zu erstellen. Ein weiterer Vorteil besteht darin, dass nur ein Teil des gesamten Systems getestet wird, wodurch die Antwortzeiten kürzer sind und man schneller Ergebnisse erhält.

Wenn bei Benchmarks schlechte Designentscheidungen getroffen werden, kann dies zu einer falschen Interpretation des Systems führen, da die Ergebnisse nicht die Realität widerspiegeln. Die Größe des Datensatzes und des Workloads muss realistisch sein. Idealerweise verwendet

[1] Gemeint ist ein Engpass beim Transport von Daten oder Waren, der maßgeblichen Einfluss auf die Abarbeitungsgeschwindigkeit hat. Optimierungsversuche an anderer Stelle führen oft nur zu geringen oder gar keinen messbaren Verbesserungen der Gesamtsituation. (Vogel, 2009)

man einen Snapshot[2] des tatsächlichen produktiven Datensatzes. Gibt es keine Produktionsdaten, sollten die Daten und der Workload simuliert werden, da realistische Benchmarks komplex und zeitaufwendig sein können.

Häufige Fehler beim Durchführen von Benchmarks sind unter anderem, dass nur ein kleiner Teil der tatsächlichen Datensatzgröße verwendet wird und die Datensätze unkorrekt gleichmäßig verteilt sind. In der Realität können Hotspots auftreten. Bei zufällig generierten Werten kommt es hingegen häufig zu unrealistisch gleichmäßig verteilten Datensätzen. Ein weiterer Fehler besteht darin, dass man beim Testen einer Anwendung nicht das tatsächliche Benutzerverhalten nachstellt. Wenn gleiche Abfragen in einer Schleife ausgeführt werden, muss man außerdem auf das Caching achten, da sonst falsche Annahmen über die Performance getroffen werden können. Zudem wird oft die Warmmachphase des Systems vollständig ignoriert. Kurze Benchmarks können schnell zu falschen Annahmen über die Performance des Systems führen.

Um verlässliche Ergebnisse zu erhalten, sollte ein Benchmark ausreichend lange laufen, um den stabilen Zustand des Systems zu beobachten, insbesondere bei Servern mit großen Datenmengen und viel Speicher. Dabei ist es wichtig, so viele Informationen wie möglich zu erfassen und sicherzustellen, dass der Benchmark wiederholbar ist, da unzureichende oder fehlerhafte Tests wertlos sind. Außerdem ist es wichtig, die Ergebnisse in einem Diagramm darzustellen, da auftretende Phänomene sonst anhand einer tabellarischen Darstellung nicht erkannt werden können.

1.2 Measures

- **Durchsatz (Throughput):** Der Durchsatz ist die Anzahl an Transaktionen pro Zeiteinheit. Er ist standardisiert, und Datenbankanbieter versuchen, diesen zu optimieren. Meistens werden Transaktionen pro Sekunde (oder manchmal pro Minute) als Einheit verwendet.
- **Antwortzeiten (Latenz):** Die Antwortzeit misst die gesamte Zeit, die für eine Abfrage benötigt wird. Diese kann, abhängig von der Applikation, in Mikrosekunden (µs), Millisekunden (ms), Sekunden oder Minuten angegeben werden. Von dieser Zeit können aggregierte Antwortzeiten wie Durchschnitt, Maximum, Minimum und Perzentile abgeleitet werden. Das Maximum ist oft eine weniger sinnvolle Metrik, da es sich nicht gut wiederholen lässt. Daher nutzt man eher Perzentile bei den Antwortzeiten. Wenn beispielsweise das 95. Perzentil der Antwortzeit bei 5 ms liegt, bedeutet dies, dass mit einer Wahrscheinlichkeit von 95 % die Abfrage in weniger als 5 ms abgeschlossen ist.

[2] Snapshots bestehen größtenteils aus Metadaten, die den Zustand Ihrer Daten definieren, und sind keine vollständige Duplikation der Daten auf Ihrer Festplatte. Snapshots werden häufig für Test-/Entwicklungsaufgaben verwendet. (Germany, 2024)

- **Nebenläufigkeit (Concurrency):** Die Nebenläufigkeit auf dem Webserver lässt sich nicht zwangsläufig auf den Datenbankserver übertragen. Eine genauere Messung der Gleichzeitigkeit auf dem Webserver besteht darin, zu bestimmen, wie viele gleichzeitige Anfragen zu einem bestimmten Zeitpunkt ausgeführt werden. Es kann auch geprüft werden, ob der Durchsatz sinkt oder die Antwortzeiten steigen, wenn die Gleichzeitigkeit zunimmt. Beispielsweise benötigt eine Website mit „50.000 Benutzern gleichzeitig“ vielleicht nur 10 oder 15 gleichzeitig laufende Abfragen.
- **Skalierbarkeit (Scalability):** Skalierbarkeit ist wichtig für Systeme, die ihre Performance unter unterschiedlich starken Workloads beibehalten müssen. Ein ideales System würde doppelt so viele Abfragen beantworten (Throughput), wenn doppelt so viele „Arbeiter“ versuchen, die Aufgaben zu erfüllen. Die meisten Systeme sind jedoch nicht linear skalierbar und zeigen Leistungseinbußen, wenn die Parameter variieren.

1.3 Tools

1.3.1 Einführung

Als Haupttool, um Benchmarktests durchzuführen, habe ich mich für Sysbench akopytov, 2024 entschieden. Sysbench ist ein Open-Source-Tool, das ein skriptfähiges, multi-threaded Benchmark-Tool ist, das auf LuaJIT basiert. Es wird auch hauptsächlich für Datenbankbenchmarks verwendet, kann jedoch auch dazu eingesetzt werden, beliebig komplexe Arbeitslasten zu erstellen, die keinen Datenbankserver erfordern. Dabei werden Tests auf verschiedenen Systemressourcen, wie CPU, S peicher, I/O und Datenbanken wie MySQL Reimers, 2017 verwendet.

Im Zuge der Recherchearbeit habe ich mir auch andere Benchmarking-Tools betrachtet, wie z.B. Benchbase Difallah et al., 2013 oder mybench Shopify, 2024. Die größten Vorteile von Sysbench habe ich in der Skriptfähigkeit und Flexibilität gesehen. D.h. dass ich benutzerdefinierte Benchmarks schneller und unkompliziert erstellen kann. Außerdem hat sich Sysbench als de facto Standard im Bereich der Datenbankbenchmarks etabliert Shopify, 2022b. Dadurch stehen eine breite Nutzerbasis und viele verfügbare Ressourcen zur Verfügung. Im Vergleich zu den anderen Tools bietet allerdings Sysbench eine weniger präzise Steuerung der Ergebnisrate und der Transaktionen. Außerdem haben Tools wie mybench die Möglichkeit, in Echtzeit umfassende Visualisierungen darzustellen. Damit können Metriken live in einem Diagramm angezeigt werden Shopify, 2022a. Dieses Feature ist sicherlich hilfreich, aber in meinem Fall habe ich abgewogen und bin zu dem Entschluss gekommen, dass die einfachere Bedienung für mich der ausschlaggebende Grund, neben dem Fakt, dass Sysbench der de facto Standard ist.

Trotzdem kann man nicht komplett auf Graphen verzichten, da beispielweise Entwicklungen im Laufe einer Zeitmessung in einem Kurvenverlauf deutlich besser zu erkennen sind als in einer CSV-Datei. Anhand der reinen Zahlen aus diesen Tabellen fallen diese wiederkehrende Trends unter anderem nicht direkt auf. Die Kennzahlen, die mithilfe von Sysbench ermittelt werden, werden in einer CSV-Datei gespeichert. Um diese tabellarische Form in eine Grafische umzuwandeln, gibt es unterschiedliche Tools, die widerum eigene Vor- und Nachteile bieten.

Die erste mögliche Alternative stellt das Tool Gnuplot Williams et al., 2024 dar. Mit diesem lassen sich CSV-Dateien sehr gut darstellen. Wenn man aber beispielweise nur bestimmte Spalten aus der Tabelle anzeigen lassen will, dann kommt man schnell an seine Granzen. Um besser Anpassungsfähig sein zu können, habe ich mich letztlich dazu entschieden ein eigenes Python-Script zu schreiben, die mithilfe der Libraries pandas (//TODO: find source) matplotlib.pyplot (//TODO: find source) die Graphen erstellt.

1.3.2 Einführung in die Tools

Als allererstes muss der MySQL-Server (oder eine anderes relationales Datenbankverwaltungssystem, das von Sysbench unterstützt wird) lokal auf dem Rechner gestartet sein. Wichtig ist dabei die User -und Passwortdaten zu merken, da diese von den Sysbench - Benchmarks benötigt werden. Nachdem das RDBMS gestartet worden ist, muss zudem eine Datenbank erstellt werden. Dies könnte unter anderem so aussehen:

```
1 CREATE DATABASE sbtest;
```

Nachdem man die Datenbank erstellt hat, muss das Tool Sysbench zunächst installiert werden. Als nächstes machen wir uns mit dem Tool und den verschiedenen Argumenten, die beim Aufruf mitübergeben werden müssen oder können, vertraut. Hier ist eine Auflistung mit den übergebenen Argumenten:

- `--db-driver`: Gibt den Treiber für die Datenbank an, die Sysbench verwenden soll. In diesem Fall `mysql`, um MySQL-Datenbanken zu testen.
- `--mysql-host`: Der Hostname oder die IP-Adresse des MySQL-Servers. Standardmäßig wird `localhost` verwendet, wenn nichts angegeben wird.
- `--mysql-user`: Der Benutzername, mit dem Sysbench auf die MySQL-Datenbank zugreift.
- `--mysql-password`: Das Passwort für den MySQL-Benutzer. Falls der Benutzer kein Passwort hat oder der Zugriff über eine andere Authentifizierungsmethode erfolgt, kann dieses Argument weggelassen werden.

- `--mysql-db`: Der Name der MySQL-Datenbank, auf die zugegriffen wird. In diesem Beispiel `sbtest`.
- `--time`: Gibt die Laufzeit des Benchmarks in Sekunden an und muss immer mit angegeben werden.
- `--report-interval`: Gibt das Intervall in Sekunden an, in dem Zwischenergebnisse während des Tests ausgegeben werden. Sofern `--report-interval` nicht gesetzt wird, werden die Ergebnisse erst am Ende des Tests angezeigt.
- `--tables`: Die Anzahl der Tabellen, die für den Test erstellt werden sollen. Standardmäßig wird nur eine Tabelle erstellt.
- `--table-size`: Die Anzahl der Datensätze (Zeilen) pro Tabelle. Muss auch nicht zwingend angegebend werden.

Neben den sieben aufgelisteten Argumenten gibt es zwei weitere wichtige Optionen:

1. Wie im Abschnitt **??** erwähnt, kann ein Lua-Skript angegeben werden, um eigene Tabellen zu erstellen, Beispieldaten einzufügen und bestimmte Abfragen durchzuführen. Dazu muss am Ende der Sysbench-Befehlszeile lediglich der Pfad zur Lua-Datei hinzugefügt werden. Ein erklärendes Beispiel dazu folgt weiter unten in diesem Abschnitt.
2. Die Methode, den Sysbench ausführen soll, muss ebenfalls spezifiziert werden. Auch dieser wird am Ende der Sysbench-Befehlszeile angehängt.

Zunächst schauen wir ein kurzes Demo-Beispiel, denn es gibt die Möglichkeit die Datenbank auf Performance zu testen, ohne selbst eigene SQL-Befehle zu schreiben. Dafür gibt es vordefinierte Testtypen von Sysbench. Auf diese Weise kann man schnell die Korrektheit der Einrichtung des Tools überprüfen, bevor man Lua-Scripts für die eigenen Bedürfnisse schreibt.

Man kann unter anderen zwischen diesen Testtypen wählen:

- **oltp_insert**: Prüft die Fähigkeit der Datenbank, Daten schnell und effizient einzufügen und simuliert eine Umgebung, in der viele Schreiboperationen ausgeführt werden.
- **oltp_read_only**: Fokussiert sich auf die Performance bei Leseoperationen und eignet sich, um die Leistung bei einer rein lesenden Arbeitslast zu testen.
- **oltp_read_write**: Simuliert eine realistische Arbeitslast, bei der sowohl Lese- als auch Schreiboperationen gleichzeitig durchgeführt werden.

Des Weiteren gibt es auch unterschiedliche Methoden, die mit den Testtypen kombiniert werden können.

-

- **prepare**: Bereitet die Datenbank für den Test vor, u.a. das Einfügen von benötigten Datensätze.
- **run**: Ist die Ausführungsphase des Tests. Je nach Testtyp führt diese Methode die spezifizierten Operationen aus, wie etwa das Einfügen von Daten (oltp_insert), das Abfragen von Daten (oltp_read_only) oder beides (oltp_read_only). Dabei wird die Performance der Datenbank unter der angegebenen Arbeitslast gemessen.
- **cleanup**: Diese Methode sorgt dafür, dass nach Abschluss des Tests alle Testdaten entfernt werden. Sie stellt die Datenbank in ihren ursprünglichen Zustand zurück und stellt sicher, dass keine Testdaten zurückbleiben, die eine mögliche produktive Umgebung beeinträchtigen könnten.

Für das Demo-Beispiel wählen wir den Testtypen **oltp_read_write** und allen Methoden aus. Für die Methode run würde unsere Query so aussehen, wobei `YOUR_USER` und `YOUR_PASSWORD` entsprechend ersetzt werden müssten:

```
sysbench oltp_read_write \
  --db-driver=mysql \
  --mysql-user=YOUR_USER \
  --mysql-password=YOUR_PASSWORD \
  --mysql-db="sbtest" \
  --time=10 \
  --report-interval=1 \
  run
```

Wenn man nur diese Query ausführt, fällt er auf, dass die Query scheitert. Deshalb bietet es sich an ein Shell-Script zu schreiben, indem zuerst `prepare` aufgerufen wird und als Nächstes erst `run`. Die Ergebnisse der Log-Datei speichert man sich dann in einer Datei und aus dieser Datei erstellt man eine CSV-Datei, mit der man später die Graphen erstellen lässt. Und als letzten Schritt ruft man die `cleanup`-Methode auf, damit bei erneuter Ausführung keine Fehler entstehen, bzw. die Produktivumgebung nicht gestört wird, wenn diese sonst beeinflusst werden würde.

Dies ist das Shell-Script, dass zuständig ist für den kompletten Ablauf:

Codeblock 1.1: Sysbench Script

```
#!/bin/bash

# File Paths
GENERATE_PLOT_SCRIPT="/Users/danielmendes/Desktop/Bachelorarbeit/Ausarbeitung/Tools/
    Pandas/generateplot.py"
```

```
OUTPUT_DIR="output"
OUTPUT_FILE="output/sysbench_output.csv"
RAW_RESULTS_FILE="output/sysbench.log"
GNUPLOT_SCRIPT="plot_sysbench.gp"

# Connection parameters
DB_USER="root"
DB_PASS="password"
DB_NAME="sbtest"
TABLES=10
TABLE_SIZE=10000
DURATION=10

# Ensure output directories exist
rm -rf "$OUTPUT_DIR"
mkdir -p "$OUTPUT_DIR"

# Function to run sysbench with parameters
run_sysbench() {
  local MODE="$1"
  local EXTRA_ARGS="$2"
  local LOG_FILE="$3"

  sysbench oltp_read_write \
    --db-driver=mysql \
    --mysql-user="$DB_USER" \
    --mysql-password="$DB_PASS" \
    --mysql-db="$DB_NAME" \
    --tables="$TABLES" \
    --table-size="$TABLE_SIZE" \
    $EXTRA_ARGS \
    $MODE >> "$LOG_FILE" 2>&1

  return $?
}

echo "Preparing the database..."
run_sysbench "prepare" "" "$RAW_RESULTS_FILE"
echo "Database prepared."

echo "Running benchmark..."
run_sysbench "run" "--time=$DURATION --threads=1 --report-interval=1" "
```

```
    $RAW_RESULTS_FILE"
echo "Benchmark complete. Results saved to $OUTPUT_FILE."

# Format the results into CSV
echo "Time (s),Threads,TPS,QPS,Reads,Writes,Other,Latency (ms;95%),ErrPs,ReconnPs" > "$OUTPUT_FILE"
grep '^\[ ' $RAW_RESULTS_FILE | while read -r line; do
    time=$(echo $line | awk '{print $2}' | sed 's/s//')
    threads=$(echo $line | awk -F 'thds: ' '{print $2}' | awk '{print $1}')
    tps=$(echo $line | awk -F 'tps: ' '{print $2}' | awk '{print $1}')
    qps=$(echo $line | awk -F 'qps: ' '{print $2}' | awk '{print $1}')
    read_write_other=$(echo $line | sed -E 's/.*\(r\/w\/o: ([0-9.]+)\/([0-9.]+)\/([0-9.]+)\).*/\1,\2,\3/')
    reads=$(echo $read_write_other | cut -d',' -f1)
    writes=$(echo $read_write_other | cut -d',' -f2)
    other=$(echo $read_write_other | cut -d',' -f3)
    latency=$(echo $line | awk -F 'lat \\(ms,95%\\): ' '{print $2}' | awk '{print $1}')
    err_per_sec=$(echo $line | awk -F 'err/s: ' '{print $2}' | awk '{print $1}')
    reconn_per_sec=$(echo $line | awk -F 'reconn/s: ' '{print $2}' | awk '{print $1}')

    echo "$time,$threads,$tps,$qps,$reads,$writes,$other,$latency,$err_per_sec,$reconn_per_sec" >> "$OUTPUT_FILE"
done

echo "Cleaning up..."
run_sysbench "cleanup" "" "$RAW_RESULTS_FILE"
echo "Database cleanup complete."

# Generate plot with gnuplot
rm -rf "$OUTPUT_DIR/gnuplot"
mkdir -p "$OUTPUT_DIR/gnuplot"
echo "Generating plot with gnuplot..."
gnuplot $GNUPLOT_SCRIPT

# Generate plot with pandas and move objects
echo "Generating plots with pandas..."
python3 "$GENERATE_PLOT_SCRIPT" "$OUTPUT_FILE"
SOURCE_DIR="output/detailed_pngs"
DEST_DIR="output/pandas"
FILE_TO_MOVE="output/output_final.png"
```

```
mkdir -p "$DEST_DIR"
mv "$SOURCE_DIR"/* "$DEST_DIR"
mv "$FILE_TO_MOVE" "$DEST_DIR/Summary.png"
rm -rf "$SOURCE_DIR"

echo "Plots generated."
```

Codeblock 1.2: Gnuplot Script

```
set datafile separator ","
set title "Benchmark Results: TPS, Latency, Queries, and More"
set xlabel "Time (s)"
set ylabel "Values"
set grid
set key outside
set terminal pngcairo enhanced font 'Arial,10'
set output "/Users/danielmendes/Desktop/Bachelorarbeit/Ausarbeitung/Tools/Output/
    sysbench_output_plot.png"
set yrange [0:*]

# Plot each attribute on its own line
plot "/Users/danielmendes/Desktop/Bachelorarbeit/Ausarbeitung/Tools/Output/
    sysbench_output.csv" using 1:2 title "Threads" lt 1 lc rgb "black" with lines, \
     "" using 1:3 title "TPS" lt 2 lc rgb "green" with lines, \
     "" using 1:4 title "QPS" lt 3 lc rgb "blue" with lines, \
     "" using 1:5 title "Reads" lt 4 lc rgb "red" with lines, \
     "" using 1:6 title "Writes" lt 5 lc rgb "orange" with lines, \
     "" using 1:7 title "Other" lt 6 lc rgb "purple" with lines, \
     "" using 1:8 title "Latency (ms)" lt 7 lc rgb "cyan" with lines, \
     "" using 1:9 title "Err/s" lt 8 lc rgb "magenta" with lines, \
     "" using 1:10 title "Reconn/s" lt 9 lc rgb "brown" with lines
```

Das Python-Script, das zuständig ist für die Graphgenerierung muss als Argument zum einen die CSV-Datei übermittelt bekommen und zum anderen kann es nur eine bestimmte Auswahl an Messwerten übergeben, damit nur für diese die Graphen erzeugt werden. Dies ist das zuständige Python-Script:

Codeblock 1.3: Pandas Graph Generator

```
import pandas as pd
import matplotlib.pyplot as plt
import os
```

```
import argparse

def parse_arguments():
    parser = argparse.ArgumentParser(description='Generate plots from CSV data.')
    parser.add_argument('datafile', type=str, help='Path to the input CSV data file')
    parser.add_argument('metrics', type=str, nargs='*', help='List of metrics to plot (e.g., QPS Reads Writes). If empty, all metrics will be used.')
    return parser.parse_args()

def plot_metrics(data, measures, output_dir, detailed_pngs_dir):
    has_script_column = 'Script' in data.columns

    if has_script_column:
        scripts = data['Script'].unique()
    else:
        scripts = [None]

    plt.figure(figsize=(10, 6))

    for measure in measures:
        if has_script_column:
            # Plot each script as a separate line for each measure if 'Script' column exists
            for script in scripts:
                script_data = data[data['Script'] == script]
                plt.plot(script_data['Time (s)'], script_data[measure], label=f"{script} - {measure}")
        else:
            # Plot only the measure if no 'Script' column exists
            plt.plot(data['Time (s)'], data[measure], label=measure)

    plt.title('Metrics over Time' + (' by Script' if has_script_column else ''))
    plt.xlabel('Time (s)')
    plt.ylabel('Values')
    plt.legend(title="Script and Measure" if has_script_column else "Measure")
    plt.grid(True)

    # Save the combined plot
    output_final_path = os.path.join(output_dir, 'output_final.png')
    plt.savefig(output_final_path)
    plt.close()
```

```
42
43      for measure in measures:
44          plt.figure(figsize=(10, 6))
45          if has_script_column:
46              # Plot each script as a separate line for each measure if 'Script'
    column exists
47              for script in scripts:
48                  script_data = data[data['Script'] == script]
49                  plt.plot(script_data['Time (s)'], script_data[measure], label=f"{
    script} - {measure}")
50          else:
51              # Plot only the measure if no 'Script' column exists
52              plt.plot(data['Time (s)'], data[measure], label=measure)
53
54          # Plot settings for individual figures
55          plt.title(f'{measure} over Time by Script')
56          plt.xlabel('Time (s)')
57          plt.ylabel(measure)
58          plt.legend(title="Script")
59          plt.grid(True)
60
61          # Save the detailed plot to a PNG file in the specified output directory
62          detailed_output_file_path = os.path.join(detailed_pngs_dir, f"{measure}.png"
    )
63          plt.savefig(detailed_output_file_path)
64          plt.close()
65
66  def main():
67      args = parse_arguments()
68
69      # Load CSV data
70      datafile = args.datafile
71      if not os.path.isfile(datafile):
72          raise FileNotFoundError(f"The file {datafile} does not exist.")
73
74      data = pd.read_csv(datafile)
75
76      # Determine metrics to plot
77      if args.metrics:
78          measures = args.metrics
79      else:
80          # Use all columns except 'Time (s)' and 'Script' as metrics
```

```
81          measures = [col for col in data.columns if col not in ['Time (s)', 'Script'
    ]]
82
83      # Define output directory for plots
84      output_dir = os.path.dirname(datafile)
85      detailed_pngs_dir = os.path.join(output_dir, 'detailed_pngs')
86
87      # Create output directories if they don't exist
88      os.makedirs(detailed_pngs_dir, exist_ok=True)
89      plot_metrics(data, measures, output_dir, detailed_pngs_dir)
90
91  if __name__ == '__main__':
92      main()
```

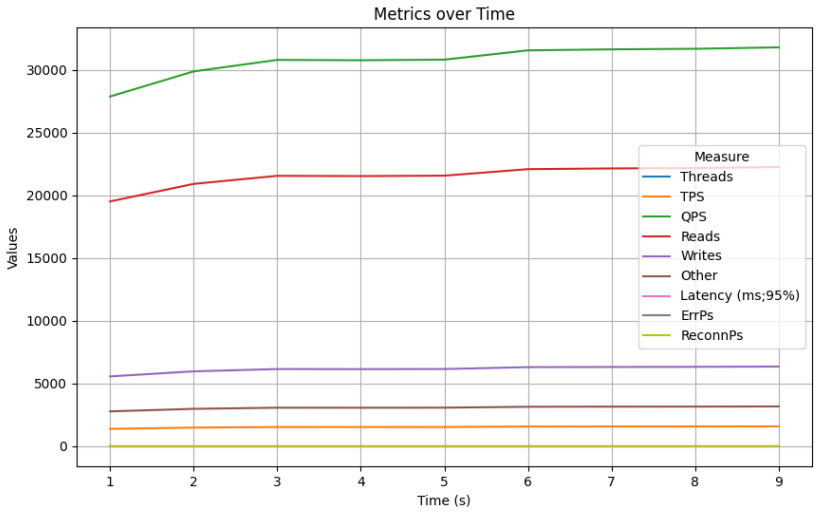

Abbildung 1.1: Grafik generiert mithilfe des Pyhtontools Pandas

- **Threads**: Die Anzahl der gleichzeitig verwendeten Threads. Mehr Threads können die Parallelität erhöhen, jedoch kann eine zu hohe Anzahl die Leistung beeinträchtigen, wenn das System überlastet wird.
- **TPS (Transactions Per Second)**: Die Anzahl der Transaktionen pro Sekunde. Ein höherer Wert deutet auf eine bessere Datenbankleistung hin.
- **QPS (Queries Per Second)**: Die Anzahl der Abfragen pro Sekunde. Ein höherer Wert ist besser und zeigt die Effizienz bei der Verarbeitung von Abfragen.

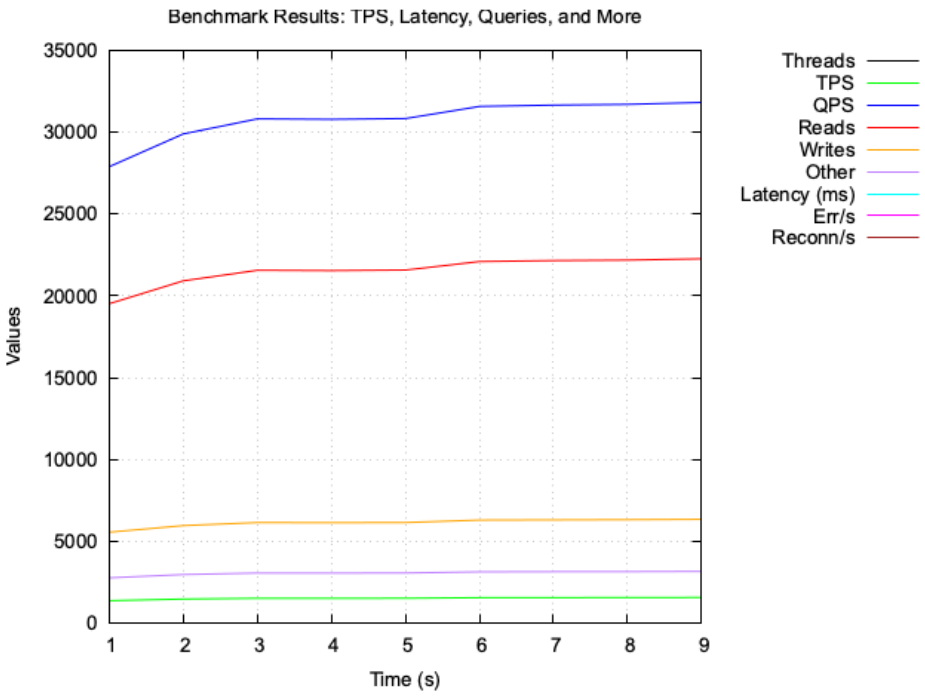

Abbildung 1.2: Grafik generiert mithilfe von Gnuplot

- **Reads**: Die Anzahl der Leseoperationen. Mehr Leseoperationen sind im Allgemeinen besser, da sie eine höhere Datenauslastung anzeigen, was jedoch auch vom spezifischen Anwendungsfall abhängt.
- **Writes**: Die Anzahl der Schreiboperationen. Ähnlich wie bei den Leseoperationen: Mehr Schreibvorgänge sind besser, solange die Performance erhalten bleibt.
- **Other**: Bezieht sich auf andere Arten von Operationen, die weder als Reads noch als Writes kategorisiert werden. Ein höherer Wert ist gut, solange er nicht zu einer Überlastung führt.
- **Latency (ms; 95%)**: Die durchschnittliche Zeit in Millisekunden, die benötigt wird, um Anfragen zu bearbeiten, wobei der Wert im 95. Perzentil betrachtet wird. Niedrigere Werte sind besser, da sie auf schnellere Reaktionszeiten hinweisen.
- **ErrPs (Errors Per Second)**: Die Anzahl der Fehler pro Sekunde. Ein niedriger Wert ist wünschenswert, da er auf eine höhere Stabilität und Zuverlässigkeit des Systems hinweist.
- **ReconnPs (Reconnects Per Second)**: Die Anzahl der Wiederverbindungen pro Sekunde. Ein niedrigerer Wert ist ebenfalls besser, da häufige Wiederverbindungen auf Stabilitätsprobleme hindeuten können.

2 Indexierung und Einfluss auf die Performance

2.1 Grundlagen der Indexierung

Das folgende Thema befasst sich mit der Indexierung und den damit verbundenen Performance-Optimierungen, die näher erläutert werden. Zunächst betrachten wir die Grundlagen der Indexierung, anschließend die verschiedenen Arten von Indizes und schließlich deren Auswirkungen auf die Performance.

Indizes (oder auch Indexes) sind Datenstrukturen, die von Speicher-Engines (engl. storage engines) verwendet werden, um unter anderem Zeilen schneller zu finden. Sie haben einen großen Einfluss auf die Performance der Datenbank und werden umso wichtiger, je größer die Datenbank wird. Weniger ausgelastete Datenbanken können ohne ordnungsgemäße Indizes gut funktionieren, aber die Leistung kann rapide sinken, wenn die Datenmenge wächst. Wenn ein solches Problem auftritt, ist die Index-Optimierung oft der effektivste Weg, die Abfrageleistung zu verbessern. Um wirklich optimale Indizes zu erstellen, ist es häufig notwendig, Abfragen umzuschreiben. Wie genau Indizes erstellt werden müssen, wird im weiteren Verlauf der Arbeit betrachtet.

Um die Funktionsweise eines Indexes zu verdeutlichen, betrachten wir ein Beispiel aus einem wissenschaftlichen Fachbuch. Am Ende solcher Bücher gibt es meist ein Stichwortverzeichnis oder Register. Dieses Register besteht aus einer alphabetisch geordneten Liste von Begriffen, Themen und Stichworten. Möchte man einen Begriff nachschlagen, sucht man ihn in der Liste und erhält die Seitenzahlen, auf denen er vorkommt. In MySQL verwendet die Storage-Engine Indizes auf ähnliche Weise. Sie durchsucht die Datenstruktur des Indexes nach einem Wert. Wird ein Treffer gefunden, kann die Engine die Zeile ermitteln, die den Treffer enthält. Betrachten wir dazu folgendes Beispiel:

Codeblock 2.1: Variationen

```
1 SELECT name FROM customer WHERE cust_id = 7;
```

Es gibt einen Index auf der Spalte `cust_id`, sodass MySQL diesen Index nutzt, um Zeilen zu finden, deren `cust_id` gleich 7 ist. Mit anderen Worten wird eine Suche innerhalb der Indexwerte durchgeführt, und alle entsprechenden Zeilen werden zurückgegeben.

Ein Index kann Werte aus einer oder mehreren Spalten einer Tabelle enthalten. Bei mehreren Spalten ist die Reihenfolge der Spalten im Index entscheidend, da MySQL nur effizient auf ein linkes Präfix des Indexes zugreifen kann. Ein Index über zwei Spalten ist nicht gleichbedeutend mit zwei separaten einspaltigen Indizes. Es gibt verschiedene Typen von Indizes, die jeweils für unterschiedliche Zwecke optimiert sind und die im nächsten Abschnitt behandelt werden.

2.2 B-Baum-Index

Indizes werden auf der Ebene der Storage-Engine und nicht auf der Serverebene implementiert. Daher sind sie nicht standardisiert und unterscheiden sich je nach Engine. Zudem unterstützen nicht alle Engines alle Index-Typen. Eine Storage-Engine ist eine Kernkomponente eines Datenbankmanagementsystems (DBMS), die für die Speicherung und Verwaltung der Daten zuständig ist. Sie entscheidet, wie Daten physisch organisiert, gespeichert und abgerufen werden. Verschiedene Storage-Engines unterscheiden sich in ihrer Indexfunktionalität sowie in der Unterstützung von Transaktionen und Sperrmechanismen.

Der erste zu betrachtende Indextyp ist der B-Baum-Index (engl. B-Tree Index), der auf einer speziellen Baum-Datenstruktur basiert. Diese Struktur wird von den meisten MySQL-Storage-Engines unterstützt. Die Implementierung und Nutzung des B-Baum-Indexes kann jedoch je nach verwendeter Storage-Engine variieren.

Das Grundprinzip eines B-Baums ist, dass alle Werte in einer bestimmten Reihenfolge gespeichert werden und jede Blattseite den gleichen Abstand zum Wurzelknoten hat. Ein B-Baum-Index beschleunigt den Datenzugriff, da die Storage-Engine nicht die gesamte Tabelle durchsuchen muss, um die gewünschten Daten zu finden. Stattdessen beginnt die Suche beim Wurzelknoten.

Die Slots im Wurzelknoten enthalten Zeiger auf Kindknoten, und die Storage-Engine folgt diesen Zeigern. Der richtige Zeiger wird durch Vergleich der Werte in den Knoten-Seiten (engl. node pages) ermittelt, die die oberen und unteren Grenzen der Werte in den Kindknoten definieren. Letztlich stellt die Storage-Engine fest, ob der gewünschte Wert existiert, oder sie erreicht erfolgreich eine Blattseite (engl. leaf page).

Blattseiten sind besonders, da sie Zeiger auf die indexierten Daten enthalten, anstatt auf andere Seiten zu verweisen. Zwischen dem Wurzelknoten und den Blattseiten können viele Ebenen von Knoten-Seiten existieren. Die Tiefe des Baumes hängt von der Größe der Tabelle ab.

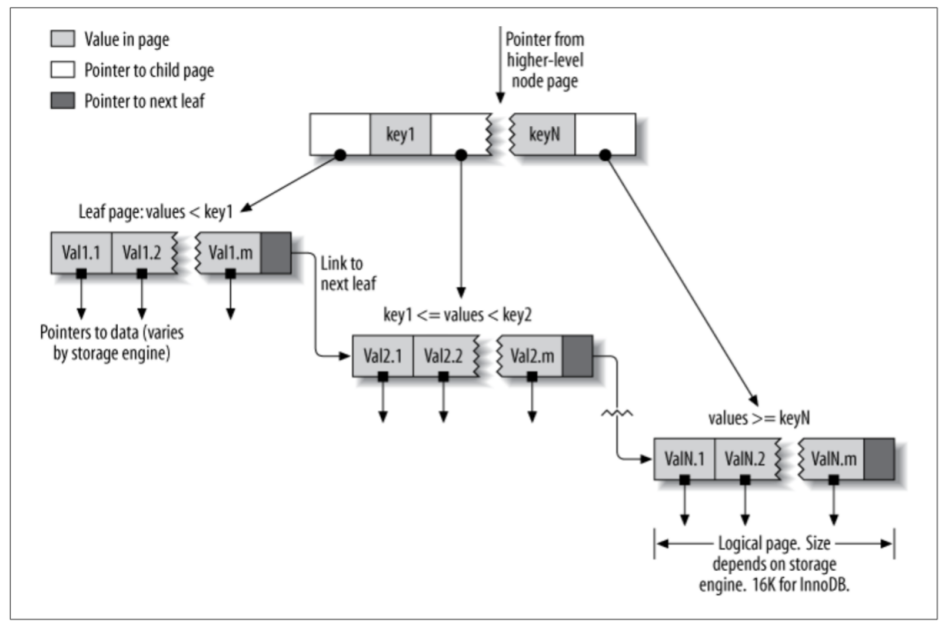

Abbildung 2.1: Darstellung des binären Baums mit Knoten und Blättern

Außerdem speichern B-Bäume die indexierten Spalten in einer festgelegten Reihenfolge, was sie besonders nützlich für die Suche nach Datenbereichen macht. Beispielsweise kann ein Index auf einem Textfeld (z.B. vom Typ `VARCHAR`) effizient alle Namen finden, die mit „K“ beginnen, da die Werte in alphabetischer Reihenfolge gespeichert sind.

Der Index sortiert die Werte entsprechend der Reihenfolge der in der `CREATE TABLE`-Anweisung angegebenen Spalten, beispielsweise des Primärschlüssels (`last_name, first_name, b_day`). B-Baum-Indizes eignen sich gut für Suchen mit dem vollständigen Schlüsselwert (engl. full key value), einem Schlüsselbereich (engl. key range) oder einem Schlüsselpräfix (engl. full key prefix). Beim Schlüsselpräfix ist dies jedoch nur der Fall, wenn die Suche das linkeste Präfix des Indexes verwendet.

Als nächstes betrachten wir die möglichen Abfragen, bei denen B-Baum-Indizes besonders hilfreich sind, um ein besseres Verständnis für ihre optimale Nutzung zu erlangen. Eine Übereinstimmung mit dem vollständigen Schlüsselwert liefert Werte für alle Spalten im Index. Eine beispielhafte Abfrage wäre die Suche nach allen Einträgen für Max Mustermann, geboren am 2000-01-01, wenn der Schlüssel aus Nachname, Vorname und Geburtsdatum besteht. Für diesen Index sind auch Abfragen nützlich, die nur mit dem linken Präfix übereinstimmen, beispielsweise die Suche nach „Mustermann“. Eine weitere Möglichkeit ist die Übereinstimmung mit einem Spaltenpräfix, also dem ersten Teil eines Spaltenwerts, etwa alle Nachnamen, die mit „M“ beginnen. Ebenso effizient ist der Index bei der Übereinstimmung mit einem Wertebereich, z.B. Nachnamen zwischen „Mustermann“ und „Müller“.

Ein B-Baum-Index kann auch genutzt werden, um Abfragen effizient zu unterstützen, bei denen eine Spalte exakt und eine andere innerhalb eines Wertebereichs abgefragt wird. Beispielsweise könnte dies eine exakte Übereinstimmung mit dem Nachnamen „Mustermann“ und eine Bereichsabfrage für Vornamen, die mit „Ma“ beginnen, umfassen. Der letzte Anwendungsfall sind Abfragen, die nur den Index verwenden und nicht die gespeicherten Zeilen, etwa wenn alle benötigten Daten im Index enthalten sind.

Ein weiterer Vorteil von B-Baum-Indizes ist, dass sie aufgrund der sortierten Baumstruktur nicht nur Abfragen, sondern auch `ORDER BY`-Bedingungen effizient unterstützen können. Wenn ein B-Baum für die Suche genutzt werden kann, kann er auch für die Sortierung der Ergebnisse verwendet werden.

Es gibt jedoch Einschränkungen von B-Baum-Indizes, die dazu führen, dass andere Indextypen für bestimmte Szenarien besser geeignet sind. Eine Einschränkung ist, dass die Suche nicht am linken Ende des Indexes beginnen kann. Beispielsweise ist ein Index, der aus Nachname, Vorname und Geburtsdatum besteht, nicht geeignet, um alle Personen zu finden, die vor dem Jahr 2000 geboren wurden, ohne dass der Nachname und Vorname ebenfalls spezifiziert werden.

Für optimale Leistung sollten Indizes mit den gleichen Spalten, jedoch in unterschiedlicher Reihenfolge erstellt werden, um die häufigsten Abfragen zu optimieren. Eine Analyse der am häufigsten verwendeten Abfragen kann dabei helfen zu entscheiden, ob zusätzliche Indizes erforderlich sind.

3 Allgemeines

Diese Vorlage zur Verwendung mit LaTeX[1] kann für die Formatierung Ihrer Bachelorarbeit verwendet werden, ist jedoch keine verbindliche Vorgabe; bitte stimmen Sie sich hier mit Ihrer betreuenden Erstprüferin bzw. Ihrem betreuenden Erstprüfer ab.

Den aktuellen Stand dieser Vorlage entnehmen Sie bitte dem Datum auf der Titelseite.

Das Textsatzsystem LaTeX ist für die Erstellung druckfertiger wissenschaftlicher Arbeiten unabhängig von deren Umfang hervorragend geeignet, weil es explizit dafür konzipiert ist. Im Allgemeinen sollten Sie damit bessere Ergebnisse als mit Textverarbeitungsprogrammen wie Word oder Pages erzielen. Allerdings handelt es sich nicht um ein mit der Maus zu bedienendes WYSIWYG-Programm und erfordert eine gewisse Einarbeitungszeit. Die Verwendung von LaTeX für eine Abschlussarbeit ist daher nicht unbedingt zu empfehlen, wenn Sie das System erst kurz vor dem Schreiben der Arbeit zum ersten Mal benutzen.

In diesem Sinne ist das vorliegende Dokument auch explizit *nicht* als Einführung in LaTeX gedacht, sondern setzt voraus, dass Sie schon ausreichende Kenntnisse mitbringen. Die können Sie beispielsweise durch die Lektüre von Büchern wie (Voß, 2021) oder (Schlosser, 2021) erwerben.

[1] In diesem Dokument wird den üblichen Gepflogenheiten entsprechend nicht zwischen dem zugrundeliegenden Textsatzsystem TeX und dem weitverbreiteten Makropaket LaTeX für dieses System unterschieden, obwohl das rein technisch falsch ist.

4 Die einzelnen Teile der Arbeit

In diesem Kapitel geht es um die wesentlichen Teile, die eine Abschlussarbeit haben sollte. Die Dateien, auf die hier Bezug genommen wird, werden in in Tabelle 4.1 vorgestellt und in Kapitel 5 näher erläutert. Die Vorlage wurde absichtlich in relativ viele Dateien zerlegt, um zu demonstrieren, wie man mit dem Befehl `\includeonly` nur Teile der Arbeit kompilierern kann, um Zeit zu sparen. Vor dem Druck muss natürlich das komplette, aus allen Dateien bestehende Dokument kompiliert werden (ggf. mehrfach), damit alle Querverweise (siehe Abschnitt 5.5.1) korrekt sind.

Tabelle 4.1: Dateien der Vorlage

Dateiname	Zweck
`VorlageBA.tex`	Hauptdatei
`defs.tex`	Laden von Paketen und Setzen von Optionen
`title.tex`	Titelseite und Abstract
`toc.tex`	Inhaltsverzeichnis und optionale Verzeichnisse
`chap1.tex` bis `chap3.tex`	erstes, zweites und drittes Kapitel
`appendix.tex`	Literaturverzeichnis, Anhang und Eigenständigkeitserklärung
`demo.bib`	exemplarische Bibliographie
`HAW_Marke_RGB_300dpi.jpg`	HAW-Logo für Titelseite
`bitmap.png`, `vector.pdf` und `euler.py`	Beispieldateien

4.1 Titelei

Mit dem Begriff *Titelei* bezeichnet man im Buchwesen den Teil eines Buches, der dem eigentlichen Inhalt vorangestellt ist. In dieser Vorlage dient der Absatz, den Sie gerade lesen, im Wesentlichen aber nur als Vorwand, eine weitere Unterebene einzufügen.

4.1.1 Titelseite

Die Titelseite befindet sich in der Datei `title.tex` und ist der Vorlage der HAW nachempfunden. Sie müssen natürlich in dieser Datei den Titel, die Namen und das Datum anpassen. Außerdem müssen die Hausschriften der HAW installiert sein.

4.1.2 Abstract

Nach der Titelseite folgt der Abstract, der sich ebenfalls in der Datei `title.tex` befindet. Hier ersetzen Sie den Beispieltext durch eine möglichst aussagekräftige Zusammenfassung Ihrer Arbeit.

4.2 Inhaltsverzeichnis und andere Verzeichnisse

Nach dem Abstract (siehe Abschnitt 4.1.2) folgt das Inhaltsverzeichnis, das von LaTeX automatisch erzeugt wird. Exemplarisch wird in dieser Vorlage auch gezeigt, wie man Abbildungs- und Tabellenverzeichnisse generieren könnte; siehe dazu die Datei `toc.tex`. Das ist in der Regel aber nur dann nötig, wenn es in Ihrer Arbeit sehr viele Abbildungen bzw. Tabellen gibt. Bei Bedarf können auch Verzeichnisse von Codeblöcken, mathematischen Formeln oder anderen Objekten erzeugt werden; mehr dazu in (Voß, 2021, Kap. 12). Hier gilt aber ebenfalls, dass das normalerweise nicht nötig sein sollte. Auch ein Glossar oder ein Abkürzungsverzeichnis ist in einer Bachelorarbeit eher unüblich. Sprechen Sie ggf. mit Ihrer Erstprüferin oder Ihrem Erstprüfer ab, was wirklich gebraucht wird.

4.3 Gliederungsebenen

Eine Bachelorarbeit sollte in der Regel mit maximal drei Gliederungsebenen auskommen. In der Dokumentenklasse der Vorlage entspricht das den Befehlen `\chapter`, `\section` und `\subsection`, die auch für eine automatische Aufnahme der jeweiligen Abschnitte ins Inhaltsverzeichnis sorgen. Weitere Gliederungsebenen verringern typischerweise die Lesbarkeit des Dokuments. Falls Sie das bei Ihrer Arbeit trotzdem für nötig halten, sprechen Sie es vorher mit der Erstprüferin bzw. dem Erstprüfer ab.

Auf jeder Gliederungsebene sollte es mindestens zwei Abschnitte derselben Hierarchie geben, da ansonsten eine Gliederung auf dieser Ebene sinnlos wäre. Wenn Sie also einen Abschnitt 2.3.1 haben, dann muss es mindestens einen weiteren Abschnitt 2.3.2 geben; anderenfalls sollte alles unter 2.3 stehen.

Außerdem sollten einzelne Abschnitte einen Umfang haben, der die entsprechende Gliederung rechtfertigt. Dieser Text, in dem Abschnitte meistens nur aus wenigen Sätzen bestehen, ist dafür ein schlechtes Beispiel![1] Es handelt sich allerdings auch um eine Vorlage und nicht um eine Bachelorarbeit …

In der Vorlage ist jedes Kapitel in eine eigene Datei ausgelagert. Beachten Sie, dass der Befehl `\include` grundsätzlich eine neue Seite anfängt. Unterabschnitte von Kapiteln müssen daher mit `\input` eingefügt werden, wenn sie in separaten Dateien liegen sollen.

4.4 Literaturverzeichnis

Die Vorlage enthält ein exemplarisches Literaturverzeichnis, das nach dem sogenannten APA-Standard formatiert ist und das als Beispiele einige Bücher, einen Fachartikel und eine Internetquelle umfasst. Das Literaturverzeichnis befindet sich am Ende der Arbeit vor dem Anhang. Falls Ihre Erstprüferin oder Ihr Erstprüfer ein anderes Format wünscht, ist das mit dem verwendeten Paket BibLaTeX ohne große Probleme zu realisieren, siehe z. B. (Voß, 2021, Kap. 13). Wichtig ist, dass das Literaturverzeichnis vollständig ist und Sie nur die Quellen angeben, die Sie im Rahmen Ihrer Bachelorarbeit wörtlich zitiert oder sinngemäß wiedergegeben haben. Literatur, die Sie lediglich zur Vorbereitung genutzt haben, gehört nicht in das Literaturverzeichnis.[2]

Wenn Sie die von Ihnen verwendeten Texte nach dem Muster der Datei `demo.bib` eingeben, wird das Literaturverzeichnis automatisch einheitlich dargestellt und alphabetisch sortiert. Die Literaturverwaltung Citavi, für die es eine Hochschullizenz gibt, kann Daten im sogenannten BibTeX-Format ausgeben. (Das ist das in `demo.bib` verwendete Format.)

Verwenden Sie wissenschaftliche Quellen. (Hinweis: Wikipedia gilt nicht als wissenschaftliche Quelle.) Wenn die Angabe von Internetquellen unumgänglich ist, muss das Datum des letzten Aufrufs angegeben werden. Ein Beispiel dafür finden Sie in der Vorlage.

Wie man zitiert, wird in Abschnitt 5.5.2 gezeigt.

4.5 Anhang

Falls zu Ihrer Arbeit Datenreihen, Quelltexte, transkribierte Interviews oder weitere ergänzende Informationen gehören, dann gehören diese in einen Anhang ganz am Ende. Ob ein Anhang

[1] Darum wirken die Abstände und Seitenaufteilungen in dieser Vorlage auch vergleichsweise unruhig. Bei längeren Texten wird es besser aussehen.

[2] Wenn Sie die Voreinstellungen der Vorlage verwenden, werden aber ohnehin nur die Quellen ins Literaturverzeichnis übernommen, die auch zitiert werden.

notwendig ist und welchen Umfang er haben sollte, sollten Sie vorab mit Ihrer Erstprüferin oder Ihrem Erstprüfer absprechen. Häufig ist es sinnvoller, die Daten auf einem Datenträger der Arbeit beizulegen.

Die Vorlage enthält einen exemplarischen Anhang in der Datei `appendix.tex`.

4.6 Eigenständigkeitserklärung

Die letzte Seite der Arbeit ist die Eigenständigkeitserklärung, die Sie in der Datei `appendix.tex` finden. Tragen Sie hier den Titel Ihrer Arbeit sowie den Ort und das Datum ein. Alle gedruckten Exemplare werden dann oberhalb des Datums von Ihnen unterschrieben.

5 TEXnik und Typographie

Die grundsätzliche Philosophie von LATEX ist, dass sich die Autoren auf den Inhalt konzentrieren und um das Erscheinungsbild des Textes keine großen Gedanken machen sollen. Die typographischen „Entscheidungen“, die das System bzw. die Dokumentenklasse trifft, sind in der Regel sinnvoll und führen zu besseren Ergebnissen als manuelle Eingriffe ungeübter Benutzer.

Wenn Sie sich dabei erwischen, dass Sie Abstände, Positionen, Größen oder andere Parameter manipulieren, dann machen Sie sehr wahrscheinlich etwas falsch. (Eine Ausnahme sind Dinge wie die Titelseite, die einer Vorgabe nachempfunden werden sollen.)

In diesem Kapitel werden einige *Best Practices* aufgeführt, die besonders die Leser aufmerksam studieren sollten, die noch nicht so erfahren im Umgang mit LATEX sind. Auf diverse typische Fehler im Umgang mit LATEX weist das Video https://youtu.be/OSzs9K6-fRQ hin, aber es ist kein Ersatz für ein einführendes Buch.

5.1 Verwendung der Vorlage

Die Vorlage setzt ein installiertes TEX-System wie MiKTEX oder TEX Live voraus. Sie ist für das Kompilieren mit LuaTEX gedacht (und wird daher mit pdfTEX nicht ohne entsprechende Änderungen funktionieren). Die Bibliographie wird mit Biber bearbeitet.

Es ist nicht möglich, in diesem Rahmen für jede Entwicklungsumgebung die richtigen Optionen anzugeben. Daher nur Anmerkungen zu drei weitverbreiteten Tools:[1]

- Für TEXstudio muss man in der Konfiguration als *Standardcompiler* die Option `LuaLaTeX` auswählen und als *Standard Bibliographieprogramm* die Option `Biber`.
- Für TEXworks benötigt man einen Durchlauf mit der Einstellung `LuaLaTeX` und dann einen mit der Einstellung `Biber`. Danach lässt man das Dokument noch einmal mit der Einstellung `LuaLaTeX` kompilieren und kann in dieser Einstellung bleiben. Einen weiteren `Biber`-Durchlauf zwischendurch benötigt man nur dann, wenn in der Bibliographie etwas geändert wurde.

[1] Die ExpertInnen wissen natürlich, wie man so etwas automatisieren kann.

- Wenn Sie LaTeXmk benutzen, dann können Sie die Vorlage einfach mit

  ```
  latexmk -lualatex -bibtex VorlageBA.tex
  ```

  kompilieren und alles sollte ohne weiteren Eingriff funktionieren.

Nach diesen Schritten sollte das erzeugte PDF exakt so aussehen wie das, das Sie gerade lesen. Die Vorlage besteht aus mehreren Dateien, die in Tabelle 4.1 auf Seite 19 aufgeführt sind. Für LuaTeX müssen die Quelldateien (also die mit der Endung `.tex`) UTF-8-kodiert sein. Die Vorlage ist als Gerüst für Ihre Arbeit gedacht und als Vorschlag zu betrachten. In den meisten Fällen gibt es alternative Methoden, um zu vergleichbaren Ergebnissen zu kommen. Wenn Sie sich gut mit LaTeX auskennen, spricht nichts dagegen, beispielsweise andere Pakete, andere Befehle oder andere Optionen zu verwenden.

Die Dateien sind mit ausführlichen Kommentaren versehen. Viele Informationen zur Verwendung der Vorlage finden Sie nur in diesen Kommentaren und *nicht* in dem Text, den Sie gerade lesen!

Die Datei, die kompiliert werden muss und die die anderen lädt, ist `VorlageBA.tex`. Diese Datei sollten Sie zunächst umbenennen (z. B. in `BANameVorname.tex`), damit das erzeugte PDF-Dokument keinen generischen Namen hat, sondern Ihnen zugeordnet werden kann. Wenn Sie die anderen Dateien umbenennen, dann müssen in den entsprechenden Aufrufen von Befehlen wie `\include` oder `\addbibresource` auch die Namen angepasst werden.

5.2 Textformatierung

In der Vorlage sind, abgesehen von der Titelseite, die freien Schriftarten *Libertinus* für Fließtext und *Anonymous Pro* für Code in einer Größe von 12 Punkt eingestellt. Diese Schriften sind im Druck und auf dem Bildschirm gut lesbar und aufeinander abgestimmt. Außerdem bietet diese Kombination volle Unterstützung für mathematischen Formelsatz. Wenn Sie kein Experte für Typographie sind, sollten sie keine Zeit damit verschwenden, andere Schriftsätze auszuprobieren. Der Zeilendurchschuss ist etwas großzügiger bemessen, als man es für den Druck eines Buches machen würde. Auch diesen Parameter sollten Sie nicht ohne triftigen Grund ändern.

5.3 Mehrsprachiger Text

Die Vorlage ist für Text in deutscher Sprache angelegt, in dem auch englische Zitate vorkommen können. Dafür wird das Standardpaket BABEL eingesetzt. Damit die automatische Silbentrennung sowie einige andere Details richtig funktionieren, muss LaTeX „wissen“, welche Sprache gerade verwendet wird. Beispiele dafür, wie man dem System den Wechsel der Sprache mitteilt, findet man in den Dateien `chap3.tex` und `title.tex`. Weitere Sprachen kann man in der Datei `defs.tex` hinzufügen.

This paragraph only serves to demonstrate that ”quotation marks” will be treated correctly depending on the choice of language.

5.4 PDF-Ausgabe

Wenn Sie die Vorlage wie vorgesehen verwenden, werden durch das Paket HYPERREF automatisch Lesezeichen zur bequemen Navigation in den gängigen PDF-Programmen gesetzt. Außerdem können (und sollten) dem Dokument auch Metadaten zugewiesen werden. Dies geschieht am Ende der Datei `defs.tex`. Dort müssen Sie Ihren Namen und den Titel Ihrer Arbeit eingeben. Ebenfalls am Ende von `defs.tex` wird erklärt, wie Sie HYPERREF so umkonfigurieren, dass für die gedruckte Version Ihrer Arbeit Links nicht farbig dargestellt werden.

5.5 Verweise

5.5.1 Querverweise

Wenn Sie für Querverweise auf Abschnitte, Formeln, Seiten, Abbildungen usw. die Standardbefehle `\label`, `\ref`, `\eqref` und `\pageref` benutzen, sorgt LaTeX automatisch dafür, dass alles richtig nummeriert wird und im PDF zusätzlich Links gesetzt werden. In der Vorlage finden Sie diverse Beispiele für den Einsatz dieser Befehle. Sie sollten Querverweise auf keinen Fall manuell in der Form „`siehe Kapitel 3`“ oder „`auf Seite 42`“ eintippen!

5.5.2 Zitate und Literaturverweise

Bei Verweisen auf die Literatur werden immer Autor und Jahr angegeben, typischerweise in der Form (Weitz, 2021b). Wörtliche Zitate werden zusätzlich durch Anführungszeichen oder durch kursive Schrift gekennzeichnet und mit der Seitenzahl versehen. Es folgen ein paar Beispiele:

(i) Man kann die Leibnizformel mit zahlentheoretischen Methoden herleiten. Das ist allerdings relativ umständlich (Weitz, 2021a).

(ii) Wie Weitz in (2021a) schreibt, kann man die Leibnizformel mit zahlentheoretischen Methoden herleiten. Das sei allerdings relativ umständlich

(iii) Das Fazit des Autors nach Abwägen aller Alternativen: ”The crux of the biscuit is the apostrophe” (Zappa, 1974, S. 42).[2]

(iv) Zappa kommt in (1974) schließlich und endlich zum Ergebnis: „The crux of the biscuit is the apostrophe“ (S. 42).

(v) Man kann die Leibnizformel mit zahlentheoretischen Methoden herleiten. Das ist allerdings relativ umständlich.[3]

Die Versionen (i) und (iii) sind am übersichtlichsten und werden empfohlen. Die Alternative (v) – Quellenangaben in den Fußnoten – ist für Bachelorarbeiten in unserem Department eher unüblich und sollte nur nach Absprache verwendet werden.

5.6 Mathematische Formeln

Mathematischer Formelsatz ist die Spezialität von LaTeX. Während man allerdings Texte in LaTeX fast so wie in einem Textverarbeitungsprogrammen schreiben kann, muss man für Formeln eine eigene Sprache lernen. Einen „Crashkurs“ für diese mathematische Syntax, die man inzwischen auch in Word und vielen anderen Programmen verwenden kann, bietet das Video https://youtu.be/7ovgNXRiJ6g. (Siehe auch das am Anfang von Kapitel 5 erwähnte Video zu typischen Fehlern.) Auch hier gilt aber die Anmerkung, dass ein Lehrbuch die bessere Quelle sein wird.

Die Vorlage ist so eingerichtet, dass eingerückte Formeln nicht zentriert werden, sondern wie beispielsweise

$$\int_a^b g(t)\,\mathrm{d}t = \int_a^b \Re(g(t))\,\mathrm{d}t + \mathrm{i}\int_a^b \Im(g(t))\,\mathrm{d}t \tag{5.1}$$

etwas versetzt am linken Rand anfangen. Diese Einstellung kann man beim Aufruf des Befehls `\documentclass` ändern, indem man die Option `fleqn` entfernt.

[2] Hier werden die Anführungszeichen des fremdsprachlichen Zitates verwendet, während im nächsten Beispiel deutsche Anführungszeichen um den fremdsprachlichen Text herum benutzt werden. Insbesondere für kurze Zitate in anderen Sprachen gibt es dafür keine einheitlichen Regeln. Sie sollten aber zumindest konsistent vorgehen und im gesamten Dokument nur eine von beiden Alternativen verwenden.

[3] Weitz, 2021a.

Im mathematischen Kontext ist es üblich, dass Formeln wie Satzbestandteile der normalen Sprache behandelt werden. Das hat insbesondere Auswirkungen auf die Interpunktion. Zum Beispiel besagt die verallgemeinerte Kontinuumshypothese, dass für jede Ordinalzahl α die Gleichung $\aleph_\alpha = \beth_\alpha$ gilt. In der Sprache der Zermelo-Fraenkel-Mengenlehre heißt das

$$\forall \alpha \in \mathrm{ON}\ 2^{\aleph_\alpha} = \aleph_{\alpha+1}.$$

Dieses Beispiel sollte nur demonstrieren, dass hier an das Ende der Formel ein Punkt gehört, weil der Satz dort endet. Abgesetzte Formeln wie die gerade gezeigte sollten normalerweise nicht nummeriert werden. Eine Ausnahme davon sind natürlich Formeln wie (5.1), auf die später mit `\eqref` verwiesen wird.

Die Vorlage lädt automatisch die Erweiterung $\mathcal{AMS}$-LaTeX, die viele hilfreiche Erweiterungen für mathematische Ausdrücke enthält. Ohne dieses Paket wäre es beispielsweise wesentlich aufwendiger, Kettenbrüche wie

$$\mathrm{e} = \sum_{k=0}^{\infty} \frac{1}{k!} = \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = 2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{2}{3 + \cfrac{3}{4 + \cfrac{4}{5 + \cfrac{5}{\ddots}}}}}}$$

darzustellen.

5.7 Elemente in Gleitumgebungen

Tabellen und Abbildungen sollten grundsätzlich mit einer Beschriftung (`\caption`) versehen sein. Dazu gehört auch eine Quellenangabe, wenn es sich um Daten handelt, die Sie übernommen haben. Außerdem sollten solche Objekte immer in eine sogenannte Gleitumgebung wie `table` oder `figure` gesetzt werden, damit LaTeX für eine richtige Platzierung sorgen kann und die Objekte ggf. in Verzeichnissen (siehe Abschnitt 4.2) erfasst werden können. Viel mehr zu diesem Thema findet man u. a. in (Voß, 2021, Kap. 11).

In der Vorlage stehen Beschriftungen für Tabellen und Codeblöcke (siehe Abschnitt 5.8) jeweils links eingerückt über den Tabellen und Beschriftungen für Abbildungen zentriert unter diesen. Achten Sie darauf, den `\caption`-Befehl entsprechend am Anfang bzw. am Ende der Gleitumgebung aufzurufen.

5.7.1 Tabellen

Für typographisch anspruchsvoll gesetzte Tabellen verwendet die Vorlage das Paket BOOKTABS zusammen mit der Standardumgebung `tabular`. Beispiele für Tabellen finden Sie in den Dateien `chap2.tex` und `chap3.tex`. Grundsätzlich gilt, dass Linien in Tabellen im Allgemeinen *nicht* der Übersichtlichkeit dienen, sondern bezüglich der Lesbarkeit kontraproduktiv sind. In den meisten Ratgebern zur Typographie wird insbesondere empfohlen, auf vertikale Linien komplett zu verzichten! In Tabelle 5.1 gibt es eher zu viele Linien. Die wurden dort lediglich eingefügt, um ein paar Gestaltungsmöglichkeiten von BOOKTABS zu demonstrieren.

Tabelle 5.1: Durchfallquoten Mathematik

Fantasiewerte!	Jahr			
	2018	2019	2020	2021
Sommersemester	16,5 %	15,4 %	14,3 %	13,2 %
Wintersemester	26,5 %	25,4 %	24,3 %	23,2 %
Gesamt	21,5 %	20,4 %	19,3 %	18,2 %

Rein technisch ist es möglich, mit `\includegraphics` Tabellen einzufügen, die in anderen Programmen erzeugt wurden. Das sollte aber eigentlich immer vermieden werden, weil es sehr häßlich aussieht. Siehe dazu auch die Anmerkungen in Abschnitt 5.7.2. Für Tabellen, die länger als eine Seite sind, werden spezielle Pakete wie beispielsweise LONGTABLE benötigt.

Es gibt diverse Tools, die Daten aus Programmen wie EXCEL einlesen und diese in LaTeX-Syntax als Tabellen ausgeben können. Manche Programme, z. B. MATHEMATICA, können auch direkt nach LaTeX exportieren.

5.7.2 Abbildungen

Abbildungen können mit dem Befehl `\includegraphics` eingefügt werden. Wenn es sich nicht um Fotos handelt, ist jedoch darauf zu achten, dass auf jeden Fall skalierbare Vektorgrafiken verwendet werden. Rastergrafiken (wie sie beispielsweise durch Screenshots oder Scans erzeugt werden) führen, insbesondere im Druck, fast immer zu suboptimalen Ergebnissen. Das wird etwas übertrieben in Abbildung 5.1 dargestellt.

Abbildung 5.2 zeigt im Vergleich eine Vektorgrafik. Das ist schon besser, hat aber in den meisten Fällen den Nachteil, dass die Schriften nicht mit denen Ihrer Arbeit übereinstimmen werden, was unter typographischen Aspekten sehr unschön ist.[4] (Der zugehörige Code in der Datei `chap3.tex` zeigt nebenbei, wie man eingefügte Grafiken beschneiden kann.)

[4] Das gilt „erst recht“ für Tabellen, die aus externen Quellen eingefügt werden.

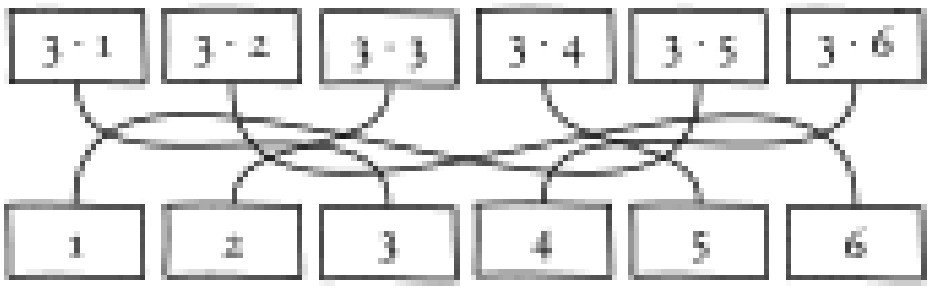

Abbildung 5.1: Ganz schlechtes Beispiel! (Quelle: Weitz, 2021a, S. 184)

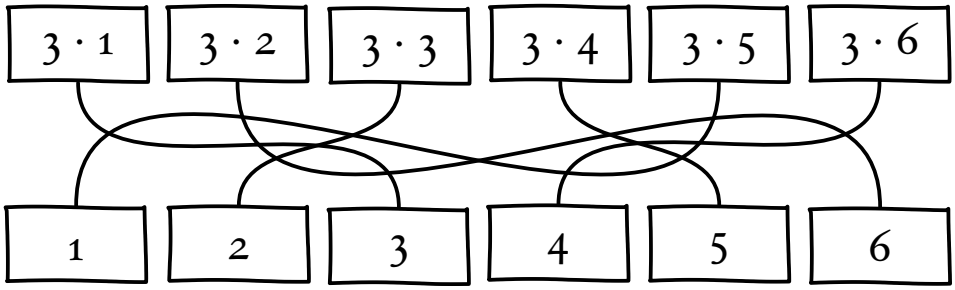

Abbildung 5.2: Besser, aber nicht optimal (Quelle: Weitz, 2021a, S. 184)

Die beste Lösung ist das Erstellen der Grafiken direkt in LaTeX. Dazu eignet sich hervorragend das Paket PGF/TikZ, das nahezu unbegrenzte Möglichkeiten bietet, jedoch eine steile Lernkurve hat. Dafür ist die mitgelieferte Dokumentation allerdings auch hervorragend. Als Einführung kann man auch die Videos in der Playlist

https://www.youtube.com/playlist?list=PLb0zKSynM2PBbpe9x6LgOZkSCl2yJAOQY

verwenden.

Diese Vorgehensweise hat zudem den Vorteil, dass man potentiellen Problemen mit dem Urheberrecht aus dem Weg geht, weil dieses nicht auf eine nach einer Vorlage selbst erstellte Grafik anwendbar ist. Nichtsdestotrotz muss aber auch in diesem Fall die Herkunft benannt werden!

Die Abbildungen 5.3 und 5.4 zeigen Beispiele für solche Grafiken. Den zugehörigen Quellcode findet man in der Datei `chap3.tex`.

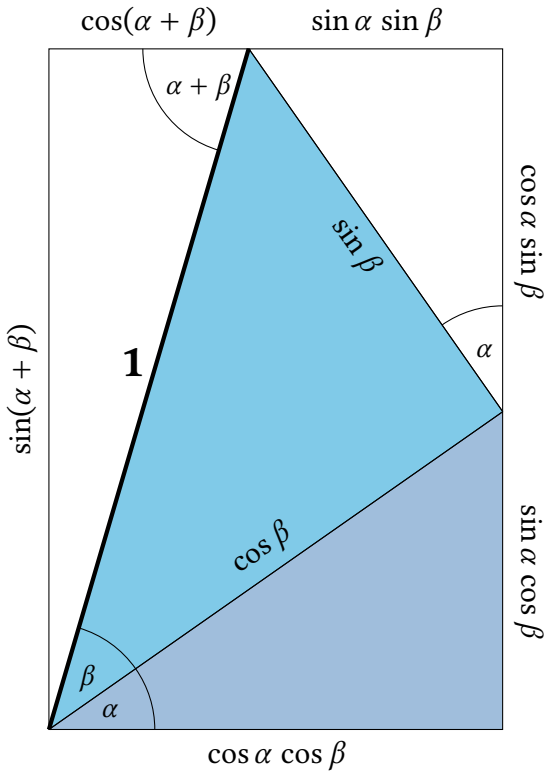

Abbildung 5.3: Mit TikZ [eigene Grafik nach (Weitz, 2021a, S. 271)]

5.8 Code

Für die Darstellung von Programmcode wie in Codeblock 5.1 wird in der Vorlage das Paket LISTINGS verwendet.

Codeblock 5.1: Eulersches Polygonzugverfahren

```
1 def euler(f,x0,y0,h,n):
2     x, y, result = x0, y0, [(x0,y0)]
3     for i in range(n):
4         y += f(x,y) * h
5         x += h
6         result.append((x,y))
7     return result
```

In der Datei `chap3.tex` kann man sehen, wie der Code aus einer externen Datei eingelesen wird. Durch diese Vorgehensweise kann man dafür sorgen, dass auch tatsächlich die aktuelle Version des eigenen Codes verwendet wird, und man vermeidet potentielle Fehler beim Abtippen. Man kann den Code aber auch wie in Codeblock 5.2 direkt eintippen.

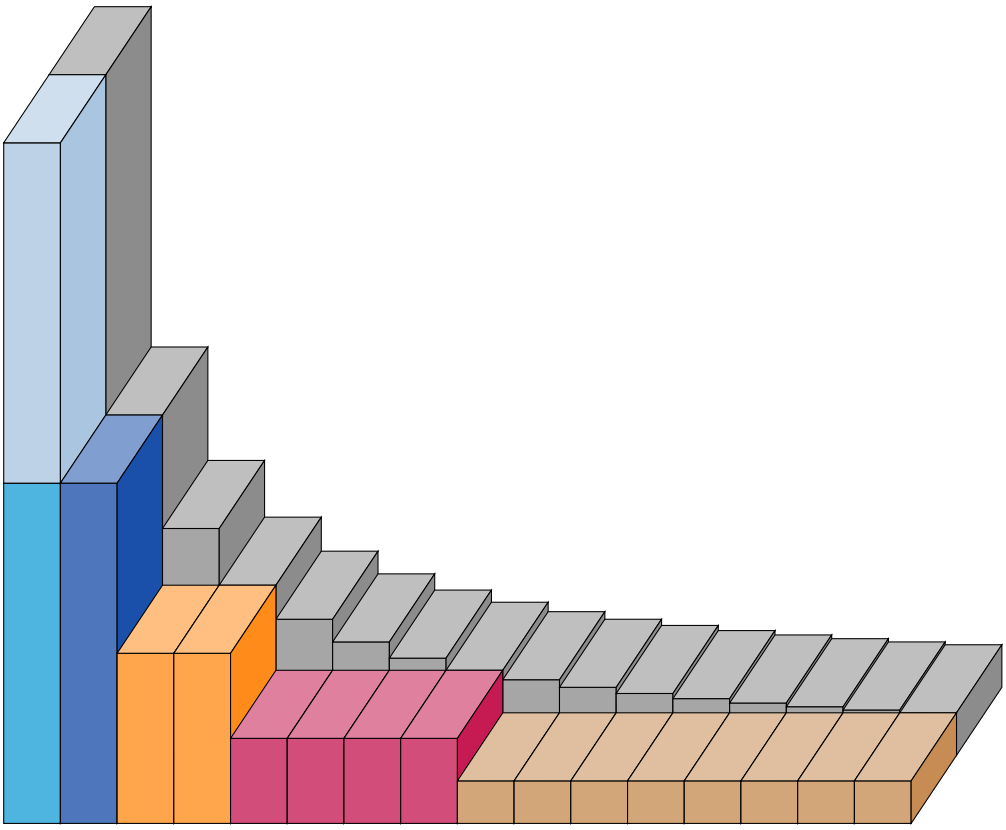

Abbildung 5.4: Auch mit TikZ [eigene Grafik nach (Weitz, 2021a, S. 533)]

Codeblock 5.2: Variationen

```
1 def kperm(L, k):
2     if k == 0:
3         return [[]]
4     return [[L[i]]+P for i in range(len(L))
5                      for P in kperm(L[:i] + L[i+1:], k-1)]
```

In der Datei `defs.tex` wird exemplarisch gezeigt, wie man das Aussehen der Codeblöcke individuell gestalten kann. Es ist auch möglich, Codeblöcke wie Abbildungen und Tabellen „gleiten“ zu lassen.

Ein Verzeichnis der Codeblöcke kann mit dem Paket LISTINGS bei Bedarf erzeugt werden, wenn Ihre Arbeit sehr viele Codeblöcke enthält. Umfangreiche Codeblöcke gehören aber nicht in die Arbeit und auch nicht in den Anhang, sondern sollten – ggf. nach Absprache mit der Erstprüferin bzw. dem Erstprüfer – auf einem Datenträger zusammen mit der Arbeit eingereicht werden.

Literatur

akopytov. (2024). *Sysbench Github Repository*. Verfügbar 28. Oktober 2024 unter https://github.com/akopytov/sysbench

Difallah, D. E., Pavlo, A., Curino, C., & Cudré-Mauroux, P. (2013). OLTP-Bench: An Extensible Testbed for Benchmarking Relational Databases. *PVLDB*, *7*(4), 277–288. http://www.vldb.org/pvldb/vol7/p277-difallah.pdf

Germany, R. (2024). *Was ist ein Snapshot Backup?* Verfügbar 28. Oktober 2024 unter https://www.rubrik.com/de/insights/what-is-a-snapshot-backup

Reimers, N. (2017). *Virtuelle, dezidierte und Cloud-Server: MySQL-Benchmark mittels sysbench*. Verfügbar 28. Oktober 2024 unter https://www.webhosterwissen.de/know-how/server/mysql-benchmark-mittels-sysbench/

Schlosser, J. (2021). *Wissenschaftliche Arbeiten schreiben mit LaTeX: Leitfaden für Einsteiger* (7. Aufl.). mitp.

Shopify. (2022a). *Detailed design documentation*. Verfügbar 28. Oktober 2024 unter https://shopify.github.io/mybench/detailed-design-doc.html#live-monitoring-user-interface

Shopify. (2022b). *What is mybench?* Verfügbar 28. Oktober 2024 unter https://shopify.github.io/mybench/introduction.html

Shopify. (2024). *Mybench Github Repository*. Verfügbar 28. Oktober 2024 unter https://github.com/Shopify/mybench

Vogel, M. (2009). *EDV-Lexikon: Bottleneck*. Verfügbar 28. Oktober 2024 unter https://martinvogel.de/lexikon/bottleneck.html

Voß, H. (2021). *Die wissenschaftliche Arbeit mit LaTeX: unter Verwendung von LuaTeX, KOMA-Script und Biber/BibLaTeX* (2. Aufl.). Lehmanns Media.

Weitz, E. (2021a). *Pi und die Primzahlen: Eine Entdeckungsreise in die Mathematik*. Springer.

Weitz, E. (2021b). Hausdorff's forgotten proof that almost all numbers are normal. *Math. Semesterberichte*, *68*(2), 273–282. https://doi.org/10.1007/s00591-021-00303-w

Williams, T., Kelley, C., & many others. (2024). *Gnuplot Repository*. Verfügbar 28. Oktober 2024 unter https://github.com/gnuplot/gnuplot

Zappa, F. (1974). *Apostrophe ('): Stink-Foot*. Verfügbar 11. März 2022 unter http://www.donlope.net/fz/lyrics/Apostrophe_(').html

Anhang

Hier beginnt der Anhang. Siehe die Anmerkungen zur Sinnhaftigkeit eines Anhangs in Abschnitt 4.5 auf Seite 21.

Der Anhang kann wie das eigentliche Dokument in Kapitel und Abschnitte unterteilt werden. Der Befehl `\appendix` sorgt im Wesentlichen nur für eine andere Nummerierung.

Eigenständigkeitserklärung

Hiermit versichere ich, dass ich die vorliegende Bachelorarbeit mit dem Titel

Viele zufällige Zahlen

selbstständig und nur mit den angegebenen Hilfsmitteln verfasst habe. Alle Passagen, die ich wörtlich aus der Literatur oder aus anderen Quellen wie z. B. Internetseiten übernommen habe, habe ich deutlich als Zitat mit Angabe der Quelle kenntlich gemacht.

Hamburg, 21. Dezember 1940