

Fachhochschule der Wirtschaft

-FHDW-

Paderborn

Bachelorarbeit

Thema:

**Untersuchung von Einflüssen auf die Performance von Datenbanken und
Datensynchronisation auf mobilen Endgeräten**

Prüfer:

Herr Prof. Dr. Reus

Herr Dr. Baum

Verfasser:

Jens Milbredt

Matrikelnummer: 1615328

6. Semester

Studiengang Angewandte Informatik

Schwerpunkt: Technik

Eingereicht am:

28.04.2009

Inhaltsverzeichnis

Inhaltsverzeichnis.....	ii
Abbildungsverzeichnis.....	iii
Tabellenverzeichnis.....	iii
Abkürzungsverzeichnis.....	iv
1. Einführung.....	1
2. Grundlagen der Kommunikation mobiler Geräte.....	3
2.1. Technologien für mobile Netzwerkkommunikation.....	3
2.2. Konsequenzen der Beschränkungen mobiler Netzwerktechnologien für mobile Clients.....	10
3. Darstellung mobiler Datenhaltung.....	12
3.1. Anforderungen an Datenbanksysteme.....	12
3.2. Anforderungen an mobile Datenbanksysteme.....	14
3.3. Grundlegender Aufbau von Datenbanksystemen.....	14
3.4. Aufbau von mobilen Datenbanksystemen.....	16
3.5. Grenzen mobiler Datenbanksysteme.....	20
4. Modellierung von Client-Systemen für mobile Geräte.....	22
4.1. Anforderungen durch die Netzwerkschnittstellen für mobile Geräte.....	22
4.2. Einführung in verteilte Systeme.....	23
4.3. Einführung in verteilte Datenbanksysteme.....	27
4.3.1. Grundlagen.....	27
4.3.2. Replikationsmodelle für den Einsatz mit mobilen Geräten.....	31
4.3.3. Vergleich von Architekturen für verteilte DBS mit mobilen Geräten.....	33
5. Performance-Tests	36
5.1. Notwendigkeit von Performance-Tests	36
5.2. Herangehensweise an Performance-Tests	37
6. Durchführung eines Performance-Tests eines mobilen Datenbanksystems und einer Synchronisation.....	44
6.1. Einführung.....	44
6.2. Hardware- und Softwareumgebung.....	44
6.3. Abgrenzungen.....	45
6.4. Schlüsselszenarien.....	46
6.5. Arbeitslast.....	50
6.6. Metriken.....	57
6.7. Testfälle.....	57
6.7.1. Testfälle Szenario 1.....	57
6.7.2. Testfälle Szenario 2 und 3.....	59
6.8. Durchführung des Tests und Modell der Testapplikation.....	61
6.8.1. Struktur des Gesamtsystems.....	61
6.8.2. Anwendungsfälle.....	62
6.8.3. Abläufe.....	64
6.8.4. Paketaufbau.....	71
6.8.5. Modell Server.....	73
6.8.6. Modell mobiler Client.....	76
6.9. Darstellung der Messergebnisse.....	79
6.9.1. Ergebnisse Szenario 1 (Synchronisation von Daten).....	79
6.9.2. Abschließende Betrachtung der Synchronisationsergebnisse.....	81

6.9.3. Ergebnisse Szenario 2 (Anfragen zur Ermittlung von Daten aus der mobilen Datenbank).....	83
6.9.4. Ergebnisse Szenario 3 (Datenmanipulationen).....	84
6.9.5. Abschließende Betrachtung der Datenbankergebnisse.....	86
7. Ergebnis und Fazit.....	88
Literaturverzeichnis.....	90
Ehrenwörtliche Erklärung.....	97
Anhang.....	98

Abbildungsverzeichnis

Abbildung 1: Clusteraufbau im WWAN.....	7
Abbildung 2: Charakteristika der Funktechnologien der einzelnen Generationen.....	9
Abbildung 3: Allgemeine Architekturen mobiler Datenbanksysteme.....	17
Abbildung 4: Architektur des SQL Server CE.....	19
Abbildung 5: Peer-To-Peer-Modell und Client-Server-Modell.....	23
Abbildung 6: Schemata einer verteilten Datenbank.....	28
Abbildung 7: Architekturen eines verteilten DBS.....	34
Abbildung 8: Beispieldiagramm.....	43
Abbildung 9: Datenbankmodell für den Performance-Test.....	48
Abbildung 10: Verteilungsdiagramm der Testkomponenten.....	61
Abbildung 11: Datenbankschema der Datenbank für die Speicherung der Messergebnisse.....	62
Abbildung 12: Anwendungsfälle.....	63
Abbildung 13: Aktivitätsdiagramm – Testzustand herstellen.....	64
Abbildung 14: Aktivitätsdiagramm – DBS-Test durchführen	65
Abbildung 15: Aktivitätsdiagramm – Synchronisationstest.....	66
Abbildung 16: Unteraktivitäten für den Synchronisationstest.....	68
Abbildung 17: Aktivitätsdiagramm – Basisdatenmenge erstellen.....	68
Abbildung 18: Aktivitätsdiagramm – Überprüfung der Messergebnisse durchführen. .	69
Abbildung 18: Aktivitätsdiagramm – Überprüfung der Messergebnisse durchführen. .	69
Abbildung 19: Aktivitätsdiagramm – Diagramme erstellen.....	70
Abbildung 20: Aktivitätsdiagramm – Datenbanken anlegen.....	70
Abbildung 21: Paketdiagramm.....	71
Abbildung 22: Modell der Serverseite.....	74
Abbildung 23: Modell mobile Testapplikation.....	77
Abbildung 24: Modell Clientbibliothek.....	78
Abbildung 25: Ergebnis Szenario 1 – Testfälle für GPRS.....	79
Abbildung 26: Ergebnis Szenario 1 – Testfälle für WLAN.....	80
Abbildung 27: Ergebnis Szenario 1 – Gegenüberstellung WLAN und GPRS.....	80
Abbildung 28: Ergebnis Szenario 2.....	83
Abbildung 29: Ergebnis Szenario 3 – Testfall „Aktualisierung von Daten“.....	84
Abbildung 30: Ergebnis Szenario 3 – Testfall „Einfügen von Daten“.....	85
Abbildung 31: Ergebnis Szenario 3 – Testfall „Löschen von Daten“.....	86

Tabellenverzeichnis

Tabelle 1: OSI-Modell.....	6
Tabelle 2: Darstellung der verschiedenen WWAN-Generationen.....	9

Tabelle 3: Vorgehensmodell für Performance-Tests.....	38
Tabelle 4: Metriken für Performance-Tests.....	40
Tabelle 5: Daten des mobilen Gerätes.....	45
Tabelle 6: Daten des Servers.....	45
Tabelle 7: Spaltendefinition für die Testdatenbank.....	49
Tabelle 8: Szenarien.....	50
Tabelle 9: Berechnung der Größe der Datenblöcke für das Erhöhen der Last (alle Szenarien).....	54
Tabelle 10: Berechnung der Größe der Datenblöcke für Veränderungen (Szenario 1). ..	55
Tabelle 11: Testfälle Szenario 1 (Synchronisation).....	58
Tabelle 12: Testfälle Szenario 2 (Anfragen) und Szenario 3 (Datenmanipulationen)....	61

Abkürzungsverzeichnis

ACID: Atomicity, Consistency, Isolation and Durability
AMPS: Advanced Mobile Phone Service
bps: Bits pro Sekunde
Bps: Bytes pro Sekunde
CDMA: Code Division Multiple Access
DB: Datenbank
DBMS: Datenbankmanagementsystem
DBS: Datenbanksystem
Ds: Datensatz/Datensätze
FDM: Frequenz Division Multiplexing
FTP: File Transfer Protocol
GPRS: General Packet Radio Services
GPS: Global Positioning System
GSM: Global System For Mobile Communications
GUI: Graphical User Interface
GUID: Globally Unique Identifier
HTTP: Hypertext Transport Protocol
HTTPS: Hypertext Transport Protocol Secure
IEEE: Institute Of Electrical And Electronics Engineers
ISO: International Standard Organisation
LBS: Location-Based Services
LDBS: Lokales Datenbanksystem
MDBS: Multidatenbanksystem
IIS: Microsoft Internet Information Services
OSI: Open Systems Interconnection

RDA: Remote Database Access
RPC: Remote Procedure Call
SMS: Short Message Service
SMTP: Simple Mail Transfer Protocol
SOAP: Simple Object Access Protocol
SQL: Structured Query Language
TCP/IP: Transmission Control Protocol/Internet Protocol
UMTS: Universal Mobile Telecommunications Service
TDM: Time Division Multiplexing
WAP: Wireless Application Protocol
WIMAX: Worldwide Interoperability For Mikrowave Access
WLAN: Wireless Local Area Network
WWAN: Wireless Wide Area Network
XML: Extensible Markup Language
Z: Zeichen

1. Einführung

Moderne Technologien haben zu einer Verkleinerung von Computersystemen geführt. Dadurch sind kompakte und, im Verhältnis zur Größe, leistungsfähige mobile Geräte entstanden. Beispiele dafür sind intelligente Mobiltelefone (Smartphones), Personal Digital Assistants (PDA, kleine Computer, die man in der Hand tragen kann), Spezialgeräte wie Barcodescanner für Lagerhäuser, welche direkt mit zentralen Systemen verbunden sind, oder Navigationsgeräte. Diese mobilen Geräte bilden die Grundlage, um Benutzer unabhängig von ihrem Aufenthaltsort zu unterstützen. Solche Geräte stehen häufig vor dem Problem, lokal Daten speichern zu müssen und diese verschiedenen Anwendungen zur Verfügung zu stellen. Um eine strukturierte Datenhaltung mit standardisierten Schnittstellen für kleine Geräte zu ermöglichen, sind mobile Datenbanksysteme entwickelt worden, welche Datenbestände innerhalb von Datenbanken auf mobilen Geräten verwalten. Es besteht häufig die Notwendigkeit, die Daten von mobilen Geräten an einem zentralen Ort zu sammeln und zu speichern, bzw. Daten aus einem zentralen System an mehrere mobile Geräte zu verteilen. Daher spielen Synchronisationsverfahren und Modelle für die Verteilung von Daten bei der mobilen Datenhaltung eine zentrale Rolle. Synchronisationen werden dabei häufig mithilfe von Funktechnologien durchgeführt, die den mobilen Geräten eine ortsunabhängige Kommunikation mit zentralen Systemen ermöglichen.

Mobile Geräte sind jedoch nicht so leistungsfähig wie stationäre Computersysteme. Innerhalb dieser Arbeit wird die These vertreten, dass die Leistungsfähigkeit der mobilen Datenbanksysteme aufgrund der geringen Leistungsfähigkeit der mobilen Geräte eingeschränkt ist. Weiterhin wird vertreten, dass die Leistungsfähigkeit von Synchronisationsschnittstellen durch die Verwendung von Funktechnologien für mobile Geräte ebenfalls eingeschränkt ist. Um die Thesen zu prüfen, soll mithilfe eines Performance-Tests gezeigt werden, wie genau sich eine mobile Datenbank sowie Synchronisation unter Last verhält und wie sich die geringe Leistungsfähigkeit auf die Performance auswirkt.

Um darzustellen, welchen Beschränkungen die Netzwerktechnologien für mobile Geräte unterliegen, werden in dieser Arbeit zunächst die Grundlagen der mobilen Kommunikation dargestellt. Dabei wird speziell auf die Funktechnologien eingegangen, da diese die mobile (also ortsunabhängige) Kommunikation ermöglichen. Weiterhin werden die Schwachpunkte in den Technologien aufgezeigt, ein Überblick über die

einzelnen Technologien gegeben sowie die Konsequenzen aus den Schwächen dargestellt.

Im weiteren Verlauf wird beschrieben, wie mobile Datenbanksysteme strukturiert und aufgebaut werden. Dazu gehören die Anforderungen, welche an Systeme gestellt werden, die eine strukturierte Speicherung von Informationen realisieren. Dabei werden allgemeine Anforderungen wie auch Anforderungen an mobile Systeme dargestellt. Weiterhin wird der Aufbau von Datenbanksystemen dargestellt. Dabei werden zunächst allgemeine Konzepte und dann speziell mobile Architekturen betrachtet. Abgeschlossen wird dieses Kapitel mit einer kompakten Darstellung der Grenzen von mobilen Datenbanksystemen.

Fortgesetzt wird die Arbeit mit der Beschreibung von verteilten Systemen und verteilten Datenbanken in Kapitel 4, welche mobile Geräte mit einbeziehen. Diese bilden die Grundlage für die Sammlung von Daten in einem zentralen System. Um verteilte Systeme zu beschreiben, wird zunächst auf die Anforderungen der mobilen Netzwerkschnittstellen eingegangen, die an ein Gesamtsystem aus mehreren mobilen und stationären Geräten gestellt werden. Danach werden die Grundlagen zur Verteilung von Applikationen und basierend darauf zur Verteilung von Datenbeständen und Datenbanksystemen dargestellt. Der Schwerpunkt liegt dabei auf Modellen und Techniken, die auf die Probleme und Anforderungen der mobilen Geräte angepasst sind. Die Grundlagen für die Tests, die die Leistungsfähigkeit mobiler Datenbanken und Synchronisationen ermitteln sollen, werden in Kapitel 5 dargestellt. Darin wird gezeigt, welche Schritte notwendig sind, um einen solchen Performance-Test durchzuführen.

Entsprechend der Schritte in Kapitel 5 wird ein Testsystem aufgebaut. Dazu gehört die Definition von Szenarien sowie einzelner detaillierter Testfälle, die die zu testenden Operationen darstellen. Basierend darauf wird eine verteilte Applikation aufgebaut, welche die automatisierte Durchführung dieser Testfälle ermöglicht. Kapitel 6 stellt diesen Testaufbau sowie den Aufbau der Testapplikation dar. Dieses Kapitel wird abgeschlossen durch die Darstellung der Testergebnisse.

Kapitel 7 fasst das Ergebnis zusammen und zieht ein Fazit. Weiterhin gibt es einen Ausblick auf mögliche Weiterentwicklungen des Testsystems bzw. der Testszenarien sowie einen kurzen Ausblick auf die Konsequenzen zukünftiger Entwicklungen der mobilen Geräte bzw. der Funktechnologien.

2. Grundlagen der Kommunikation mobiler Geräte

Innerhalb dieses Kapitels werden die Grundlagen der Netzwerkkommunikation mobiler Geräte dargestellt. Dabei beschränkt sich diese Arbeit auf kabellose Technologien, die mobilen Applikationen eine begrenzt ortsunabhängige Kommunikation erlauben. Diese werden im Normalfall von mobilen Geräten zur Kommunikation verwendet.

2.1. Technologien für mobile Netzwerkkommunikation

Bei Technologien für mobile Kommunikation kann man zwischen zwei grundlegenden Systemen unterscheiden. Erstens gibt es das **Wireless Local Area Network (WLAN)** für die Verwendung in lokalen Bereichen wie z. B. einem Firmengebäude. Zweitens gibt es das **Wireless Wide Area Network (WWAN)** zur Abdeckung großer Bereiche, wie z. B. die Funknetzwerke für Mobiltelefone.¹

Solche WWANs geben dem Benutzer eine wesentlich höhere Ortsunabhängigkeit. Gleichzeitig ist der Aufbau komplexer und unterliegt zusätzlichen Beschränkungen. Die Strukturen des WWANs werden genauer dargestellt, um zu zeigen, wo im Gegensatz zu WLANs zusätzliche Beschränkungen auftreten. Weiterhin ist die Gesamtstruktur komplexer und es gibt im Gegensatz zu WLAN mehr Technologien mit unterschiedlichen Charakteristiken.

Die für die kabellosen Netzwerke verwendeten Technologien basieren auf elektromagnetischen Wellen. Charakterisierend für diese Wellen ist die Frequenz (Anzahl Schwingungen pro Sekunde). Dabei steigt mit höheren Frequenzen die Anzahl der Daten, die man übertragen kann. Gleichzeitig steigt die Fehleranfälligkeit und sinkt die Entfernung, über die gesendet werden kann. Ebenfalls zeigt sich an dieser Stelle ein Bruch mit der digitalen Informationsverarbeitung von Computersystemen. Funkwellen sind analog und müssen in digitale Signale umgewandelt werden, damit diese weiterverarbeitet werden können.²

Neben dem Einfluss der Frequenz und der Wellenlänge wird die Qualität der Wellen durch weitere äußere Einflüsse gestört. Zunächst werden sie schwächer, sobald sie feste Materie durchdringen oder sobald sich die Entfernung zwischen Sender und Empfänger erhöht. Die Fähigkeit der Durchdringung von Materie wird dabei stark von der Frequenz beeinflusst (um so höher, desto schlechter). Ein weiteres großes Problem ist, dass Wellen, die sich auf derselben Frequenz befinden, sich gegenseitig stören. Auch

¹ Vergl. Murthy, C. Siva Ram et al. (2004), S. 63f und S. 109f

² Vergl. Murthy, C. Siva Ram et al. (2004), S. 2ff sowie S. 10f

andere Störsignale wie ein laufender Motor oder das Grundrauschen der Atmosphäre beeinflussen die Wellen. Reflektionen von Wellen, z. B. auf den Boden oder an Wänden, führen zu einer Verlangsamung von Teilstücken der Wellen oder zu einer Verdoppelung der Signale bis zum Erreichen der Basisstation. Die Auswirkung dessen ist eine fehlerhafte Interpretation der Wellen beim Empfänger. Reflektionen treten häufiger und stärker auf, um so höher die Frequenz ist. Das heißt, die den verschiedenen Technologien zur Verfügung gestellten Frequenzbereiche müssen sorgfältig ausgewählt werden, damit eine Kombination aus großer Reichweite und geringen Fehlerraten erreicht werden kann.³

Innerhalb der Verwendung von Funktechnologien gibt es ein weiteres Problem. Viele Systeme mit einer hohen Benutzeranzahl (Funkverkehr, Mobiltelefone, Radio) basieren auf elektromagnetischen Wellen. Der Benutzeranzahl gegenüber stehen eine begrenzte Anzahl von Frequenzen, die die einzelnen Benutzer verwenden können. Dies begründet die Verwaltung der verschiedenen Frequenzen durch Regulierungsbehörden. Diese teilen den verschiedenen Technologien bestimmte Frequenzbereiche zu, die diese verwenden dürfen.⁴

Diese Aufteilung ist allerdings nicht ausreichend. Innerhalb einer Technologie kann die Notwendigkeit auftreten, den verfügbaren Frequenzbereich unter vielen Benutzern aufzuteilen (*Frequenzteilung*). Zwei grundlegende Techniken dafür stellen das **Time Division Multiplexing** (TDM) sowie das **Frequency Division Multiplexing** (FDM) dar, welche kombiniert werden können. Innerhalb der TDM-Technik senden und empfangen mehrere Benutzer auf einem Frequenzbereich, wobei nur ein Benutzer zu einer Zeit sendet. Um dies zu gewährleisten, werden einzelne Zeitabschnitte definiert, in denen die einzelnen Teilnehmer den Frequenzbereich verwenden dürfen. Bei entsprechend kleinen Zeitabschnitten werden Unterbrechungen, z. B. bei Telefongesprächen oder bei einem Datenaustausch, nicht bemerkbar. Bei FDM wird der vorhandene Frequenzbereich in mehrere disjunkte Bereiche (so genannte Kanäle) geteilt, auf dem Benutzer unabhängig voneinander senden und empfangen können. Dies bedeutet, wenn z. B. eine Technologie einen Frequenzbereich zwischen 100 kHz und 200 kHz zugeteilt bekommt und zwei Benutzer diese am gleichen Ort verwenden, erhält ein Benutzer den Frequenzbereich zwischen 100 kHz und 140 kHz und der zweite den zwischen 160 kHz

3 Vergl. Kurose, James F. et al.(2008), S. 519; Tse, David et al. (2006), S. 18f sowie Murthy, C.Siva Ram et al. (2004), S. 2ff und S. 10f

4 Vergl. Mutschler, Bela et al. (2004), S. 26f

und 200 kHz. Der Teil zwischen 140 kHz und 160 kHz wird dabei als Trennschicht verwendet, damit sich beide Teilnehmer bei starken Schwankungen der Wellen nicht stören.⁵ Weiterführende Techniken verfeinern die Frequenzteilung unter den Benutzern. Die Verwaltung der Verbindungen und die Verwendungen dieser Techniken beinhalten einen Overhead sowie das Risiko zusätzlicher Störungen. Weiterhin sinkt die Übertragungsrate pro zusätzlichen Benutzer.⁶

Die verschiedenen Lösungsansätze, die den Umgang mit dem unsicheren physikalischen Medium erlauben, werden von den verschiedenen Funktechnologien umgesetzt. Diese Funktechnologien gliedern sich dabei in die verschiedenen Schichtenmodelle ein, in die Netzwerke strukturiert werden. Diese sind entstanden, um die verschiedenen Probleme bei einer Netzwerkkommunikation zwischen Computersystemen zu kapseln und zu lösen. Ein solches Schichtenmodell zeigt, welche Aufgaben in einem Netzwerk gelöst werden müssen. Tabelle 1 zeigt das von der ISO (**I**nternational **S**tandard **O**rganisation) entwickelte Modell für eine Schichtenarchitektur eines Netzwerkes. Das Modell trägt den Namen OSI-Modell (**O**pen **S**ystems **I**nterconnection). Neben diesem Modell gibt es das TCP/IP-Modell (**T**ransmission **C**ontrol **P**rotocol/**I**nternet **P**rotocol), das sich in der Praxis im Internet oder lokalen Netzwerken durchgesetzt hat sowie das **W**ireless **A**pplication **P**rotocol (WAP), welches speziell für Funktechnologien entwickelt worden ist. Das OSI-Modell besitzt gegenüber dem TCP/IP- bzw. dem WAP-Protokoll mehr Schichten und zeigt deutlicher, welche Aufgaben innerhalb eines Netzwerks gelöst werden müssen. Daher wird in dieser Arbeit das OSI-Modell dargestellt. Das größte Problem der Schichtenarchitektur besteht dabei in dem Overhead, welcher sich aus dem Durchlaufen der Daten durch mehrere Schichten ergibt.⁷

Nr.	Schicht	Aufgabe
7	Applikation (Application Layer)	Schnittstellen für Applikationen
6	Darstellung (Presentation Layer)	Diese Schicht übersetzt system-spezifische Datendarstellungen in eine rein logische Darstellung und umgekehrt.
5	Sitzung (Session Layer)	Die Sitzungsschicht sorgt für eine geordnete Kommunikation zwischen zwei Systemen. Dazu gehört der Auf- und Abbau von Verbindungen und das Wiederherstellen von Verbindungen.
4	Transport (Transport Layer)	Diese Schicht übernimmt den Versand und Empfang der Daten. Für die Versendung werden die Daten in bestimmte Strukturen (Pakete) verpackt,

5 Dies ist ein rein fiktives Beispiel. Wie die Zahlenwerte in der Realität aussehen, hängt von der jeweiligen Realisierung und dem verwendeten Frequenzbereich ab.

6 Vergl. Murthy, C. Siva Ram et al. (2004), S. 19 ff, als weiterführende Literatur im Bereich der Frequenzteilung können die Bücher Murthy, C. Siva Ram et al. (2004) sowie Tse, David et al. (2006) verwendet werden.

7 Vergl. Shanmugam, Ramadas et al. (2002), S. 16 und Lehner, Franz (2003), S. 142f

Nr.	Schicht	Aufgabe
		welche beim Empfänger wieder in die ursprüngliche Darstellung umgewandelt werden.
3	Vermittlung (Network Layer)	Die Vermittlungsschicht übernimmt das Auffinden von Kommunikationspartnern im Gesamtnetzwerk.
2	Sicherung (Link Layer)	Diese Schicht übernimmt die Umwandlung von Signalen in Datenblöcke und umgekehrt. Sie stellt Sicherungsmechanismen zur Verfügung, um die Vollständigkeit der Übertragung zu garantieren.
1	Bitübertragung (Physical Layer)	Die Aufgabe dieser Schicht ist der Umgang mit physikalischen Verbindungen und das Versenden und Empfangen von Signalen.

Tabelle 1: OSI-Modell⁸

Die Realisierungen für den Umgang mit Funk befinden sich in der Schicht eins bis drei.⁹ Während Schicht eins den Umgang mit dem physikalischen Medium kapselt, übernimmt Schicht zwei neben den Sicherungsmechanismen auch die Techniken zum Mehrfachzugriff auf das Medium.¹⁰ Innerhalb von Schicht drei wird die Auffindung des jeweiligen Kommunikationspartners realisiert. Daneben ermöglicht sie innerhalb von WWANs die Übergabe von Verbindungen zwischen zwei Basisstationen (bzw. Zellen, diese werden im folgenden noch näher erläutert). Innerhalb von Schicht sieben können spezielle Probleme durch die mobile Umgebung auftreten. Dazu gehört z. B. die Auffindung von Diensten in mobilen Ad-Hoc-Netzwerken und die Verwendung verschiedener Geräte und Betriebssysteme.¹¹

WLANs unterteilen sich dabei in zwei unterschiedliche Arten von Netzwerken. Zunächst gibt es Funknetze, die auf einer stationären Basisstation basieren, welche die Verwaltung des drahtlosen Netzwerkes übernimmt. Diese kann eine Weiterleitung in kabelbasierte Netzwerke oder dem Internet anbieten. Um eine größere Abdeckung des WLANs zu erreichen, können mehrere Basisstationen eines WLANs miteinander verbunden werden. Dem gegenüber stehen die Ad-Hoc-Netzwerke, in denen sich die teilnehmenden Geräte selbst organisieren. Diese Art des WLANs hat keine Infrastruktur, auf die sie sich stützen kann. Somit verbleibt der Verwaltungsaufwand des Netzwerkes bei den einzelnen Teilnehmern. Welche von diesen beiden Kategorien von Netzwerken unterstützt werden, hängt von der jeweiligen Technologie ab.¹²

Die verschiedenen Techniken und Probleme haben zu mehreren Standards geführt.

⁸ Vergl. Shanmugam, Ramadas et al. (2002), S. 17ff und Kurose, James F. et al. (2008), S. 48

⁹ Vergl. Kurose, James F. et al. (2008), S. 513 und S. 48 sowie Lehner, Franz (2003), S. 141 und S. 147 (Beide beziehen sich auf das TCP/IP-Protokoll, allerdings zeigen beide im Vergleich von TCP/IP und OSI, dass im OSI-Modell diese drei Schichten betroffen sind), vergl. auch Mutschler, Bela (2004), S. 23

¹⁰ Vergl. Mutschler, Bela (2004), S. 23 und Häckelmann, Heiko et. al (2000), S. 115

¹¹ Vergl. Mutschler, Bela (2004), S. 23

¹² Vergl. Tse, David et al. (2006), S. 4f

Diese legen die Charakteristika und Schnittstellen der jeweiligen Technologie fest. Im Rahmen von lokalen Netzwerken mit Basisstationen haben sich Technologien nach dem Standard IEEE 802.11 (Institute of Electrical and Electronics Engineers) durchgesetzt. Diesen Standard gibt es in mehreren Ausführungen, die sich in Hinsicht auf Frequenz und Datenrate unterscheiden. Dabei unterstützt der IEEE 802.11-Standard auch Ad-Hoc-Netzwerke. Eine weitere Technologie innerhalb der Ad-Hoc-Netzwerke stellt z. B. Bluetooth dar.¹³

Den WLAN-Technologien gegenüber stehen zellenbasierte Netzwerke, auf denen die gängigen kabellosen WWANs für mobilen Datenaustausch basieren. Diese Netzwerke bezeichnet man als zellulare Netzwerke. Sie bieten eine flächendeckende Lösung an, um den Benutzern dieser Netzwerke unabhängig von deren Ort und deren Bewegungen eine unterbrechungsfreie Verbindung zu ermöglichen. Speziell die höhere Mobilität stellt einen zentralen Unterschied gegenüber den WLAN-Lösungen dar.

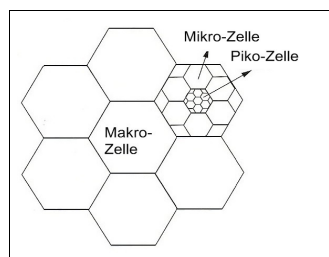


Abbildung 1: Clusteraufbau im WWAN¹⁴

Ein wichtiger Bestandteil dieser Netze sind *Cluster*, die eine Zusammenfassung von mehreren Zellen mit jeweils einer eigenen stationären Basisstation darstellen (siehe Abbildung 1). Dabei kann die Anzahl der Zellen variieren. Idealerweise nehmen Zellen eine hexagonale Form an, welche in der Praxis durch örtliche Gegebenheiten verzerrt werden. Die Verbindungen in einer Zelle (und somit zu einer Basisstation) kann an eine zweite Zelle übergeben werden. Auch zwischen verschiedenen Clustern können Verbindungen weitergegeben werden. Das gesamte Netzwerk wird aus solchen Clustern gebildet.¹⁵

Innerhalb eines Clusters wird der verfügbare Frequenzbereich unter den einzelnen Zellen aufgeteilt. Jeder Cluster kann wiederum denselben Frequenzbereich verwenden. Dies funktioniert, da sich Funkwellen mit der Zeit abschwächen und die Cluster eine solche Abdeckung erreichen, dass sie sich nicht gegenseitig stören (*Frequency reusing*).

¹³ Vergl. Kurose, James F. et al. (2008), S. 526f und S. 544f

¹⁴ Vergl. Murthy, C. Siva Ram et al. (2004), S. 112

¹⁵ Vergl. Murthy, C. Siva Ram et al. (2004), S. 110 ff und Tse, David et al. (2006), S. 121 sowie Walke, Bernhard (2001), S. 53

Je weniger Zellen in einem Cluster vorhanden sind, desto mehr Frequenzen stehen pro Zelle zur Verfügung. Um einen fließenden Übergang zwischen den einzelnen Zellen und Clustern zu ermöglichen, gibt es so genannte *Handover-* oder *Handoff-Mechanismen*. Diese Mechanismen ermöglichen den Übergang, ohne vorhandenen Verbindungen komplett abbrechen zu lassen. Diese sind verbunden mit einem Overhead sowie mit einem kurzen Verbindungsabbruch im Millisekundenbereich.¹⁶

Eine einzelne Zelle in einem solchen Cluster wird als *Makro-Zelle* bezeichnet und deckt einen Bereich mit einem Durchmesser im zweistelligen Kilometerbereich ab. Eins der größten Probleme einer solchen großen Zelle ist die Benutzeranzahl. Die Aufteilung der Frequenzen reicht nicht aus, um eine hohe Anzahl von Benutzern, wie sie z. B. in Ballungsgebieten vorkommt, zu bedienen. Um die Anzahl der Benutzer, die von einer solchen Zelle unterstützt werden, zu erhöhen, kann eine weitere Unterteilung der Makro-Zellen in *Mikro-Zellen* (Bereiche mit einem Durchmesser unter einem Kilometer) und weiter von Mikro-Zellen in *Piko-Zellen* (Bereiche mit einem Durchmesser von mehreren Metern) durchgeführt werden. Innerhalb dieser kleineren Einheiten werden die in der größeren Zelle verfügbaren Frequenzen feiner aufgeteilt und erhöhen somit die Anzahl der möglichen Verbindungen, wobei sich Fehlerraten erhöhen und die Datenraten sinken.¹⁷

Basierend auf dieser Zellenarchitektur gibt es WWAN-Technologien, die in verschiedene Generationen aufgeteilt sind. Tabelle 2 stellt die verschiedenen Generationen mit einigen darin entstandenen Technologien dar.

Generation	Beschreibung	Technologiebeispiele
1. Generation	Die 1. Generation der Funktechnologien dienen der reinen, analogen Übertragung von Gesprächen über eine zellulare Netzwerkarchitektur und unterstützen von sich aus keine Datenübertragung. Technologien der 1. Generationen sind inzwischen selten im Einsatz.	C 450 (Auch bekannt als C-Netz), Advanced Mobile Phone Service (AMPS)
2. Generation	Um digitale Anwendungen besser zu unterstützen, sind die Technologien der 2. Generation entstanden. Diese bieten, im Gegensatz zur 1. Generation, digitale Schnittstellen an. Die 2. Generation bietet weitere Dienste wie z. B. dem Short Message Service (SMS) an.	Global System For Mobile Communications (GSM) , IS-05 CDMA (Code Division Multiple Access) , IS-95
2.5 Generation	Die Generation 2.5 stellt eine Erweiterung und Verbesserung der 2. Generation dar. Diese Stufe ist entstanden, da der Aufbau von einer flächendeckenden Versorgung von Technologien der 3. Generation bis heute andauert. Unterschiede zur 2. Generation stellen die	GPRS (General Packet Radio Services) , basiert auf GSM), CDMA-2000 Phase 1

¹⁶ Vergl. Murthy, C. Siva Ram et al. (2004), S. 110 ff sowie Saranqapani, Jagannathan (2007), S. 17f

¹⁷ Vergl. Murthy, C. Siva Ram et al. (2004), S. 110 ff

Generation	Beschreibung	Technologiebeispiele
	verwendeten Signale und Frequenzbereiche dar, sodass zum Teil wesentlich höhere Datenraten ermöglicht werden.	
3. Generation	Die 3. Generation stellen Technologien dar, die vollständig auf Datenübertragung ausgerichtet sind. Sie erreichen im Vergleich zu Technologien der 2. Generation wesentlich höhere Datenraten, liegen aber noch deutlich unter den Datenraten der WLANs.	Universal Mobile Telecommunications Service (UMTS, basiert auf GSM), CDMA-2000
3. Generation Enhanced	Die Technologien der 3. Generation Enhanced stellen größtenteils WLAN-Standards dar, die unter Verringerung der Datenraten eine wesentlich höhere Abdeckung erreichen. Abgegrenzt vom WLAN-Standard gibt es die Worldwide Interoperability Microwave Access-Technologie (WiMAX), welche eine von den vorhergehenden Generationen unabhängige Entwicklung auf Mikrowellen-Basis darstellt. Eine zellulare Struktur wird von WiMAX unterstützt, jedoch nicht von den WLAN-Standards. Allerdings können Technologien, die den Standard 802.11 a sowie g realisieren, eine ähnliche Abdeckung wie die einzelnen Makro-Zellen von WWANs unter Reduzierung der Datenrate erreichen.	ISO 802.11 a und g, Point-To-Point, WiMAX

Tabelle 2: Darstellung der verschiedenen WWAN-Generationen¹⁸

Je nach Land können unterschiedliche Technologien zum Einsatz kommen. Europa z. B. setzt größtenteils auf GSM und deren Nachfolger, während die USA CDMA und deren Nachfolger einsetzt. Das Problem dabei stellen mobile Geräte dar, die evtl. nicht alle Technologien unterstützen und nicht überall eine Kommunikation ermöglichen.¹⁹

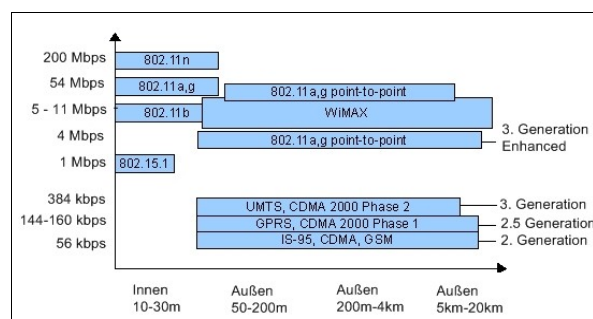


Abbildung 2: Charakteristika der Funktechnologien der einzelnen Generationen²⁰

Abbildung 2 zeigt einen Überblick über die Eigenschaften der einzelnen Generationen und stellt diese im Vergleich zu den verschiedenen WLAN-Standards dar. Die inzwischen praktisch bedeutungslose 1. Generation wird innerhalb der Abbildung nicht gezeigt. Die Werte der Datenraten sind in der Graphik in **bits per second (bps)** angegeben. Der Abbildung entnimmt man, dass WLAN-Technologien bei einer

¹⁸ Vergl. Walke, Bernhard et al. (2001), S. 5; Kurose, James F. et al. (2008), S. 517f, S. 343ff und S. 551f sowie Green, James Harry (2005), S. 359f

¹⁹ Vergl. Kurose, James F. et al. (2008), S. 551ff

²⁰ Vergl. Kurose, James F. et al. (2008), S. 517 und Stein, Erich (2008), S. 343

geringeren Datenrate eine höhere Abdeckung erreichen. Innerhalb der Graphik sind die Maximalwerte der Technologien angegeben. Durch die innerhalb der Cluster verwendeten Frequenzteilungen und den Störungen in den Technologien werden diese allerdings selten erreicht. Dies zeigt sich z. B. bei GPRS. Während die Maximalgeschwindigkeit bei 160 kbps liegt, stellen 115 kbps die durchschnittliche Geschwindigkeit bei GPRS dar.²¹ Diese kann bei einer hohen Benutzeranzahl in einer Zelle allerdings stark unter diesen Durchschnitt sinken.

2.2. *Konsequenzen der Beschränkungen mobiler Netzwerktechnologien für mobile Clients*

Die Begrenzungen durch die Probleme und niedrigen Übertragungsraten der Funktechnologien stellen eine Herausforderung an mobile Systeme dar, welche auf einer Verbindung zu einem zentralen System basieren. Eine ständige und leistungsfähige Verbindung zu einem zentralen System, wie sie bei stationären Systemen (beispielsweise einem PC in einem Netzwerk) gegeben ist, besteht bei mobilen Geräten aufgrund der vorhergehend dargestellten Probleme nicht immer. Zum einen decken die WLANs und WWANs nicht alle Flächen ab. Weiterhin brechen innerhalb von WLANs die Verbindungen ab, sobald deren Sendebereich verlassen wird. Zum anderen sorgen die verschiedenen Probleme auf physikalischer Ebene zu schwachen Verbindungen bzw. zu Verbindungen, die aufgrund von Störungen abbrechen können. Dies hat bei der Entwicklung mobiler Systeme zu einer Kategorisierung in zwei verschiedene Szenarien geführt. Die erste Kategorie sind Online-Szenarien. Dort wird davon ausgegangen, dass das mobile Gerät eine ständige Verbindung besitzt. Dies kann z. B. beim Einsatz in Lagerhäusern zur Erfassung von Waren der Fall sein, wo ein WLAN für die mobilen Geräte bereitgestellt wird. Dabei muss auf eine ausreichende Überdeckung des Bereichs sowie einer ausreichenden Stärke und Qualität der Signale geachtet werden. Störquellen wie Maschinen beim Einsatz in Produktionszentren o. ä. stellen weitere zu beachtende Punkte beim Aufbau eines solchen Netzwerkes dar. Die mobilen Applikationen, die in einem solchen Szenario eingesetzt werden, brauchen lediglich eine Benutzersteuerung zu realisieren. Die Verarbeitung und Speicherung der Daten kann in einem zentralen System geschehen, welches über das WLAN mit der Oberfläche auf dem mobilen Gerät verbunden ist. Solche Clients werden als Thin-Clients bezeichnet. Eine komplette Ortsunabhängigkeit wird aber in diesen Szenarien

²¹ Vergl. Stein, Erich (2008), S. 343

nicht erreicht. Dem gegenüber steht das Offline-Szenario, in dem immer eine Unterbrechung der Verbindung geschehen kann. Ein Beispiel für dieses Szenario wäre ein mobiler Client, der Mitarbeiter im Vertrieb bei Kundenbesuchen an beliebigen Orten unterstützt, um direkt vor Ort Verkäufe durchzuführen (Dieses Beispiel stellt die Grundlage für die in dieser Arbeit durchgeführten Tests). In dem Fall muss auf die WWANs zurückgegriffen werden, da bei diesem Szenario keine Verbindung zu WLANs garantiert werden kann.

Durch Störungen wie z. B. Gebäude mit einer schlechten Durchlässigkeit von Funkwellen oder laufende Maschinen (Falls auch Geschäftskunden mit entsprechenden Anlagen zum Kundenstamm gehören) kann allerdings die Verbindung zum WWAN abbrechen. Weiterhin decken auch WWANs nicht alle Bereiche ab. Somit müssen die Applikationen auf dem Gerät eine Datenhaltung und -verarbeitung für eine Offline-Verwendung realisieren. Solche Clients werden als Thick-Clients bezeichnet. Da die Daten in vielen Szenarien später zentral verfügbar gemacht werden müssen und die leistungsschwachen Geräte nicht immer eine komplette Verarbeitung übernehmen können, müssen Strategien für eine Synchronisation mit dem zentralen System und einer Verteilung von Daten realisiert werden.²²

Daraus resultiert ein erhöhter Entwicklungsaufwand für mobile Applikationen und höhere Kosten der einzelnen Applikationen. Mit der zukünftigen Weiterentwicklung der mobilen Funknetzwerke ebenso wie der zunehmenden Leistungsfähigkeit von mobilen Geräten entschärfen sich die Probleme der mobilen Geräte. Die Unsicherheit der Verbindungen bleibt aber bestehen. Den Möglichkeiten der Applikationen, die durch mobile Datenbanken eine gewisse Unabhängigkeit erlangt haben, stehen ebenfalls Probleme gegenüber. Eingeschränkt werden die mobilen Applikationen durch eine geringe Leistungsfähigkeit der Geräte sowie die Komplexität, die durch die Synchronisation und Datenverteilung entstehen. Neben der geringen Leistungsfähigkeit der mobilen Geräte kommen auch zusätzliche Fragestellungen auf, wenn ein mobiler Thick-Client entwickelt wird. Eine Frage ist z. B., wie bei mobilen Thick-Clients auf die Heterogenität der Geräte reagiert wird. Einige mobile Geräte und mobile Betriebssysteme unterstützen z. B. die Laufzeitumgebung von Java und .Net, bei anderen werden eigene Modelle angeboten.²³

²² Vergl. Hanhart, Daniel (2008), S. 12f und Coulouris, George et al. (2005), S. 40 f

²³ Vergl. B'Far, Reza et al. (2005), S. 33f und Höpfner, Hagen et al. (2005), S. 2f

3. Darstellung mobiler Datenhaltung

Das vorhergehende Kapitel hat die Notwendigkeit einer Datenhaltung auf mobilen Geräten aufgezeigt. Um diese Datenhaltung zu ermöglichen, gibt es heutzutage Datenbanksysteme, die auf die beschränkten Möglichkeiten mobiler Geräte wie Handhelds (PDAs und Mobiltelefone) und kleinere Geräte angepasst sind.²⁴

Mobile Datenbanksysteme stellen eine wesentliche Grundlage zur Implementation von Thick-Clients dar. Sie ermöglichen eine zentrale Verwaltung von Datenbeständen auf mobilen Geräten und besitzen im Idealfall bereits Schnittstellen zu zentralen Systemen. Allerdings stellt die Verwaltung von Datenbeständen in mobilen Umgebungen auch eine Stelle dar, an der Performance-Probleme auftreten können. Dieses Kapitel zeigt, welche Anforderungen an mobile Datenbanksysteme gestellt werden und wie solche Systeme aufgebaut werden können. Weiterhin stellt es das Datenbanksystem vor, mit dem die mobile Datenhaltung getestet werden soll.

In den folgenden Kapiteln wird dabei zwischen *Datenbanksystem* (DBS), *Datenbankmanagementsystem* (DBMS) und *Datenbank* (DB) unterschieden. Eine Datenbank stellt einen strukturierten Datenbestand dar, während ein DBMS eine Software für den Zugriff auf diese Daten darstellt. Das DBMS ermöglicht das Ermitteln sowie Manipulieren von Daten sowie das Erstellen und Manipulieren der Datenbanken selbst. Dazu gehört die applikationsneutrale und effiziente Realisierung dieser Zugriffsmöglichkeiten. Ein DBS ist die Summe aus einem DBMS und dessen Datenbanken.²⁵

3.1. Anforderungen an Datenbanksysteme

Die zentrale Aufgabe von Datenbanksystemen besteht in der strukturierten, konsistenten und effizienten Datenhaltung. Weiterhin müssen sie Schnittstellen für Operationen auf dem Datenbestand zur Verfügung stellen. Für die Operationen müssen Transaktionen mit den ACID-Eigenschaften bereitgestellt werden. Transaktionen stellen eine Zusammenfassung von Operationen dar, welche als Ganzes durchgeführt werden sollen. Eine Operation stellt dabei eine atomare Aktion auf einer Datenbank dar, wie zum Beispiel eine Anfrage zur Ermittlung bestimmter Datensätze (*Abfrage*).²⁶

Die Abkürzung ACID steht hier für Atomarität (*Atomicity*), Konsistenz (*Consistency*), Isoliertheit (*Isolation*) und Dauerhaftigkeit (*Durability*). Atomarität bedeutet die

²⁴ Vergl. Mutschler, Bela et al. (2004), S. 178f. und Höpfner, Hagen et al. (2005), S. 11

²⁵ Vergl. Schneider, Markus (2003), S. 3ff

²⁶ Vergl. Saake, Gunter et al. (2008), S. 8ff

Unteilbarkeit der einzelnen Schritte sowie eine vollständige Abhängigkeit der Teilschritte voneinander. In Fall einer fehlgeschlagenen Teiloperation müssen alle vorhergehenden Schritte wieder rückgängig gemacht werden. Dadurch wird ein fehlerhafter Stand der Daten verhindert. Konsistenz bedeutet die Einhaltung der Beziehungen der Daten untereinander. Diese Abhängigkeit muss geprüft werden und die Änderungen verworfen werden, falls diese Prüfung ein negatives Ergebnis aufweist. Somit garantieren Transaktionen mit den ACID-Eigenschaften das korrekte und vollständige Einfügen von Daten in die DB. Die Isolierung schreibt die Unabhängigkeit verschiedener Transaktionen zueinander vor. Zwei parallel durchgeführte Transaktionen dürfen nicht auf den Teiländerungen der jeweils anderen Transaktion arbeiten, da dies zu einem fehlerhaften Datenbestand führen kann. Dies bedeutet, die verschiedenen Transaktionen müssen entsprechend synchronisiert werden. Die letzte Bedingung ist die dauerhafte Speicherung der Daten, sobald die Transaktion durchgeführt wurde (Dauerhaftigkeit).²⁷

Transaktionen mit ACID-Eigenschaften stellen ein wichtiges Mittel dar, um einen konsistenten Aufbau der Daten im laufenden Betrieb zu garantieren. Sie stellen die fehlerfreie und komplette Speicherung voneinander abhängiger Daten sicher. Das Problem bei mobilen Systemen ist der Aufwand zur Erfüllung der Transaktionsregeln, welche die Laufzeit von Anfragen an das System erhöht.²⁸

Eine weitere Anforderung besteht aus der Realisierung einer Zugriffskontrolle. Diese verwaltet die Benutzerrechte für Benutzer und Applikationen, die auf die Daten zugreifen wollen. Dadurch wird ein unberechtigter Zugriff auf Daten verhindert.²⁹

Um auf die Möglichkeit eines Systemabsturzes oder eines Fehlers des Speichermediums zu reagieren, müssen DBS eine Sicherungsschicht realisieren, die die Entstehung von Inkonsistenzen verhindert.³⁰ Da nach einem Ausfall des Gerätes eine Verhinderung von Inkonsistenzen evtl. nicht möglich ist, müssen entsprechende Möglichkeiten geboten werden, um nach einem erneuten Start des Systems die Konsistenz wieder herzustellen. Neben den vorhergehenden Anforderungen werden an Datenbanksystemen der Anspruch gestellt, dass diese Backup-Funktionen anbieten, mit denen in regelmäßigen Abständen der Datenbestand gesichert werden kann.

27 Vergl. Saake, Gunter et al. (2008), S. 388f und Höpfner, Hagen, et al. (2005), S. 553f

28 Vergl. Saake, Gunter et al. (2008), S.385 und Mutschler, Bela et al. (2004), S. 115f.

29 Vergl. Saake, Gunter et al. (2008), S. 7f

30 Vergl. Türker, Can et al.(2006), S. 4f

3.2. Anforderungen an mobile Datenbanksysteme

An die mobilen Datenbanksysteme werden dieselben grundlegenden Anforderungen gestellt, die auch an normale Datenbanksysteme gestellt werden. Mobile Systeme stehen dabei weiterhin vor Herausforderungen, die in normalen Datenbanksystemen nicht oder nicht in einer vergleichbaren Stärke auftreten. Letztere agieren im Normalfall in einer Umgebung, die große Datenmengen aufnehmen kann, komplexe Anfragen nach Daten und die Optimierung dieser Anfragen durch hohe Rechenkraft ermöglichen sowie weiterhin auf eine Breitbandverbindung zurückgreifen kann.

Mobile Umgebungen stellen solche Möglichkeiten nicht zur Verfügung. Aufgrund geringer Leistungsfähigkeit und geringer Speichergrößen mobiler Geräte sind starke Optimierungen von Anfragen und die Verwaltung großer Datenmengen nicht möglich. Trotz dessen müssen mobile DBS eine strukturierte und fehlerfreie Speicherung der Daten ermöglichen.

Backup-Funktionen, welche in normalen DBS eine wichtige Anforderung darstellen, spielen bei mobilen Systemen, speziell wenn die Daten auch in einem zentralen Informationspool (z. B. ein zentraler Datenbankserver) gespeichert werden, eine untergeordnete Rolle. Dies liegt u. a. an der geringen Speichergröße der mobilen Geräte, welche u. U. nicht genügend Platz für Backups bietet.

Falls das mobile DBS Anbindungen an zentrale Systeme von sich aus unterstützt, kommen weitere Faktoren hinzu, die beachtet werden müssen. Dazu gehört z. B. die Unterstützung heterogenen Umgebungen (unterschiedliche Betriebssysteme, unterschiedliche unterstützte Programmiersprachen etc.). Die Bereitstellung solcher Anbindungen kann als Anforderung an ein mobiles DBS angesehen werden. Dies begründet sich darin, dass in vielen Szenarien die Notwendigkeit besteht, Daten zu synchronisieren, um einerseits die Daten auf dem Client aktuell zu halten und andererseits die im Client ermittelten Daten an eine zentrale Stelle weiter zu leiten. Kapitel 4 beschäftigt sich eingehender mit der Möglichkeit, mobile DBS mit zentralen Systemen zu verbinden.

3.3. Grundlegender Aufbau von Datenbanksystemen

Datenbanksysteme benötigen einen Aufbau, der den dargestellten Anforderungen genügt. Um die Daten sowie die Datenbanken selbst zu strukturieren und die Daten von den Speicherstrukturen von Computersystemen zu abstrahieren, werden Datenmodelle

verwendet.³¹ Das am weitesten verbreitetste Modell stellt das relationale Modell von Edgar Codd dar.³² Neben diesem gibt es weitere, in der Praxis größtenteils in Spezialfällen angewandte Modelle wie z. B. das objektrelationale Modell oder das objektorientierte Modell.³³

Im relationalen Modell von Edgar Codd werden die Daten in verschiedene Relationen eingeteilt. Der Grundgedanke entstammt den mathematischen Relationen, daher auch der Name des Modells. Aus technischer Sicht wird dabei von Tabellen gesprochen. Die Attribute (oder Spalten) einer Relation tragen zur Identifizierung einen in der Relation eindeutigen Namen und enthalten Werte eines atomaren Datentyps. DBS stellen solche Datentypen zur Verfügung.³⁴

Um Selektionen und Beziehungen der Daten zu ermöglichen, ist im relationalen Modell die Schlüsselbedingung eingeführt worden. Dieser Bedingung zufolge müssen Zeilen einer Relation über die Werte einzelner Attribute eindeutig identifizierbar sein. Welche Attribute dies sind, hängt von der Definition der Relation ab. Diese Bedingung wird als Primärschlüsselbedingung bezeichnet.³⁵

Als Beispiel für Relationen kann man, basierend auf dem Beispiel der Vertriebsmitarbeiter aus Kapitel 2.2, Aufträge und Artikel nehmen. Die Aufträge werden von den Vertriebsmitarbeitern erstellt und enthalten verschiedene verkaufte Artikel. In diesem Beispiel ergeben sich die Relationen Auftrag und Artikel. An dieser Stelle fehlt aber bisher eine Möglichkeit, die Beziehung dieser beiden Relationen zueinander abzubilden. Um solche Beziehungen darzustellen, führt das relationale Modell die Fremdschlüsselbedingung ein. Ein mit dieser Bedingung definiertes Attribut einer Relation darf nur Werte enthalten, die in der Menge der Werte eines Attributs aus einer anderen Relation auftreten. Über diesen Weg können in einer Relation Attribute definiert werden, die nur Werte aus den Schlüsselattributen einer zweiten Relation enthalten darf. Durch diese Einschränkung wird ermöglicht, dass mehrere Zeilen der ersten Relation einer Zeile der zweiten Relation über einen Vergleich zugeordnet werden können.³⁶

31 Vergl. Vossen, Gottfried (2008), S. 26

32 Vergl. Connolly, Thomas et al. (2002), S. 69

33 Vergl. Matthiessen, Günter et al. (2003), S. 311 und Rob, Peter et al. (2008), S. 51. Weiterführende Literatur für objektrelationale Datenbanken stellt Türker, Can et al. (2006) dar. Meier, Andreas (2003) behandelt neben objektrelationalen Datenbanken auch objektorientierte Datenbanken.

34 Vergl. Connolly, Thomas et al. (2002), S. 69; Saake, Gunter et al. (2008), S. 11 und Lockemann, Peter C. et al. (2004), S. 79

35 Vergl. Lockemann, Peter C. et al. (2004), S. 79ff

36 Vergl. Lockemann, Peter C. et al. (2004), S. 79ff

Somit kann der Datenbestand in einer DB über den Aufbau von Relationen und deren Beziehung zueinander strukturiert werden. Der Aufbau der Relationen und der Beziehungen wird auch das *Schema* der DB genannt. Um eine möglichst geringe Redundanz von Daten und ein Minimum an Datenanomalien zu erreichen, gibt es Normalformen, nach denen ein Schema aufgebaut werden kann.³⁷ Über diesen Weg ist die Anforderung nach der Strukturierung der Daten erfüllt.

Um zum einen die Definition dieser Schemata und zum anderen einen Umgang mit dem Datenbestand (z. B. Selektieren und Manipulieren der Datenbestände) zu ermöglichen, wurde die deskriptive Sprache SQL (Structured Query Language) entwickelt. Diese Sprache ist standardisiert und gilt als Standardsprache, um DBS zu programmieren.³⁸ Deskriptive Sprachen geben lediglich das Ergebnis einer Anfrage an, nicht jedoch die eigentliche Berechnungsvorschrift wie z. B. eine Funktion. Eine Umsetzung der deskriptiven Anfragen in entsprechende Ausführungsschritte wird durch die Erstellung eines sogenannten Ausführungsplans realisiert. Dieser enthält die Berechnungen, die für eine Anfrage notwendig sind.³⁹

Eine Möglichkeit, um stark ressourcenverbrauchende Anfragen an mobile DBS zu vermeiden, besteht in der nicht vollständigen Umsetzung des SQL-Standards. Komplexere Operatoren und Schlüsselwörter kann man somit von vornherein ausschließen. Einerseits schränkt man damit die Möglichkeiten des DBS ein, andererseits werden somit komplexere und das Gerät stark belastende Operationen bis zu einem gewissen Grad verhindert.

3.4. Aufbau von mobilen Datenbanksystemen

Die im vorhergehenden Kapitel vorgestellten Konzepte werden auch von mobilen Datenbanksystemen implementiert. Dieses Kapitel beschreibt, wie ein mobiles DBS aufgebaut werden kann.

Die allgemeinen Architekturen in Abbildung 3 stellen zwei Modelle für eine strukturierte Datenhaltung auf mobilen Geräten dar. Ein zentrales System dient dabei als gemeinsamer Informationspool. Die Architektur auf der linken Seite stellt einen Ansatz für mächtigere Geräte wie PDAs dar, während die rechte Seite einen Ansatz für kleine Geräte wie einfache Mobiltelefone oder Barcodescanner darstellt. Die Architekturen

37 Vergl. Saake, Günther et al. (2008), S. 159 und Elmasri, Ramez et al. (2007), S. 355. Für eine Einführung in Normalformen, vergl. Vossen, Günther (2008), Kapitel 8 (S. 251ff).

38 Vergl. Steiner, René (2003), S. 155

39 Vergl. Connolly, Thomas et al. (2002), S. 110f; Elmasri, Ramez et al. (2007), S. 233ff, 325ff und 346f sowie Burnus, Heinz (2008), S. 84

verwenden, wie auch die OSI-Architektur der Netzwerke, eine Aufteilung in Schichten. Unterschiede dieser beiden Architekturen bestehen im Umgang mit den Daten sowie in der Optimierung von Anfragen. Innerhalb des Modells für die einfachen Endgeräte wird auf eine Optimierung der Anfragen verzichtet. Weiterhin wird in der Architektur für mächtigere Geräte eine Verwaltung der Daten im transienten Speicher des Gerätes mithilfe von Seiten realisiert (Schicht fünf in der linken Architektur).⁴⁰

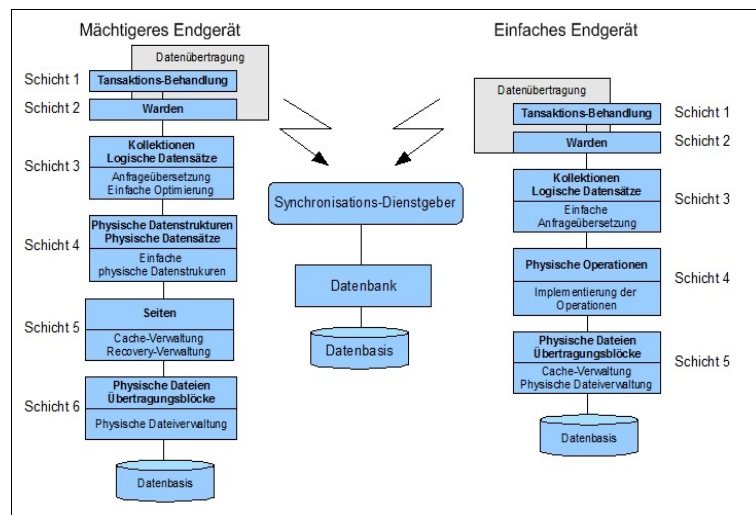


Abbildung 3: Allgemeine Architekturen mobiler Datenbanksysteme⁴¹

Die Verwendung einer Seitenverwaltung entstammt den Techniken von Betriebssystemen zur Speicherverwaltung. Diese Seiten bilden dort die Grundlage des virtuellen Speichers, welcher eine Trennung der Applikationen vom realen Speicher ermöglicht.⁴²

Nach der Referenzarchitektur von Härder und Reuter⁴³ von DBS verwenden diese ebenfalls die Einteilung der Daten in Seiten. Dies bedeutet, der Datenfluss aus dem persistenten Datenspeicher wird zu Blöcken zusammengefasst (den Seiten) und innerhalb solcher Einheiten im transienten Speicher gehalten. Die Beschaffung der Daten aus dem persistenten Speicher wird so von der Verwaltung des transienten Speichers gekapselt. Die physikalischen Datenstrukturen müssen somit lediglich mit dem Aufbau des Speichers umgehen können und nicht mit heterogenen Geräten zur persistenten Speicherung von Daten. Weiterhin wurden auf Grundlage von Seiten entsprechende Pufferstrategien entwickelt, die hier zum Einsatz kommen können. In kleinen mobilen Geräten kommen aber die Nachteile dieses Vorgehens zum Tragen. Die

40 Vergl. Lockemann, Peter C. et al. (2004), S. 51ff und S. 346ff

41 Vergl. Lockemann, Peter C. et al. (2004), S. 346

42 Vergl. Tanenbaum, Andrew S. (2003), S. 209ff

43 Vergl. Lockemann, Peter C. et al. (2004), S. 51ff

Aufteilung in Seiten sowie die entsprechenden Pufferstrategien sorgen für einen erhöhten Verwaltungsaufwand, Speicherbedarf und Rechenkraft, der in diesen Geräten nicht vorhanden ist. Ebenfalls stellen die Betriebssysteme für diese Geräte evtl. keine Aufteilung in Seiten von sich aus zur Verfügung, womit das DBS dies selber implementieren müsste. Somit fällt in kleinen Geräten dieser Teil weg und Schicht vier arbeitet direkt mit dem persistenten Datenspeicher.⁴⁴

Mithilfe der vierten Schicht in beiden Architekturen wird eine Kapselung der realen Speicherstrukturen mithilfe von physikalische Datenstrukturen ermöglicht. Diese Strukturen stellen die einzelnen Operationen zur Änderung und Ermittlung der Daten zur Verfügung und trennen die logische Sicht, welche in Schicht drei realisiert ist, von der realen Speicherverwaltung. Die logische Sicht realisiert z. B. die Aufteilung in Datensätze, Relationen, etc. dar, die im relationalen Modell verwendet werden. Die dritte Schicht verwendet somit lediglich die Schnittstellen der vierten Schicht, um auf die Daten zuzugreifen. Innerhalb dieser dritten Schicht werden die Anfragen an das DBS verarbeitet und ggf. optimiert. Danach werden die Anfragen mithilfe der vierten Schicht durchgeführt.⁴⁵

Schicht zwei stellt eine Möglichkeit dar, die Anbindung an ein Serversystem zu ermöglichen. Hier übernimmt ein so genannter Warden die Position eines Stellvertreters, um zum einen Synchronisationen von sich aus auszuführen und zum anderen die Anwendungen, je nach Verbindungsstatus, auf das mobile DBS oder den Server zu lenken. Ein Dienstgeber auf Serverseite stellt die Schnittstellen für die Clients zur Verfügung.⁴⁶

Der Warden sorgt also für Transparenz, ob ein lokales oder entferntes DBS verwendet wird und verbirgt die Durchführung von Synchronisationen. Einen anderen Ansatz stellen Architekturen dar, die eine Anbindung an ein zentrales System getrennt von der normalen Abfragebearbeitung realisieren. Applikationen verbinden sich bei diesem Vorgehen immer mit dem mobilen DBS und davon getrennt wird der Abgleich mit dem zentralen Informationspool durchgeführt. In diesem Fall kann auf die Schicht mit dem Warden verzichtet werden. Je nach verwendetem DBS kann die Kommunikation mit dem Server sehr unterschiedlich realisiert worden sein – falls das mobile DBS eine solche Anbindung überhaupt bereitstellt.⁴⁷

44 Vergl. Tanenbaum, Andrew S. (2003), S. 209ff und Lockemann, Peter C. et al. (2004), S. 93f und S. 51ff

45 Vergl. Lockemann, Peter C. et al. (2004), 51ff und Mutschler, Bela et al. (2004), S. 235ff

46 Vergl. Lockemann, Peter C. et al. (2004), S. 347

47 Vergl. Lockemann, Peter C. et al. (2004), S. 347f

Transaktionen und Abfragen stellen in dieser Schicht ein Problem dar, da ein oder mehrere Partner auf anderen Computersystemen betroffen sind. Daher gibt es mobile Transaktionen (welche auf die Schwächen der Funktechnologien eingehen) und, bei mehreren Partnern, verteilte Transaktionen und verteilte Abfragen.⁴⁸

Innerhalb der ersten Schicht wird die Transaktionsverwaltung realisiert. Bei einer Anbindung an ein zentrales System sind lokale Transaktionen auf dem mobilen Gerät nur vorläufig. Bei einem Abgleich mit dem zentralen System können Konflikte auftreten, welche entsprechend gelöst werden müssen.

Die vorhandenen mobilen DBS auf dem Markt unterscheiden sich mehr oder minder stark von den Architekturen aus Abbildung 3. Je nach Geräteart, die verwendet wird, ist weiterhin fraglich, ob sich ein klares Schichtenmodell einhalten lässt. Ein DBS, welches der Architektur für mächtige Endgeräte am nächsten kommt, stellt das DBS „DB2 Everyplace“ von IBM dar. Dort wird allerdings auf einen Warden verzichtet. Weitere verbreitete mobile DBS stellen das „OracleLite“ von Oracle und der „Microsoft SQL Server CE“ dar. Das OracleLite-System verwendet eine Art Warden und verbirgt somit vor dem Benutzer, ob eine zentrale oder lokale DB verwendet wird.⁴⁹

Neben den beiden vorgestellten Architekturen gibt es weiterführende Modelle, die eine Datenspeicherung auf extrem kleinen Systemen wie z. B. Smartcards ermöglichen. Smartcards gehören zu den kleinsten, zur Zeit verfügbaren, mobilen Computersystemen. Solche DBS werden als *Pico-Datenbanksysteme* bezeichnet.⁵⁰

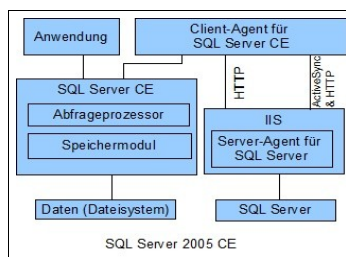


Abbildung 4: Architektur des SQL Server CE⁵¹

Innerhalb dieser Arbeit wird das mobile DBS „SQL Server 2005 CE“ in der Version 3.5 untersucht. Dieses System ist ein gängiges DBS auf Geräten, die das Betriebssystem „Windows Mobile“ von Microsoft verwenden. Die genaue interne Architektur wurde nicht von Microsoft veröffentlicht. Abbildung 4 zeigt die grobe Architektur des

48 Vergl. Mutschler, Bela et al. (2004), S. 115 und Connolly, Thomas et al. (2002), S. 688f

49 Vergl. Lockemann, Peter C. et al. (2004), S. 347ff sowie Mutschler, Bela et al. (2004), S. 199 und S. 223

50 Vergl. Mutschler, Bela et al. (2004), S. 173 und 178f. Weiterführende Literatur für Pico-Datenbanken stellt Mutschler, Bela et al. (2004), Kapitel 8 (S. 173ff) dar.

51 Vergl. Microsoft (2009 b); Microsoft (2009 c) sowie Mutschler, Bela et al. (2004), S. 257f

mobilen Systems mit der Verwendung eines zentralen SQL Servers als Informationspool für die Clients.⁵²

Dieses DBS ist für mächtigere Clients gedacht und bringt ein eigenes Speichermodul sowie einen Abfrageprozessor mit, der die Clientanfragen verarbeitet. In dieser Struktur ist kein in den Server integrierter Warden wie in der allgemeinen Architektur vorgesehen. Stattdessen sollen RDA-Zugriffe oder Replikationsmechanismen verwendet werden, um einen Abgleich mit einem zentralen System, getrennt von den Anfragen an das mobile DBS, zu ermöglichen.⁵³ Für die Replikationsmechanismen wird ein Agent-System verwendet. Replikationen und Agent-Systeme werden in Kapitel 4 näher erläutert. RDA steht für **Remote Database Access** und ist ein DBS-unabhängiges Protokoll, um entfernte Datenzugriffe auf DB zu ermöglichen.⁵⁴ Die Replikationsmechanismen des SQL Server 2005 haben an dieser Stelle gegenüber den RDA-Zugriffen den Vorteil, dass bereits Algorithmen zur Synchronisation implementiert worden sind und ein Kommunikationsmodell mit Agents zur Verfügung steht, welches für mobile Umgebungen angepasst wurde. Um möglichst viele heterogene Geräte zu unterstützen, läuft dieses Kommunikationsprotokoll auf Basis des **Hyper Text Transport Protocol (HTTP)**. Da die Architektur des SQL Server CE für mächtigere Endgeräte gedacht ist, sollen im weiteren Verlauf dieser Arbeit speziell die Handhelds betrachtet werden.

Der SQL Server CE implementiert das relationale Modell und verwendet SQL, um Abfragen, Schemadefinitionen und Datenmanipulationen zu ermöglichen. Der SQL Server CE arbeitet mit einzelnen Datenbankdateien.

3.5. Grenzen mobiler Datenbanksysteme

In wie weit die begrenzte Möglichkeit zur Optimierung sowie die durch kleine Speichergrößen begrenzten Puffermöglichkeiten die Leistungsfähigkeit der mobilen DBS beschränken, soll in dieser Arbeit neben der Leistungsfähigkeit der Synchronisation untersucht werden. Dabei liegt die Vermutung nahe, dass Operationen auf geringen Datenmengen, die sich in einer mobilen Datenbank befinden, performant behandelt werden können. Bei einer wiederholten Erhöhung der Datenmenge ist zu erwarten, dass das DBS auf einen Punkt zusteuert, an dem es entweder nicht mehr reaktionsfähig ist oder zumindest so lange Antwortzeiten aufweist, dass dies die

⁵² Vergl. Mutschler, Bela et al. (2004), S. 256f, S. 199 und S. 223

⁵³ Vergl., Lockemann, Peter C. et al. (2004), S. 350f

⁵⁴ Vergl. Häckelmann, Heiko et al. (2000), S. 423ff

Verwendung in Applikationen stark einschränken würde bzw. unmöglich machen würde.

Die Antwortzeit hängt dabei neben der Datenmenge auch stark von der Komplexität der Abfragen ab. Speziell die Verknüpfungen von Tabellen, Sortierungen und Unterabfragen (Abfragen innerhalb einer Abfrage) stellen leistungsfordernde Operationen dar. Filteroperationen, die eine zentrale Rolle bei der Verwendung eines Datenbestands spielen, sind ebenfalls ressourcenverbrauchend.⁵⁵

Dementsprechend müssen bei der Untersuchung der Grenzen der Belastbarkeit eines mobilen DBS entsprechende Abfragen formuliert werden, die auf einer zunehmenden Datenmenge ausgeführt werden. Dafür können zunächst einfache Abfragen formuliert werden, die auf einer oder zwei Relationen basieren. Daneben können komplexe Abfragen mit Verknüpfungen mehrerer Relationen sowie Sortierungen und Unterabfragen verwendet werden, um einen Vergleich zwischen einfachen und komplexeren Operationen zu ermöglichen.⁵⁶

Neben der reinen Datenselektion sind weiterhin Schreib-, Aktualisierungs- und Löschvorgänge (*Datenmanipulationen*) abhängig von der sich in einer mobilen DB befindlichen Datenmenge und werden somit in eine Betrachtung des Verhaltens des mobilen DBS unter einer zunehmenden Datenmenge mit einbezogen. Beim Einfügen neuer Daten wird die Einhaltung der Primär- und Fremdschlüsselbedingungen überprüft, die an Zeilen der jeweiligen Relation gestellt werden. Zu diesen Zweck müssen die bereits vorhanden Primärschlüssel durchsucht werden, um auszuschließen, dass der neue Datensatz nicht schon vorhanden ist. Weiterhin müssen die Spalten aus den Relationen durchsucht werden, auf die die Fremdschlüsselbedingungen verweisen. Je nach Anzahl der Zeilen, die vorhanden sind, dauert eine Überprüfung entsprechend lange. Die Überprüfung der Fremdschlüsselbedingungen muss bei Aktualisierungen und Löschungen ebenfalls durchgeführt werden, um die Datenintegrität zu gewährleisten. Weiterhin kommt beim Löschen und Aktualisieren die Suche der betroffenen Zeilen hinzu. Die zentralen zu testenden Funktionen der mobilen DBS stellen somit die Datenselektion sowie die Datenmanipulation dar.

⁵⁵ Vergl. Gulutzan, Peter et al.(2006), S. 12ff, 40f und 87f

⁵⁶ Vergl. auch das Kapitel 5 für die Herangehensweise an Performance-Tests

4. Modellierung von Client-Systemen für mobile Geräte

In diesem Kapitel werden Modelle und Verfahren zum Aufbau eines Gesamtsystems aus mobilen Clients und einem zentralen System dargestellt.

4.1. Anforderungen durch die Netzwerkschnittstellen für mobile Geräte

Bei der Verwendung mobiler Geräte ergeben sich Probleme, die durch die Unsicherheit der Funktechnologien ausgelöst werden. Ein Beispiel stellen Transaktionen zu einem zentralen System dar. Die Verbindungen sind nicht sicher vor einem plötzlichen Verbindungsabbruch und sind geprägt von Datenverlusten. Klassische Techniken zur Umsetzung von ACID-Transaktionen über eine Funktechnologie können somit nur begrenzt eingesetzt werden, da diese mit solchen Problemen nicht umgehen können.⁵⁷

Mobile Thick-Clients in einem Gesamtsystem, die mit den Daten aus diesem System ohne Netzwerkverbindung arbeiten sollen, müssen auf Techniken zurückgreifen, die eine lokale Speicherung von Daten des Gesamtsystems (*Replikationsverfahren*) und den Abgleich von Änderungen, die während einer getrennten Verbindung auftreten, ermöglichen (*Synchronisationsverfahren*).⁵⁸

Neben solchen Verfahren muss die Gesamtarchitektur des Systems auf die Verbindungsunsicherheit der mobilen Geräte angepasst sein. Weiterführend müssen die verschiedenen Netzwerktechnologien und deren unterschiedliche Bandbreite beachtet werden. Je nach Anwendungsfall kommen evtl. verschiedene Geräte mit unterschiedlichen Netzwerktechnologien, verschiedene Betriebssysteme und Programmiersprachen zum Einsatz. Zusätzlich können innerhalb mobiler Umgebungen unterschiedliche Protokolle zur Kommunikation über ein Netzwerk zum Einsatz kommen. Das Gesamtsystem muss mit einer solchen Heterogenität umgehen können.⁵⁹

Für die Übertragung sensibler Daten muss weiterhin eine Sicherung vorgenommen werden, damit lediglich Befugte Zugriff auf diese Daten erhalten. Dabei können unterschiedliche Techniken und Protokolle, wie z. B. das **Hyper Text Markup Protocol Secured (HTTPS)** für eine sichere Verbindung in das Internet, verwendet werden.⁶⁰

⁵⁷ Vergl. Reichwald, Ralf (2002), S. 139 und Höpfner, Hagen et al. (2005), S. 145ff sowie Mutschler, Bela et al. (2004), S. 115

⁵⁸ Vergl. Reichwald, Ralf (2002), S. 139f und Höpfner, Hagen et al. (2005), S. 2f

⁵⁹ Vergl. Gschwind, Thomas et al. (2005), S. 64f

⁶⁰ Vergl. Eckert, Claudia (2007), Kapitel 18 (S. 785ff) für eine Einführung in sichere mobile Kommunikation.

4.2. Einführung in verteilte Systeme

Bei einer Verteilung werden einzelne Teile (*Dienste*) einer Software auf verschiedenen Computersystemen ausgeführt und über Netzwerkschnittstellen vernetzt. Dabei kann innerhalb eines Computersystems auch auf die eigenen Dienste zugegriffen werden. In diesem Sinne stellen die Verwendung mobiler Clients mit eigenem Datenbestand sowie die Anbindung an ein zentrales System (und auch das in dieser Arbeit aufzubauende Testsystem) ein verteiltes System dar. Beim Aufbau eines solchen verteilten Gesamtsystems stellt sich die Frage, welche Rollen die einzelnen Teilnehmer spielen und wie die Kommunikation zwischen den Teilnehmern realisiert wird. Dabei stehen dezentralisierte Modelle (*Peer-To-Peer-Modelle*) den zentralisierten Modellen (*Client-Server-Modelle*) gegenüber (siehe Abbildung 5). Weiterhin existieren kombinierte Modelle aus beiden Ansätzen (*Hybride Modelle*).⁶¹

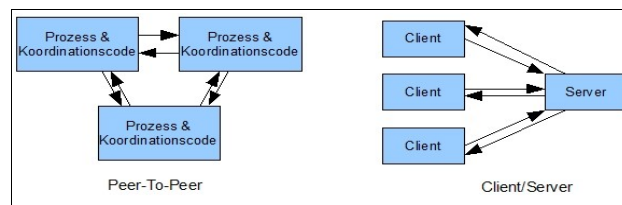


Abbildung 5: Peer-To-Peer-Modell und Client-Server-Modell⁶²

Innerhalb des Peer-To-Peer-Modells werden alle Teilnehmer (*Knoten* oder *Nodes*) gleichberechtigt behandelt. Jeder Knoten kann für die Lösung einer Aufgabe alle anderen Knoten mit einbeziehen. Dies führt jedoch zu einem erhöhten Aufwand und einer höheren Komplexität des Gesamtsystems. Bei der Verwendung mobiler Geräte kommt erschwerend die Verbindungsunsicherheit hinzu, was gegen dieses Modell spricht. Um dem erhöhten Verwaltungsaufwand zu entgehen, können Dienste zentralisiert werden. Die benötigten Dienste laufen bei einem zentralisierten Ansatz auf einem bestimmten Knoten, der als *Server* für die anderen Knoten (die *Clients*) agiert. Somit kommunizieren die Clients nicht mehr untereinander, um bestimmte Dienste aufzurufen, sondern wenden sich an den Server. Abbildung 5 verdeutlicht die zwei verschiedenen Ansätze. Weitere Ansätze stellen hybride Architekturen dar, bei denen z. B. Peer-To-Peer-Netze hinter Servern verborgen werden. Für die Verwendung mobiler Knoten innerhalb eines verteilten Systems bieten sich Client-Server-Modelle oder hybride Modelle an. Bei reinen Peer-To-Peer-Modellen stellen geringe

⁶¹ Vergl. Bengel, Günther (2004), S. 4f und S. 29f sowie Tanenbaum, Andrew S. et al (2008), S. 55f, S. 62f und S. 70f

⁶² Vergl. Bengel, Günther (2004), S. 30 und Coulouris, George et al. (2005), S. 36

Übertragungsraten und vorübergehend nicht verbundene Knoten eine große Beschränkung dar. Diese Arbeit konzentriert sich daher auf Client-Server-Modelle. Auch aus inhaltlichen Gründen kann eine Zentralisierung gewünscht sein. Zum Beispiel kann die Sammlung von Daten an einer Stelle die Weiterverarbeitung und Weiterleitung in andere Systeme stark erleichtern.⁶³

Beim Client-Server-Modell können Variationen auftreten. Das in Abbildung 5 beschriebene Client-Server-Modell stellt eine 2-Schichten-Architektur dar, bestehend aus einer Client- und einer Serverschicht. Um eine Verringerung der Komplexität zu erreichen und den zentralen Knoten zu entlasten, können weitere Schichten eingezeichnet werden. Diese Schichten können Teilaufgaben lösen, die Infrastruktur des Gesamtsystems stärken o. ä. Neben der Verwendung von Schichten kann eine Lastverteilung durch die Verwendung mehrerer Geräte in einer Schicht vorgenommen werden. Dabei wird der Dienst, der in einer Serverschicht bereitgestellt wird, entweder in kleinere Teile auf die Maschinen verteilt oder er wird repliziert, d. h. mehrere Maschinen führen denselben Dienst aus. Im letzteren Fall kann eine zusätzliche Schicht die Aufteilung der Clients auf die einzelnen Server übernehmen (*Balancing-Schicht*).⁶⁴

Eine *Proxy-Schicht* ermöglicht eine Pufferung zwischen Client und Server. Anfragen vom Client werden zuerst an die sogenannten *Proxyserver* gestellt. Diese versuchen, selber die Clientanfrage aus bereits zwischengespeicherten Ergebnissen von vorherigen Anfragen zu beantworten. Falls kein passendes zwischengespeichertes Ergebnis vorhanden ist, fragt der Proxy beim jeweiligen Server an, puffert dessen Antwort und beantwortet die Client-Anfrage. Dieses Verfahren wird speziell im Internet eingesetzt.⁶⁵

Um die Clients vom Server unabhängig zu machen, können *Broker* verwendet werden. Diese haben Kenntnis über den Aufenthaltsort der Server und leiten die Anfragen der Clients an den Server weiter und geben die Antwort an den Client zurück.⁶⁶

Eine Änderung der Serveradresse braucht somit lediglich am Broker bekannt gemacht zu werden. Weiterhin kann der Broker zur dynamischen Dienstermittlung verwendet werden bzw. neben der reinen Vermittlung noch zusätzliche Dienste wie Filterung anbieten. Problematisch sind an dieser Stelle Ausfälle des Brokers oder der Broker, da dann die dahinter liegende Serverschicht nicht mehr erreichbar ist. Dieses Modell ähnelt

63 Vergl. Bengel, Günther (2004), S. 4 und Coulouris, George et al. (2005), S. 35f sowie Tannenbaum, Andrew S. et al. (2008), S. 70ff

64 Vergl. Coulouris, George et al. (2005), S. 31 und S. 37ff sowie Bengel, Günther (2004), S. 67f

65 Vergl. Bengel, Günther (2004), S. 69

66 Vergl. Bengel, Günther (2004), S. 68ff

dem Proxy-Modell. Der Unterschied ist die Zielsetzung. Bei der Verwendung eines Brokers müssen alle, die auf die von den Brokern verwalteten Server zugreifen wollen, die Brokerschicht verwenden. Der Broker vermittelt zwischen den Clients und den Servern. Auf diese Weise können Dienste in einem verteilten System entkoppelt werden und sind weniger stark von den restlichen Teilen des Systems abhängig. Das Ziel des Proxys wiederum besteht darin, eine Pufferung durchzuführen. Er dient als Stellvertreter des Servers, die Clients könnten sich auch direkt an den Server wenden.⁶⁷

Um dynamisch verschiedene Serverdienste für eine Clientanfrage zu kombinieren, kann eine Schicht aus *Agents* eingerichtet werden. Ein Agent sucht aus den ihm bekannten Diensten diejenigen heraus, die für die Clientanfrage notwendig sind und ruft sie auf. Aus den Ergebnissen der einzelnen Aufrufe wird ein Ergebnis für den Client erstellt. Dieses Modell kann erweitert werden durch Agents auf der Client-Seite. Diese suchen innerhalb des verfügbaren Netzwerkes die benötigten Dienste heraus, z. B. Dienste für das Beziehen der Daten von einem oder mehreren Datenbankservern, um einen lokalen Datenbestand aufzubauen. Dieses Modell kommt speziell bei der Verbindung des mobilen DBS von Microsoft mit einem zentralen System zum Einsatz. Der Microsoft SQL Server 2005 CE kommuniziert über clientseitige Agents mit stationären Replikationsservern, um Daten aus den Servern lokal zu speichern (für Replikation, vergl. Kapitel 4.3).⁶⁸ Microsoft stellt weiterhin ein serverseitiges Agentsystem zur Verfügung, mithilfe dessen den Clients eine einheitliche Schnittstelle zur Replikation mit mehreren Quellservern angeboten wird. Eine Variante des Agent-Systems stellt der *mobile Code* dar. Dort sammeln Agents ausführbaren Code für die Clients.⁶⁹

Um auf die Probleme der mobilen Geräte in Hinsicht auf schwache und abbrechende Verbindungen wie auch auf das Auftreten heterogener Geräte mit verschiedenen Kommunikationstechnologien und Betriebssystemen zu reagieren, ist die Einführung einer Schicht aus *Mediatoren* denkbar. Dieses Modell weist eine Verwandtschaft zum Broker- bzw. Agentmodell auf und ist speziell auf mobile Geräte angepasst. Wie das Agent-Modell rufen Mediatoren, passend zur Anfrage, mehrere Dienste auf und erstellen daraus ein Ergebnis. Weiterhin entkoppelt es wie die Broker die Server von den Clients. Dadurch müssen Server keine verschiedenen Schnittstellen für heterogene mobile Clients bereitstellen und umgekehrt mobile Clients keine heterogenen Server

⁶⁷ Vergl. Buschmann, Frank et al. (2000), S. 99 und Bengel, Günther (2004), S. 68f

⁶⁸ Vergl. Microsoft (2009 b)

⁶⁹ Vergl. Bengel, Günther (2004), S. 78f, Mutschler, Bela (2004), S. 73ff und Bauder, Irene (2006), S. 440ff

unterstützten. Um Verbindungsverluste mobiler Clients zu unterstützen, können Mediatoren Ergebnisse für den Client zwischenspeichern. Bei einer wiederhergestellten Verbindung kann dann das zwischengespeicherte Ergebnis übertragen werden.⁷⁰

Neben den vorgestellten Grundmodellen gibt es weitere Modelle, mit denen je nach Anwendungsfall weitere Strukturen realisiert werden können. Mithilfe der Erweiterungen des einfachen Client-Server-Modells, welche kombinierbar sind, können komplexe Strukturen entstehen, die eine feine Aufteilung der Applikation auf verschiedene Knoten erlaubt und die Wiederverwendbarkeit von Diensten ermöglicht.⁷¹

Die Modelle für die Verteilung basieren auf *Middlewares*. Diese realisieren die Kommunikation zwischen den einzelnen Teilnehmern und ermöglichen die eigentliche Verteilung sowie die mehrschichtigen Architekturen. Bis heute sind verschiedene Kommunikationsmodelle wie z. B. *Remote Procedure Calls* (RPC) oder *verteilte Objekte* entwickelt worden. Sie übernehmen die reale Netzwerkkommunikation und vereinfachen dadurch die Entwicklung eines verteilten Systems. Diese verschiedenen Ansätze ermöglichen jeweils eine unterschiedliche Granularität in der Verteilung (z. B. einzelne Funktionen bei RPC oder Objekte, die auf die Teilnehmer aufgeteilt werden) und verwenden unterschiedliche Protokolle, um eine Kommunikation zu realisieren.⁷²

Eine Middleware, welche für mobile Clients geeignet ist, stellen *Web Services* dar. Im Gegensatz zu RPC oder verteilten Objekten verwenden Web Services Transportprotokolle, welche standardisiert sowie technologie- und plattformübergreifend sind. Der Standard des W3C, welcher Web Services definiert, erwähnt als Transportprotokoll explizit HTTP und das **Simple Mail Transfer Protocol** (SMTP), aber andere standardisierte Transportprotokolle wie z. B. das **File Transfer Protocol** (FTP) sind ebenfalls standardkonform.⁷³ Web Services kommunizieren über Nachrichten in der **Extensible Markup Language** (XML), die nach dem **Simple Object Access Protocol** (SOAP) aufgebaut sind.⁷⁴ Die Verwendung systemunabhängiger Technologien wie XML und standardisierten Transportprotokollen ermöglicht eine Kommunikation zwischen heterogenen Teilnehmern. Da mobile Geräte wie Handhelds Web Services unterstützen, gibt es inzwischen mobile Web Services, die z.B. eine

⁷⁰ Vergl. Mertens, Stefan (2003), S. 145ff

⁷¹ Für eine detaillierte Einführung in weitere Client-Server-Modelle, vergl. Bengel, Günther (2003), Kapitel 2 (S. 27ff).

⁷² Vergl. Bengel, Günther (2004), S. 85f und S. 137 sowie Tannenbaum, Andrew S. et al. (2008), S. 147 und Mutschler, Bela et al. (2004), S. 71

⁷³ Vergl. Heutschi, Roger (2007), S. 53

⁷⁴ Für eine Einführung in den genauen Aufbau von Web Services, vergl. Coulouris, George et al. (2005), S. 783ff

Integration von mobilen Geräten in bereits vorhandene Systeme erlauben.⁷⁵

Bei der Verteilung von Datenbeständen und DBS kann auf die verschiedenen Modelle für die Aufteilung und Techniken zur Kommunikation und Verteilung zurückgegriffen werden. Dasselbe gilt für den Aufbau des Tests eines mobilen DBS mit einer Anbindung an ein zentrales Serversystem. In dieser Arbeit wird für den Aufbau des Testsystems ein einfaches Client-Server-Modell (Vergl. Abbildung 5), welches auf Web Services basiert, verwendet. Das mobile Testgerät wird die Testdaten ermitteln und an den Server übertragen. Dort werden dann die Messergebnisse aufbereitet und angezeigt. Weiterhin wird über dieses Modell der Client in die Lage versetzt, das Serversystem während der Tests zu steuern.

4.3. Einführung in verteilte Datenbanksysteme

Beim Aufbau eines Thick-Clients, welcher in Verbindung mit einem zentralen System arbeiten soll, stellt sich die Frage, wie ein Datenbestand auf verschiedene Geräte verteilt werden kann. Das vorhergehende Kapitel hat grundlegende Modelle vorgestellt, um allgemein eine Verteilung von Software vorzunehmen. Dieses Kapitel stellt Modelle und Techniken vor, die bei einer Aufteilung eines Datenbestandes und somit bei der Verteilung von *Daten* verwendet werden können. Für die Verwaltung der Daten auf mehreren Geräten werden weiterhin verteilte Datenbanksysteme benötigt, die hier ebenfalls dargestellt werden.

4.3.1. Grundlagen

Verteilte Datenbanken sind charakterisiert durch die Aufteilung auf mehrere Teilnehmer, die so genannten *Sites* oder *Knoten*, die selber ein eigenes Modul mit DBMS-Funktionalitäten bereithalten. Sie werden nach drei verschiedenen Dimensionen klassifiziert. Nach der Heterogenität der einzelnen Teilnehmer (Unterschiedliche Datenbankschemata oder Datenbanksysteme o. ä.), Verteilung (Anzahl der Computersysteme mit einem eigenen DBS) und die Autonomie der einzelnen Teilnehmer (In wie weit können sie unabhängig voneinander agieren).⁷⁶

Die einzelnen Sites können über ein globales Schema, nach dem die Daten im Gesamtsystem strukturiert werden, als Gesamtheit auftreten. Dies bezeichnet man als *globales konzeptionelles Schema*. Um die einzelnen Knoten, unabhängig von ihrem internen, lokalen Schema der Datenbank, in dieses globale Schema zu integrieren,

⁷⁵ Vergl. Ferstl, Otto K. et al. (2005), S. 377ff und Bengel, Günther (2004), S. 267ff sowie Tanenbaum, Andrew S. et al. (2008), S. 603ff

⁷⁶ Vergl. Özsu, M. Tamer et al. (1999), S. 82ff und Mutschler, Bela et al. (2004), S. 64ff

können sie ein *lokales konzeptionelles Schema* bereitstellen. Die Verteilung der Daten auf die einzelnen Knoten wird durch ein *Verteilungsschema* auf globaler Ebene beschrieben. Durch dieses Vorgehen entsteht die in Abbildung 6 dargestellte Struktur einer verteilten Datenbank.

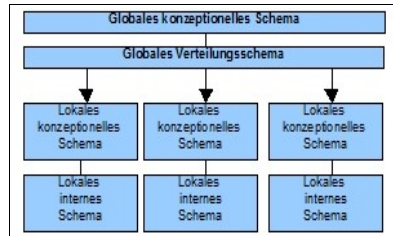


Abbildung 6: Schemata einer verteilten Datenbank⁷⁷

Die Verwendung eines lokalen konzeptionellen Schemas wird speziell dann benötigt, wenn bereits vorhandene Datenbanken über ein globales System miteinander verbunden werden (*föderiertes Datenbanksystem*). Da die internen Schemata vom globalen konzeptionellen Schema abweichen können, sorgen die lokalen konzeptionellen Schemata für eine entsprechende Umsetzung zwischen dem lokalen internen und dem globalen Schema. Innerhalb einer verteilten Datenbank, welche bereits von Anfang an in Hinblick auf eine Verteilung geplant wird, kann auf das lokale konzeptionelle Schema verzichtet werden. Statt dessen können die lokalen internen Schemata vom globalen Schema abgeleitet werden (*homogen verteiltes Datenbanksystem*).⁷⁸

Eine Aufteilung der Daten auf die einzelnen Knoten wird als *Fragmentierung* bezeichnet. Bei einer *vollständigen Fragmentierung* halten die verschiedenen Knoten unterschiedliche Daten. Die Ergebnisermittlung muss somit alle Knoten einbeziehen, damit ein vollständiges Ergebnis geliefert werden kann. Um die Daten zu fragmentieren, gibt es zwei verschiedene Ansätze. Zum einen können die Daten *horizontal* aufgeteilt werden. Bei dieser Art werden die einzelnen Zeilen globaler Relationen auf die verschiedenen Knoten aufgeteilt. Zum anderen können die Daten *vertikal* aufgeteilt werden. Hier werden die globalen Relationen in einzelne Spalten zerlegt. Diese Spalten werden auf die Knoten aufgeteilt. Bei der Wahl eines Vorgehens für die Fragmentierung des Datenbestandes in mobilen Umgebungen muss der Ausfall der einzelnen Knoten mit beachtet werden.⁷⁹

Gegenüber der vollständigen Fragmentierung des Datenbestandes steht die *vollständige*

⁷⁷ Vergl. Özsu, M. Tamer et al. (1999), S. 90f und Mutschler, Bela et al. (2004), S. 69f.

⁷⁸ Vergl. Höpfner, Hagen (2005), S. 219ff

⁷⁹ Vergl. Elmasri, Ramez et al. (2007), S. 857 und S. 861f; Özsu, M. Tamer et al. (1999) S. 112f und S.131 sowie Mutschler, Bela et al. (2004), S. 81 und Tannenbaum, Andrew S. et al. (2008), S. 62f

Replikation. Unter *Replikation* versteht man das redundante Halten eines Teils der Daten auf mehreren oder allen Knoten. Im Falle der vollständigen Replikation halten alle Knoten alle Daten. Methoden zur Replikation ermöglichen den Ausgleich von Ausfällen einzelner Knoten im Gesamtsystem und stellen einen Lösungsansatz dar, um Datensätze vom zentralen System lokal für den Offline-Zugriff auf den mobilen Geräten zu speichern. Somit spielen Replikationsverfahren eine zentrale Rolle beim Aufbau eines Systems mit mobilen Thick-Clients, um lokale Kopien von Daten aus dem Gesamtsystem zu erzeugen.⁸⁰

Eine vollständige Fragmentierung stellt für mobile Geräte lediglich in Online-Szenarien eine sinnvolle Lösung dar, da die Verfügbarkeit des Datenbestandes durch Geräte ohne Verbindung beschränkt wird. Zusätzlich führen schwache und langsame Verbindungen zu einer sehr langen Laufzeit bei der Verarbeitung von Abfragen, die alle verfügbaren mobilen Geräte mit einbeziehen. Umgekehrt stellt eine vollständige Replikation aufgrund geringer Speichergrößen auf mobilen Geräten ebenfalls keine Lösung dar.⁸¹

Um eine Replikation des Datenbestandes zu ermöglichen, können *Replikationsschemadefinitionen* verwendet werden. Über diese Definition werden Teile des globalen Datenbankschemas für die Replikation ausgewählt. Auf dieser Grundlage können *Replikationsdefinitionen* erstellt werden. Diese stellen eine Auswahl der Daten aus einer Replikationsschemadefinition für die einzelnen Knoten dar. Ein Problem der Replikation ist der Konflikt von drei Forderungen, welche an Replikationen gestellt werden. Zum einen sollen die Daten möglichst immer verfügbar und aktuell sein. Zum anderen soll der Datenbestand möglichst immer konsistent sein. Und als letztes sollen möglichst geringe Kommunikations- und Berechnungskosten auftreten. Daher gibt es verschiedene Verfahren zur Synchronisation und Modelle für den Gesamtaufbau der Replikation, welche die einzelnen Forderungen in unterschiedlicher Stärke erfüllen.⁸²

Synchronisationsverfahren werden in zwei unterschiedliche Typen eingeteilt. Die *konsistenzerhaltenden Verfahren* haben als Ziel die Erhaltung der Datenkonsistenz. Dem gegenüber stehen die *verfügbarkeitserhaltenden Verfahren*, die die Bedingung der Konsistenz an den Datenbestand begrenzt aufgeben, um eine höhere Verfügbarkeit von Änderungsoperationen zu erreichen. Verletzungen der Konsistenzbedingung an den Datenbestand können sich bei der Verwendung einer Replikation durch gleichzeitige

⁸⁰ Vergl. Elmasri, Ramez et al. (2007), S. 860ff

⁸¹ Vergl. Mutschler, Bela et al. (2004), S. 97f

⁸² Vergl. Mutschler, Bela et al. (2004), S. 81 und 85ff sowie Höpfner, Hagen et al. (2005), S. 226f und S. 232

Änderung an zwei verschiedenen Kopien eines Datenbestandes ergeben. Da die Aufgabe der Konsistenzbedingung zu einem fehlerhaften Zustand der Daten führen kann, konzentriert sich diese Arbeit auf konsistenzerhaltende Verfahren.⁸³

Die konsistenzerhaltenden Verfahren können noch einmal in *pessimistische* (konfliktvermeidende) und *optimistische* (konfliktauflösende) Verfahren eingeteilt werden. Pessimistische Verfahren verwenden Sperren, die während einer Änderung weitere Änderungszugriffe auf den Datensatz sowie den verschiedenen Kopien verhindern und schließen dadurch Inkonsistenzen aus. Der Vorgang der Synchronisation wird dadurch erheblich erleichtert, da kein Abgleich von Änderungen an verschiedenen Stellen in der verteilten Datenbank geschehen muss. Diese Verfahren sind für ein Szenario mit mobilen Geräten lediglich begrenzt einsetzbar. Änderungen können nur dann als vollständig durchgeführt betrachtet werden, wenn alle Knoten diese übernommen haben. Somit muss auf alle Geräte gewartet werden; auch auf die Geräte, welche zu dem Zeitpunkt keine Verbindung zum Gesamtsystem besitzen. Andererseits können auf Knoten ohne eine Verbindung keine Änderungen vorgenommen werden, da keine Sperrung im Gesamtsystem vorgenommen werden kann. Weiterhin steigt der Aufwand der Erzeugung von Sperren mit der Größe des Systems.⁸⁴

Günstigere Ansätze für die Verwendung von mobilen Knoten stellen die optimistischen Verfahren dar. Diese gehen von einem seltenen Auftreten von Inkonsistenzen aus und verzichten daher auf Sperren. Sie lassen lokale Änderungen auf den einzelnen Kopien zu und führen später eine Synchronisation mit dem gesamten System durch. Bei der Verwendung eines Client-Server-Modells kann in diesem Fall mit dem Server synchronisiert werden, welcher die Änderungen an die anderen Kopien verteilt.⁸⁵

Ein Problem bei optimistischen Verfahren stellen die erwähnten Inkonsistenzen durch voneinander unabhängige Änderungen an verschiedenen Kopien eines Datensatzes dar. Die optimistischen Verfahren gehen zwar von einer geringen Anzahl solcher Konflikte aus, müssen aber auf das mögliche Auftreten vorbereitet sein. Deshalb wenden optimistische Verfahren eine Konflikterkennung und -lösung an. Für die Erkennung wird eine Validierungsphase vor der Übernahme von Änderungen aus lokalen Knoten verwendet. Innerhalb dieser Phase wird ebenfalls entschieden, wie mit einem möglichen Konflikt umgegangen wird (z. B. die neueste Änderung setzt sich durch). Während die

⁸³ Vergl. Mutschler, Bela et al. (2004), S. 89 sowie S. 95

⁸⁴ Vergl. Mutschler, Bela et al. (2004), S. 89, Höpfner, Hagen et al. (2005), S. 221f und Tanenbaum, Andrew S. et al. (2008), S. 349

⁸⁵ Vergl. Höpfner, Hagen et al. (2005), S. 232f und Mutschler, Bela et al. (2004), S. 95

optimistischen Verfahren den Zustand der Inkonsistenz als Fehlerzustand ansehen, werden die vorher erwähnten verfügbarkeitserhaltenden Verfahren, die eine Toleranz von Inkonsistenzen in einem definierten Rahmen vorsehen, als eine Variante der optimistischen Verfahren angesehen.⁸⁶

Weiterhin wird bei Synchronisationen zwischen verschiedenen Szenarien unterschieden. Zwei häufig verwendete Szenarien ist die *bidirektionale Synchronisation* sowie die *unidirektionale Synchronisation*. Bei der bidirektionalen Synchronisation können auf Client und Server Änderungen auftreten, daher müssen Daten in beide Richtungen synchronisiert werden. Dem Gegenüber steht die unidirektionale Synchronisation, bei der nur auf einer Seite Änderungen entstehen (z. B. weil die Clients nur lesen dürfen).⁸⁷ Innerhalb des Beispiels der Vertriebsmitarbeiter (Vergl. Kapitel 2.2) ist von Änderungen im zentralen Datenbestand sowie auf dem mobilen Gerät auszugehen. Neue Kunden oder Artikel müssen an die mobilen Geräte verteilt werden, während z. B. neu angelegte Aufträge in das zentrale System übertragen werden müssen, um dort weiterverarbeitet zu werden. Daher wird in dieser Arbeit eine bidirektionale Synchronisation verwendet.

Bei der Kommunikation zwischen den einzelnen Teilnehmern des verteilten DBS stellen Transaktionen ein Problem dar. Um globale Transaktionen in einer verteilten Datenbank oder einem verteilten System über mehrere Teilnehmer hinweg durchzuführen, sind Modelle für *verteilte Transaktionen* entstanden. Für die Einbindung mobiler Geräte in verteilte Systeme, bzw. verteilte Datenbanksysteme, sind diese Modelle erweitert worden, um Transaktionen über die schwache und unsichere Verbindung eines mobilen Gerätes zu ermöglichen. Einige können mit kurzzeitigen Abbrüchen im Millisekundenbereich während eines Wechsels der Basisstation umgehen (*Transaktionen Klasse eins*). Andere können mit vollständigen Abbrüchen umgehen (*Transaktionen Klasse zwei*).⁸⁸

4.3.2. Replikationsmodelle für den Einsatz mit mobilen Geräten

Um eine Replikation zu ermöglichen, muss ein *Replikationsmodell* mit einem entsprechenden *Synchronisationsverfahren* ausgewählt werden. Da sich innerhalb dieser Arbeit auf konsistenzhaltende Verfahren konzentriert, wird jeweils ein pessimistisches

⁸⁶ Vergl. Höpfner, Hagen et al. (2005), S. 232f und Mutschler, Bela et al. (2004), S. 95

⁸⁷ Vergl. Mutschler, Bela (2004), S. 99ff und S. 105 (dort werden auch weitere Arten der Synchronisation, zusätzlich zu bi- und unidirektionalen Synchronisationen erläutert) und Höpfner, Hagen (2005), S. 218f

⁸⁸ Vergl. Mutschler, Bela et al. (2004), S. 209 ff., dieses Buch kann auch als weiterführende Literatur für mobile Transaktionen verwendet werden. Für verteilte Transaktionen, siehe Coulouris, George et al. (2005), Kapitel 14 oder Özsu, M. Tamer et al. (1999), Kapitel 10 und 11

und ein optimistisches Verfahren für mobile Umgebungen vorgestellt. Diese beiden Modelle haben, im Gegensatz zu klassischen Konzepten, den Vorteil, dass sie an mobile Umgebungen angepasst sind.⁸⁹

Das erste Modell soll zeigen, auf welche Probleme Replikationsmodelle mit pessimistischen Verfahren in mobilen Umgebungen stoßen können. Das zweite Modell ist innerhalb des Microsoft SQL Server-System die Grundlage für eine bidirektionale Synchronisation.⁹⁰ Daher wird in dieser Arbeit das zweite Modell verwendet. Neben diesen Modellen gibt es weitere Synchronisationsmodelle für mobile Umgebungen.⁹¹

Ein Modell, welches an mobile Geräte angepasst ist und ein pessimistisches Synchronisationsverfahren verwendet, ist das *Virtual-Primary-Copy-Modell*. Dieses Modell stellt eine Weiterentwicklung des *Primary-Copy-Modells* dar. Das Primary-Copy-Modell stellt einen Peer-To-Peer-Ansatz dar, bei dem von jedem zu replizierenden Datensatz eine *Masterkopie* auf einem Knoten innerhalb des Gesamtsystems existiert. Änderungen werden zuerst an der Masterkopie durchgeführt. Bevor die Änderungen durchgeführt werden, wird die Masterkopie für weitere Änderungen gesperrt. Die an der Masterkopie vorgenommenen Änderungen werden dann an den Kopien vorgenommen und danach wird die Sperre wieder entfernt. Falls die Masterkopie auf einem mobilen Gerät ohne Konnektivität liegt, können Änderungen nicht durchgeführt bzw. beendet werden, solange die Verbindung nicht wiederhergestellt ist. Daher wird beim Virtual-Primary-Copy-Verfahren von den Masterkopien jeweils eine virtuelle Masterkopie angelegt. An diese wird die Anforderung gestellt, dass sie ständig erreichbar ist (z. B. indem sie sich auf einem stationären Teilnehmer befindet). Änderungen werden, falls die Masterkopie nicht erreichbar ist, an der virtuellen Kopie vorgenommen, welche später mit der Masterkopie wieder vereint wird. Das Problem der Unerreichbarkeit der Masterkopie wird damit entschärft. Geräte ohne Konnektivität zur virtuellen und regulären Masterkopie können trotz dessen keine Änderungen an diesem Datensatz vornehmen.⁹²

Ein für Offline-Szenarien geeignetes, auf optimistischen Synchronisationsverfahren basierendes Replikationsmodell stellt das *Snapshot-Modell* dar. Das Snapshot-Modell

89 Vergl. Mutschler, Bela et al. (2004), S. 98 und S. 108. Für eine Einführung in klassische Modelle wie z. B. der Device-Master- oder der Peer-To-Peer-Replikation, vergl. Terry, Douglas B. et al. (2007), Kapitel 2 (S. 7ff) und Mutschler, Bela et al. (2004), S. 88ff.

90 Vergl. Bauder, Irene (2006), S. 441f und 443ff. Innerhalb des SQL Servers wird anstelle von einer bidirektionalen Synchronisation von einer Merge-Synchronisation gesprochen.

91 Vergl. für weiterführende mobile Modelle Terry, Douglas B. et al. (2007), Kapitel 2 (S. 7ff).

92 Vergl. Elmasri, Ramez et al. (2007), S. 880f und Mutschler, Bela et al. (2004), S. 98f

folgt einem Client-Server-Modell, bei dem die Daten innerhalb eines oder mehrerer zentraler Datenbankserver (*Master-Sites*) gespeichert werden. Basierend auf den zentralen Datenbeständen werden Snapshots auf Grundlage einer (*einfacher Snapshot*) oder mehrerer (*komplexer Snapshot*) Relationen aus einer Datenbank definiert. Innerhalb des Snapshot-Modells sind lokale Änderungen der replizierten Daten möglich. Synchronisationen ermöglichen den Abgleich der Daten zwischen den Clients und dem zentralen System. Bei komplexen Snapshots kann durch die Verbindung mehrerer Relationen die Möglichkeit zur Änderung der Daten wegfallen. Um mehrere Snapshots zu bündeln und den Clients zur Verfügung zu stellen, kann das Snapshot-Modell mit einem Publish-Subscribe-Modell kombiniert werden.⁹³

Beim Publish-Subscribe-Modell werden *Publikationen* erstellt, die aus einem oder mehreren Snapshots (*Publikationsartikel*) bestehen. Bei diesem Modell bietet sich eine Replikationsschicht an, die die Verwaltung dieser Publikationen übernimmt. Clients können sich für die verschiedenen Publikationen anmelden, um diese zu erhalten und lokal zu speichern (*Subskription*). Dies kann auch dynamisch während der Ausführung einer Applikation geschehen, um Daten zu erhalten, die zum Zeitpunkt des Abgleichs benötigt werden.⁹⁴ Das Snapshot-Verfahren stellt eine für Thick-Clients gut angepasste Struktur dar, da lokale Änderungen ohne Verbindung durchgeführt werden können. Das SQL Server-System verwendet für bidirektionale Synchronisation eine Kombination aus diesen beiden Modellen.⁹⁵

4.3.3. Vergleich von Architekturen für verteilte DBS mit mobilen Geräten

Innerhalb dieses Kapitels wird eine Architektur für das in dieser Arbeit aufzubauende verteilte DBS ausgewählt. Für die Verteilung von Datenbanksystemen gibt es mehrere Architekturen, um die einzelnen Komponenten als Gesamtsystem auftreten zu lassen.

Abbildung 7 stellt zwei grundlegende Ansätze vor. Die Architektur mit einem mobilen *Multidatenbanksystem* (MDBS) stellt eine Architektur dar, die dem logischen Aufbau einer verteilten Datenbank, wie sie in Abbildung 6 im Kapitel 4.3.1 dargestellt wurde, am ehesten entspricht.⁹⁶ Dort übernimmt eine globale Komponente die Verwaltung der einzelnen Knoten. Dem gegenüber steht die zweite Architektur, welche von einer Replikation ausgehend entwickelt worden ist und daher besser auf die Gegebenheit eines mobilen Thick-Clients angepasst ist.

⁹³ Vergl. Mutschler, Bela et al. (2004), S. 99f

⁹⁴ Vergl. Höpfner, Hagen et al. (2005), S. 223f

⁹⁵ Vergl. Mutschler, Bela (2004), S. 99f und Bauder, Irene (2006), S. 439ff und 443ff.

⁹⁶ Vergl. Mutschler, Bela et al. (2004), S. 70

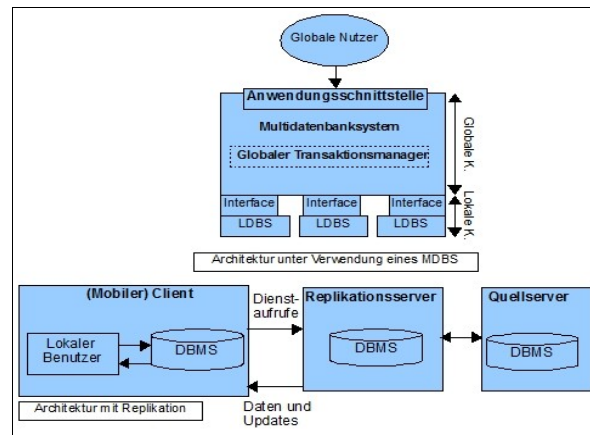


Abbildung 7: Architekturen eines verteilten DBS⁹⁷

Das MDBS übernimmt die Aufgabe, Anfragen zu verarbeiten, ein globales Schema bereitzustellen und die Existenz der beteiligten Datenbanksysteme vor dem Benutzer zu verbergen (*Verteilungstransparenz*). Die lokalen Datenbanksysteme (LDBS) werden vom globalen MDBS angesprochen, während sich Benutzer ausschließlich direkt an das MDBS wenden. Ein globaler Transaktionsmanager übernimmt die Durchsetzung der ACID-Eigenschaften im globalen System. Anfragen müssen innerhalb dieser Architektur aufgeteilt werden, um aus den verschiedenen Knoten die Daten zu ermitteln und daraus ein Ergebnis zu generieren (*verteilte Abfragen*). Ein Vorteil dieses Vorgehens besteht in der Transparenz der Verteilung. Benutzer des MDBS müssen lediglich das globale System ansprechen, ohne die Verteilung berücksichtigen zu müssen. Allerdings können mobile Knoten ohne Verbindung nicht auf das MDBS zugreifen. Das MDBS tritt als Client der verschiedenen Knoten auf, während die Clients des gesamten DBS sich an das MDBS selbst wenden.⁹⁸

Der Begriff Multidatenbanksystem entstammt dem Umstand, dass die einzelnen Knoten eigene Datenbanksysteme haben, welchen ihren Inhalt mit der globalen Komponente teilen. Diese Architektur kann auch mit einem verteilten DBS aufgebaut werden, welches auf die einzelnen Knoten verteilt wird und somit die Kontrolle über den lokalen Datenbestand auf jedem Knoten hat. Die beiden Ansätze ähneln sich, allerdings gibt es im zweiten Fall nur ein verteiltes DBS, während das MDBS mehrere lokale Datenbanksysteme zu einem Gesamtsystem verbindet.⁹⁹

Innerhalb der zweiten Architektur geht man von der direkten Verwendung des LDBS

⁹⁷ Vergl. Mutschler, Bela et al. (2004), S. 70 und S. 101ff; Höpfner, Hagen (2005), S. 219 sowie Özsu, M. Tamer (1999), S. 98

⁹⁸ Vergl. Mutschler, Bela et al. (2004), S. 70f

⁹⁹ Vergl. Özsu, Tamer M. (1999), S. 94

durch dem Benutzer aus. Das LDBS wird durch eine Replikation mit einem Synchronisationsverfahren auf einem möglichst aktuellen Stand gehalten. Dabei können mehrere Quellserver verwendet werden. Der Einsatz einer Replikationsschicht stellt bei dieser Architektur eine Möglichkeit dar, den Datenbankserver zu entlasten sowie die Client- und Serverschicht voneinander abzukoppeln (Broker-Modell, vergl. Kapitel 4.2). Die Abkoppelung stellt dabei die Hauptaufgabe der Replikationsschicht dar. Ein zusätzliches DBS auf dem Replikationsserver ermöglicht eine Zwischenspeicherung und Verarbeitung der Daten der Clients und der Server. Weiterhin kann die Replikationsschicht eine Sicherung der Datenbankserver vornehmen, ähnlich einer Firewall.¹⁰⁰

Um bei einer Synchronisation Daten aus mehreren Quellservern zu ermitteln bzw. die Clients mit mehreren Replikationsschichten zu verbinden, bietet sich zusätzlich der Einsatz von Agents an. Mithilfe von Agents auf Clientseite können mehrere Replikationsserver mit unterschiedlichen Inhalten eingebunden werden. Weiterhin können Agents auf der Serverseite auf Synchronisationsanfragen von den Clients eine Antwort aus den Daten mehrerer Quellserver erstellen. Dieses Vorgehen ist auch im Microsoft SQL Server-System vorgesehen. Agents werden im Kapitel 4.2 näher erläutert.

Diese Architektur ist gleichzeitig die Grundlage für eine weitere Form der Replikation. Die Dienste, die in der Replikationsschicht bereitgestellt werden, können auch orts- und zeitabhängig implementiert werden. Damit können mobile Knoten Informationen abrufen, abhängig vom Ort des Benutzers und dem Zeitpunkt der Synchronisation. Eine solche Replikation wird als *benutzerdefinierte Replikation* bezeichnet.¹⁰¹

Die zweite Architektur stellt eine für mobile Umgebungen geeignete Struktur dar. Es müssen keine Abfragen unter Einbeziehung des Gesamtsystems durchgeführt werden und es werden Verbindungsabbrüche unterstützt. Diese Architektur in Verbindung mit dem Snapshot- und Publish-Subscribe-Modell für die Replikation ermöglicht den Benutzern, unabhängig voneinander zu arbeiten. Für das Beispiel der Vertriebsmitarbeiter (Vergl. Kapitel 2.2) bedeutet dies, dass die einzelnen Mitarbeiter nicht auf langwierige verteilte Abfragen o. ä. warten müssen. Weiterhin wird diese Architektur vom Microsoft SQL Server-System unterstützt. Daher wird im weiteren Verlauf die zweite Architektur (nicht orts- und zeitabhängig) verwendet.

¹⁰⁰ Vergl. Mutschler, Bela et al. (2004), S. 101ff und Höpfner, Hagen (2005), S. 219

¹⁰¹ Vergl. Mutschler, Bela et al. (2004), S. 101ff und Weikum, Gerhard et al. (2003), S. 453ff

5. Performance-Tests

Um Auswirkungen geringer Leistungsfähigkeit und Speichergröße mobiler Geräte und geringe Datenraten der Netzwerktechnologien zu bestimmen, können Performance-Tests aufgebaut werden. Dieses Kapitel stellt dar, welche Ziele Performance-Tests haben können und welche Punkte bei der Modellierung eines solchen Tests beachtet werden müssen.

5.1. *Notwendigkeit von Performance-Tests*

An die Performance eines Systems werden bei der Entwicklung einer Software im Normalfall Anforderungen gestellt. Diese stellen einen Teil der so genannten nicht funktionalen Anforderungen dar. Nicht funktionale Anforderungen stellen Qualitätsansprüche dar, welche häufig schwer messbar sind. Dazu gehören Ansprüche wie z. B. einfache und intuitive Bedienung oder ein schnell reagierendes System. Um Performance-Ansprüche messbar zu machen, müssen quantitative Aussagen zu bestimmten Performance-Merkmalen getroffen werden. Solche Merkmale stellen z. B. die Geschwindigkeit, in der Anfragen verarbeitet werden oder die Anzahl von Operationen, die in einer bestimmten Zeit möglich sind, dar. Diese Merkmale nennt man *Metriken*. Anforderungen an solche Eigenschaften (z. B. max. Antwortzeit oder ein Minimum an Transaktionen, welche innerhalb eines bestimmten Zeitfensters verarbeitet werden müssen) können gemessen und überprüft werden. Performance-Tests sind inzwischen Teil von standardisierten Testmodellen wie z. B. dem ISO-Standard 9126.¹⁰² Performance-Tests sind ein Mittel, um festzustellen, ob ein Gesamtsystem mit verschiedenen Softwarebestandteilen den Anforderungen an die Performance genügt. Falls solche Tests nicht vor der Einführung eines Systems durchgeführt werden, kann dies zu einem Fehlschlag des Projektes führen. Weiterhin können Performance-Tests zeigen, für welche Last ein System geeignet ist, um Kunden oder den eigenen Marketing-Experten Informationen zu einzelnen Metriken bereitzustellen. Damit kann vor dem Kauf eines Systems entschieden werden, ob dieses für den jeweiligen Einsatzzweck geeignet ist. Performance-Tests werden ebenfalls verwendet, um Teile innerhalb eines Systems zu finden, die einen wesentlichen negativen Einfluss auf die Performance haben und Entwicklern aufzeigen, an welcher Stelle sie ansetzen können, um auftretende Performance-Probleme zu beheben.¹⁰³

¹⁰² Vergl. Spillner, Andreas et al. (2007), S. 71F; Sneed, Harry M. et al. (2009), S. 25F; Baker, Paul et al. (2008), S. 106ff; Dirlewanger, Werner (2006), S. 14 und Ford, Chris et al. (2008), S. 1

¹⁰³ Vergl. Ford, Chris et al. (2008), S. 1ff und Beydeda, Sami (Hrsg.) et al. (2004), S.79

Das nachfolgende Kapitel beschreibt anhand eines Vorgehensmodells, wie Performance-Tests aufgebaut werden können, speziell in Hinsicht auf den Test des mobilen Datenbanksystems und der Synchronisation.

5.2. Herangehensweise an Performance-Tests

Vor der Entwicklung eines Performance-Tests stellt sich die Frage nach der Art dieses Tests. Je nach Ziel gibt es unterschiedliche Arten von Performance-Tests. Um z. B. zu zeigen, wie sich ein System unter der zu erwartenden Maximallast einer Anwendung oder sogar einer darüber hinausgehenden Last verhält, können *Stresstests* durchgeführt werden. Dagegen zeigen *Lasttests*, wie sich ein System unter der für das System gedachten Last verhält. Dafür wird bis zur maximalen geplanten Last getestet (oder leicht darüber, um eine Grenze für Abweichungen zu haben).¹⁰⁴

Um zu bestimmen, wo die reale Maximallast eines Systems liegt, bieten sich *Skalierungstests* an. Unter dem Begriff Skalierungstest versteht man verschiedene Arten von Tests. Sie können zeigen, wie sich die Performance eines Systems verbessert, wenn weitere Teile, wie z. B. zusätzliche Prozessoren, dem ausführenden Computersystem hinzugefügt werden. Unter einem Skalierungstest versteht man aber auch einen Test, der die realen Leistungsgrenzen eines Systems ermitteln soll, z. B. indem die Last, unter der das System arbeitet, zunehmend erhöht wird, bis Anomalien auftreten.¹⁰⁵

Je nach Literatur werden diese verschiedenen Varianten als spezielle Art eines Performance-Tests angesehen, während andere mit Performance-Test speziell die Messung von Laufzeiten meinen (auch *Laufzeittest* genannt) und unter den anderen Varianten jeweils eigene Arten von Tests meinen. In dieser Arbeit wird der ersten Ansicht gefolgt, in der Performance-Test als globaler Begriff verwendet wird, welcher in verschiedenen Variationen auftreten kann.¹⁰⁶

Innerhalb dieser Arbeit wird ein Lasttest aufgebaut. Da reale Vorgaben der maximalen Last fehlen, soll untersucht werden, wie sich das mobile DBS und die Synchronisation unter verschiedenen Laststufen verhält. Tabelle 3 beschreibt, wie ein Performance-Test aufgebaut werden kann.

¹⁰⁴ Vergl. Beydeda, Sani (Hrsg.) (2004), S. 87f

¹⁰⁵ Vergl. Baker, Paul et al. (2008), S. 106f und Yang, Laurence et al. (2005), S. 696

¹⁰⁶ Vergl. Baker, Paul et al. (2008), S.106f; Spillner, Andreas et al. (2005), S. 72 und Hoffmann, Dirk W. (2008), S. 173

Schritt	Beschreibung
1: Schlüsselszenario/ -en entwickeln	Innerhalb des ersten Schritts werden einzelne Szenarien aufgebaut, welche getestet werden sollen. Damit wird eine Grundlage geschaffen, auf der detaillierte Testfälle definiert werden können. Zum Beispiel kann ein Szenario das Einfügen von Daten in die Datenbank darstellen.
2: Arbeitslast identifizieren	Schritt zwei stellt die Definition der Last, unter der der Test durchgeführt wird, dar. Dabei können unterschiedliche Arten von Lasten mit in die Tests übernommen werden. Dies kann z. B. die Anzahl von Benutzern, zu verarbeitende Daten oder die Anzahl an Prozessen, die ein Computersystem belasten, sein. Die Last hängt dabei von der zu testenden Software sowie den Anforderungen, die an diese gestellt werden, ab.
3: Metriken identifizieren	Um einen Performance-Test durchzuführen, muss festgelegt werden, welche Performance-Merkmale getestet werden sollen.
4: Testfälle entwickeln	Auf Grundlage der Szenarien, der Last sowie der Metrik oder den Metriken werden einzelne Testfälle entwickelt. Ein Beispiel stellt eine detailliert beschriebene Anfrage, die Daten in die Datenbank einfügt, dar.
5: Testdurchführung	Nach der Definition der Testfälle werden diese durchgeführt. Dies kann z. B. per Hand, unter Verwendung von bereits vorhandenen Testtools oder durch eine eigens entwickelte Testsoftware geschehen. Um Abweichungen auszugleichen, die eine Messung durch nicht oder wenig beeinflussbaren Veränderungen der Messbedingungen verursacht werden (z. B. Prozesse, die vom Betriebssystem benötigt werden und während der Testdurchführung unterschiedlich stark arbeiten), können die Messungen wiederholt und ein Mittelwert gebildet werden sowie Überprüfungen durchgeführt werden, ob die Wiederholungen ausreichend waren. ¹⁰⁷
6: Analyse der Ergebnisse	Nach einer Ermittlung der Ergebnisse müssen diese interpretiert werden. Um dies zu ermöglichen, können die Ergebnisse graphisch aufbereitet werden. Auf Grundlage der Ergebnisse können dann Aussagen über die Leistungsfähigkeit des getesteten Systems getroffen werden. Innerhalb von Projekten können die Ergebnisse verwendet werden, um festzustellen, ob das System die Performance-Vorgaben einhält.

Tabelle 3: Vorgehensmodell für Performance-Tests¹⁰⁸

Innerhalb von Schritt eins werden einzelne Szenarien aufgebaut, welche getestet werden sollen. Hier werden die Rahmenbedingungen angegeben, unter denen die einzelnen Szenarien durchgeführt werden sowie die groben Schritte beschrieben, aus denen die einzelnen Testszenarien bestehen. Weiterhin werden hier die zu testenden Funktionalitäten ausgewählt. Bei der Auswahl dieser Funktionalitäten sollen speziell die Teile gewählt werden, bei denen Einbrüche in der Performance oder Fehler in der Software durch eine hohe Belastung zu erwarten sind. Weitere zu testende Teile sind die Funktionen, welche kritisch für die Verwendung des Systems sind. Für das in dieser Arbeit zu testende System bedeutet dies zunächst die Definition eines DB-Schemas, welches für einen Test des verteilten DBS verwendet werden soll, sowie die Auswahl der Operationen (Datenänderungen, Synchronisation von neuen Daten o. ä.), deren

¹⁰⁷ Vergl. Borg, Ingwer et al. (2007), S. 313ff; Malle, H. (2006), Teil 3, S. 10 und Bronstein, I.N. et al. (2008), S. 853

¹⁰⁸ Vergl. Meier, J.D. et al. (2006), S.745ff und S. 752 ff. und Ford, Chris et al. (2008), S. 151ff

Verhalten unter Last untersucht werden soll. Das DB-Schema bildet dabei die Grundlage für alle Szenarien. Aus den einzelnen Operationen werden dann Testszenarien definiert.¹⁰⁹

Schritt zwei besteht aus der Definition der Arbeitslast, unter der getestet wird. Innerhalb des in dieser Arbeit aufzubauenden Tests stellt die Arbeitslast Datenmengen dar, mit der das mobile DBS und die Synchronisation belastet werden. Zunächst stellt sich bei der Definition der Arbeitslast die Frage, wie die Daten für den Test beschafft werden. Ein Vorgehen für den Test eines DBS besteht aus der Verwendung von zufällig generierten Werten, die in die Datenbank eingefügt werden. Damit wird der Aufbau eines Tests erleichtert, da die Daten nicht per Hand eingegeben werden müssen, sondern automatisiert in die Datenbank eingefügt werden können. Eine andere Möglichkeit stellt die Verwendung von Realdaten dar, falls diese für das zu testende System vorhanden sind. Um das Verhalten des Gesamtsystems unter zunehmender Belastung zu untersuchen, werden Laststufen benötigt, auf deren Grundlage gemessen wird. Ein in der Praxis geläufiger Ansatz stellt die Erhöhung der Last um einen fest definierten Wert dar. Bei einem Lasttest z. B. kann leicht unter der Last angefangen werden, für die das System gedacht ist und dann linear erhöht werden.¹¹⁰

Die Erhöhung der Datenmenge kann auch anders vorgenommen werden, je nach Anforderungen an den Test.¹¹¹ Um das Verhalten des Systems auch unter geringer und mittlerer Last zu ermitteln, kann bei einer kleinen Startmenge angefangen werden. Damit wird eine Abschätzung möglich, wie sich die Performance auch oberhalb der maximal getesteten Last verhält. Innerhalb dieser Arbeit soll gezeigt werden, wie sich die zu testenden Teile unter niedrigen, mittleren und hohen Belastungen verhält. Dabei stellt die Verwendung einer linearen Erhöhung eine ungünstige Wahl dar, da für kleine Datenmengen eine geringe Schrittweite benötigt wird, während bei höheren Datenmengen eine solche Schrittweite einerseits eine Genauigkeit liefert, die nicht benötigt wird und andererseits zu einer hohen Laufzeit des Gesamttests führt. Eine Alternative stellt die Verwendung einer logarithmischen Schrittweite der Datenerhöhung dar. Diese kann mit einer linearen Schrittweite kombiniert werden. Bei diesem Vorgehen werden logarithmisch Laststufen ausgewählt. Ausgehend von diesen

109 Vergl. Ford, Chris et al. (2008), S. 19f; Meier, J.D. et al. (2006), S. 752 ff. und Beydeda, Sami (Hrsg.) et al. (2004), S. 79

110 Vergl. Bowers, David (Hrsg.) (1994), S. 114f und S. 117f, Vossen, Gottfried (2008), S. 58f; Gulutzan, Peter et al. (2003), S. 7f sowie Ford, Chris et al. (2008), S. 20

111 Vergl. Ford, Chris et al. (2008), S. 19f

Laststufen können, mithilfe einer linearen Erhöhung, mehrere Messpunkte pro Laststufe gewählt werden. Formel 5.2.1 zeigt dieses Vorgehen in einem Beispiel.

$$D_1=10, D_2=20, \dots, D_n=90, D_{n+1}=100, D_{n+2}=200, \dots, D_N$$

*Formel 5.2.1: Beispiel
einer Lasterhöhung*

Damit werden Messungen durchgeführt, die das Verhalten des Systems unter verschiedenen hohen Lasten zeigt. Um weiterhin einen automatisierten Test zu ermöglichen, wird eine Maximaldatenmenge benötigt, die als Abbruchbedingung verwendet werden kann. Diese Grenze kann so gewählt werden, dass ein Trend im Verhalten der Laufzeit in Abhängigkeit von der Arbeitslast erkennbar wird. Weiterhin muss die lineare Schrittweise für einen automatisierten Test festgelegt werden, die pro logarithmische Laststufe verwendet wird. Je nachdem, wie diese gewählt wird, ergibt sich ein entsprechend detailliertes Ergebnis.

Innerhalb von Schritt 3 müssen die Metriken festgelegt werden, die getestet werden sollen. Einige gängige Metriken sind in Tabelle 4 dargestellt. Diese zeigen verschiedene Aspekte der Performance eines Systems oder einer Software.¹¹²

Metrik	Beschreibung
Verarbeitungszeit	Dauer von Anfragen an ein System, Dauer einer Funktionsdurchführung.
Durchsatz	Gibt die Rate von Aktionen pro Zeiteinheit an, wie z. B. Anzahl Transaktionen pro Sekunde.
Verfügbarkeit	Die Zeitdauer, in der ein Dienst oder System durchschnittlich ohne Unterbrechung (z. B. wegen Wartung) verfügbar ist.
Verlässlichkeit	Gibt an, wie verlässlich das System unter einer bestimmten Last ist, gemessen in der Wahrscheinlichkeit von Fehlern bzw. der Zeit zwischen zwei Fehlern.

Tabelle 4: Metriken für Performance-Tests¹¹³

Der nächste Schritt stellt die Ausarbeitung von detaillierten Testfällen dar. Auf Grundlage der Szenarien, Arbeitslast und der zur testenden Metrik werden die einzelnen Testfälle aufgebaut. Diese stellen in dieser Arbeit konkrete Datenbankabfragen an das mobile DBS sowie die Synchronisationsoperationen dar, die mithilfe verschiedener Netzwerktechnologien ausgeführt werden sollen. Zu den Testfällen gehört weiterhin die Beschreibung des erwartenden Verhalten des Systems, bzw. der Software, für den jeweiligen Testfall, um Anomalien vom normalen Verhalten unterscheiden zu können. Somit stellen die Testfälle konkrete Beschreibungen dar, welche Operationen pro Messung durchgeführt werden, unter welchen Bedingungen

¹¹² Vergl. Beydeda, Sani (Hrsg.) (2004), S. 87f und Jain, Raji (1991), S. 37f

¹¹³ Vergl. Beydeda, Sani (Hrsg.) (2004), S. 87f und Jain, Raji (1991), S. 37ff

diese durchgeführt werden und wie ein korrektes Ergebnis aussehen soll. Innerhalb eines Projektes stellt dieser Schritt einen Teil der Planung dar, damit vor der Realisierung klar ist, welche Teile wie getestet werden. Damit wird verhindert, dass die Implementierung eines Projektes den Aufbau der Tests beeinflusst. Innerhalb eines normalen Projektes gehören zu den Anforderungen auch Aussagen zum Performanceverhalten des Systems, wie z. B. die maximale Reaktionszeit. Innerhalb dieser Arbeit werden keine solchen Vorgaben gemacht, da hier gezeigt werden soll, wie das Performanceverhalten der zu testenden Systemteile ist. Reale Vorgaben, die als maximale Grenze dienen, gibt es nicht.¹¹⁴

Nach der Definition der Szenarien, der Arbeitslast sowie der einzelnen Testfälle werden diese in einer Testumgebung ausgeführt. Beim Aufbau einer Testumgebung muss auf eine Isolierung der zu testenden Software oder des zu testenden Systems vor äußeren Einflüssen geachtet werden, um eine verlässliche Aussage zur Performance der Software zu treffen. Dies bedeutet z. B. das Beenden aller unnötigen Prozesse auf den beteiligten Geräten. Damit wird die Performance des Gerätes lediglich durch benötigte Prozesse sowie dem Testsystem belastet. Dadurch können Messungenauigkeiten reduziert werden.

Trotz der Isolierung werden Messungen von verschiedenen, nicht beeinflussbaren Faktoren, wie z. B. unterschiedliche Laufzeiten der Netzwerkkommunikation oder Festplattenzugriffe, beeinflusst. Wiederholungen der Messungen können diese Schwankungen ausgleichen. Solche wiederholten Messungen werden in dieser Arbeit als *Messreihe* bezeichnet. Aufgrund solcher Abweichungen stellt ein aus den Messungen gebildeter Mittelwert eine Annäherung an ein reales Verhalten dar.¹¹⁵

Um Messergebnisse einer bestimmten Güte zu erhalten, gibt es verschiedene Methoden. Innerhalb dieser Arbeit wird eine Überprüfung der Güte nach einem in der ISO 14756-Norm für Performance-Tests definierten Vorgehen durchgeführt werden. Dort wird eine mathematische Verteilung zur Überprüfung einer Messreihe verwendet werden. Eine Verteilung zeigt im Fall des in dieser Arbeit durchgeführten Tests, mit welcher Wahrscheinlichkeit sich die Messergebnisse in einem bestimmten Bereich um den realen Wert befinden. Innerhalb der Berechnungen zur Überprüfung, ob die Ergebnisse nicht zu stark durch äußere Einflüsse verzerrt worden sind, wird ein *Quantil* benötigt.

114 Vergl. Borg, Ingwer et al. (2007), S. 313ff und Meier, J.D. et al. (2006), S. 755 f.

115 Vergl. Malle, H. (2006), Teil 3, S. 10 und Bronstein, I.N. et al. (2008), S. 853

Dies bezeichnet die Umkehrfunktion zur Verteilung.¹¹⁶ Um die Güte einer Messreihe zu bestimmen, wird eine Verteilung der Messreihe angenommen, da die reale Verteilung unbekannt ist.¹¹⁷

Innerhalb der ISO-Norm wird die Normalverteilung $\Phi(0,1)$ für die realen Laufzeiten und die Messwerte angenommen. Weiterhin wird eine Irrtumswahrscheinlichkeit α festgelegt (in der Praxis normalerweise zwischen 5 % und 10 %, Werte über 20 % lassen eine zu hohe Irrtumswahrscheinlichkeit zu), mit der die Überprüfung falsch liegen kann, sowie ein Wert d , mit dem ein Intervall um den Mittelwert T_{me} der Messwerte gebildet wird (das Intervall berechnet sich aus $T_{me} \pm d$). Für d hat sich gezeigt, dass die Definition $d = b \cdot (\text{Mittelwert der Messreihe})$ gute Ergebnisse liefert, wobei b zwischen 0,05 und 0,1, max. jedoch 0,2 gewählt werden sollte (ansonsten wird die Überprüfung zu ungenau). Der Test prüft mit einer Irrtumswahrscheinlichkeit α , ob sich die durchschnittliche reale Laufzeit im definierten Intervall um die gemessene durchschnittliche Laufzeit befindet. Für diese Überprüfung wird das so genannte $u_{1-\alpha/2}$ -Quantil der Normalverteilung $\Phi(0,1)$ benötigt. Dies bedeutet die Verwendung des Quantils der Normalverteilung, welches sich aus der Wahrscheinlichkeit $1-\alpha/2$ ergibt. Basierend auf diesen Parametern kann überprüft werden, ob die Anzahl der Wiederholungen ausreicht, um einen aussagekräftigen Mittelwert zu erhalten, oder ob die Messungen durch äußere Einflüsse zu stark verzerrt worden sind.¹¹⁸

Die genauen Berechnungen, die in dem Test implementiert werden, sind in Anhang 1 dargestellt. Das Quantil der Normalverteilung zu einer Wahrscheinlichkeit α kann aus Anhang 2 entnommen werden.

Ein weiterer Schritt bei der Testdurchführung stellen vorbereitende Läufe des Systems dar. Je nach zu testendem System benötigt dieses eine gewisse Zeit, bis z. B. Puffer aufgebaut sind und sämtliche Dienste geladen worden sind.¹¹⁹ Um das mobile DBS sowie die Synchronisation entsprechend vorzubereiten, soll in dieser Arbeit zunächst ein Testlauf unter geringer Last durchgeführt werden, welcher nicht als Messreihe mit aufgenommen wird. Danach wird mit dem eigentlichen Test begonnen.

¹¹⁶ Vergl. Mosler, Karl et al. (2006), S. 105

¹¹⁷ Vergl. Dirlewanger, Werner (2006), S. 34f und S. 55ff

¹¹⁸ Z. B. zu entnehmen aus Graf, Ulrich et al. (1998), S. 458 oder Bronstein, I.N. et al (2008), S. 1143f., wobei bei der letzteren Quelle die Wahrscheinlichkeit genommen werden muss, die die gewünschte Toleranz gerade überschreitet, vergl. für $u_{1-\alpha/2}$ auch Graf, Ulrich et al. (1998), S. 112

¹¹⁹ Vergl. Dirlewanger, Werner (2006), S. 42f

Der letzte Schritt besteht in der Darstellung und Analyse der Messergebnisse. Die in den einzelnen Testfällen ermittelten Werte, bestehend aus den Wertepaaren Laufzeit und Datenmenge, können in einem Kurvendiagramm abgebildet werden. Aus diesem Diagramm kann man ermitteln, für welche Performanceanforderungen das getestete System geeignet ist.

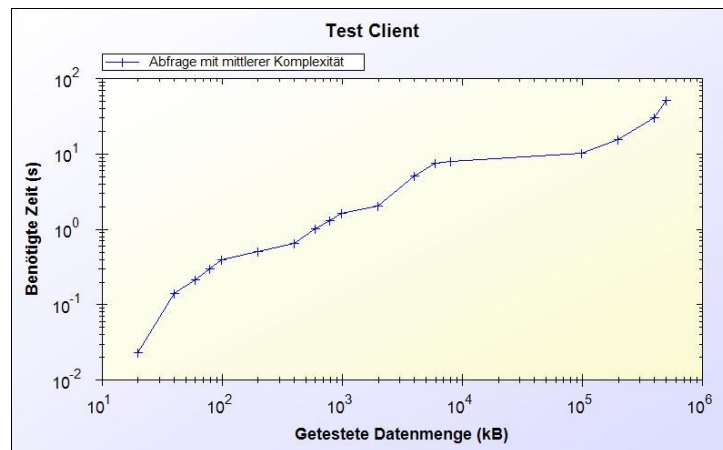


Abbildung 8: Beispieldiagramm

Abbildung 8 stellt für dieses Vorgehen ein Beispieldiagramm dar. Bei der Erstellung der Diagramme ist eine Auswahl von aussagekräftigen Achseneinheiten wichtig (z. B. Anzahl Datensätze für die Datenlast). Innerhalb dieser Arbeit ist die logarithmische Einteilung der Achsen zu beachten. Im folgenden wird ein Testsystem aufgebaut, welches zeigen soll, für welche Anforderungen und Lasten die Synchronisationen über Funktechnologien sowie mobile DBS geeignet sind.

6. Durchführung eines Performance-Tests eines mobilen Datenbanksystems und einer Synchronisation

Innerhalb dieses Kapitels wird durch die Realisierung eines Performance-Tests, bestehend aus mehreren Schlüsselszenarien, gezeigt werden, wie stark sich die Einflüsse der mobilen Umgebung auswirken. Dafür wird eine Testapplikation bereitgestellt. Der Aufbau dieser Applikation wird ebenfalls dargestellt.

6.1. Einführung

Für den Performance-Test wird eine Testumgebung benötigt, die ein mobiles Gerät mit einem mobilen DBS, einem zentralen DBS sowie einem Replikationsserver, der den Datenabgleich zwischen mobilen DBS und zentralem DBS übernimmt, enthält. Um ein solches System automatisiert testen zu können, wird eine verteilte Applikation entwickelt, welche die Durchführung vorher definierter Testfälle ermöglicht sowie zusätzliche Dienste bereitstellt. Dazu gehört z. B. das Herstellen eines Testzustandes auf der Server- und Clientseite durch das Beenden von unnötigen Prozessen.

Um die Testapplikation aufzubauen, wird festgelegt, welche Szenarien sowie welche Arbeitslast und Metriken innerhalb des Tests verwendet werden. Dies entspricht dem Vorgehensmodell aus Kapitel 5. Auf Basis dieser Vorgaben werden Testfälle definiert, welche von der Testapplikation durchgeführt werden. Abgeschlossen wird mit einer Darstellung der Messergebnisse.

6.2. Hardware- und Softwareumgebung

Für den Aufbau des in den vorhergehenden Kapiteln beschriebenen Gesamtsystems wird ein PDA als mobiler Client und ein Serversystem verwendet, welches die Replikationsdienste und das zentrale DBS zur Verfügung stellt. Weiterhin werden im zentralen DBS die Messergebnisse gespeichert und von einer Applikation ausgewertet. Dies geschieht unabhängig von den Relationen, die für den Synchronisationstest benötigt werden. Die eigentliche Steuerung der Tests wird durch eine mobile Anwendung auf dem PDA realisiert. Um eine Steuerung des Servers zu ermöglichen, werden über Web Services entsprechende Dienste auf dem Server bereitgestellt. Dazu gehören zum einen Schnittstellen zur Vorbereitung und Durchführung der Synchronisation und zum anderen Schnittstellen für die Übertragung von Messergebnissen. Innerhalb dieser Arbeit wird der PDA „T-Mobile Ameo“ verwendet. Die Daten des Gerätes sind in Tabelle 5 dargestellt.

Eigenschaft	Beschreibung
Modell	T-Mobile Ameo
Prozessor	Intel ® PXA 270 624 MHz
Speicher	256 MB ROM sowie 128 MB SDRAM
Festplatte	8 GB
Funkmodul	UMTS, GSM, GPRS, EDGE
Weitere drahtlose Zugänge	Bluetooth, WLAN
Betriebssystem	Microsoft Windows Mobile 6
Mobiles DBS	Microsoft SQL Server 2005 CE V. 3.5

Tabelle 5: Daten des mobilen Gerätes¹²⁰

Dieser PDA stellt ein mächtiges Endgerät dar, welcher genügend Speicherplatz und Rechenkraft hat, um ein mobiles DBS wie den SQL Server CE zu betreiben. Somit sind die Tests nicht aussagekräftig für einfachere Geräte wie kleinere Mobiltelefone o. ä.

Tabelle 6 beschreibt die Daten des Servers. Wie beim mobilen Gerät sind die Testergebnisse lediglich für ähnliche Computersysteme aussagekräftig. Der Server verwendet dabei die Version 6 der Internet Information Services (IIS), um die Web Services sowie die Replikationsdienste bereitzustellen. Als IIS wird das Webserverssystem von Microsoft bezeichnet.

Eigenschaft	Beschreibung
Prozessor	Intel® 3.00 GHz
Speicher	1,5 GB RAM
Betriebssystem	Microsoft Windows Server 2003 Enterprise Edition
DBS	Microsoft SQL Server 2005
Webserver	Microsoft IIS V. 6

Tabelle 6: Daten des Servers

Um die verschiedenen Komponenten zu entwickeln, kommt das Microsoft .Net Framework 2.0 für die Serverseite zum Einsatz. Passend dazu wird auf der Seite des mobilen Clients das Microsoft .Net Compact Framework 2.0 verwendet. Die Entwicklung wird in der .Net-Sprache C# durchgeführt.

6.3. Abgrenzungen

Dieses Kapitel stellt die Funktionalitäten und Tests dar, die über die in dieser Arbeit aufgebauten Testsystem sowie Testszenarien hinausgehen und gibt weiterhin einen Ausblick, welche Erweiterungen zukünftig an der Testapplikation vorgenommen werden können.

¹²⁰ Vergl. T-Mobile (2009), S. 327f

Innerhalb des in dieser Arbeit aufgebauten Performance-Tests wird eine bidirektionale Synchronisation getestet (Vergl. dazu Kapitel 4.3.1). Eine zukünftige Erweiterung wäre die Aufnahme von Szenarien, welche auf anderen Arten von Synchronisationen basieren. Zum Beispiel kann mithilfe des Tests einer unidirektionalen Synchronisation gezeigt werden, in wie weit sich die Performance der Synchronisation verändert, wenn lediglich Änderungen von einer Seite des Systems synchronisiert werden müssen.

Aufgrund des ausgewählten Aufbaus des verteilten DBS stellen verteilte und mobile Transaktionen ebenfalls keinen Teil des Testsystems dar. Der Test von verteilten Anfragen und verteilten sowie mobilen Transaktionen und deren Verhalten unter Last wäre für die Betrachtung von Thin-Clients oder bestimmten Modellen von verteilten DBS interessant.

Weiterhin wird in dieser Arbeit genau ein mobiles Gerät verwendet, um zu testen. Für die Performance der Synchronisation wäre die Verwendung von mehreren mobilen Geräten interessant, um herauszufinden, wie viele Clients das Serversystem bewältigen kann.

6.4. Schlüsselszenarien

In diesem Kapitel werden die Schlüsselszenarien für die Performance-Tests dargestellt. Als zentrale Grundlage für alle Szenarien wird zunächst ein Datenbankschema definiert, nach der die zu testende verteilte Datenbank aufgebaut wird. Dabei stellen die folgenden Punkte Grundüberlegungen für den Aufbau dar.

- 1) Größe der Datensätze: Die Erhöhung der Datenmenge basiert auf dem Einfügen von Datensätzen, die sich aus den Relationen und ihren Attributen ergibt. Da für die Übertragung von Daten vom Server zum Client und zurück die Größe der Datenmengen in Byte interessant ist, wird in diesem Kapitel zusätzlich zum Aufbau der Datenbank die Größe der Attribute angegeben. Innerhalb der Definition der Arbeitslast stellen diese Größen die Grundlage für die Berechnung der zu übertragenden Datenmengen dar.
- 2) Aufteilung der Daten in mehrere Tabellen: Um Abfragen, basierend auf mehreren Tabellen, zu ermöglichen, sollen über Fremdschlüsselbedingungen verbundene Relationen entstehen.
- 3) Die einzelnen Relationen sehen einen Erstellungszeitpunkt für die einzelnen Datensätze vor. Datensätze, die für eine Messung der Synchronisation eingefügt worden sind, können damit von anderen Datensätzen unterschieden und gezielt

gelöscht werden, um den ursprünglichen Zustand herzustellen. Damit können die einzelnen Messungen der Synchronisation wiederholt werden, um eine höhere Genauigkeit der Ergebnisse zu erreichen.

- 4) Um eine Erhöhung der Datenmenge im Gesamtsystem durch die Aktualisierung von Daten zu ermöglichen, werden die einzelnen Relationen ferner mit einem Änderungszeitpunkt versehen. Dadurch kann man verhindern, dass bereits aktualisierte Datensätze erneut überschrieben werden. Dies ist wichtig, damit die Menge an Datensätzen, die geändert werden, berechnet werden kann.¹²¹

Das Datenbankmodell basiert neben den obigen Grundüberlegungen auf dem Beispiel aus Kapitel 2.2, bei denen Vertriebsmitarbeiter von mobilen Geräten bei Verkaufsprozessen unterstützt werden. Dafür wird ein Datenbankschema aufgebaut, welches als Basis für ein einfaches Bestellsystem dienen kann. Dieses Schema besteht aus Kunden, zu denen Aufträge mit verschiedenen Positionen erstellt werden können. Die Positionen geben dabei eine Menge an und verweisen auf einen Artikel. Dabei übernehmen die Positionen den Preis aus dem Artikel, falls dieser sich im Artikel nach der Bestellung ändern sollte. Eine Position stellt dabei eine Verbindung aus genau einem Auftrag und einem Artikel dar. Der Primärschlüssel der Position setzt sich somit aus dem Artikel- und Auftragsschlüssel zusammen. Dadurch kann ein einzelner Artikel mehreren Aufträgen zugeordnet werden sowie ein Auftrag mehrere Artikel enthalten. Um weitere Einschränkungen bzw. komplexere Abfragen innerhalb der Testfälle möglich zu machen, gibt es Produktarten sowie davon abhängige Kollektionen. Jeder Artikel soll einer Kollektion zugeordnet sein. Dies führt zu dem in Abbildung 9 dargestellten Datenbankmodell. Dieses Modell stellt die Grundlage für die einzelnen Szenarien dar. Für eine vollständige Datendefinition, siehe Anhang 3. Dieses Datenbankschema soll auf Server- und Clientseite verwendet werden. Somit stellt die verteilte Datenbank eine homogen verteilte Datenbank dar (Vergl. Kapitel 4.3.1).

¹²¹ Vergl. auch Kapitel 6.5 für die Abläufe der Aktualisierung

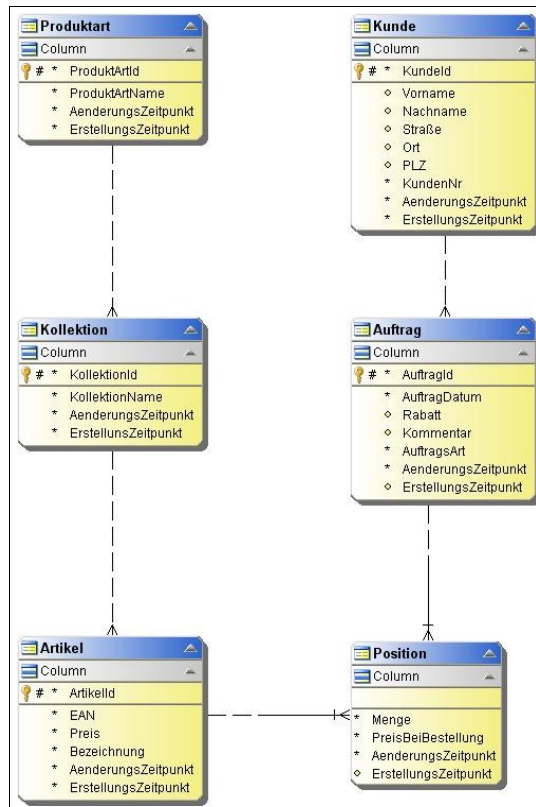


Abbildung 9: Datenbankmodell für den Performance-Test

Tabelle 7 gibt an, welche Größen die Spalten einer Relation haben.

Tabelle	Spalte	Datentyp	Größe
Kunde	KundeId	GUID	16 Byte
	Vorname	Varchar	2 Byte pro Stelle
	Nachname	Varchar	2 Byte pro Stelle
	Straße	Varchar	2 Byte pro Stelle
	Ort	Varchar	2 Byte pro Stelle
	PLZ	Char(10)	10 Byte
	KundenNr	Char(20)	20 Byte
	ErstellungsZeitpunkt	Datetime	8 Byte
	ÄnderungsZeitpunkt	Datetime	8 Byte
Auftrag	AuftragId	GUID	16 Byte
	KundeId	GUID	16 Byte
	AuftragDatum	Datetime	8 Byte
	Rabatt	Smallint	2 Byte
	Kommentar	Varchar	2 Byte pro Stelle
	AuftragsArt	Char(4)	4 Byte
	ErstellungsZeitpunkt	Datetime	8 Byte
	ÄnderungsZeitpunkt	Datetime	8 Byte
Position	AuftragId	GUID	16 Byte

Tabelle	Spalte	Datentyp	Größe
	ArtikelId	GUID	16 Byte
	Menge	Integer	4 Byte
	PreisBeiBestellung	Money	8 Byte
	ErstellungsZeitpunkt	Datetime	8 Byte
	ÄnderungsZeitpunkt	Datetime	8 Byte
Artikel	ArtikelId	GUID	16 Byte
	EAN	Char(26)	26 Byte
	Preis	Money	8 Byte
	Bezeichnung	Varchar	2 Byte pro Stelle
	KollektionID	GUID	16 Byte
	ErstellungsZeitpunkt	Datetime	8 Byte
	ÄnderungsZeitpunkt	Datetime	8 Byte
Kollektion	KollektionID	GUID	16 Byte
	KollektionName	Char(20)	20 Byte
	ErstellungsZeitpunkt	Datetime	8 Byte
	ProduktartId	GUID	16 Byte
	ÄnderungsZeitpunkt	Datetime	8 Byte
Produktart	ProduktartId	GUID	16 Byte
	ProduktartName	Char(20)	20 Byte
	ErstellungsZeitpunkt	Datetime	8 Byte
	ÄnderungsZeitpunkt	Datetime	8 Byte

Tabelle 7: Spaltendefinition für die Testdatenbank¹²²

Auf Grundlage dieser DB-Struktur werden verschiedene Testszenarien entwickelt.

Das erste Szenario besteht aus der Synchronisation von Daten. Um die Synchronisation zu testen, wird zunächst eine Datenmenge in das verteilte DBS eingefügt. Damit ist ein Datenbestand gegeben, mit der die Synchronisation neue Datensätze abgleicht. Weiterhin können diese Datensätze verwendet werden, um einen Teil der Last für die Synchronisation durch Aktualisierungen zu erzeugen. Dies stellt ein normaler Zustand für eine Replikation dar, da sich im Normalfall bereits synchronisierte Daten im Gesamtsystem befinden. Um zu zeigen, in wie weit diese Basisdatenmenge sich auf die Performance der Synchronisation auswirkt, werden in Kapitel 6.7.1 Testfälle mit unterschiedlichen Basisdatenmengen definiert. Nach der Erstellung der Basisdatenmenge wird eine Menge an neuen Daten in die mobile Datenbank und das zentrale System eingefügt sowie eine Menge an Daten auf Client- und Serverseite verändert. Darauf basierend wird die Laufzeit der Synchronisierung dieser neuen bzw.

¹²² Vergl. Microsoft (2008 a) für die jeweiligen Größen.

veränderten Daten gemessen. Um Wiederholungen für die Erzeugung von Messreihen zu ermöglichen, wird nach einer Messung der Synchronisierung die neu erstellten Datensätze wieder gelöscht und somit der Grundzustand wiederhergestellt. Damit werden Wiederholungen der Messungen unter denselben Voraussetzungen ermöglicht. Weiterhin wird davon ausgegangen, dass alle Clients alle Daten sehen dürfen und somit alle Daten synchronisiert werden müssen.

Das zweite Szenario ist das Durchführen von Anfragen an das mobile DBS, welche Daten ermitteln sollen (*Abfragen*). Dabei sollen Testfälle mit verschiedenen komplexe Abfragen aufgebaut werden. Damit soll gezeigt werden, wie sich die Komplexität der Abfragen auf die Laufzeiten auswirkt und bis zu welcher Datenmenge welche Komplexität möglich ist.

Szenario drei besteht aus der Manipulation von Daten in der mobilen Datenbank unter zunehmender Last. Dazu zählt das Einfügen, Löschen und Aktualisieren von Daten. Da bei solchen Operationen im Normalfall einzelne Zeilen betroffen sind, sollen die Testfälle des Szenario 3 mit einzelnen Datensätzen durchgeführt werden. Tabelle 8 fasst die einzelnen Szenarien zusammen. Vergleiche für Szenario zwei und drei auch Kapitel 3.5. Da in dieser Arbeit ein Lasttest durchgeführt werden soll, werden die einzelnen Szenarien unter zunehmender Last durchgeführt.

Name des Szenarios	Kurzbeschreibung
Szenario 1	Das Synchronisieren von Datenänderungen sowie neuen Daten.
Szenario 2	Abfragen von Datenmengen aus der mobilen Datenbank
Szenario 3	Datenmanipulationen des Datenbestands in der mobilen Datenbank

Tabelle 8: Szenarien

Der nächste Schritt stellt die Definition der Arbeitslast dar. Darin wird festgelegt, wie die Datenmengen aufgebaut werden, die für die Synchronisation benötigt werden, sowie aus denen die Datenbasis für die einzelnen Anfragen in Szenario zwei und drei aufgebaut werden. Weiterhin wird die Erhöhung der Last dargestellt. Im weiteren Verlauf der Arbeit wird auf die Namen der Szenarien in Tabelle 8 verwiesen.

6.5. Arbeitslast

Dieses Kapitel definiert die Arbeitslast des Performance-Tests für die Synchronisation und dem mobile Datenbanksystem. Die Last besteht aus Datenmengen, die für den Test des DBS in die Datenbank eingefügt werden und die die Grundlage der Anfragen an das DBS stellen bzw. aus Datenmengen, welche bei der Synchronisation übertragen werden.

Um die Arbeitslast zu modellieren, kann aus der Schemadefinition der Datenbank die Größen einzelner Testdatensätze für die einzelnen Relationen ermittelt werden. Um die Datenmenge kontrolliert erhöhen zu können und um die Komplexität der Generierung der Daten zu verringern, wird eine Grundmenge an Datensätzen definiert, die jeweils zusammen als Datenblock in die Datenbank eingefügt werden. Tabelle 9 stellt die Anzahl der Datensätze sowie ihre Größe in Byte dar, die in einem solchen Datenblock enthalten sind. Innerhalb der Berechnungen der Größe des Blocks ist das Änderungsdatum nicht mit eingerechnet, da dieses Feld bei neuen Datensätzen nicht gefüllt sein soll. Die Abkürzung Ds. in der Tabelle steht für **Datensatz** bzw. für **Datensätze**.

Tabelle 9 ist wie folgt zu lesen: Pro Relation, mit Ausnahme der Position, Produktart und Kollektion, gibt es mehrere Zeilen mit Datensätzen. Für diese Datensätze werden die Größen für Felder mit einer variablen Länge festgelegt (in dem in dieser Arbeit verwendeten Datenbankschema lediglich Werte vom Datentyp Varchar, also alphanumerische Werte variabler Länge). Die Anzahl der Zeichen wird mit zwei multipliziert. Die Multiplikationen begründen sich in dem Datentyp Varchar, der im Microsoft SQL Server 2005 CE und Microsoft SQL Server 2005 pro Zeichen (mit Z. in der Tabelle abgekürzt) zwei Byte Speicherplatz benötigt. Pro Datensatz wird dessen Gesamtgröße angegeben. Innerhalb der letzten Spalte wird die Größe aller Zeilen der jeweiligen Tabelle in einem Datenblock angegeben. Da die Relationen Position, Kollektion und Produktart keine variablen Datentypen verwenden, wird bei ihnen direkt die Gesamtgröße aus der Anzahl der Datensätze, welche in diese Relation pro Block eingefügt werden, sowie der Größe eines Datensatzes in Byte berechnet.

Wie aus dem Datenbankschema in Abbildung 9 entnommen werden kann, stellen die variablen Felder die einzigen Attribute dar, bei denen der Wert NULL eingefügt werden darf. Dies bedeutet, für dieses Attribut der Relation wird zugelassen, dass kein Wert angegeben wird. Dementsprechend wird an dieser Stelle kein Speicherplatz mit Nutzdaten belegt. Auf der anderen Seite müssen somit alle Attribute, welche keinen variablen Datentyp haben, gefüllt werden und können als fester Wert in die Berechnung der Datensätze aufgenommen werden. Einzige Ausnahme ist das Änderungsdatum, welches erst bei einer Änderung (dazu zählt nicht die Erstellung) gefüllt wird.

Welches variable Attribut einer Relation mit dem jeweiligen berechneten Werten gefüllt wird, ist lediglich für die Testdatensätze relevant, nach denen innerhalb der

Abfragetestfälle gesucht wird. Dort müssen die jeweiligen Kriterien, nach denen gesucht werden, erfüllt werden.

Aus der Größenberechnung in Tabelle 9 ergibt sich ein 5 kB-Block bzw. eine Bündelung von 50 Datensätzen, die für den Test zusammen eingefügt werden. Die Betrachtung der Anzahl von Datensätzen, auf denen gearbeitet wird, ist speziell für den Test des mobilen Datenbanksystems interessant. Da relationale Datenbanken mit einer logischen Sicht auf Basis von Datensätzen in Tabellen arbeiten, ist die Performance des Datenbanksystems abhängig von der Menge der Zeilen. Die Anzahl der Datensätze bestimmt z. B. die Anzahl der für Sortierungen notwendigen Operationen oder die Anzahl von Vergleichen für Verbindungen.

Für die Synchronisation sind beide Größen wichtig. Die Anzahl der Datensätze spielen eine Rolle, wenn die Änderungen auf dem Server und dem Client verglichen werden, während die reine Datenübertragung von der Größe der zu übertragenden Daten abhängig ist. Da in dieser Arbeit speziell die Funktechnologien und somit die Datenübertragung betrachtet werden sollen, wird für die Synchronisation die Datenmenge in Byte betrachtet. Über die Berechnungen in Tabelle 9 kann berechnet werden, wie viele Datensätze pro Datenmenge eingefügt werden. Somit können beide Größen betrachtet werden.

Tabelle	Nr. des Ds. oder Anzahl Ds.	Größenberechnung der Felder	Gesamtgröße Datensatz/ Datensätze	Gesamtgröße Relation
Kunde	DS. 1	18 Z. * 2 = 36 B	184 B	1654 B
		18 Z. * 2 = 36 B		
		20 Z. * 2 = 40 B		
		9 Z. * 2 = 18 B		
		58 B		
	DS. 2	10 Z. * 2 = 20 B	232 B	
		27 Z. * 2 = 54 B		
		20 Z. * 2 = 40 B		
		32 Z. * 2 = 64 B		
		54 B		
	DS. 3	16 Z. * 2 = 32 B	134 B	
		NULL = 0 B		
		10 Z. * 2 = 20 B		
		14 Z. * 2 = 28 B		
		54 B		
	DS. 4	8 Z. * 2 = 16 B		
		NULL = 0 B		

Tabelle	Nr. des Ds. oder Anzahl Ds.	Größenberechnung der Felder	Gesamtgröße Datensatz/ Datensätze	Gesamtgröße Relation
		8 Z. * 2 = 16 B	128 B	
		21 Z. * 2 = 42 B		
		54 B		
	DS. 5	56 Z. * 2 = 112 B	270 B	
		15 Z. * 2 = 30 B		
		19 Z. * 2 = 38 B		
		18 Z. * 2 = 36 B		
		54 B		
	DS. 6	49 Z. * 2 = 98 B	360 B	
		17 Z. * 2 = 34 B		
		36 Z. * 2 = 72 B		
		51 Z. * 2 = 102 B		
		54 B		
	DS. 7	26 Z. * 2 = 52 B	202 B	
		NULL = 0 B		
		NULL = 0 B		
		52 Z. * 2 = 104 B		
		54 B		
	DS. 8	26 Z. * 2 = 52 B	144 B	
		NULL = 0 B		
		19 Z. * 2 = 38 B		
		NULL = 0 B		
		54 B		
Auftrag	DS. 1	NULL = 0 B	54 B	518 B
		54 B		
	DS. 2	36 Z. * 2 = 72 B	126 B	
		54 B		
	DS. 3	24 Z. * 2 = 48 B	102 B	
		54 B		
	DS. 4	64 Z. * 2 = 128 B	182 B	
		54 B		
	DS. 5	NULL = 0 B	54 B	
		54 B		
Artikel	DS. 1	20 Z. * 2 = 40 B	114 B	1352 B
		74 B		
	DS. 2	12 Z. * 2 = 24 B	98 B	
		74 B		
	DS. 3	17 Z. * 2 = 34 B	108 B	
		74 B		
	DS. 4	NULL = 0 B	74 B	
		74 B		

Tabelle	Nr. des Ds. oder Anzahl Ds.	Größenberechnung der Felder	Gesamtgröße Datensatz/ Datensätze	Gesamtgröße Relation
	DS. 5	NULL = 0 B	106 B	
		74 B		
	DS. 6	16 Z. * 2 = 32 B	106 B	
		74 B		
	DS. 7	15 Z. * 2 = 30 B	104 B	
		74 B		
	DS. 8	30 Z. * 2 = 60 B	134 B	
		74 B		
Position	12 DS.	12 Ds. * 52	-	624 B
Produktart	3 DS.	3 Ds. * 44	-	132 B
Kollektion	14 DS.	14 Ds. * 60	-	840 B
Gesamtgröße: 5120 Bytes				

Tabelle 9: Berechnung der Größe der Datenblöcke für das Erhöhen der Last (alle Szenarien)¹²³

Die Datenblöcke aus Tabelle 9 werden für *sämtliche Szenarien* verwendet. Die reale Datenmenge, die in die zentrale und mobile Datenbank eingefügt wird, wird weiterhin durch einen Overhead für die Verwaltung der Datensätze im SQL Server sowie durch die Replikationstechniken des SQL Servers erhöht. Die Verwaltung der Daten belegt pro Feld 1 Byte und pro Zeile 6 Byte. Für die Verwaltung der Replikation vergrößert sich jede Zeile der einzelnen Tabellen um einen Integer-Wert (4 Byte), einen **Globally Unique Identifier** (GUID, Werte dieses Datentyps sind weltweit eindeutig und sind 16 Byte groß¹²⁴) sowie ein Binärfeld mit einer Größe von 24 Byte. In dieser Arbeit soll die Größe der reinen Nutzdaten betrachtet werden, daher werden die Größen für die Verwaltung der Daten sowie der Replikation nicht mit in die Berechnungen für die Datenmengen aufgenommen. Die einzelnen Attribute der Datensätze werden mit Werten aus einem Zufallsgenerator gefüllt.¹²⁵

Für Aktualisierungen, die beim Test der Synchronisation einen weiteren Teil der zu synchronisierenden Daten ausmachen, ändert sich die Berechnung aus Tabelle 9, da die Werte, die die jeweilige Zeile identifizieren, nicht verändert werden sollen (*Szenario 1*). Dasselbe gilt für die Fremdschlüssel sowie für das Erstellungsdatum. Dafür wird das Änderungsdatum gefüllt. Damit wird die Komplexität des Update-Vorgangs gesenkt. Tabelle 10 zeigt die Änderungen in der Größenberechnung. Darin werden dieselben Abkürzungen wie in Tabelle 9 verwendet.

¹²³ Die 5 kB setzen sich aus 5 mal 1024 Byte zusammen, vergl. dazu Appelrath, Hans-Jürgen et al. (2000), S. 35f

¹²⁴ Vergl. Delaney, Kalen (2007), S. 193f (hier wird auch das grundlegende Vorgehen zur Erzeugung dieser global eindeutigen Werte dargestellt) und Microsoft (2007)

¹²⁵ Vergl. Microsoft (2007)

Innerhalb der Berechnung in Tabelle 9 werden auf unterschiedliche Längen der variablen Attribute der Relationen geachtet, um für die Testfälle der Datenermittlung, die nach diesen Attributen sortieren und Bedingungen stellen, Realbedingungen zu simulieren. Aktualisierungsblöcke, die nach dem Schema in Tabelle 10 aufgebaut sind, werden lediglich für die Synchronisation benötigt. Innerhalb der Synchronisierung spielt die Länge einzelner Werte in den einzelnen Attribute eine untergeordnete Rolle. Daher wird auf eine unterschiedliche Länge der variablen Felder verzichtet.

Pro Tabelle werden somit eine Anzahl von Datensätzen mit einer bestimmten Größe angegeben. Weiterhin werden die Größen der Datensätze von Relationen ohne variable Datentypen direkt angegeben (Position, Kollektion und Produktart). Auch hier ergeben sich 5 kB-Blöcke bzw. eine Zusammenfassung aus 50 Datensätzen.

Tabelle	Anzahl Datensätze	Größenberechnung der Felder	Gesamtgröße Datensätze	Gesamtgröße Relation
Kunde	8 Ds.	4 * (30 Z. * 2 B) = 240 B	8 * 286 B = 2288 B	2288 B
		46 B		
Auftrag	5 Ds.	68 Z. * 2 = 136 B	5 * 158 B = 958 B	958 B
		22 B		
Artikel	7 Ds.	60 Z. * 2 = 120 B	7 * 182 B = 1274 B	1286 B
		42 B		
	1 Ds.	75 Z. * 2 = 150 B	1 * 192 B = 192 B	
		42 B		
Position	12 Ds.	12 Ds. * 20 = 240 B	-	240 B
Produktart	3 Ds.	3 Ds. * 28 = 84 B	-	84 B
Kollektion	14 Ds.	14 Ds. * 28 = 392 B	-	392 B
Gesamtgröße: 5120 Bytes				

Tabelle 10: Berechnung der Größe der Datenblöcke für Veränderungen (Szenario 1)

Innerhalb des Synchronisationsszenario soll jeweils eine Hälfte der Last aus neuen Datensätzen und eine Hälfte aus aktualisierten Datensätzen bestehen. Wie das Verhältnis zwischen neuen und aktualisierten Datensätzen in der Realität aussieht, ist abhängig vom Anwendungsfall.

Um eine Filterung innerhalb von Abfragen zu ermöglichen, werden Suchblöcke verwendet. Innerhalb solcher Blöcke werden in der Kundenrelation im Attribut Nachname der Wert „Testkunde“ in sämtliche Datensätze des Blocks eingetragen (und mit zufälligen Zeichen vor und hinter dem Wert aufgefüllt werden, falls die Länge nicht zu den Berechnungen in Tabelle 9 passen), um eine Suche nach einem Zeichenkriterium zu ermöglichen. Weiterhin wird in den Tests des DBS das Erstellungsdatum der

Suchblöcke gespeichert. Damit können die Anfragen für Szenario zwei immer nach derselben Menge an Daten suchen, indem das Erstellungsdatum mit als Kriterium übernommen wird. Die einzelnen Messreihen eines Testlaufs bleiben somit vergleichbar. Da in dieser Arbeit ein Lasttest durchgeführt wird, muss die Datenmenge, mit der getestet wird, erhöht werden. Im folgenden wird dargestellt, nach welchen Schemata die Last für die einzelnen Szenarien erhöht werden soll.

Vorgehen der Lasterhöhung für Szenario 1

Wie in Kapitel 5 dargestellt, soll eine logarithmische Erhöhung der Datenmengen verwendet werden. Weiterhin sollen pro logarithmischer Laststufe Messpunkte mit einer linearer Schrittweise bestimmt werden. Für die automatisierte Erhöhung der Datenlast wird die Schrittweite der linearen Erhöhung für die Szenarien festgelegt. Innerhalb des Synchronisationstest soll bei einer Datenmenge von 30 kB gestartet werden. Für die Synchronisation ergeben sich die verschiedenen, zu synchronisierenden Datenmengen D_N in kB, wobei die Schrittweite der Lasterhöhung nach dem Schema in Formel 6.5.1 aufgebaut wird.¹²⁶ Pro Punkt wird eine Messreihe pro Testfall des Szenario 1 erstellt.

$$\begin{array}{l} D_1=30, D_2=60, D_3=90, \\ D_4=300, D_5=600, \dots, D_N \end{array}$$

Formel 6.5.1: Schrittweite der Datenerhöhung für die Synchronisation

Diese Auswahl an Punkten stellt ein Kompromiss aus Aussagekraft und Laufzeit der einzelnen Testfälle der Synchronisation dar, da der Durchlauf eines Testfalls der Synchronisation mit entsprechenden Wiederholungen, um eine Messreihe zu erstellen, ein langwieriger Vorgang ist. Die logarithmische Erhöhung ermöglicht einen starken Sprung der Last auf dem System. Somit kann einfach eine Untersuchung von leichten, mittleren und hohen Lasten realisiert werden.

Vorgehen der Lasterhöhung für Szenario 2 und 3

Da der Test des mobilen Datenbanksystems nicht so langwierig ist, sollen für die Testfälle des Szenario zwei und drei eine feinere Schrittweite gewählt werden. Die geringere Laufzeit der Szenarien für den DBS-Test begründet sich in der fehlenden Notwendigkeit, für jede Messung Daten zu generieren und zu löschen. Statt dessen wird pro Messreihe einmal die Datenmenge erhöht. Darauf basierend können die einzelnen Messungen von Szenario 1 und Szenario 2 durchgeführt werden. Mithilfe der feineren Schrittweise kann das Verhalten des mobilen Datenbanksystems unter Last deutlicher dargestellt werden. Hier kommt eine Erhöhung der Last entsprechend dem Schema in

¹²⁶ Vergl. Kapitel 5 für die Kombination einer logarithmischen und linearen Schrittweise.

Formel 6.5.2 zum Einsatz. Im Fall von Szenario 2 und 3 steht D_N für die Datenmenge in kB, welche sich bei der Durchführung der Testfälle in der mobilen Datenbank befindet.¹²⁷ Auch hier wird pro Testfall und Messpunkt eine Messreihe erstellt.

$$D_1=20, D_2=40, D_3=60, D_4=80, D_5=100, \\ D_6=200, D_7=400, D_8=600, \dots, D_N$$

*Formel 6.5.2: Schrittweite der
Datenerhöhung für den
Datenbanksystem-Test*

6.6. Metriken

Wie in Kapitel 5 dargestellt, gibt es verschiedene Metriken, die innerhalb eines Performance-Tests betrachtet und gemessen werden können. Die innerhalb dieser Arbeit zu testende Metrik soll die Verarbeitungszeit der einzelnen Testfälle sein.

Innerhalb von *Szenario eins* wird die Zeit gemessen, die für eine Synchronisation von Daten benötigt wird. In *Szenario zwei* wird die Laufzeit der jeweiligen Abfrage und das Auslesen der ermittelten Daten aus der Datenbank gemessen. In *Szenario drei* wird die Ausführungsgeschwindigkeit der jeweiligen Datenmanipulation gemessen.

Bei der späteren Betrachtung der Ergebnisse werden die Datenmengen, die von der Synchronisation verarbeitet worden sind bzw. auf der die verschiedenen Anfragen an das mobile DBS ausgeführt werden, in Relation zu der getesteten Metrik gestellt.

6.7. Testfälle

Dieses Kapitel stellt die einzelnen Testfälle für die verschiedenen Szenarien dar.

6.7.1. Testfälle Szenario 1

Das erste Szenario, für das in diesem Kapitel Testfälle definiert werden, stellt das Szenario der Synchronisation dar. Für die Synchronisationstestfälle stellt sich zunächst die Frage, mithilfe welcher Funktechnologien synchronisiert werden soll. Wie in Kapitel 2 bereits geschildert wurde, soll in dieser Arbeit speziell die Auswirkungen der WWAN-Technologien untersucht werden. Da die Technologien der 2. Generation abgelöst werden sollen, die Technologien der 3. Generation aber noch keine vollständige Überdeckung erreicht haben, wird eine Technologie der Generation 2.5 untersucht. Da der verwendete PDA als einzige 2.5-Generation-Technologie GPRS anbietet, wird diese verwendet. Um zu zeigen, wie stark der Unterschied von WWAN-Technologien zu lokalen Lösungen ist, werden weiterhin Testfälle auf Basis einer WLAN-Technologie nach dem IEEE-Standard 802.11g für Innenräume (max. 54 Mbs, vergl. Abbildung 2 in Kapitel 2.1) aufgebaut. Geräte, die auf diesem Standard basieren,

¹²⁷ 5 kB sind 50 Datensätze.

sind zur Zeit stark verbreitet.¹²⁸ Vergl. zu den beiden Technologien auch Kapitel 2.

Da eine bidirektionale Synchronisation getestet wird, wird eine zu synchronisierende Datenmenge auf dem Client und dem Server erstellt. Innerhalb des in dieser Arbeit verwendeten Modell eines verteilten Datenbanksystems ist von einer höheren Menge an neuen Daten auf dem Server auszugehen. Die Begründung liegt in der Sammlung aller Veränderungen im zentralen System, während die Clients untereinander keine Daten austauschen. Wie das Verhältnis der Datenänderungen auf Client und Server letztendlich in der Realität aussieht, wird vom jeweiligen Anwendungsfall bestimmt. In den einzelnen Synchronisationstestfällen wird jeweils ein Zehntel der zu synchronisierenden Daten auf dem mobilen Client und neun Zehntel der Daten auf dem Server erzeugt.

Tabelle 11 zeigt die verschiedenen Testfälle für Szenario 1. Die zu synchronisierenden Datenmengen ergeben sich aus den Punkten in Formel 6.5.1, wobei jeweils bis zu der in Tabelle 11 gezeigten Datengrenze erhöht wird.¹²⁹

Bezeichnung des Testfalls	Funk-technologie	Basisdatenmenge (kB)	Basisdatenmenge (Datensätze)	Datengrenze (kB)	Datengrenze (Datensätze)
WLAN klein	WLAN	1.000	10.000	5.000	50.000
WLAN Mittel	WLAN	2.000	20.000	5.000	50.000
WLAN Groß	WLAN	5.000	50.000	5.000	50.000
GPRS klein	GPRS	1.000	10.000	5.000	50.000
GPRS Mittel	GPRS	2.000	20.000	5.000	50.000
GPRS Groß	GPRS	5.000	50.000	5.000	50.000

Tabelle 11: Testfälle Szenario 1 (Synchronisation)

Innerhalb von Tabelle 11 wird eine Basisdatenmenge pro Testfall angegeben. Diese stellt eine Grundbelastung für die Synchronisation dar und wird vor der Durchführung eines Synchronisationstests in die verteilte Datenbank eingefügt. Dabei zeigen die in den einzelnen Testfällen verwendeten Basisdatenmengen, wie sich die Synchronisation bei einer Verdoppelung und dann bei einem fünffachen einer Basismenge verhält. Da die Synchronisation die neuen Daten mit diesen vorhandenen Daten abgleichen muss, liegt die Vermutung nahe, dass die Laufzeit der Synchronisation bei einer höheren Basisdatenmenge länger dauern wird. Daher kann über diese sechs Testfälle gezeigt werden, wie sich eine Synchronisation unter einer unterschiedlich hohen Basisdatenmenge sowie einer steigenden Datenmenge für die Synchronisation verhält.

¹²⁸ Vergl. Kurose, James F. et al. (2008), S. 526

¹²⁹ Vergl. Kapitel 6.5

Weiterhin zeigt sich der Unterschied zwischen WLAN und GPRS.

Die Datengrenze der einzelnen Testfälle wurde so gewählt, dass sich durch die Schrittweite der Synchronisation drei verschiedene Laststufen ergeben (10, 100, 1000). Damit kann das Verhalten der Synchronisation bei einer steigenden Last dargestellt werden und lässt Abschätzungen zu, wie sich die Datenübertragung bei höheren Datenmengen verhält. Gleichzeitig bleibt die Laufzeit der Testfälle in einem durchführbaren Rahmen.¹³⁰

Um das Transaktionslog der mobilen Datenbank, welches die Operationen auf einer Datenbank für die Realisierung von ACID-Transaktionen enthält und bei einer zunehmenden Größe Einfluss auf die Performance nimmt, zu kontrollieren, soll nach jeder Messreihe das Transaktionslog automatisch verkleinert werden.¹³¹ Auf Serverseite wird dies per Hand vom Tester durchgeführt (Pro Test ein mal). Dies wird lediglich in diesem Szenario durchgeführt, da das Generieren und Löschen von Daten für jede Messung das Transaktionslog unnatürlich stark wachsen lässt.

6.7.2. Testfälle Szenario 2 und 3

Um das mobile Datenbanksystem zu testen, sind im Kapitel 6.4 zwei Szenarien vorgesehen. Die Aufgabe der Testfälle in *Szenario zwei* ist das Ermitteln und Auslesen von Daten aus der Datenbank. Dabei sollen Abfragen einfacher, mittlerer und hoher Komplexität verwendet werden. Das folgende Listing zeigt den ersten Testfall des zweiten Szenario.

```
01: SELECT      Kunde.KundeId, Kunde.Vorname, Kunde.Nachname,
02:             Kunde.Strasse, Kunde.Ort, Kunde.PLZ, Kunde.KundenNr
03: FROM        Kunde
04: WHERE        ErstellungsZeitpunkt < @ErstellungsZeitpunkt
05: ORDER BY    Nachname
```

Innerhalb dieser Abfrage ist lediglich eine Relation betroffen. Weiterhin wird nach einem Attribut sortiert sowie nach dem Erstellungszeitpunkt gefiltert, um die Suchblöcke zu ermitteln. In Zeile vier im SQL-Befehl steht das „@Erstellungsdatum“ für den Zeitpunkt, nachdem das Einfügen von Suchdaten abgeschlossen worden ist.¹³²

Das folgende Listing beschreibt die mittelschwere Abfrage für Szenario 2. Diese verbindet die Relationen Kunde, Auftrag, Position und Artikel miteinander, um Informationen über Kunden und Artikel zu erhalten. Die Filterung des Nachnamens stellt eine zusätzliche Herausforderung an das DBS dar. Mithilfe dieser Abfrage wird

¹³⁰ Es hat sich gezeigt, dass die einzelnen Testfälle zwischen eineinhalb und zwei Tagen benötigen.

¹³¹ Während der Testdurchführung haben sich unterschiedliche Laufzeiten bei stark unterschiedlichen Log-Größen gezeigt

¹³² Vergl. Kapitel 6.5

getestet, wie das mobile Datenbanksystem auf das Verbinden von Tabellen reagiert und wie sich die Laufzeit im Gegensatz zu einer Abfrage auf lediglich einer Tabelle verhält.

```

01: SELECT  Kunde.Vorname, Kunde.Nachname, Kunde.Strasse, Kunde.Ort,
02:         Kunde.PLZ, Kunde.KundenNr, Kunde.ErstellungsZeitpunkt,
03:         Position.MengeArtikel.Bezeichnung, Auftrag.AuftragDatum
04: FROM Kunde
05:     INNER JOIN Auftrag ON Kunde.KundeId = Auftrag.KundeId
06:     INNER JOIN Position ON Auftrag.AuftragId = Position.AuftragId
07:     INNER JOIN Artikel ON Artikel.ArtikelId = Position.ArtikelId
08: WHERE Kunde.Nachname LIKE '%TESTKUNDE%' AND
09:        Kunde.ErstellungsZeitpunkt < @ErstellungsZeitpunkt
10: ORDER BY Nachname, Bezeichnung

```

Der dritte Testfall für Szenario 2 stellt eine komplexe Abfrage dar. Diese ist in dem folgendem Listing dargestellt. Neben der Verbindung aller sechs Relationen der Datenbank wird die Einschränkung der Datensätze über eine Unterabfrage (Zeile 21-23) realisiert. Weiterhin wird nach drei Attributen sortiert. Bei dieser Abfrage ist aufgrund der aufwändigen Operationen eine lange Laufzeit auf dem mobilen Gerät zu erwarten.¹³³

```

01: SELECT Kunde.Vorname, Kunde.Nachname, Kunde.Strasse, Kunde.Ort,
02:        Kunde.PLZ, Kunde.KundenNr, Kunde.ErstellungsZeitpunkt AS
03:        KundeErstellung, Kunde.AenderungsDatum,
04:        Auftrag.AuftragDatum, Auftrag.Rabatt, Auftrag.Kommentar,
05:        Auftrag.AuftragsArt, Auftrag.ErstellungsZeitpunkt AS
06:        AuftragErstellung, Position.Menge,
07:        Position.PreisBeiBestellung, Position.ErstellungsZeitpunkt
08:        AS PositionErstellung, Artikel.EAN, Artikel.Preis,
09:        Artikel.Bezeichnung, Artikel.KollektionId,
10:        Produktart.ProduktArtName, Kollektion.KollektionName
11: FROM Kunde INNER JOIN
12:        Auftrag ON Kunde.KundeId = Auftrag.KundeId
13:        INNER JOIN Position ON Auftrag.AuftragId =
14:        Position.AuftragId
15:        INNER JOIN Artikel ON Position.ArtikelId = Artikel.ArtikelId
16:        INNER JOIN Kollektion ON Artikel.KollektionId =
17:        Kollektion.KollektionId
18:        INNER JOIN Produktart ON Kollektion.ProduktArtId =
19:        Produktart.ProduktArtId
20: WHERE Kunde.KundeId in
21:        (SELECT Kunde.KundeId FROM Kunde WHERE
22:         Produktart.ErstellungsZeitpunkt <= @ErstellungsZeitpunkt AND
23:         Kunde.Nachname LIKE '%TESTKUNDE%')
24: ORDER BY Kunde.Nachname, Auftrag.AuftragDatum,
25:         Artikel.Bezeichnung

```

Für *Szenario drei* ergeben sich drei verschiedene Testfälle, das Einfügen, Aktualisieren und Löschen von Datensätzen. Dabei wird jeweils mit einer Datenzeile in der Tabelle Kunde gearbeitet.

Zugehöriges Szenario	Name	Beschreibung	Datengrenze (Datensätze)
Szenario 2	Anfrage mit niedriger Komplexität	Durchführen der ersten SQL-Anfrage, welche auf der Relation Kunde arbeitet.	500.000
Szenario 2	Anfrage mit mittlerer	Durchführen der zweiten SQL-Anfrage, welche	500.000

¹³³ Vergl. Gulutzan, Peter et al. (2003), S. 11 und S. 113

Zugehöriges Szenario	Name	Beschreibung	Datengrenze (Datensätze)
	Komplexität	vier Tabellen miteinander verbindet.	
Szenario 2	Anfrage mit hoher Komplexität	Durchführen der dritten SQL-Anfrage, welche alle Tabellen verbindet.	500.000
Szenario 3	Aktualisierung von Daten	Änderung einer Zeile in der Tabelle Kunde.	500.000
Szenario 3	Einfügen von Daten	Einfügen einer Zeile in der Tabelle Kunde.	500.000
Szenario 3	Löschen von Daten	Löschen einer Zeile in der Tabelle Kunde.	500.000

Tabelle 12: Testfälle Szenario 2 (Anfragen) und Szenario 3 (Datenmanipulationen)

Tabelle 12 fasst die verschiedenen Testfälle für den DBS-Test zusammen. Diese Testfälle werden bis zu einer Gesamtdatenmenge von 500.000 Datensätze (50.000 kB) durchgeführt. Die einzelnen Datenmengen, auf denen die verschiedenen Operationen durchgeführt werden, ergeben sich aus Formel 6.5.2 sowie dieser maximalen Datenmenge.¹³⁴ Durch den Test bis zu dieser Grenze soll, wie beim Synchronisationstest, der Trend der Performance unter einer zunehmenden Belastung gezeigt werden. Diese Grenze liegt weit über der Grenze der Synchronisationstestfälle, da die Gesamtlaufzeit der DBS-Tests stark unter der der Synchronisationstest liegt.

6.8. Durchführung des Tests und Modell der Testapplikation

Dieses Kapitel beschreibt, wie die Testfälle aus dem vorherigen Kapitel mit einer verteilten Testapplikation durchgeführt werden. Dafür wird zunächst der grobe Aufbau der Testapplikation dargestellt. Weiterhin werden die einzelnen Anwendungsfälle, die mit der Applikation möglich sind, sowie deren Ablauf dargestellt. Abgeschlossen wird mit einer detaillierten Darstellung der Testapplikation.

6.8.1. Struktur des Gesamtsystems

Dieses Kapitel zeigt, in welche Einzelteile die verteilte Anwendung zerlegt ist. Das Verteilungsdiagramm in Abbildung 10 zeigt die Komponenten des Systems.

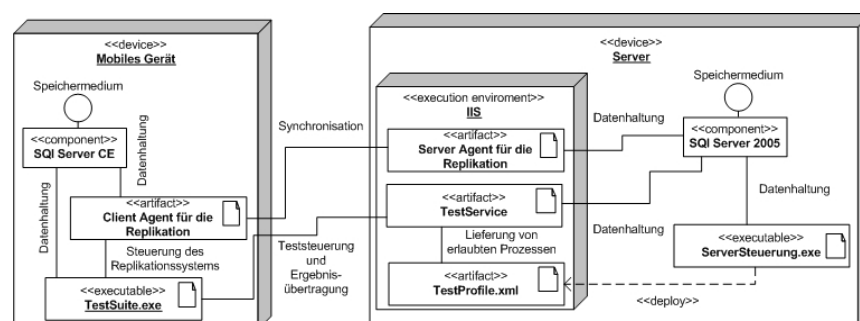


Abbildung 10: Verteilungsdiagramm der Testkomponenten¹³⁵

¹³⁴ Vergl. Kapitel 6.5

¹³⁵ Zum Aufbau von Verteilungsdiagrammen, vergl. Rupp, Chris et al. (2007), S. 211ff und S. 223ff sowie

Die Testapplikation besteht aus einer Anwendung auf Clientseite (*TestSuite.exe*) sowie Serverseite (*ServerSteuerung.exe*) und einer Webapplikation (*TestService*), welche Web Services für die Steuerung des Servers sowie Schnittstellen zur Übertragung von Messergebnissen enthält. Zusätzlich kommt das mobile DBS und das Server-DBS mit dem Agent-System für die Replikation zum Einsatz. Die Agents ermöglichen die Erstellung von Publikationen, die Subskription von Clients sowie die Synchronisation.¹³⁶ Während die Clientapplikation die Tests steuert, übernimmt die Applikation auf der Serverseite das Erstellen von Prozessprofilen für den Server sowie die Anzeige und Auswertung von Ergebnissen. Anhang 4 beschreibt, welche Schritte zur Installation und Einrichtung dieses Systems unternommen werden müssen.

Für die Speicherung der Messergebnisse werden die Relationen aus dem Schema verwendet, welches in Abbildung 11 dargestellt ist. Die Relation *Master_Test* nimmt dabei die einzelnen Szenarien auf, während die Testfälle in der Relation *Testfall* und die Messungen in der Tabelle *Test* gespeichert werden. Eine genaue Datendefinition ist in Anhang 5 zu finden. In Anhang 6 wird die Bedienung des Systems dargestellt.



Abbildung 11: Datenbankschema der Datenbank für die Speicherung der Messergebnisse

6.8.2. Anwendungsfälle

Dieses Kapitel stellt die einzelnen Anwendungsfälle dar, die ein Benutzer mit der Testsoftware durchführen kann. Abbildung 12 zeigt diese in einem Use-Case-Diagramm¹³⁷.

Die Anwendungsfälle für die Vorbereitung der Tests stellen das Anlegen der mobilen Datenbank (Nach einer Installation der mobilen Testsoftware ist diese nicht vorhanden) sowie die Subskription bei der Replikation des zentralen Systems dar. Bei der Erstellung der mobilen Datenbank gilt, dass die Serverdatenbank bereits vorhanden ist. Ansonsten scheitert die Erstellung, da das Schema der mobilen Datenbank nicht von der Serverdatenbank abgeleitet werden kann. Weitere Vorbereitungen sind das Herstellen eines Testzustandes auf dem mobilen Gerät und die Erstellung einer Basis für die Synchronisation. Da diese Aktionen nicht für jeden Test durchgeführt werden müssen

Oestereich, Bernd (2005), S. 110f

¹³⁶ Vergl. auch Kapitel 4.2 für eine Erläuterung von Agents

¹³⁷ Zum Aufbau von Use-Case-Diagrammen, vergl. Rupp, Chris et al. (2007), S. 237ff und Oestereich, Bernd (2005), S. 20ff

und teilweise zeitaufwändig sind, stellt die Software diese Funktionalitäten getrennt von den eigentlichen Tests zur Verfügung. Falls beim Start eines Tests noch nicht alle nötigen vorbereitenden Anwendungsfälle durchlaufen worden sind, werden diese um die vorbereitenden Fälle erweitert.

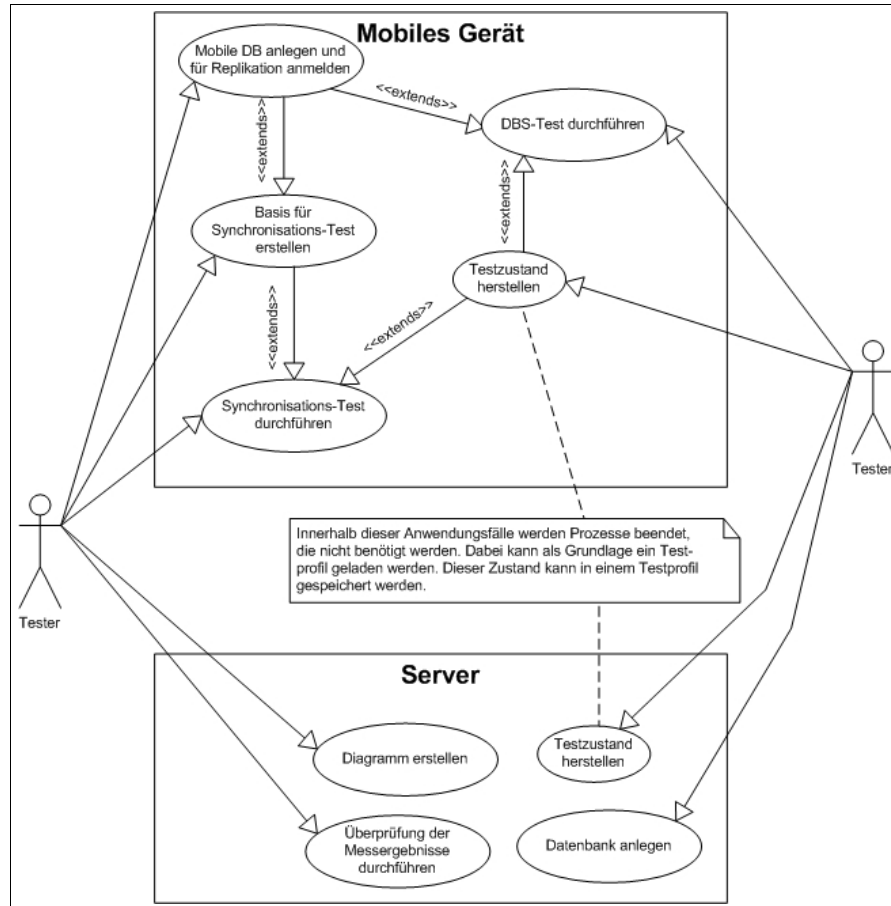


Abbildung 12: Anwendungsfälle

Die Anwendungsfälle auf der Serverseite sind durch die Serversteuerung realisiert. Um die Serverdatenbank auf der Serverseite einzurichten, stellt die Serverapplikation eine Möglichkeit zur Verfügung, ein SQL-Erstellungsscript für die Erstellung der Datenbank auszuführen.¹³⁸ Für das Herstellen eines Testzustands auf Serverseite existiert ebenfalls ein entsprechender Anwendungsfall. Die Testprofile, die hier entstehen, müssen den Web Services zur Verfügung gestellt werden, damit der Client das Laden eines solchen Profils steuern kann. Dies ist notwendig, da die Teststeuerung auf dem mobilen Gerät realisiert ist. Weitere Möglichkeiten auf Serverseite ist die Erstellung von Diagrammen aus den Messergebnissen der Tests sowie die Überprüfung, ob die Messergebnisse aussagekräftig sind.

¹³⁸ Für eine vollständige Installationsanleitung, siehe Anhang 4.

6.8.3. Abläufe

Dieses Kapitel stellt mithilfe von Aktivitätsdiagrammen¹³⁹ dar, aus welchen Teilschritten die einzelnen Anwendungsfälle aus Kapitel 6.8.2 bestehen und zeigt, wie die eigentliche Testdurchführung von der Applikation realisiert worden sind.

Abbildung 13 beschreibt, aus welchen Teilschritten der Anwendungsfall „*Testzustand herstellen*“ besteht. Dieser Ablauf wird auf Clientseite wie auf Serverseite verwendet. Dabei gilt auf der Serverseite, dass im Unterverzeichnis „*Profile*“ im Ausführungsordner der Web Services ein Testprofil abgelegt werden muss. Falls mehrere Testprofile vorhanden sind, wird von den Web Services das erste Profil genommen, welches gefunden wird.

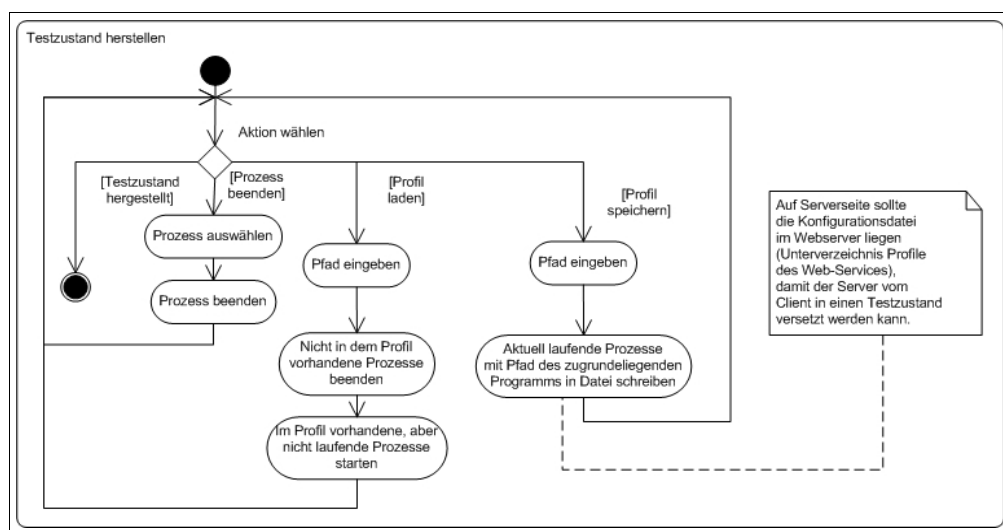


Abbildung 13: Aktivitätsdiagramm – Testzustand herstellen

Abbildung 14 stellt die einzelnen Schritte dar, aus denen der Anwendungsfall „*DBS-Test durchführen*“ besteht.

Der Test startet mit der Überprüfung, ob sich das mobile Gerät im Testzustand befindet. Dabei wird die Aktivität „*Testzustand herstellen*“ (Vergl. Abbildung 13) aufgerufen und durchgeführt, falls die Überprüfung ein negatives Ergebnis liefert. Dann werden die Parameter des Tests aus der Maske der Teststeuerung ermittelt und auf Vollständigkeit sowie Korrektheit geprüft. Weiterhin wird geprüft, ob die mobile Datenbank vorhanden ist. Falls nicht, wird sie erstellt (Vergl. dazu die Aktivität „*Mobile Datenbank anlegen und für Replikation anmelden*“, Abbildung 20).

Mit der Durchführung eines Testlauf ohne Erstellung von Messergebnissen wird das DBS in einen Zustand gebracht, in dem Puffer bereits vorbereitet sind und die Dienste

¹³⁹ Für Aktivitätsdiagramme, vergl. Rupp, Chris et al. (2007), S. 259ff sowie Oestereich, Bernd (2005), S. 112ff

für ein Herstellen einer Verbindung gestartet sind (Vergl. Kapitel 5.2). Somit werden die Testergebnisse nicht durch das Starten aller benötigten Dienste verfälscht. Danach werden die einzelnen Messpunkte (Datenmengen) durchlaufen. Pro Messpunkt werden alle Testfälle von Szenario zwei und drei durchgeführt. Dabei gilt, dass im Testfall „Löschen von Daten“ der Datensatz gelöscht wird, der für den Testfall „Einfügen von Daten“ eingefügt wurde. Dadurch verzerren die für den letzteren Testfall eingefügten Datensätze nicht die Datenlast. Nach Erreichen der Datengrenze wird der Test beendet.

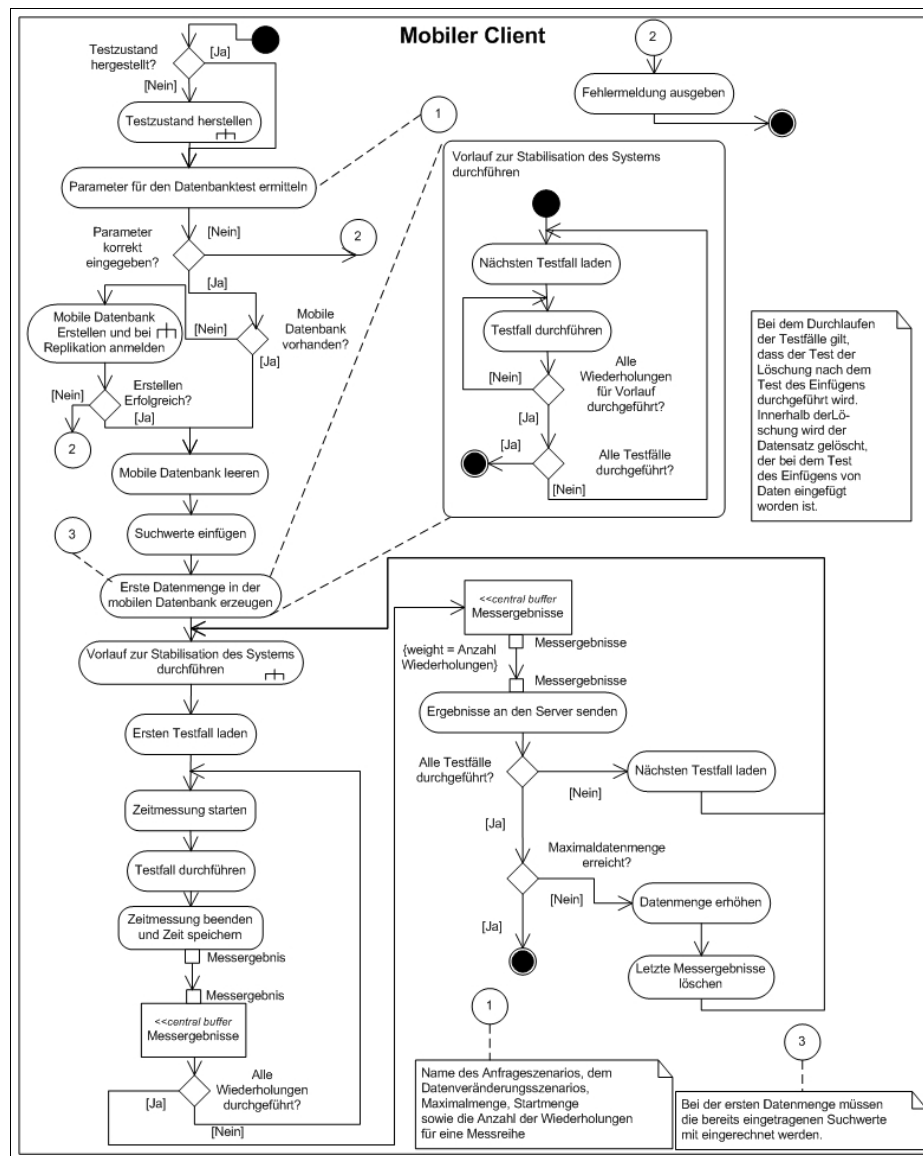


Abbildung 14: Aktivitätsdiagramm – DBS-Test durchführen

Abbildung 15 zeigt den vollständigen Ablauf eines Synchronisationstestes. Auch hier werden zunächst vorbereitende Schritte unternommen. Der Tester muss zunächst dafür sorgen, dass das Gerät für eine Verbindung mit dem Server die zu testende Technologie verwendet. Dies kann man über die Netzwerkeinstellungen des Betriebssystems

einstellen. Wie beim DBS-Test wird die Aktivität „Testzustand herstellen“ durchgeführt, wenn auf dem Client noch nicht alle unnötigen Prozesse beendet worden sind. Weiterhin werden die Parameter für den Test ermittelt und geprüft sowie sichergestellt, dass eine Basisdatenmenge vorhanden ist.

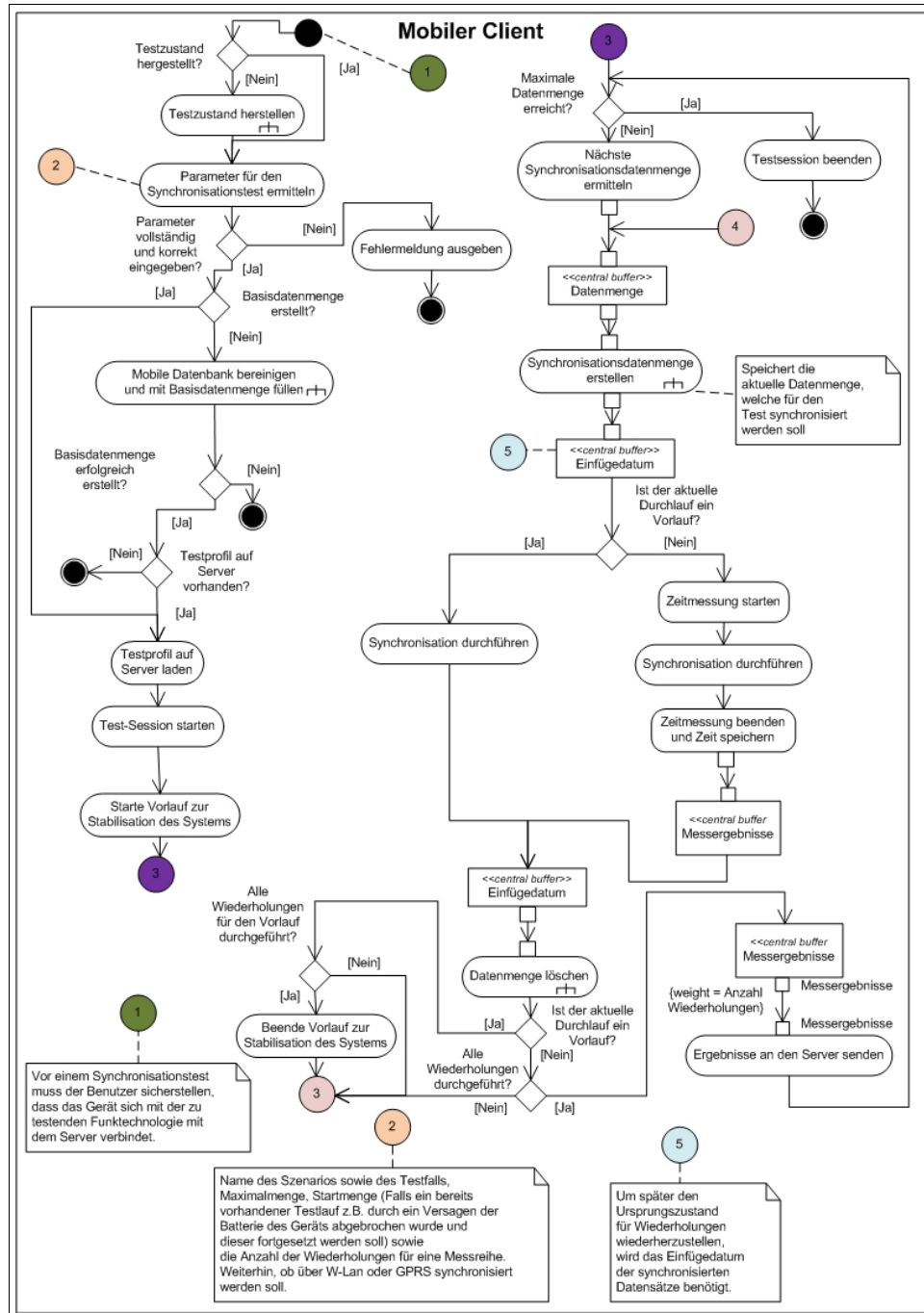


Abbildung 15: Aktivitätsdiagramm – Synchronisationstest

Nachdem auch der Server in einen Testzustand versetzt wurde (durch das Laden eines Testprofils auf dem Server; falls keins vorhanden ist, wird der Vorgang abgebrochen) und alle für den Synchronisationstest benötigte serverseitige Session-Variablen erstellt

worden sind (also Variablen, welche mehrere Web Service-Aufrufe überleben und zu einem Synchronisationstestlauf gehören), kann mit der eigentlichen Testdurchführung gestartet werden.

Dann wird, äquivalent zum DBS-Test, eine Messreihe simuliert, um mögliche Schwankungen in der Laufzeit durch einen Start der Replikation und ein Aufbauen der entsprechenden Verbindungen zu vermeiden. Dabei werden keine Messungen vorgenommen, sondern lediglich die Synchronisation durchgeführt. Weiterhin wird eine Anzahl von Wiederholungen verwendet, die lediglich für den Vorlauf gilt. Damit soll vermieden werden, dass eine zu hohe Anzahl von Wiederholungen zur Vorbereitung den Test unnötig verzögern.

Der erste Schritt, um eine Messreihe für den Synchronisationstest zu erhalten, besteht in der Ermittlung und Erstellung der für die Messreihe benötigten Datenmenge, die synchronisiert werden soll (für die Erstellung der Datenmenge, vergl. Abbildung 16). Danach wird die Messung der Laufzeit der Synchronisation durchgeführt.

Nach der Durchführung der Synchronisation wird die Laufzeit gespeichert und die neu eingefügten Daten gelöscht, um den Ursprungszustand für die nächste Messung herzustellen. Dieser Vorgang wird so lange wiederholt, bis die vom Benutzer eingegebene Anzahl von Wiederholungen erreicht ist. Nach der Durchführung der Wiederholungen werden die Ergebnisse der Messreihe an den Server übertragen. Dann wird mit der nächsten Messreihe fortgesetzt. Pro Messreihe wird die zu synchronisierende Datenmenge erhöht und es werden so lange Messreihen erzeugt, bis die vom Benutzer eingegebene Maximaldatenmenge erreicht ist.

Abbildung 16 zeigt die im Synchronisationstest verwendeten, zusammengeklappten Abläufe „*Synchronisationsdatenmenge löschen*“ und „*Synchronisationsdatenmenge erstellen*“. Die Aktivität „*Synchronisationsdatenmenge erstellen*“ erzeugt eine zu synchronisierende Datenmenge durch das Einfügen neuer Datensätze sowie dem Aktualisieren von vorhandenen Datensätzen auf der Client- sowie der Serverseite. Weiterhin liefert die Aktivität das Erstellungsdatum von neuen Datensätzen, die später in der Aktivität „*Datenmenge löschen*“ verwendet wird, um die bei der Datenerhöhung eingefügten, neuen Datensätze zu löschen und den Grundzustand wieder herzustellen. Die Datenmenge wird zur Hälfte aus neuen Datensätzen und zur Hälfte aus aktualisierten Datensätzen erzeugt (Vergl. Kapitel 6.5).

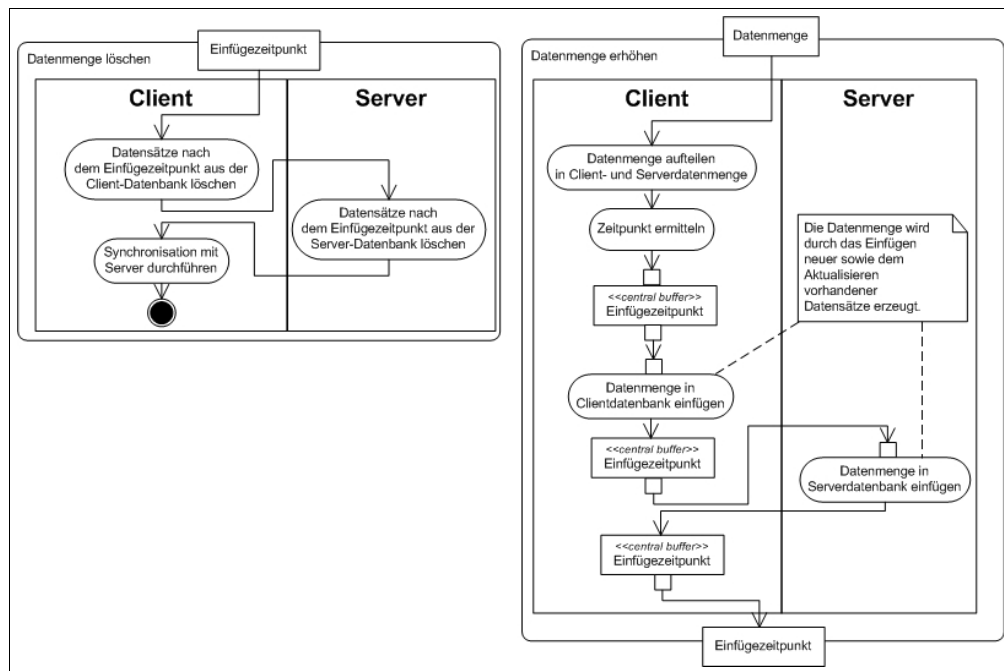


Abbildung 16: Unteraktivitäten für den Synchronisationstest

Abbildung 17 zeigt, wie die Basisdatenmenge für die Synchronisation, unter Vorgaben des Benutzers, in die gesamte verteilte Datenbank eingefügt wird. Dabei wird sichergestellt, dass die mobile Datenbank vorhanden ist (falls nein, wird der Ablauf des Anwendungsfall „Mobile Datenbank erstellen und für Replikation anmelden“ aufgerufen und durchgeführt).

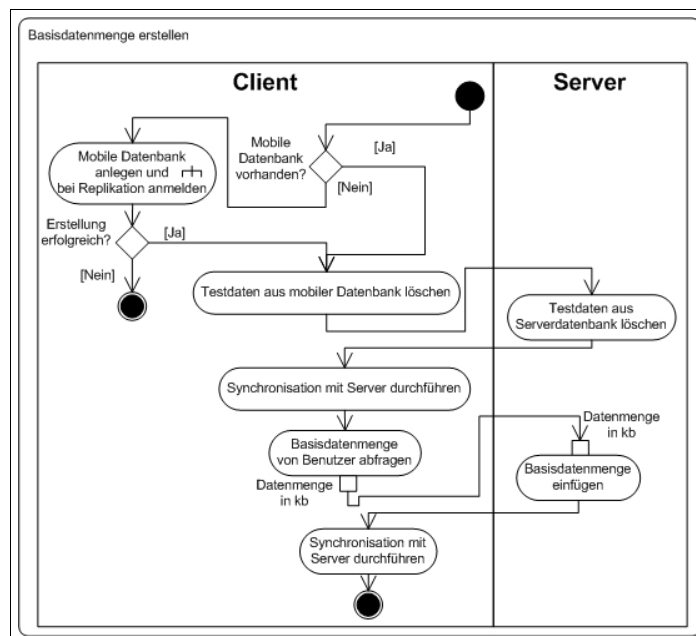


Abbildung 17: Aktivitätsdiagramm – Basisdatenmenge erstellen

Der Ablauf des Anwendungsfall „Überprüfung der Messergebnisse durchführen“ wird

in Abbildung 18 dargestellt. Um die Überprüfung zu starten, wird das Szenario sowie der zu überprüfende Testfall ausgewählt. Weiterhin wird der Wert d , aus dem das Intervall berechnet wird, sowie das $u_{1-\alpha/2}$ -Quantil, basierend auf der Irrtumswahrscheinlichkeit α (Vergl. für diese Parameter Kapitel 5 sowie Anhang 1), vom Benutzer eingegeben. Die Eingaben können parallel und in beliebiger Reihenfolge geschehen. Die Ausnahme davon ist die Berechnung der Wahrscheinlichkeit $1-\alpha/2$, welche zusammenhängend realisiert worden ist.

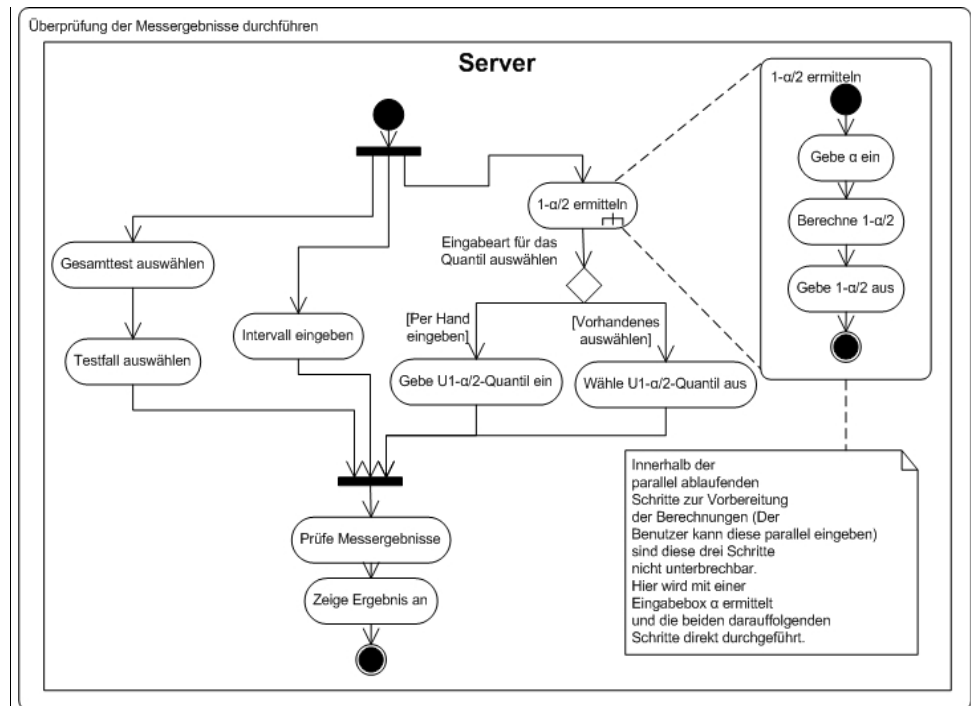


Abbildung 18: Aktivitätsdiagramm – Überprüfung der Messergebnisse durchführen

Daraufhin werden die einzelnen Messreihen des Testfalls überprüft und angezeigt, bei welchen die Abweichungen zu stark sind. Das $u_{1-\alpha/2}$ -Quantil kann dabei zum einen per Hand eingegeben werden. Zum anderen bietet die Serverapplikation eine Auswahl an Quantilen an, aus der der Benutzer eine auswählen kann.

Abbildung 19 stellt den Ablauf des Anwendungsfalls „Diagramm erstellen“ dar. Der Tester kann dabei zunächst das Szenario auswählen. Darauf basierend stehen ihm die Testfälle dieses Szenario zur Verfügung, deren Graphen er dem Diagramm hinzufügen kann. Weiterhin wird ihm die Möglichkeit gegeben, den Grundzustand der Maske über das Löschen aller Graphen aus dem Diagramm wiederherzustellen. Neben der Auswahl der darzustellenden Graphen gibt es die Möglichkeit, die Einheiten der Achsen zu bestimmen und ein erstelltes Diagramm als Bild abzuspeichern. Mithilfe der

Speicherung des Diagramms als Bild kann dieses in Dokumente übernommen werden.

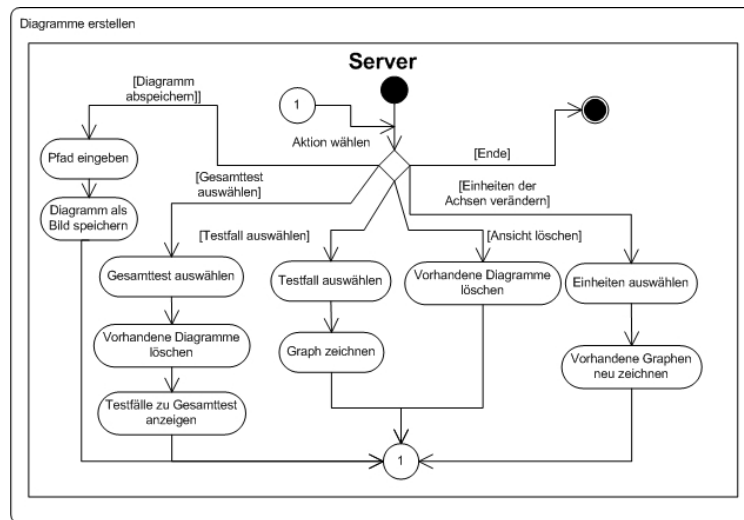


Abbildung 19: Aktivitätsdiagramm – Diagramme erstellen

Abbildung 20 stellt den Ablauf der Datenbankerstellung (Client wie Server) dar. Dabei gilt, dass die mobile Datenbank nach der Serverdatenbank (sowie der Erstellung der Publikationen auf Serverseite) erstellt werden muss, da ansonsten die mobile Datenbank nicht vom Replikationsschema abgeleitet werden kann.¹⁴⁰ Auf Serverseite wird dafür ein SQL-Script bereitgestellt, welches über die Serverapplikation ausgeführt werden kann. Auf Clientseite wird das Replikationsschema verwendet, welches auf Serverseite bereitgestellt wird. In den folgenden Kapiteln wird dargestellt, wie das Testsystem aufgebaut ist. Die Benutzeranleitung in Anhang 6 verdeutlicht die einzelnen Vorgänge, speziell die Einbindung des Benutzers in die einzelnen Abläufe.

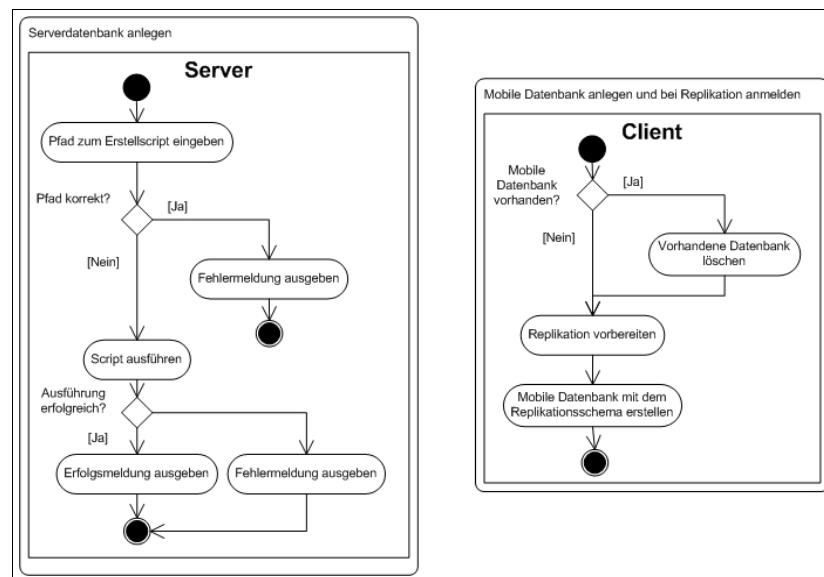


Abbildung 20: Aktivitätsdiagramm – Datenbanken anlegen

¹⁴⁰ Vergl. auch die Installationsanleitung in Anhang 4.

6.8.4. Paketaufbau

Dieses Kapitel stellt die Pakete vor, in die die einzelnen Klassen des Testsystems aufgeteilt sind. Abbildung 21 zeigt diese Struktur in einem Paketdiagramm.¹⁴¹

Das Paket *TestService* stellt dabei die Grenze zwischen dem Server- und Clientsystem dar und enthält die Web Services, welche im Unterpaket *TestSession* gekapselt sind. Diese Web Services stellen *Fassadenklassen* dar, welche Schnittstellen zu der im Paket *TestLibraryServer* befindlichen Testlogik auf Serverseite bereitstellen.¹⁴²

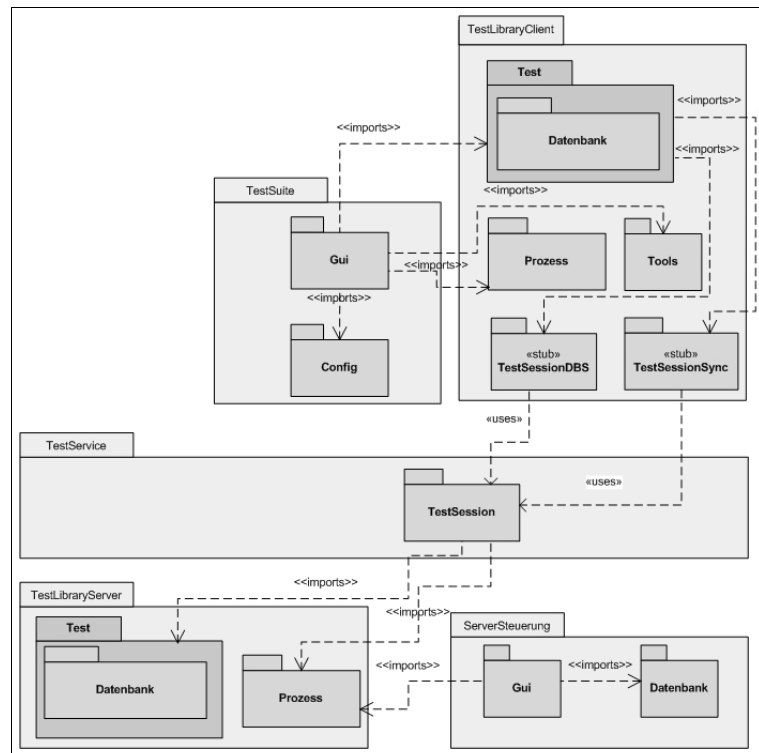


Abbildung 21: Paketdiagramm

Die Client-Applikation ist im Paket *TestSuite* gekapselt und enthält neben einem GUI-Paket (Graphical User Interface) noch ein Paket, in dem Konfigurationslogik für den Client enthalten ist (*Config*). Die Konfigurationslogik im Paket *Config* erlaubt es, die Testapplikation mit Parametern aus einer XML-Datei zu versehen. Diese Parameter werden für die Verbindung mit dem zentralen System benötigt. Bei Veränderungen (z. B. eine Änderung der Adresse des Replikationsservers) muss lediglich die XML-Datei angepasst werden.¹⁴³

Daneben gibt es das Paket *TestLibraryClient*, welches die eigentliche Steuerungslogik des Tests im Unterpaket *Test* bereitstellt und Pakete mit Stub-Klassen enthält, die für

¹⁴¹ Für Paketdiagramme, vergl. Rupp, Chris et al. (2007), S.165ff und Oestereich, Bernd (2005), S. 104ff

¹⁴² Vergl. für Fassadenklassen Gamma, Erich et al. (2004), S. 212

¹⁴³ Vergl. auch die Installationsanleitung in Anhang 4, dort wird die Konfigurationsdatei näher dargestellt.

die Kommunikation mit den Web Services verantwortlich sind (*TestSessionDBS* und *TestSessionSync*).¹⁴⁴ Das Paket *TestSessionDBS* enthält entsprechende Klassen für die Kommunikation, die für den DBS-Test benötigt wird, während das Paket *TestSessionSync* die Kommunikation mit dem Web Service für den Synchronisationstest übernimmt. Innerhalb der Teststeuerung ist ein weiteres Unterpaket untergebracht, welches die Kommunikation mit dem mobilen Datenbanksystem übernimmt (*Datenbank*).

Weiterhin enthält die Clientbibliothek ein Paket für den Umgang mit Prozessen (*Prozess*) sowie das Paket *Tools* für allgemeine Aufgaben. Somit ist zum einen die eigentliche Testlogik und zum anderen die Datenhaltung von der Testoberfläche getrennt in der Clientbibliothek zu finden.

Auf Serverseite gibt es ebenfalls eine Bibliothek mit dem Namen *TestLibraryServer*. Diese enthält zum einen, äquivalent zur Clientseite, ein Unterpaket *Test*, welches die vom mobilen Client für die Tests benötigten Serverfunktionalitäten kapselt. Die Web Services stellen dem mobilen Client diese Funktionalitäten zur Verfügung. Zum anderen gibt es ebenfalls ein Paket *Prozess*, welche den Umgang mit Prozessen für das Herstellen eines Testzustandes auf Serverseite bereitstellt. Diese werden von der Serverapplikation verwendet, welche sich im Paket *ServerSteuerung* befindet. Dieses Paket wird in das Unterpaket *Gui* für die Oberfläche sowie dem Unterpaket *Datenbank*, welche der Serversteuerung den Zugriff auf die Messergebnisse in der Serverdatenbank ermöglicht, aufgeteilt.

Die vorgestellte Paketstruktur teilt die mobile Testapplikation in eine GUI-, Logik- und Datenschicht, welche einen Austausch einer der Schichten ermöglicht sowie die Komplexität der Gesamtanwendung senkt.

Für den Wechsel des zu testenden mobilen DBS, des zentralen DBS oder des Replikationssystems müssen lediglich die Testlogik in den Testpaketen sowie die Datenbankpakete auf beiden Seiten angepasst werden. Einzige Ausnahme davon können die Konfigurationsparameter in der GUI auf Clientseite sein, die sich bei einem Wechsel des Serversystems ändern können (z. B. zusätzliche Verbindungsangaben).

Auf der Serverseite werden innerhalb der einzelnen Oberflächenelemente nicht so starke Änderungen erwartet wie auf der Clientseite, wo ein Wechsel des Gerätes eine komplett

¹⁴⁴ Vergl. für eine kurze Einführung in Stub-Klassen bzw. in der Realisierung der Kommunikation bei Web Services Rechenberg, Peter (Hrsg.) et al. (2006), S. 1132ff oder (für eine vollständige Einführung in Web Services) Dunkel, Jürgen et al. (2008), Kapitel 5.2 (S. 92ff). Da diese Pakete lediglich automatisch generierte Gegenstücke zu den Web Services enthalten, tauchen sie nicht in der Quellcodedokumentation auf.

neue GUI nötig machen kann. Die Logik für die Prozesssteuerung, der Diagrammerstellung und der Überprüfung der Messergebnisse wird auch bei einem Wechsel des Datenbanksystems gleich bleiben, da der Umgang mit dem DBS im Paket *ServerSteuerung::Datenbank* gekapselt ist. Weiterhin werden Anpassungen der graphischen Oberfläche lediglich begrenzt Auswirkungen auf die Berechnungen haben. Daher wird innerhalb der Serverapplikation auf die Logikschicht verzichtet. Stattdessen wird die Anwendungslogik in einzelnen graphischen Elementen gekapselt, welche in anderen Testprojekten wiederverwendet werden können.

6.8.5. Modell Server

Dieses Kapitel stellt das Klassenmodell der Serverseite vor. Abbildung 22 zeigt die verschiedenen Klassen des Servers sowie deren Beziehungen zueinander in einem Klassendiagramm.¹⁴⁵ Um eine bessere Lesbarkeit der Klassendiagramme zu erreichen, werden die Pakete mit in die Klassendiagramme übernommen.¹⁴⁶

Im Diagramm stechen die Web Services im Paket *TestSession* hervor, welche lediglich über Standard-.Net-Klassen (aus dem Paket *System::Web* bzw. dessen Unterpaket *Services*) eine Verbindung zur Serverbibliothek aufnehmen.

Dieser Aufbau dient dem Zweck, eine Neuerstellung der Datenklassen bei jedem Web Service-Aufruf zu verhindern. Statt dessen werden Instanzen dieser Datenklassen von der Klasse *Global* erzeugt, die auf Events der Umgebung, welche die Web Services ausführen, reagiert (in diesem Fall auf das Starten und Beenden der Webapplikation, in der sich die Web Services befinden).¹⁴⁷

Diese Instanzen werden den Web Services über eine Instanz der Klasse *HttpApplicationState* zur Verfügung gestellt, in dem neben Informationen zur Webapplikation auch globale Variablen gespeichert werden. Pro Webapplikation wird ein solches *HttpApplicationState*-Objekt von der ausführenden Umgebung erzeugt und den einzelnen Elementen zur Verfügung gestellt. Bei dem Beenden der Webapplikation werden die Instanzen wieder gelöscht.

Die Test-Datenklassen auf Serverseite, welche sich im Unterpaket *Datenbank* der Serverbibliothek befinden, sind in zwei Teile geteilt. Die Klasse *ServerDatenGenerator* übernimmt die Generierung der Arbeitslast auf Serverseite. Mithilfe des

¹⁴⁵ Vergl. für Klassendiagramme Rupp, Chris et al (2007), S. 101 ff und Oestereich, Bernd (2005), S.48ff.

¹⁴⁶ Vergl. für dieses Vorgehen Booch, Grady et al. (2006), S. 139 und S. 202 sowie Oestereich, Bernd (2005), S.106 und Rupp, Chris et al (2007), S. 169

¹⁴⁷ Vergl für die ausführende Umgebung Abbildung 10 im Kapitel 6.8.1, welches die Verteilung der einzelnen Komponenten darstellt.

```

    packageDiagram
        package SystemWeb {
            class HttpApplicationState
            class Service {
                class WebService
            }
        }
        package TestService {
            class Global
        }
        package TestLibraryServer {
            class Test
            class Datenbank {
                class ServerDatenGenerator
                class TestdatenbankZugang
                class ServerWerteGenerator
            }
        }
        package Prozess {
            class ProzessHandling
            class ProzessProfil
        }
        package ServerSteuerung {
            class Gui {
                class ucProzessListe
                class frmTestAuswertung
                class ucTestAuswahl
                class ucStatistikAuswertung
                class ucEinzelAuswertung
            }
        }
        package Datenbank {
            class SQLScriptAusfuehrer
            class TestDaten
        }

        SystemWeb.HttpApplicationState "1" -- "1..*" SystemWeb.Service::WebService
        TestService.Global "1" -- "1..*" TestLibraryServer.Test
        TestLibraryServer.Datenbank::ServerDatenGenerator "1..*" -- "1" TestLibraryServer.Datenbank::ServerWerteGenerator
        TestLibraryServer.Datenbank::TestdatenbankZugang "1" -- "1" TestLibraryServer.Datenbank::ServerWerteGenerator
        TestLibraryServer.Datenbank::ServerWerteGenerator "1" -- "1" Prozess::ProzessHandling
        TestLibraryServer.Datenbank::TestdatenbankZugang "1" -- "1" Prozess::ProzessProfil
        ServerSteuerung.Gui::ucProzessListe "1..*" -- "1..*" ServerSteuerung.Gui::frmTestAuswertung
        ServerSteuerung.Gui::frmTestAuswertung "1" -- "1" ServerSteuerung.Gui::ucTestAuswahl
        ServerSteuerung.Gui::ucTestAuswahl "1" -- "1" ServerSteuerung.Gui::ucStatistikAuswertung
        ServerSteuerung.Gui::ucStatistikAuswertung "1" -- "0..*" ServerSteuerung.Gui::ucEinzelAuswertung
        ServerSteuerung.Gui::ucProzessListe "1..*" -- "1" Datenbank::SQLScriptAusfuehrer
        ServerSteuerung.Gui::frmTestAuswertung "1" -- "1" Datenbank::TestDaten
        ServerSteuerung.Gui::ucTestAuswahl "1" -- "1" Datenbank::TestDaten
        Datenbank::SQLScriptAusfuehrer "1" -- "1" Datenbank::TestDaten
    
```

System.Web

- HttpApplicationState** (1 instance)
 - Ruft Applikationsinformationen und globale Variablen ab aus einem
 - Speichert Applikationsinformationen und applikationsweite Informationen in einem
- Service** (1..* instances)
 - WebService**

TestService

- Global** (1 instance)
 - #Application_Start(ein sender : object, ein e : EventArgs)
 - #Application_End(ein sender : object, ein e : EventArgs)

TestLibraryServer

- Test** (1 instance)
 - Ermöglicht den Umgang mit Testdaten mit einem
- Datenbank** (1 instance)
 - ServerDatenGenerator** (1..* instances)
 - Füllt Serverdatenbank mit Werten aus einem
 - TestdatenbankZugang** (1 instance)
 - Ermöglicht die Eintragung von Messergebnissen in die Serverdatenbank mit einem
 - ServerWerteGenerator** (1 instance)
 - Ermöglicht den Umgang mit Testdaten mit einem

Prozess

- ProzessHandling** (1 instance)
- ProzessProfil** (1 instance)

ServerSteuerung

- Gui**
 - ucProzessListe** (1..* instances)
 - Stellt Testzustand her mithilfe einer
 - Erstellt Serverdatenbank mithilfe eines
 - Lädt und speichert Prozessprofile mit einem
 - frmTestAuswertung** (1 instance)
 - Führt statistische Überprüfung durch mit einer
 - ucTestAuswahl** (1 instance)
 - verarbeitet
 - ucStatistikAuswertung** (1 instance)
 - testet
 - ucEinzelAuswertung** (0..* instances)

Datenbank

- SQLScriptAusfuehrer** (1 instance)
- TestDaten** (1 instance)

Damit die Serverapplikation Prozessprofile erstellen kann, stellt die Serverbibliothek die Klasse *ProzessProfil* für das Laden und Speichern von Prozessprofilen zur Verfügung. Über die Klasse *ProzessHandling* werden Pfade zu laufenden Prozessen

ermittelt, um bei einem Testlauf Prozesse zu starten, die für den Test benötigt werden. Das Ermitteln der Pfade ist keine Funktionalität, die direkt vom .Net-Framework bereitgestellt wird. Dafür werden C-Bibliotheken vom Betriebssystem verwendet. Die Klasse *ProzessHandling* stellt eine *Wrapper-Klasse* für diese C-Bibliotheken dar.¹⁴⁸ Sie kapselt den Aufruf von externen C-Bibliotheken und stellt den aufrufenden Klassen C-Funktionen als .Net-Methoden zur Verfügung. Über das Starten von Prozessen, die bei vorherigen Tests schon aktiv waren, wird eine Vergleichbarkeit erreicht. Andererseits muss klar sein, dass die Aussagekraft der Performance-Tests geringer wird, wenn nicht für die Tests benötigte Prozesse aktiv sind. Um dies zu vermeiden, kann man sich auf notwendige Prozesse beschränken. Allerdings kann über die Möglichkeit, zusätzliche Prozesse zu erlauben, eine reale Umgebung simuliert werden (falls z. B. auf einem Server zusätzliche Applikationen aktiv sein sollen). Innerhalb dieser Arbeit wird sich auf zwingend notwendige Betriebssystemprozesse beschränkt.

Das zentrale Element der Serverapplikation, die sich im Paket *ServerSteuerung* befindet, stellt das Hauptfenster dar. Dieses Hauptfenster wird realisiert durch die Klasse *frmTestAuswertung* im Unterpaket *Gui*. Die Klasse *Programm* dient als Einstiegspunkt in die Applikation.

Das Hauptfenster besteht aus einer Prozessliste (*ucProzessListe*), welches das graphische Element für den Umgang mit Prozessen darstellt. Weiterhin kapselt es ein Element für die Überprüfung der Messergebnisse (*ucDatenUeberpruefer*). Zusätzlich wird ein Element verwendet, welches eine Auswahl des Szenarios und dessen Testfälle sowie das Zeichnen der einzelnen Graphen, basierend auf dieser Auswahl, übernimmt (*ucTestAuswahl*). Für die Darstellung der Graphen wird das Open Source-Element Zed Graph verwendet.¹⁴⁹ Dieses liegt in Codeform als Visual Studio Projekt¹⁵⁰ vor und stellt einen Teil des Codes auf der CD zu dieser Arbeit dar. Die Testauswahl kann mit einem bereits schon vorhandenen Zed Graph-Element erzeugt werden oder sie erzeugt selber eins, wenn sie ohne ein solches Element erzeugt wird. In beiden Fällen muss die erzeugende Klasse für die Anzeige des Zed Graph-Elements sorgen. Dadurch wird bei zukünftigen Weiterentwicklungen der Oberfläche den Entwicklern die Möglichkeit gegeben, selber zu entscheiden, wo und wie die Diagrammfläche erscheint.

148 Vergl. für Wrapper-Klasse Gamma, Erich et al. (2004), S. 171ff

149 Vergl. ZedGraph (2009)

150 Visual Studio ist eine Entwicklungsumgebung von Microsoft, welches primär für .Net-Entwicklung gedacht ist und für die Entwicklung in dieser Arbeit verwendet worden ist. Dabei wurde die Version Visual Studio 2005 verwendet. Vergl. Skibo, Craig et al. (2006) für eine Einführung in Visual Studio 2005.

Die Klasse *ucDatenUeberpruefer* ermöglicht die Überprüfung, ob die Wiederholungen der Messreihen ausreichend waren und ob die Messergebnisse nicht zu stark durch die äußeren Einflüsse verzerrt worden sind. Um die Ergebnisse der Messreihen anzuzeigen, legt die Prüfklasse Instanzen der Klasse *ucEinzelAuswertung* an, welche die Anzeige der Überprüfungsergebnisse der einzelnen Messreihen übernimmt.

Innerhalb des Paketes *Datenbank* in der Serversteuerung werden Klassen zur Verfügung gestellt, welche die Datenbeschaffung für die Serverapplikation übernimmt. Über die Klasse *SQLScriptAusführung* wird die Erstellung der Serverdatenbank mithilfe der Ausführung eines Erstellungsscript ermöglicht. Die Klasse *TestDaten* liefert der GUI die Testergebnisse für die Auswertung.¹⁵¹

6.8.6. Modell mobiler Client

In diesem Kapitel wird das Klassenmodell des mobilen Clients dargestellt. Auch hier werden die Pakete mit in die Klassendiagramme übernommen, um eine bessere Lesbarkeit zu erreichen.¹⁵² Weiterhin ist das gesamte Clientmodell in zwei Klassendiagramme geteilt. Innerhalb von Abbildung 23 wird die Struktur der mobilen Applikation (Paket *TestSuite*) vorgestellt, während Abbildung 24 die Bibliothek für die mobile Anwendung (Paket *TestLibraryClient*) vorstellt.

Die mobile Applikation enthält, wie auch die Serverapplikation, ein Hauptfenster (*frmTestSuite*) und einen Einstiegspunkt in die Applikation (*Programm*). Das Hauptfenster verwendet eine Prozessliste, um dem Benutzer die Möglichkeit zu geben, einen Testzustand auf dem mobilen Gerät herzustellen. Die Prozessliste selbst implementiert lediglich die Anzeige, während die eigentliche Logik für den Umgang mit Prozessen bzw. Prozessprofilen im Paket *TestLibraryClient::Prozess* der Bibliothek enthalten ist. Weiterhin hält das Hauptfenster eine Instanz der Klasse *ucTestSteuerung*, über die die eigentliche Teststeuerung realisiert ist.

Die Teststeuerung verwendet die Klassen *DBSTest* und *SyncTest*, welche die eigentliche Testlogik kapseln. Die Teststeuerung liefert die nötigen Parameter für die Tests und ermöglicht die Durchführung der vorbereitenden Anwendungsfälle, welche die mobile Datenbank und das Serversystem betreffen („*Mobile Datenbank anlegen*“, „*Datenbasis*

¹⁵¹ Die Klasse *TestDaten* stellt ein generiertes *DataSet* dar, welche die Datenhaltung- und Beschaffung kapselt. Vergl. für eine Einführung in *DataSets* Riordan, Rebecca M. (2006), S. 157ff. Weiterhin wird bei der Generierung eines *DataSet* ein eigenes Unterpaket angelegt, in dem die reale Kommunikation mit der Datenbank realisiert ist (*ServerSteuerung::Datenbank::TestDatenTableAdapters*, vergl. auch die Codedokumentation auf der CD).

¹⁵² Vergl. für dieses Vorgehen Booch, Grady et al. (2006), S. 139 und S. 202 sowie Oestereich, Bernd (2005), S.106 und Rupp, Chris et al (2007), S. 169

erstellen“). Dafür verwendet die Teststeuerung die Klasse *DatenbankVorbereiter* aus dem Paket *TestLibraryClient::Test::Datenbank*. Diese übernimmt die tatsächliche Durchführung der beiden Anwendungsfälle. Die beiden Klassen, welche die Testdurchführung realisieren, und die Klasse *DatenbankVorbereiter* übernehmen die Kommunikation mit dem Server sowie dem mobilen Datenbanksystem und koppeln somit die GUI von diesen beiden Systemen ab.

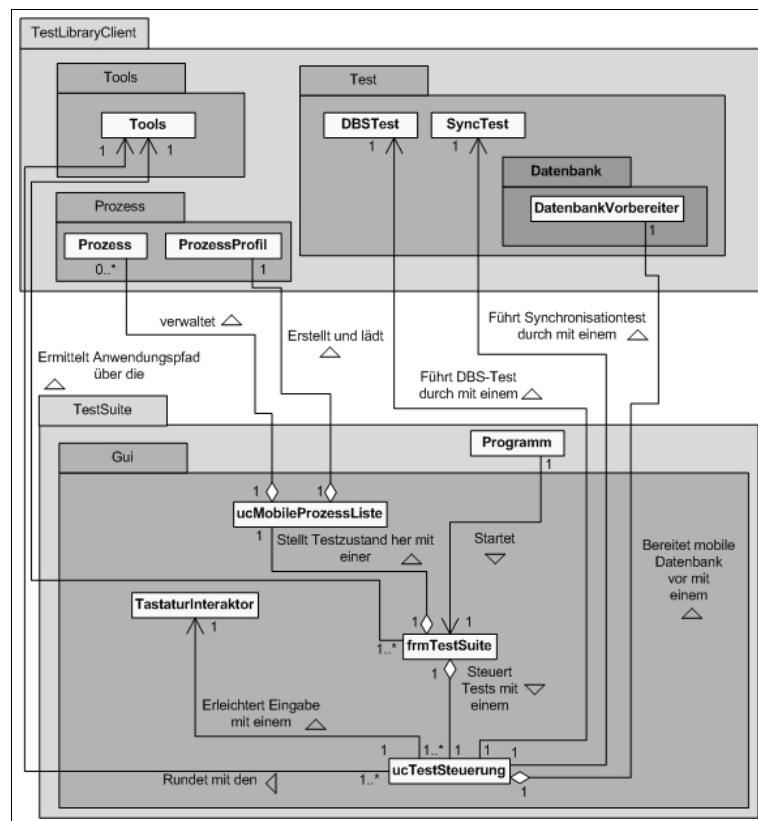


Abbildung 23: Modell mobile Testapplikation

Um die Eingabe zu erleichtern, verwendet die Teststeuerung eine Klasse zum Anzeigen der virtuellen Tastatur des Gerätes. Die virtuelle Tastatur stellt ein GUI-Element dar, mithilfe der über ein mobiles Gerät mit Touchscreen eine Eingabe über das Display ermöglicht wird. Diese virtuelle Tastatur lässt sich lediglich über eine C-Bibliothek ansprechen. Die Klasse *TastaturInteraktor* stellt eine Wrapper-Klasse für diese Bibliothek dar.

Die Erstellung von Prozessprofilen und von Testzuständen auf Clientseite wird über das graphische Element *ucProzessListe* ermöglicht. Für den Umgang mit einzelnen Prozessen greift die Prozessliste auf die Klasse *Prozess* zu. Die Klasse *Prozessprofil* übernimmt für die Prozessliste das Erstellen und Laden von Prozessprofilen.

Die Clientbibliothek, dargestellt in Abbildung 24, kapselt die eigentlichen

Funktionalitäten für die Tests und wird von der Oberfläche gesteuert. Die Klassen *DBSTest* und *SyncTest* bilden die Grundlage für die Tests. Sie halten den jeweiligen Status eines Testlaufs (z. B. welche vorbereitenden Schritte durchgeführt worden sind bzw. welche noch durchzuführen sind) und bieten jeweils die Durchführung des DBS- bzw. Synchronisationstests an. Um die Messergebnisse zu übertragen und Daten auf Serverseite zu generieren, halten sie jeweils eine Instanz der zugehörigen Stubklasse, welche die Kommunikation mit den Web Services realisieren.

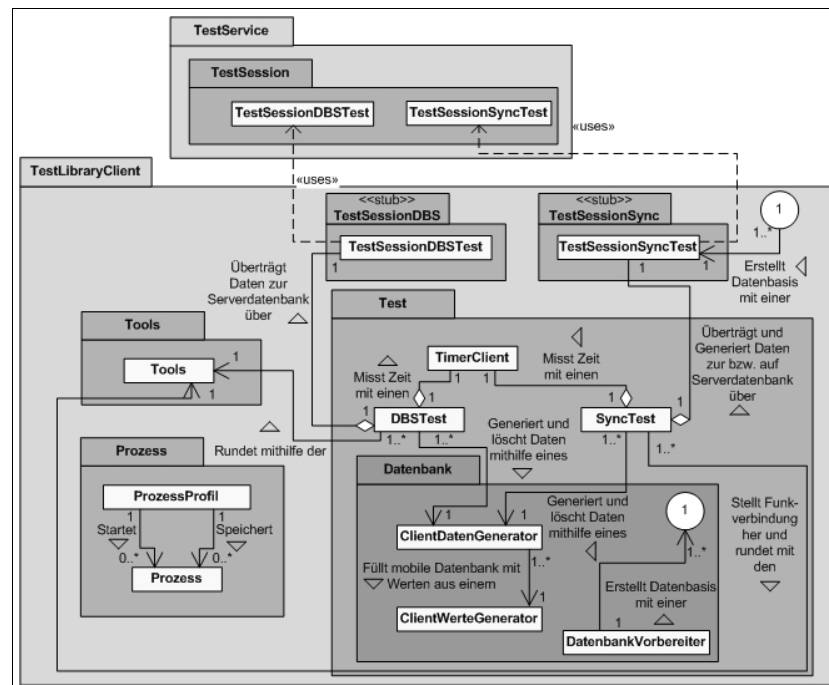


Abbildung 24: Modell Clientbibliothek

Weiterhin wird, wie auch in der Serverbibliothek, Funktionalitäten zum Umgang mit Prozessprofilen bereitgestellt (*ProzessProfil*). Für die Verwaltung der einzelnen

Prozesse wird eine andere Struktur als auf Serverseite verwendet, da das .Net Compact Framework keine Funktionalität zum Ermitteln aller aktiven Prozesse mitbringt. Zur Ermittlung dieser Prozesse wird ebenfalls die OpenNETCF-Bibliothek verwendet. Die einzelnen Prozessinformationen werden in *Prozess*-Objekten gekapselt, welche in der GUI für den Aufbau der Prozessliste verwendet werden. Innerhalb der Prozessklasse wird auch der Anwendungspfad zum Prozess ermittelt. Für das Laden und Speichern von Prozessprofilen verwendet die Klasse *ProzessProfil* Instanzen der *Prozess*-Klasse. Eine detaillierte Darstellung der Klassen ist in einer Codedokumentation auf der CD zu dieser Arbeit vorhanden.¹⁵⁴

6.9. Darstellung der Messergebnisse

In diesem Kapitel werden die Messergebnisse dargestellt. Die Überprüfungen der Messreihen sind erfolgreich durchlaufen. Dies kann auf die Korrektheit der Annahme der Normalverteilung und auf eine nicht allzu starke Verzerrung der Messungen durch die Umgebung deuten. Eine detaillierte Darstellung der Überprüfungen sämtlicher Testfälle ist in Anhang 7 zu finden. Bei den folgenden Ergebnissen wurde mit 10 Wiederholungen pro Messreihe getestet.

6.9.1. Ergebnisse Szenario 1 (Synchronisation von Daten)

Abbildung 25 zeigt das Ergebnis der Testfälle aus Szenario eins, welche die Funktechnologie GPRS verwenden, während Abbildung 26 die WLAN-Testfälle darstellt. Damit können diese Technologien zunächst einzeln betrachtet werden.

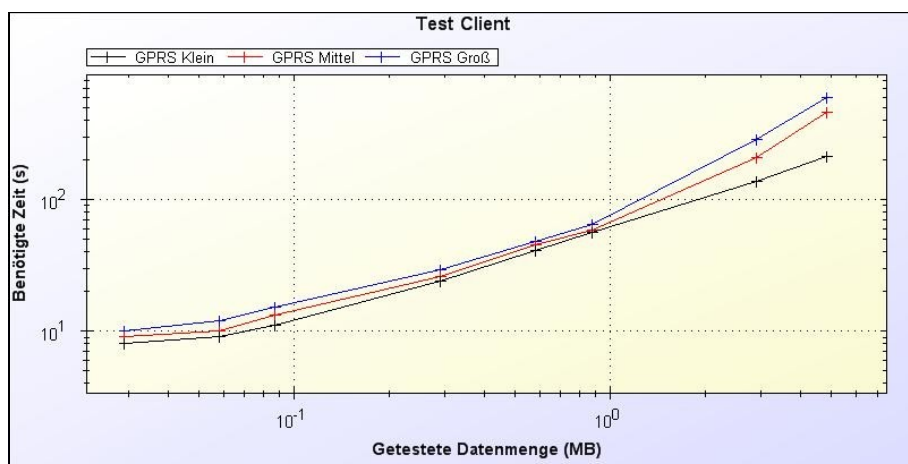


Abbildung 25: Ergebnis Szenario 1 – Testfälle für GPRS

In beiden Fällen dauert die Synchronisation bereits ab einer Datenmenge von 100 kB über eine Sekunde und steigt im MB-Bereich über 100 Sekunden. Ob solche

¹⁵⁴ Dort befindet sich auch der Quellcode sowie die Installationsdateien.

Synchronisationszeiten akzeptabel sind, hängt dabei vom jeweiligen Anwendungsfall und System ab. Weiterhin zeigt sich, dass speziell bei höheren zu synchronisierenden Datenmengen sich die Basisdatenmenge stärker auf die Laufzeit auswirkt.

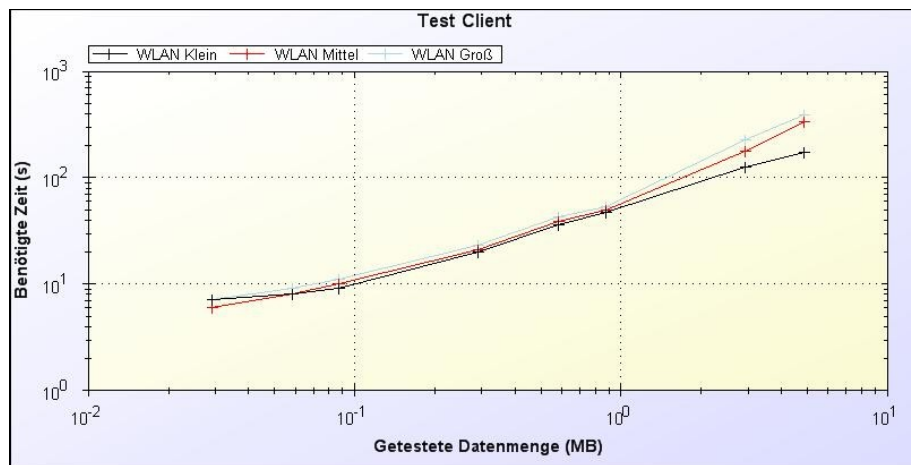


Abbildung 26: Ergebnis Szenario 1 – Testfälle für WLAN

Abbildung 27 zeigt alle sechs Testfälle des ersten Szenarios, um einen Vergleich von WLAN und GPRS zu ermöglichen.

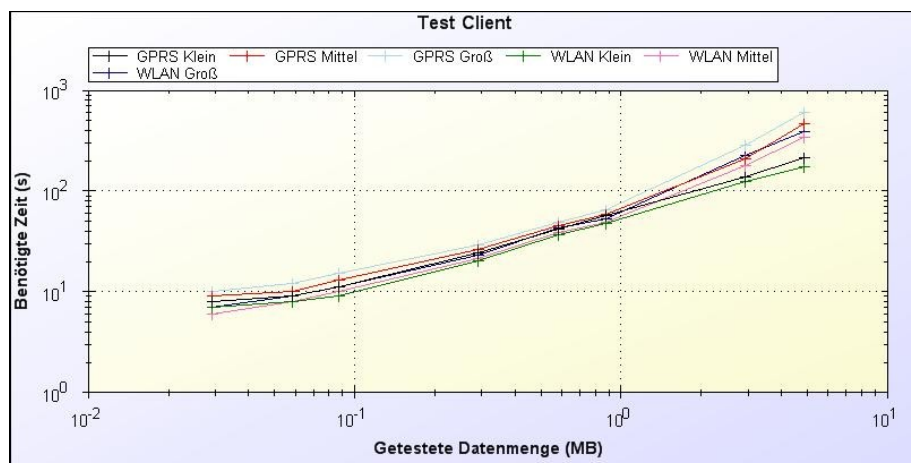


Abbildung 27: Ergebnis Szenario 1 – Gegenüberstellung WLAN und GPRS

Innerhalb von Abbildung 27 wird ersichtlich, dass es zwischen den beiden Funktechnologien zunächst keinen großen Unterschied gibt. Die GPRS-Testfälle liegen immer über ihrem Gegenstück aus den WLAN-Testfällen, aber ein drastischer Unterschied ist nicht erkennbar. Erst ab drei MB liegt der mittlere und der große GPRS-Testfall deutlicher über den mittleren und großen Testfall der WLAN-Reihe. Die geringe Übertragungsrate der GPRS-Technologie wirkt sich somit im MB-Bereich deutlicher aus. Somit ist die Grenze der Datenmenge, bis zu der die Synchronisation eingesetzt werden kann, stark abhängig von dem jeweiligen Anwendungsfall sowie die

dort auftretenden zu synchronisierenden Datenmengen.

6.9.2. Abschließende Betrachtung der Synchronisationsergebnisse

Abschließend betrachtet beschränkt die mobile Umgebung die Synchronisation je nach Datenbasis und Technologie auf maximale Datenmengen im einstelligen MB-Bereich (zwischen ca. 10.000 und 100.000 Zeilen). Bei einer darüber liegenden Menge ist durch den in den Messungen ermittelten Trend zu vermuten, dass eine Synchronisation durch Funktechnologien das Gerät im zweistelligen Minutenbereich und darüber blockiert. Hier kann eine stationäre Synchronisation mit einer kabelbasierten Netzwerktechnologie zur Vorbereitung für unterwegs oder eine Beschränkung auf Zeiträume, in denen das Gerät nicht benötigt wird, eine Lösung sein. Dieses Vorgehen verringert allerdings die Orts- und Zeitunabhängigkeit, welche den größten Vorteil eines Thick-Clients darstellen. Bei der Planung eines solchen Vorgehens muss also genau geprüft werden, ob der Einsatz eines Thin-Clients nicht besser auf den Anwendungsfall passt.

Die bidirektionale Synchronisation stellt somit eine funktionale Basis für einen aktuellen Datenbestand auf einen mobilen Thick-Client dar, solange die zu synchronisierenden Datenmengen im kB-Bereich liegen. Falls regelmäßig Datenmengen im MB-Bereich auftreten, kommt es auf die Performance-Anforderung an, die an Synchronisationen bzw. an die Applikation gestellt werden.

Mögliche Erweiterung dieses Testsystems ist die Einbeziehung von Testfällen, welche eine Synchronisation mithilfe einer stationären, also auf Kabeln basierende, Technologie durchführen. Dadurch würde ersichtlich werden, ob und in wie weit die kabelbasierten Technologien die Performance der Anbindung an ein zentrales System verbessern.

Ein weiteres Element, welches sich auf die Testfälle des Szenario 1 auswirken kann, ist die Frequenzteilung zwischen vielen Benutzern, z. B. in Ballungsgebieten. Dies stellt die Grundlage für ein weiteres zukünftiges Szenario.

Beim Trend, den die Kurven aufzeigen, wird die Laufzeit der Synchronisation bei beiden Technologien einen lediglich für Spezialfällen akzeptablen Bereich erreichen, sobald sich die zu synchronisierenden Datenmengen im zweistelligen MB-Bereich befinden. Um weiterhin die Verwendbarkeit von Synchronisationen in realen Anwendungsfällen einschätzen zu können, muss man somit bestimmen, wie häufig Datenmengen im MB-Bereich übertragen werden. Bei Fällen, in denen eine solche Last nur selten oder initial auftritt, sind somit akzeptable Laufzeiten im Sekundenbereich der

Normalfall. Weiterhin ist denkbar, die mobile Datenbank vor Auslieferung initial zu synchronisieren, sodass beim ersten Start eines Thick-Clients keine langwierige Synchronisation durchgeführt werden muss. Die Verwendung asynchroner Kommunikation sowie die Ausführung der Synchronisation in einem Thread, welcher im Hintergrund aktiv ist, erhöht ebenfalls die Verwendbarkeit des Systems. Dieses Vorgehen wurde auch in den Tests angewendet.

Trotz der Vorteile einer solchen Vorgehensweise ist der Aufwand der Synchronisation auf dem Gerät deutlich spürbar und schränkt die Leistungsfähigkeit des mobilen Geräts ein, was die Verwendung weiterer Applikationen erschwert. Dies hat sich bei der Testdurchführung gezeigt. Weiterhin können Techniken zur Verschlüsselung der Daten und zur sicheren Übertragung der Daten die Laufzeit verlängern.¹⁵⁵

Somit müssen beim Einsatz von Synchronisationen Lösungswege gefunden werden, mithilfe deren der Einsatz von mobilen Geräten möglichst wenig eingeschränkt wird. Dabei können lange Laufzeiten der Datenübertragungen umgangen werden, indem in möglichst geringen Zeitabständen synchronisiert wird und somit eine Anhäufung von zu übertragenden Daten verhindert wird.

Damit die Synchronisation nicht zu kritischen Zeitpunkten das Gerät blockiert, kann der Benutzer in die Planung und Durchführung der Synchronisation mit einbezogen werden. Dazu können von Benutzer vorgegebene Zeitpläne gehören, die Zeitbereiche angeben, in denen nicht synchronisiert wird. Abbruchmechanismen, die der Benutzer auslösen kann, sind ebenfalls ein möglicher Weg. Somit ist sichergestellt, dass das Gerät zu Zeitpunkten, an denen es dringend benötigt ist, nicht durch eine Synchronisation gestört wird bzw. in denen zumindest die Synchronisation abgebrochen werden kann. Erhöhter Stromverbrauch durch eine hohe Belastung des Geräts muss ebenfalls betrachtet werden. Somit muss ein Mittelweg zwischen der Laufzeit und der Häufigkeit der Synchronisation gefunden werden.

Wie sich bei den Messungen gezeigt hat, muss bei der Planung eines Thick-Clients ebenfalls die Datenbasis betrachtet werden, welche sich zum Zeitpunkt einer Datenübertragung in der mobilen Datenbank befindet. Diese wirkt sich direkt auf die Laufzeiten der Synchronisation aus und verzögert diese um Sekunden bis Minuten.

¹⁵⁵ Vergl. für eine Einführung in Sicherheitstechniken für Funktechnologien, vergl. Eckert, Claudia (2007), Kapitel 13 (S. 785ff)

6.9.3. Ergebnisse Szenario 2 (Anfragen zur Ermittlung von Daten aus der mobilen Datenbank)

Abbildung 28 zeigt das Ergebnis der Testfälle des Szenario 2. Da es sich bei allen drei Testfällen um Abfragen unter denselben Voraussetzungen handelt, die sich lediglich in der Komplexität unterscheiden, werden sie zum Vergleich in einem Diagramm dargestellt.

Dabei zeigt sich ein starker Unterschied der Laufzeiten der komplexen Anfrage zu den beiden anderen Testfällen. Während die Laufzeit der Abfrage mit mittlerer und niedriger Komplexität bis zu mehreren tausend Zeilen sich im Bereich der Millisekunden befinden, liegen die Laufzeiten der komplexen Abfrage bereits bei Datenmengen über 1000 Datensätze im Sekundenbereich.

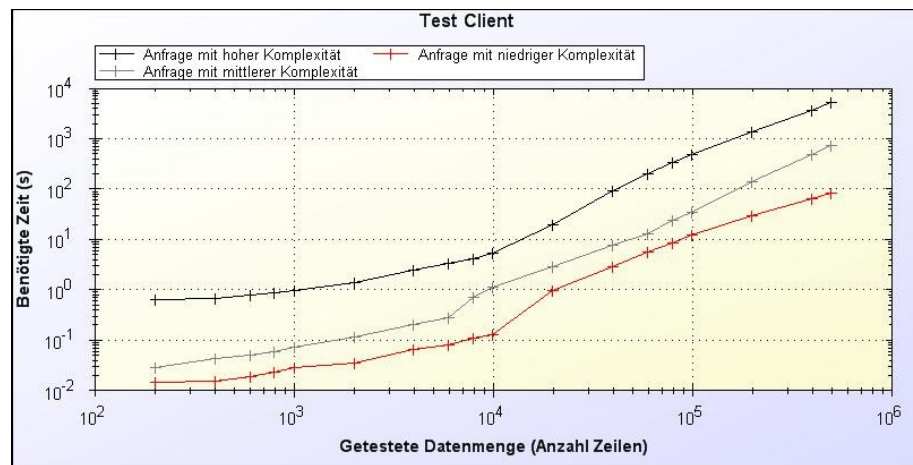


Abbildung 28: Ergebnis Szenario 2

Die Laufzeiten der Abfragen steigen im Bereich von 10.000 Datensätze und mehr stärker an. Dabei befindet sich die Laufzeit der komplexen Abfrage bei einer Datenmenge von 20.000 Datensätzen bereits im zweistelligen Sekundenbereich. Solche Laufzeiten stellen für viele Applikationen eine zu große Verzögerung durch die Ermittlung von Daten dar. Die beiden anderen Abfragen erreichen diese Zeiten erst im Bereich zwischen 60.000 und 100.000 Datensätze. Datenmengen, die über diesen Grenzen liegen, führen zu Laufzeiten, die sich dem Minutenbereich stark annähern bzw. überschreiten. Dabei ist die komplexe Abfrage hervorzuheben, bei der Laufzeiten über eine Stunde erreicht werden. Selbst die Laufzeit der einfachen Abfrage ist mit einer Laufzeit von ca. 100 Sekunden bei 500.000 Datensätzen weit über den durchschnittlichen Performance-Anforderungen, welche an eine Datenhaltung gestellt werden.

Somit ist die Laufzeit von Abfragen bis zu einer Datenmenge von 10.000 Datensätzen im akzeptablen Bereich, während speziell Abfragen, die mehrere Relationen betreffen, über einer solchen Datenmenge zu hohe Laufzeiten aufweisen. Innerhalb eines Anwendungsfalls muss somit abgeschätzt werden, welche Datenmengen auftreten sowie welche Komplexität die Abfragen erreichen können. Mithilfe einer solchen Abschätzung sowie der hier ermittelten Ergebnisse wird ersichtlich, ob das mobile DBS die an den Anwendungsfall gestellten Performance-Anforderungen erfüllt.

6.9.4. Ergebnisse Szenario 3 (Datenmanipulationen)

Im folgenden werden die Ergebnisse der Testfälle von Szenario drei dargestellt. Da die Testfälle unterschiedliche Operationen auf der Datenbank darstellen, sind sie nicht direkt vergleichbar wie die Testfälle aus Szenario eins, welche auf derselben Technologie basieren, bzw. die Testfälle aus Szenario zwei, und erhalten somit jeweils ein eigenes Diagramm.

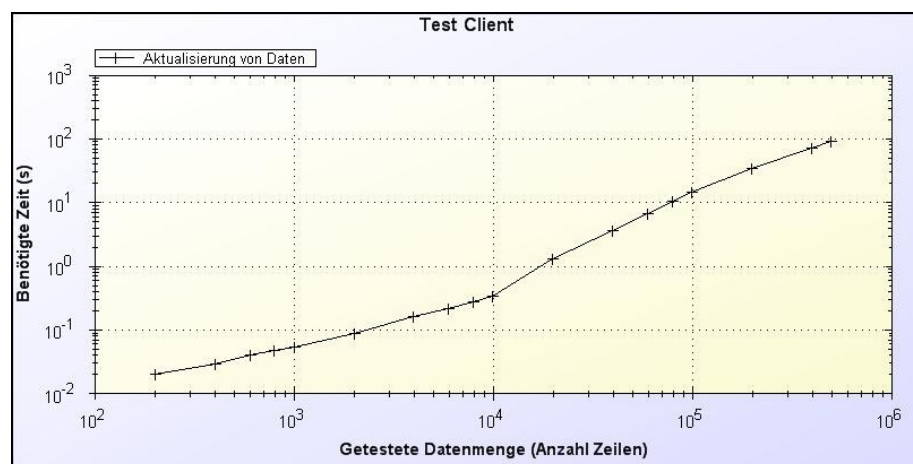


Abbildung 29: Ergebnis Szenario 3 – Testfall „Aktualisierung von Daten“

Innerhalb von Abbildung 29 werden die Ergebnisse des Testfalls „Aktualisierung von Daten“ aus Szenario drei dargestellt. Hier wird der Sekundenbereich bei 20.000 Datensätzen und der zweistellige Sekundenbereich bei 100.000 Datensätzen erreicht.

Somit verhalten sich die Aktualisierungen einzelner Zeilen ähnlich wie die einfache Abfrage aus Szenario zwei. Weiterhin zeigt sich ein stärkerer Anstieg der Laufzeiten der Aktualisierung, sobald sich die Datenmenge im Bereich über 10.000 Datenzeilen bewegt.

Laufzeiten unter einer Sekunde, welche eine gute Laufzeit für Applikationen darstellen, erreicht man bis zu einer Datenmenge von 20.000 Datensätzen, während akzeptable Laufzeiten im Sekundenbereich bis max. 100.000 Datensätze möglich sind. Dabei hängt

es wieder von den jeweiligen Performance-Anforderungen ab, welche Laufzeiten hingenommen werden können.

Innerhalb des Testfalls der Aktualisierung wird lediglich ein Datensatz geändert. Bei einer größeren Anzahl an zu aktualisierenden Zeilen in einer Anfrage bzw. bei der Kombination mit einer Anfrage zur Datenermittlung erhöht sich entsprechend die Laufzeit.

Das Ergebnis des Testfalls „*Einfügen von Daten*“ ist in Abbildung 30 dargestellt. Dies stellt das beste Ergebnis der drei Szenarien dar. Selbst bei Datenmengen über 100.000 Datensätzen bleibt die Laufzeit für das Einfügen eines Datensatzes unter einer Sekunde. Hier zeigen sich weiterhin speziell im unteren Bereich der Kurve Abweichungen durch äußere Einflüsse wie z. B. Betriebssystemprozesse.

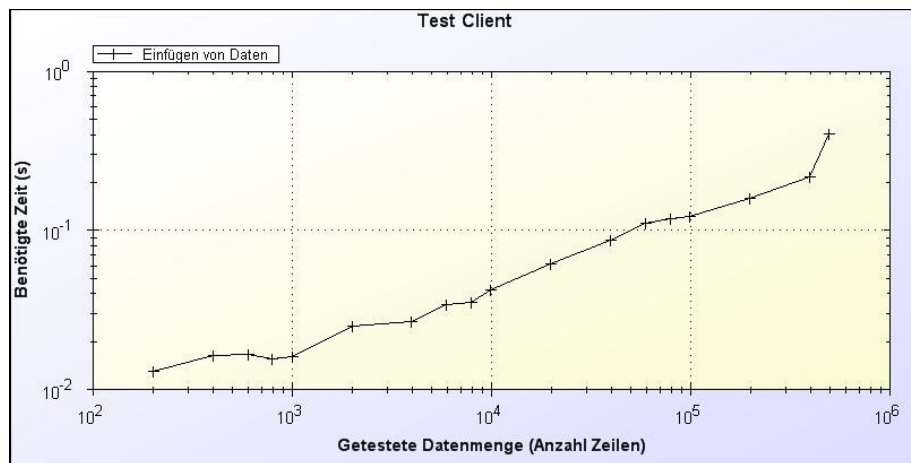


Abbildung 30: Ergebnis Szenario 3 – Testfall „Einfügen von Daten“

Somit stellt das Einfügen von einzelnen Datensätzen auch bei hohen Datenmengen kein Hindernis für den Einsatz von mobilen Datenbanksystemen dar. Lediglich das Verbinden der Operationen zum Einfügen von Daten mit Abfragen, um z. B. Daten aus einer Tabelle in eine zweite Tabelle zu übernehmen oder Fremdschlüssel zu ermitteln, kann die Laufzeit stark erhöhen. Dies ist abhängig von der Komplexität der verwendeten Anfrage.

Abbildung 31 stellt das Ergebnis des Testfalls „*Löschen von Daten*“ dar. Dabei verhält sich das Löschen ähnlich wie die Aktualisierung von Daten. Der Unterschied liegt in den geringeren Laufzeiten unter 10.000 Datenzeilen, während die Laufzeiten des Löschen über 10.000 Datenzeilen stärker ansteigen als die Laufzeiten der Aktualisierung. Der zweistellige Sekundenbereich wird bei 60.000 Datenzeilen erreicht. Somit gelten für das Löschen ähnliche Beschränkungen wie für die Aktualisierung. Bis

zu einer Datenmenge unter 20.000 Datenzeilen bleibt das Löschen bei einer Laufzeit unter einer Sekunde, während bei über 60.000 Datenzeilen der zweistellige Sekundenbereich erreicht wird.

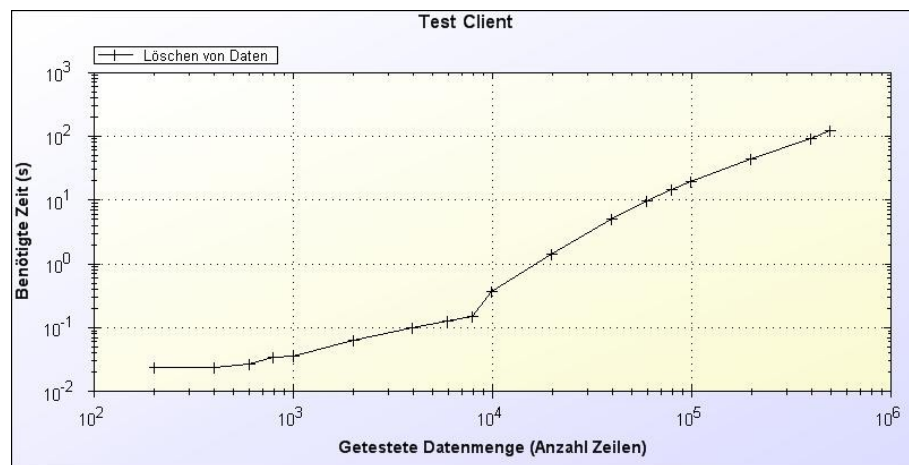


Abbildung 31: Ergebnis Szenario 3 – Testfall „Löschen von Daten“

6.9.5. Abschließende Betrachtung der Datenbankergebnisse

Zusammenfassend betrachtet unterliegt das mobile DBS ähnlichen Einschränkungen wie die Synchronisation. Bei der Verwendung von Anfragen an DBS sind Laufzeiten im Sekundenbereich bereits an der Grenze zur Unbrauchbarkeit für viele Anwendungsfälle. Dieser Bereich wird, mit Ausnahme der komplexen Abfrage und dem Einfügen von neuen Datensätzen, bereits zwischen 10.000 und 100.000 Datenzeilen erreicht. Im Bereich über 100.000 Zeilen erreichen selbst Anfragen, welche lediglich aus einer Tabelle Daten ermitteln, Laufzeiten in zweistelligen Sekundenbereich, während sich komplexere Abfragen bei einer solchen Datenmenge in der Nähe des Minutenbereichs oder bereits darüber hinaus befinden. Für einen Thick-Client, welcher seine Daten aus der mobilen Datenbank ermittelt bzw. seine Daten dort speichert, stellen solche Laufzeiten ein großes Hindernis dar bzw. sorgen für eine Unbrauchbarkeit der Software. Somit muss versucht werden, beim Einsatz eines Thick-Clients auf mobilen Geräten die Datenmenge passend zur Komplexität der Anfragen innerhalb des jeweiligen Anwendungsfalls sowie den jeweiligen Performanceansprüchen zu halten. Für diesen Zweck wäre eine Personalisierung der Daten denkbar, sodass nur benutzerspezifische Daten auf das mobile Gerät geladen werden und somit die Datenmenge reduziert wird. Für diesen Zweck muss ein entsprechendes Benutzersystem mit in die Replikation aufgenommen werden, welches die einzelnen Benutzer identifiziert und zuordnet, welche Informationen für den jeweiligen Benutzer zur Verfügung stehen.

Innerhalb von Anwendungsfällen, in denen mobile Geräte zum Sammeln von Daten für ein zentrales System verwendet werden, bietet sich eine Löschung der gesammelten Daten auf dem mobilen Gerät an, sobald sich diese im zentralen System befinden.

Ein weiterer Lösungsansatz zur Reduzierung der Datenmenge auf dem mobilen Gerät ist eine zeit- und ortsabhängige Replikation. Dabei werden die Daten auf das mobile Gerät synchronisiert, die vom Benutzer für die Erledigung von Aufgaben zu bestimmten Zeitpunkten bzw. Zeitbereichen an bestimmten Orten benötigt werden. Ein ähnliches Konzept für Thin-Clients auf mobilen Geräten stellen die Location Based Services dar, welche orts- und zeitabhängige Schnittstellen bereitstellen.¹⁵⁶ Dabei wird zur Positionsbestimmung des mobilen Gerätes das **Global Positioning System (GPS)** verwendet.

Der Unterschied der Replikation zu diesen Schnittstellen, welche zur Laufzeit den aktuellen Ort sowie die aktuelle Zeit verwenden kann, besteht in der Unkenntnis der Replikationsschnittstelle, für welche Orte bzw. welche Zeiträume der Benutzer zum Zeitpunkt der Synchronisation Daten benötigt. Daher müssen zusätzliche Informationen vom Benutzer ermittelt werden, die eine entsprechende Replikation der benötigten Daten ermöglicht und weiterhin alte bzw. nicht mehr benötigte Daten identifiziert und löscht. Die Grundlage für ein solches Vorgehen stellt das Konzept der benutzerdefinierten Replikation dar (Vergl. auch Kapitel 4.3.3). So kann eine dynamische Synchronisation realisiert werden, welche die Datenmengen auf dem mobilen Gerät verringert und somit das mobile DBS entlastet. Weiterhin können so flexiblere Anwendungsfälle möglich gemacht werden.¹⁵⁷

Der Nachteil besteht hier in der Komplexität der Replikationsschnittstellen sowie einem erhöhten Aufwand und Dauer der Synchronisation durch zusätzliche Bedingungen.

Im Falle von Thick-Clients, welche für den Einsatz ohne ein zentrales System gedacht sind, müssen dem Benutzer entsprechende Kontrollen der Datenmengen zur Verfügung gestellt werden. Dazu können Löschmasken oder Funktionen zur Übertragung von älteren Datensätzen in mobile Datenbanken, welche zur Archivierung dienen, gehören.

Das nachfolgende Kapitel fasst diese Ergebnisse zusammen und zieht ein Fazit.

¹⁵⁶ Vergl. Schiller, Jochen et al. (2004), S. 10ff sowie Kapitel für eine Einführung in Location Based Services

¹⁵⁷ Vergl. für die benutzerdefinierte Replikation zusätzlich zu Kapitel 4.3.3 Mutschler, Bela et al. (2004), S. 100ff und Weikum, Gerhard et al. (2003), S. 453ff

7. Ergebnis und Fazit

Im vorherigen Kapitel sind die Ergebnisse der Messungen dargestellt. Darin wird speziell die Begrenzung durch die geringere Leistungsfähigkeit des mobilen Gerätes sichtbar. Somit stellt das mobile DBS bis zu 10.000 Datensätzen ein stabiles System dar, welches auch komplexe Anfragen im Millisekundenbereich verarbeiten kann. Bei Datenmengen darüber hängt die Verwendbarkeit mobiler DBS von der Komplexität der Anfragen sowie der Performanceanforderungen an die jeweilige Applikation ab.

Ähnliches gilt für die Synchronisation. Bis zu einer Datenmenge von einem MB (10.000 Datensätze) bleiben die Laufzeiten der Synchronisation im Bereich um einer Minute. Darüber steigt die Laufzeit stärker an und erreicht eine Laufzeit von mehreren Minuten bei 5 MB (50.000 Datensätze), wobei der Trend der Messergebnisse eine zunehmende Steigung aufweist. Die Unterschiede in der Übertragungsrate zwischen den getesteten Funktechnologien GPRS und WLAN wirken sich erst im MB-Bereich deutlich aus.

Um Performance-Anforderungen an das mobile DBS und die Synchronisation auch bei einer großen Datenmenge erfüllen zu können, können weiterführende Modelle verwendet werden, die die Last für beiden Systemteile senkt.

Abschließend betrachtet können Thick-Clients innerhalb der ermittelten Grenzen von mobilen DBS und Synchronisationen unterstützt werden. Beim verwendeten Beispiel eines Vertriebsmitarbeiter ist z. B. die Frage, ob eine Gesamtmenge aus 10.000 Datensätzen auftritt. Dabei können verschiedene Modelle und Vorgehensweisen eingesetzt werden, um dem Benutzer eine Kontrolle über die auf dem mobilen Gerät vorhandene Datenmenge zu geben. So wird eine Anbindung an ein zentrales System ermöglicht, welches eine wesentlich höhere Datenmenge enthält als die einzelnen Clients.

Weiterhin muss bei mobilen Szenarien, bei denen Datenmengen über denen in dieser Arbeit ermittelten Grenzen auftreten, abgewogen werden, ob sich der Aufwand lohnt, den die Planung und Umsetzung von den vorgestellten komplexeren Modellen und Vorgehensweisen mit sich bringen.

Bei dieser Abwägung kann der Aufbau des zentralen Systems die Auswahl erleichtern bzw. bestimmte Modelle unterstützen. Der Microsoft SQL Server z. B. bietet von sich aus eine Zuteilung der Daten zu bestimmten Geräten an. Da häufig dasselbe Gerät vom selben Benutzer verwendet wird, kann an dieser Stelle leicht eine personalisierte

Synchronisation ermöglicht werden. Damit ließe sich das Beispiel des Vertriebsmitarbeiters erweitern, falls hier zu hohe Datenmengen auftreten. Man kann den einzelnen Mitarbeitern die Daten zuweisen, die sie benötigen und für die sie verantwortlich sind. Somit lässt sich einfacher eine Personalisierung der Daten (und somit eine Reduzierung der Datenmenge pro Gerät) realisieren, als wenn diese vollständig selbst entwickelt werden müsste.

Falls trotz der Modelle eine hohe Datenmenge nicht vermeidbar ist oder komplexere Modelle nicht gewollt sind, stellt sich die Frage, ob die Aufgabe der vollständigen Ortsunabhängigkeit und die Verwendung eines Thin-Clients nicht die bessere Lösung ist. Damit kann die Datenlast auf leistungsstarke Server ausgelagert werden. Allerdings begrenzt sich der Einsatzbereich des Clients auf Gebiete, in denen eine Funkverbindung möglich sind.

In dieser Arbeit ist gezeigt worden, wie die leistungsschwache mobile Umgebung sowie die Funktechnologien die Performance einer Anbindung an ein zentrales System beschränkt. Dabei ist davon auszugehen, dass sich die Leistungsfähigkeit der mobilen Computersysteme wie z. B. Handhelds oder Mobiltelefone in Zukunft vergrößern wird. Die Ausbreitung der Funktechnologien der 3. Generation sowie der nachfolgenden Generationen erleichtern weiterhin den Umgang mit dem zentralen System. Somit können größere Datenmengen auf das mobile Gerät geladen bzw. von dem mobilen DBS verarbeitet werden.

Zukünftig können weiterführende Szenarien aufgebaut werden, die die Performance von Thin-Clients testen. Damit können Vergleiche zwischen reinen Online-Szenarien und den Offline-Szenarien durchgeführt werden. Weiterführend können andere Modelle für die Verteilung von Daten oder der Synchronisierung bzw. andere Datenbanksysteme getestet werden. Basierend auf solchen zusätzlichen Testergebnissen kann die Leistungsfähigkeit der mobilen Geräte, Daten zu speichern und zu verarbeiten, beurteilt werden. Basierend darauf kann entschieden werden, welche Modelle und Datenbanksysteme für konkrete Anwendungsfälle und Performance-Ansprüche geeignet sind.

Literaturverzeichnis

a) Monographien

Appelrath, Hans-Jürgen; Ludewig, Jochen: Skriptum Informatik – Eine konventionelle Einführung. Teubner Verlag, Leipzig 2000

Baker, Paul; Dai, Zhen Ru; Grabowski, Jens; Hagen, Øystein; Schieferdecker, Ina; Williams, Clay: Model-Driven Testing – Using the UML Testing Profile. Springer, Berlin 2008

Bauder, Irene: Microsoft SQL Server 2005 für Administratoren. Hanser, München 2006

Bengel, Günther: Grundkurs Verteilte Systeme. Vieweg+Teubner, Wiesbaden 2004

Beydeda, Sami (Hrsg.); Gruhn, Volker (Hrsg.); Mayer, Johannes (Hrsg.); Reussner, Ralf (Hrsg.); Schweiggert, Franz (Hrsg.): Testing of Component-Based Systems and Software Quality – Net.ObjectDays Workshops on Testing of Component-Based Systems (TECOS 2004) and Software Quality (SOQUA 2004). Köllen, Bonn 2004

B'Far, Reza; Behravanfar, Reza; Lindsay, Phillip: Mobile Computing Principles – Designing and Developing Mobile Applications with UML and XML. Cambridge University Press, Cambridge 2005

Boddenberg, Ulrich B.: Windows Mobile Integrations- und Entwicklungsbuch – Architektur, Hintergründe und Programmierung der mobilen Windows-Technologien. Microsoft Press, Unterschleißheim 2007

Booch, Grady; Rumbaugh, Jim; Jacobson, Ivar: Das UML-Benutzerhandbuch. Addison Wesley, München 2006

Borg, Ingwer; Staufenbiel, Thomas: Lehrbuch – Theorien und Methoden der Skalierung. Huber, Bern 2007

Bowers, David (Hrsg.): Directions in Databases – 12th British National Conference on Databases, BNCOD 12 Guildford, United Kingdom, July 1994 – Proceedings. Springer, Berlin 1994

Bronstein, I.E.; Semendjajew, K.A.; Musiol, G.; Mühlig, H.: Taschenbuch der Mathematik. Verlag Harri Deutsch, Frankfurt am Main 2008

Burnus, Heinz: Datenbankentwicklung in IT-Berufen – Eine praktisch orientierte Einführung mit MS Access und MySQL. Vieweg+Teubner, Wiesbaden 2008

Buschmann, Frank; Meunier, Regine; Rohnert, Hans; Sommerlad, Peter; Stal, Michael: Pattern-orientierte Softwarearchitektur – Ein Pattern-System. Addison Wesley, München 2000

Bünning, Uwe; Krause, Jörg: Windows Server 2003: Einrichtung und Administration von Unternehmensnetzen mit Standard und Enterprise Edition. Hanser Fachbuchverlag, München 2006

Bünning, Uwe; Rzepka, Dirk: Microsoft Windows Server 2008 – Einrichten und Verwalten von Unternehmensnetzwerken. Hanser Fachbuchverlag, München 2008

Connolly, Thomas M.; Begg, Carolyn E.: Database Systems: A Practical Approach to Design, Implementation and Management. Pearson Education, Harlow 2002

Coulouris, George; Dollimore, Jean; Kindberg, Tim: Distributed Systems – Concepts And Design. Addison Wesley, Harlow 2005

Delaney, Kalen: Inside Microsoft® SQL Server™ 2005: The Storage Engine. Microsoft Press, Redmond 2007

Dirlewanger, Werner: Measurement and Rating of Computer Systems Performance and of Software Efficiency – An Introduction to the ISO /IEC 14756 Method and a Guide to its Application. kassel university press, Kassel 2006

Dunkel, Jürgen; Eberhart, Andreas; Fischer, Stefan; Kleiner, Carsten; Koscher, Arne: Systemarchitekturen für verteilte Anwendungen – Client-Server, Multi-Tier, SOA, Event Driven Architecture, P2P, Grid, Web 2.0. Hanser, München 2008

Eckert, Claudia: IT-Sicherheit – Konzepte – Verfahren – Protokolle. Oldenbourg, München 2007

Ferstl, Otto K.; Sinz, Elmar J.; Eckert, Sven; Isselhorst, Tilman (Hrsg.): Wirtschaftsinformatik 2005: eEconomy, eGovernment, eSociety. Physica-Verlag, Heidelberg 2005

Ford, Chris; Gileadi, Ido; Purba, Anjiv; Moerman, Mike (2008): Patterns for Performance and Operability – Building and Testing Enterprise Software. Auerbach Publications, Boca Raton 2008

Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John: Entwurfsmuster – Elemente wiederverwendbarer objektorientierter Software. Addison Wesley, München 2004

Green, James Harry: The Irwin Handbook of Telecommunications. McGraw-Hill, New York 2005

- Gschwind, Thomas; Mascolo, Cecilia (Hrsg.): Software Engineering and Middleware – 4th International Workshop, SEM 2004 Linz, Austria, September 20-21, 2004 Revised Selected Papers. Springer, Berlin 2005
- Gulutzan, Peter; Pelzer, Trudy: SQL Performance Tuning. Addison Wesley, Boston 2006
- Häckelmann, Heiko; Petzold, Hans J.; Strahinger, Susanne: Kommunikationssysteme – Technik und Anwendungen. Springer, Berlin 2000
- Hanhart, Daniel: Mobile Computing und RFID im Facility Management: Anwendungen, Nutzen und serviceorientierter Architekturvorschlag. Springer, Berlin 2008
- Hartung, Joachim; Heine, Barbara: Statistik-Übungen – Induktive Statistik. Oldenbourg, München 2004
- Henze, Norbert: Stochastik für Einsteiger – Eine Einführung in die faszinierende Welt des Zufalls. Vieweg+Teubner, Wiesbaden 2006
- Heutschi, Roger: Serviceorientierte Architektur – Architekturprinzipien und Umsetzung in die Praxis. Springer, Berlin 2007
- Hoffmann, Dirk W.: Software-Qualität. Springer, Berlin 2008
- Höpfner, Hagen; Türker, Can; König-Ries, Birgitta: Mobile Datenbanken und Informationssysteme – Konzepte und Techniken. dpunkt Verlag, Heidelberg 2005
- Jain, Raj: The Art of Computer Systems Performance Analysis – Techniques for Experimental Design, Measurement, Simulation, and Modeling. John Wiley & Sons, New York 1991
- Kurose, James E.; Rose, Keith W.: Computer Networking – A Top-Down Approach. Pearson Education, Boston 2008
- Lehner, Franz: Mobile und drahtlose Informationssysteme – Technologien, Anwendungen, Märkte. Springer, Berlin 2003
- Lockemann, Peter C.; Dittrich, Klaus R.: Architektur von Datenbanksystemen. dpunkt Verlag, Heidelberg 2004
- Malle, H.: Handbuch Mathematik. Nikol Verlag, Frankfurt am Main 2006
- Matthiessen, Günter; Unterstein, Michael: Relationale Datenbanken und SQL – Konzepte der Entwicklung und Anwendung. Addison Wesley, München 2003
- Mathur, Aditya P.: Foundations of Software Testing. Addison Wesley, Delhi 2008

Meier, Andreas; Wüst, Thomas: Objektorientierte und objektrelationale Datenbanken – Ein Kompass für die Praxis. dpunkt Verlag, Heidelberg 2003

Meier, J.D.; Vasireddy, Srinath; Babbar, Ashish; Mackman, Alex: Improving .NET Application Performance and Scalability – Patterns & Practices. Microsoft Press, Redmond 2004

Mertens, Stefan: XML-Komponenten in der Praxis. Springer, Berlin 2003

Mosler, Karl; Schmid, Friedrich: Wahrscheinlichkeitsrechnung und schließende Statistik. Springer, Heidelberg 2006

Murthy, C. Siva Ram; Manoj, B.S: Ad Hoc Wireless Networks – Architectures and Protocols. Prentice Hall, New Jersey 2004

Mutschler, Bela; Specht, Günther: Mobile Datenbanksysteme. Springer, Berlin 2004

Oestereich, Bernd: Die UML 2.0 Kurzreferenz für die Praxis – kurz, bündig, ballastfrei. Oldenbourg, München 2005

Özsu, M. Tamer; Valduriez, Patrick: Principles Of Distributed Database Systems – Second Edition. Prentice Hall, New Jersey 1999

Ramez, Elmasri; Navathe, Shamkant B.: Fundamentals of Database Systems. Addison Wesley, Boston 2007

Rechenberg, Peter (Hrsg.); Pomberger, Gustav (Hrsg.): Informatik Handbuch. Hanser, München 2006

Reichwald, Ralf: Mobile Kommunikation – Wertschöpfung, Technologien, neue Dienste. Gabler, Wiesbaden 2002

Riordan, Rebecca M.: Microsoft ADO.NET 2.0 – Schritt für Schritt. Microsoft Press, Unterschleißheim 2006

Rob, Peter; Coronel, Carlos; Crockett, Keeley: Database Systems – Design, Implementation & Management. Cengage Learning Services, London 2008

Rupp, Chris; Queins, Stefan; Zengler, Barbara: UML 2 Glasklar – Praxiswissen für die UML-Modellierung. Hanser, München 2007

Saake, Gunter; Sattler, Kai-Uwe; Heuer, Andreas: Datenbanken – Konzepte und Sprachen. Redline GmbH, Heidelberg 2008

Schiller, Jochen; Voisard, Agnès: Location-Based Services. Morgan Kaufmann Publishers, San Francisco 2004

Schneider, Markus: Implementierungskonzepte für Datenbanksysteme. Springer, Berlin 2003

Shanmugam, Ramadas; Padmini, R.; Nivedita, S.: Using TCP/IP – Special Edition. QUE, Indianapolis 2002

Skibo, Craig; Young, Marc; Johnson, Brian: Arbeiten mit Microsoft Visual Studio 2005 – Mit Makros und individueller Konfiguration Visual Studio 2005 optimieren. Microsoft Press, Unterschleißheim 2006

Sneed, Harry M.; Baumgartner, Manfred; Seidl, Richard: Der Systemtest – Von den Anforderungen zum Qualitätsnachweis. Hanser Fachbuch, München 2009

Spillner, Anderas; Linz, Tilo: Basiswissen Softwaretest – Aus- und Weiterbildung zum Certified Tester. dpunkt Verlag, Heidelberg 2007

Stein, Erich: Taschenbuch Rechnernetze und Internet. Hanser Fachbuch, München 2007

Tannenbaum, Andrew S. (2003): Moderne Betriebssysteme. Pearson Studium, München 2003

Steiner, René: Grundkurs Relationale Datenbanken – Einführung in die Praxis der Datenbankentwicklung für Ausbildung, Studium und IT-Beruf. Vieweg Verlag, Wiesbaden 2003

Tannenbaum, Andrew S.; van Steen, Maarten: Verteilte Systeme – Prinzipien und Paradigmen. Pearson Studium, München 2008

Terry, Douglas B.; Satyanarayanan, Mahadev (Hrsg): Replicated Data Management for Mobile Computing – Synthesis Lectures on Mobile and Pervasive Computing. Morgan & Claypool, San Rafael 2007

Thaller, Georg Erwin: Software-Test – Verifikation und Validation. Heise, Hannover 2002

Tse, David; Viswanath, Pramod: Fundamentals of Wireless Communication. Cambridge University Press, New York 2006

Türker, Can; Saake, Gunter: Objektrelationale Datenbanken – Ein Lehrbuch. dpunkt Verlag, Heidelberg 2006

Vossen, Gottfried: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme. Oldenbourg, München 2008

Walke, Bernhard; Bossert, Martin (Hrsg.); Fliege, Norbert (Hrsg): Mobilfunknetze und ihre Protokolle 1 – Grundlagen, GSM, UMTS und andere zellulare Mobilfunknetze. Vieweg+Teubner, Stuttgart 2001

Weikum, Gerhard; Schöning, Harald; Rahm, Erhard (Hrsg.): BTW 2003: Datenbanksysteme für Business, Technologie und Web – 10. GI-Fachtagung, Leipzig 26. - 28. Februar. Gesellschaft für Informatik, Bonn 2003

Wilrich, P-Th.; Henning, H.-J.; Graf, Ulrich: Formeln und Tabellen der angewandten mathematischen Statistik. Springer, Berlin 1998

Yang, Laurence T.; Guo, Minyi: High-Performance Computing – Paradigm and Infrastructure. Wiley & Sons, Hoboken 2006

b) Internetquellen

Microsoft: How to estimate the size of a SQL Server CE or SQL Server 2005 Mobile Edition Database. Microsoft 2007, <http://support.Microsoft.com/kb/827968/en-us> (03.01.2009)

Microsoft: Vorgehensweise: Erstellen einer Publikation und Definieren von Artikeln (SQL Server Management Studio). Microsoft 2008 a, [http://msdn.microsoft.com/de-de/library/ms151160\(SQL.90\).aspx](http://msdn.microsoft.com/de-de/library/ms151160(SQL.90).aspx) (11.02.2009)

Microsoft: Vorgehensweise: Konfigurieren von IIS für die Websynchronisierung. Microsoft 2008 b, [http://msdn.microsoft.com/de-de/library/ms152511\(SQL.90\).aspx](http://msdn.microsoft.com/de-de/library/ms152511(SQL.90).aspx) (11.02.2009)

Microsoft: Onlinedokumentation für SQL Server 2005 Compact Edition – Datentypen. Microsoft 2009 a, [http://msdn.Microsoft.com/de-de/library/ms172424\(SQL.90\).aspx](http://msdn.Microsoft.com/de-de/library/ms172424(SQL.90).aspx) (02.01.2009)

Microsoft: Onlinedokumentation für SQL Server 2005 Compact Edition – Client-/Serverumgebung. Microsoft 2009 b, [http://msdn.Microsoft.com/de-de/library/ms172075\(SQL.90\).aspx](http://msdn.Microsoft.com/de-de/library/ms172075(SQL.90).aspx) (05.01.2009)

Microsoft: Onlinedokumentation für SQL Server 2005 Compact Edition – Neuigkeiten (SQL Server Compact Edition). Microsoft 2009 c, [http://msdn.Microsoft.com/de-de/library/ms172417\(SQL.90\).aspx](http://msdn.Microsoft.com/de-de/library/ms172417(SQL.90).aspx) (05.01.2009)

Microsoft: Onlinedokumentation für SQL Server 2005 Compact Edition – Entwicklungsumgebung. Microsoft 2009 d, [http://msdn.microsoft.com/de-de/library/ms173014\(SQL.90\).aspx](http://msdn.microsoft.com/de-de/library/ms173014(SQL.90).aspx) (07.01.2009)

Microsoft: Microsoft SQL Server Compact 3.5 Service Pack 1 Server Tools. Microsoft 2009 e, <http://www.microsoft.com/downloads/details.aspx?FamilyId=FA751DB3-7685-471B-AC31-F1B150422462&displaylang=en> (11.02.2009)

Microsoft: Microsoft SQL Server Compact 3.5 Service Pack 1 and Synchronization Services for ADO.NET version 1.0 Service Pack 1 for Windows Desktop. Microsoft 2009 f, <http://www.microsoft.com/downloads/details.aspx?familyid=DC614AEE-7E1C-4881-9C32-3A6CE53384D9&displaylang=en> (11.02.2009)

Microsoft: .NET Compact Framework 2.0 Redistributable. Microsoft 2009 g, <http://www.microsoft.com/downloads/details.aspx?familyid=9655156b-356b-4a2c-857c-e62f50ae9a55&displaylang=en> (11.02.2009)

Microsoft: Understanding SQL Server CE Database Objects, Microsoft 2009 h, [http://msdn.microsoft.com/en-us/library/aa275627\(SQL.80\).aspx](http://msdn.microsoft.com/en-us/library/aa275627(SQL.80).aspx) (11.04.2009)

OpenNETCF Consulting, LLC: Smart Device Framework. OpenNETCF Consulting LLC 2009, <http://www.opennetcf.com/Products/SmartDeviceFramework/tabid/65/Default.aspx> (13.01.2009)

T-Mobile: Ameo Bedienungsanleitung. T-Mobile 2009, http://www.t-mobile.de/downloads/bedienungsanleitung_handy/t-mobile-ameo.pdf (13.01.2009)

ZedGraph: ZedGraph. ZedGraph 2009, http://zedgraph.org/wiki/index.php?title=Main_Page (13.01.2009)

Ehrenwörtliche Erklärung

Ich versichere, dass ich die beiliegende Bachelorarbeit ohne Hilfe Dritter und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt und die den benutzten Quellen wörtlich oder inhaltlich entnommenen Stellen als solche kenntlich gemacht habe. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum

Unterschrift

Anhang

1. Überprüfung einer Messreihe für das Einhalten einer bestimmten Güte nach der ISO 14756-Norm.....	99
2. Quantile-Tabelle der Normalverteilung.....	102
3. Datendefinition der Testdatenbank.....	104
4. Installationsanleitung für das Testsystem.....	105
5. Datendefinition der Datenbank für die Messergebnisse.....	109
6. Benutzeranleitung.....	110
7. Detaillierte Ergebnisse der Überprüfungen der Messreihen.....	115

1. Überprüfung einer Messreihe für das Einhalten einer bestimmten Güte nach der ISO 14756-Norm

Innerhalb von Tests sind Messungenauigkeiten ein Problem. Testergebnisse stellen eine Annäherung an einen unbekannten wahren Wert dar. Um Abweichungen auszugleichen, kann das Gesetz der Großen Zahlen verwendet werden. Dieses Gesetz sagt aus, dass ein Mittelwert genauer wird, um so mehr Werte in die Berechnung des Mittelwerts einfließen. Somit kann ein Mittelwert, erstellt aus Messergebnissen, als eine Annäherung an die reale durchschnittliche Laufzeit angesehen werden (der reale Wert).¹⁵⁸

Eine weitere Unbekannte bei Messungen stellt die Verteilung der realen Werte dar. Die Verteilung zeigt in dem Fall der realen Laufzeiten, wie wahrscheinlich eine bestimmte Abweichung der realen Werte vom Mittelwert ist bzw. wie wahrscheinlich es ist, dass die Laufzeit sich in einem bestimmten Bereich um den Mittelwert befindet. Mithilfe dieser Verteilung kann überprüft werden, ob die Wiederholungen innerhalb einer Messreihe ausreichend gewesen sind, um die innerhalb der Messreihen aufgetretene Ungenauigkeit auszugleichen. In die Überprüfung der Messergebnisse fließt die Normalverteilung als *Quantil* ein. Dies stellt die Umkehrfunktion zur Verteilung dar.¹⁵⁹ Für die Überprüfungen von Messergebnissen wird eine Verteilung angenommen. Die Annahme der Normalverteilung $\Phi(0,1)$ ist eine in der ISO-Norm verwendete Schätzung für die Verteilung von realen Laufzeiten sowie von Messergebnissen.¹⁶⁰

Um eine Überprüfung nach der ISO-Norm durchzuführen, wird zunächst ein Intervall $T_{me} \pm d$ um den Mittelwert definiert. Dann wird geprüft, ob sich der reale Mittelwert im definierten Intervall um den ermittelten Mittelwert befindet und die Wiederholungen somit ausreichend gewesen ist. T_{me} steht dabei für den Mittelwert einer Messreihe, während d die Grenzen des Intervalls bestimmt. d stellt einen Wert dar, den man innerhalb der Berechnung vorgibt. Ein weiterer Wert, der vorgegeben wird, ist die Wahrscheinlichkeit α , mit der die Überprüfung sich irrt (Irrtumswahrscheinlichkeit). Das Ergebnis der Überprüfung kann somit mit der Wahrscheinlichkeit α falsch sein. Üblicherweise wird für α 5 % gewählt. 10 % stellt

¹⁵⁸ Vergl. Henze, Norbert (2006), S. 196f

¹⁵⁹ Vergl. Mosler, Karl et al. (2006), S. 105

¹⁶⁰ Vergl. Dirlewanger, Werner (2006), S.55ff

ein weiterer, akzeptabler Wert dar, während Werte über 20 % eine zu hohe Fehlertoleranz zulassen. Für d hat sich durch Erfahrung gezeigt, dass die Definition $d = b \cdot (\text{Mittelwert der Messungen})$ eine gute Wahl darstellt. b stellt in dieser Arbeit einen Koeffizienten dar, mit dem man d weiter anpassen kann. Ein guter Bereich, aus dem b gewählt werden kann, ist ein Wert zwischen 0,05 und 0,2. Werte über 0,2 lassen einen zu großen Fehler zu. b kann dabei unabhängig von der Irrtumswahrscheinlichkeit gewählt werden.¹⁶¹

Um zu ermitteln, ob die Anzahl N der Messungen mit der Irrtumswahrscheinlichkeit α ausreichend gewesen sind oder ob weitere Durchläufe nötig sind, muss zuerst der Mittelwert M_m der einzelnen Messergebnisse x_1, \dots, x_N der Messreihe ermittelt werden. Der Mittelwert wird durch die Funktion in Formel A1.1 definiert.¹⁶²

Formel A1.1: Mittelwert¹⁶³

Innerhalb von Formel A1.2 wird die Abweichung $S_m(N)$ der einzelnen Messwerte zum Mittelwert definiert. Dadurch fließt in die Überprüfung ein, wie stark die jeweiligen Messungen sich vom Mittelwert unterscheiden.¹⁶⁴

Formel A1.2: Varianz¹⁶⁵

Basierend auf der Annahme der Normalverteilung kann nun mit der Wahrscheinlichkeit $1-\alpha$ gezeigt werden, ob sich der reale Wert im Intervall $T_{me} \pm d$ um den ermittelten Mittelwert befindet oder ob weitere Wiederholungen nötig sind. Für diesen Zweck kann die Ungleichung in Formel A1.3 verwendet werden. Dort fließt die Abweichung sowie das definierte Intervall in die Berechnung ein.

$$\left(\left(\frac{u(\alpha)}{d} \right)^2 \right) S_m < N$$

Formel A1.3: Ungleichung zur Überprüfung von Messergebnissen¹⁶⁶

Der Teil $u(\alpha)$ stellt dabei das $u_{1-\alpha/2}$ -Quantil der Normalverteilung dar.¹⁶⁷ Dies bedeutet die Verwendung eines Quantils der Normalverteilung $\Phi(0,1)$, welches sich aus der Wahrscheinlichkeit $1-\alpha/2$ ergibt.¹⁶⁸ Falls die Ungleichung in Formel A1.3 zutrifft, sind die Durchläufe des Tests ausreichend, ansonsten müssen weitere

¹⁶¹ Vergl. Dirlewanger, Werner (2006), S.56f und S. 34f

¹⁶² Vergl. Dirlewanger, Werner (2006), S.55ff

¹⁶³ Vergl. Dirlewanger, Werner (2006), S.55

Wiederholungen durchgeführt werden. Die größte Unsicherheit bei diesem Vorgehen ist die Annahme der Normalverteilung. Die tatsächliche Verteilung der Messwerte wie der realen Laufzeiten sind unbekannt, daher kann der Test an dieser Stelle evtl. scheitern. Weitere Techniken, die zusätzlich eingesetzt werden können, sind Überprüfungen der Annahme der Verteilung von den Messwerten sowie Techniken zur Schätzung solcher Parameter wie die Normalverteilung oder die Anzahl der Wiederholung.¹⁶⁹

164 Vergl. Dirlewanger, Werner (2006), S.57

165 Vergl. Dirlewanger, Werner (2006), S.55

166 Vergl. Dirlewanger, Werner (2006), S.55

167 Z. B. zu entnehmen aus Anhang 2, Graf, Ulrich et al. (1998), S. 458 oder Bronstein, I.N. et al (2008), S. 1143f., wobei bei der letzteren Quelle die Wahrscheinlichkeit genommen werden muss, die die gewünschte Toleranz gerade überschreitet, vergl. für $u_{1-\alpha/2}$ auch Graf, Ulrich et al. (1998), S. 112

168 Für ein Beispiel für eine Berechnung mit einem $u_{1-\alpha/2}$ -Quantil, vergl. Hartung, Joachim et al. (2004), S. 419f

169 Vergl. Dirlewanger, Werner (2006), S. 57

2. Quantile-Tabelle der Normalverteilung

α in %	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9
50	0,000	0,003	0,005	0,008	0,010	0,013	0,015	0,018	0,020	0,023
51	0,025	0,028	0,030	0,033	0,035	0,038	0,040	0,043	0,045	0,048
52	0,050	0,053	0,055	0,058	0,060	0,063	0,065	0,068	0,070	0,073
53	0,075	0,078	0,080	0,083	0,085	0,088	0,090	0,093	0,095	0,098
54	0,100	0,103	0,105	0,108	0,111	0,113	0,116	0,118	0,121	0,123
55	0,126	0,128	0,131	0,133	0,136	0,138	0,141	0,143	0,146	0,148
56	0,151	0,154	0,156	0,159	0,161	0,164	0,166	0,169	0,171	0,174
57	0,176	0,179	0,181	0,184	0,187	0,189	0,192	0,194	0,197	0,199
58	0,202	0,204	0,207	0,210	0,212	0,215	0,217	0,220	0,222	0,225
59	0,228	0,230	0,233	0,235	0,238	0,240	0,243	0,246	0,248	0,251
60	0,253	0,256	0,259	0,261	0,264	0,266	0,269	0,272	0,274	0,277
61	0,279	0,282	0,285	0,287	0,290	0,292	0,295	0,298	0,300	0,303
62	0,305	0,308	0,311	0,313	0,316	0,319	0,321	0,324	0,327	0,329
63	0,332	0,335	0,337	0,340	0,342	0,345	0,348	0,350	0,353	0,356
64	0,358	0,361	0,364	0,366	0,369	0,372	0,375	0,377	0,380	0,383
65	0,385	0,388	0,391	0,393	0,396	0,399	0,402	0,404	0,407	0,410
66	0,412	0,415	0,418	0,421	0,423	0,426	0,429	0,432	0,434	0,437
67	0,440	0,443	0,445	0,448	0,451	0,454	0,457	0,459	0,462	0,465
68	0,468	0,470	0,473	0,476	0,479	0,482	0,485	0,487	0,490	0,493
69	0,496	0,499	0,502	0,504	0,507	0,510	0,513	0,516	0,519	0,522
70	0,524	0,527	0,530	0,533	0,536	0,539	0,542	0,545	0,548	0,550
71	0,553	0,556	0,559	0,562	0,565	0,568	0,571	0,574	0,577	0,580
72	0,583	0,586	0,589	0,592	0,595	0,598	0,601	0,604	0,607	0,610
73	0,613	0,616	0,619	0,622	0,625	0,628	0,631	0,634	0,637	0,640
74	0,643	0,646	0,650	0,653	0,656	0,659	0,662	0,665	0,668	0,671
75	0,674	0,678	0,681	0,684	0,687	0,690	0,693	0,697	0,700	0,703
76	0,706	0,710	0,713	0,716	0,719	0,722	0,726	0,729	0,732	0,736
77	0,739	0,742	0,745	0,749	0,752	0,755	0,759	0,762	0,765	0,769
78	0,772	0,776	0,779	0,782	0,786	0,789	0,793	0,796	0,800	0,803
79	0,806	0,810	0,813	0,817	0,820	0,824	0,827	0,831	0,834	0,838
80	0,842	0,845	0,849	0,852	0,856	0,860	0,863	0,867	0,871	0,874
81	0,878	0,882	0,885	0,889	0,893	0,896	0,900	0,904	0,908	0,912
82	0,915	0,919	0,923	0,927	0,931	0,935	0,938	0,942	0,946	0,950
83	0,954	0,958	0,962	0,966	0,970	0,974	0,978	0,982	0,986	0,990
84	0,994	0,999	1,003	1,007	1,011	1,015	1,019	1,024	1,028	1,032
85	1,036	1,041	1,045	1,049	1,054	1,058	1,063	1,067	1,071	1,076
86	1,080	1,085	1,089	1,094	1,098	1,103	1,108	1,112	1,117	1,122
87	1,126	1,131	1,136	1,141	1,146	1,150	1,155	1,160	1,165	1,170
88	1,175	1,180	1,185	1,190	1,195	1,200	1,206	1,211	1,216	1,221
89	1,227	1,232	1,237	1,243	1,248	1,254	1,259	1,265	1,270	1,276
90	1,282	1,287	1,293	1,299	1,305	1,311	1,317	1,323	1,329	1,335
91	1,341	1,347	1,353	1,359	1,366	1,372	1,379	1,385	1,392	1,398
92	1,405	1,412	1,419	1,426	1,433	1,440	1,447	1,454	1,461	1,468
93	1,476	1,483	1,491	1,499	1,506	1,514	1,522	1,530	1,538	1,546
94	1,555	1,563	1,572	1,580	1,589	1,598	1,607	1,616	1,626	1,635
95	1,645	1,655	1,665	1,675	1,685	1,695	1,706	1,717	1,728	1,739
96	1,751	1,762	1,774	1,787	1,799	1,812	1,825	1,838	1,852	1,866
97	1,881	1,896	1,911	1,927	1,943	1,960	1,977	1,995	2,014	2,034
98	2,054	2,075	2,097	2,120	2,144	2,170	2,197	2,226	2,257	2,290
99	2,326	2,366	2,409	2,457	2,512	2,576	2,652	2,748	2,878	3,090

Tabelle 13: Quantile der Normalverteilung¹⁷⁰

¹⁷⁰ Vergl. Graf, Ulrich et al. (1998), S. 458

Tabelle 13 enthält die Quantile zur Normalverteilung $\Phi(0,1)$ zu einer Wahrscheinlichkeit α . Die einzelnen Spalten enthalten jeweils eine Nachkommastelle der Wahrscheinlichkeit, während die Zeilen den Wert vor dem Komma der Wahrscheinlichkeit darstellen, zu denen das jeweilige Quantil gehört. Um ein $1-\alpha/2$ -Quantil zu $\alpha=0,05$ ermitteln, muss zuerst folgende Berechnung durchgeführt werden:

$$1 - \frac{0,05}{2} = 0,975$$

*Formel A2.1: Beispiel der
Berechnung eines $1-\alpha/2$ -Quantils*

Somit muss aus der Tabelle der Wert zu der Wahrscheinlichkeit 97,5 ausgelesen werden: $u(\alpha)=1,96$ (Zeile 97, Spalte 5).¹⁷¹

¹⁷¹ Vergl. Hartung, Joachim, Heine, Barbara (2004), S. 419f und Graf, Ulrich et al. (1998), S. 458

3. Datendefinition der Testdatenbank

Kunde = KundeId + (Vorname) + (Nachname) + (Strasse) + (Ort) + (PLZ) +
KundenNr + ErstellungsZeitpunkt + (AenderungsZeitpunkt)

Auftrag = AuftragId + AuftragDatum + (Rabatt) + (Kommentar) + AuftragsArt +
ErstellungsZeitpunkt + (AenderungsZeitpunkt) + KundeId

Produktart = ProduktArtId + ProduktArtName + ErstellungsZeitpunkt +
(AenderungsZeitpunkt)

Kollektion = KollektionId + KollektionName + ErstellungsZeitpunkt +
(AenderungsZeitpunkt) + ProduktArtId

Artikel = ArtikelId + EAN + Preis + (Bezeichnung) + ErstellungsZeitpunkt +
(AenderungsZeitpunkt) + KollektionId

Position = ArtikelId + AuftragId + Menge + PreisBeiBestellung +
ErstellungsZeitpunkt + (AenderungsZeitpunkt)

Inklusionsabhängigkeiten:

Auftrag (KundeId) \subseteq Kunde (KundeId)

Kollektion (ProduktArtId) \subseteq Produktart (ProduktArtId)

Artikel (KollektionId) \subseteq Kollektion (KollektionId)

Position (ArtikelId) \subseteq Artikel (ArtikelId)

Position (AuftragId) \subseteq Auftrag (AuftragId)

4. Installationsanleitung für das Testsystem

Um das Testsystem zu installieren, müssen die Softwareteile, die in Abbildung 10 in Kapitel 6.8.1 dargestellt werden, installiert bzw. vorbereitet werden.¹⁷² Die einzelnen Komponenten auf der CD, welche in dieser Arbeit mitgeliefert werden, sind in Tabelle 14 dargestellt.

Inhalt	Beschreibung	Pfad auf der CD
Mobile Testapplikation	Dieser Ordner enthält die mobile Applikation	Installation\Mobile Applikation
Web Services	Die Webapplikation für die Tests.	Installation\Testsession
Server-Applikation	Die Server-Applikation für die Auswertung der Tests sowie für das Anlegen der Serverdatenbank.	Installation\Serversteuerung
SQL Server CE 3.5 Servertools (32 und 64 Bit)	Tools zur Erstellungen von Publikationen auf einem Microsoft IIS auf Basis einer SQL Server Datenbank	Installation\SQL Server CE 3.5 Servertools
Clientkomponenten des SQL Server 2005 CE 3.5	Diese Pakete stellen den SQL Server 2005 CE 3.5 zur Verfügung. ¹⁷³	Installation\SQL Server 2005 CE Clientkomponenten
Microsoft .Net Compact Framework 2.0 Clientkomponenten	Das Installationspaket des Compact Framework für mobile Geräte ¹⁷⁴	Installation\Microsoft .Net Compact Framework 2.0 Clientkomponenten
Erstellungsscript Serverdatenbank	Mit diesem SQL-Script kann die Testdatenbank auf dem Server erstellt werden	Installation\Erstellungsscript Serverdatenbank

Tabelle 14: Installationsdateien

Um die Serverumgebung vorzubereiten, muss zunächst die Testdatenbank auf einen Microsoft SQL Server 2005 erstellt werden. Dies kann zum einen durch das direkte Ausführen des Erstellungsscripts auf dem Server geschehen. Zum anderen befindet sich in der Serverapplikation innerhalb des Menüs „Datei“ der Punkt „Datenbank anlegen“ (Vergl. Abbildung 32). Das Erstellungsscript geht von der Existenz einer Datenbank mit dem Namen „DBTest“ aus, die auf dem SQL Server per Hand angelegt werden muss. Das Script selber legt die Tabellenstruktur an, die für die Tests und die Messergebnisse benötigt werden.

Für die Erstellung der Publikation, basierend auf den Tabellendefinitionen der

¹⁷² Die benötigten Teile befinden sich auf der CD, welche der Arbeit beiliegt.

¹⁷³ Auch als Download unter Microsoft (2009 f) verfügbar

¹⁷⁴ Als Download unter Microsoft(2009 g) verfügbar.

Testdatenbank, muss zum einen eine lokale Publikation auf dem SQL Server 2005 erstellt werden¹⁷⁵, zum anderen muss diese Publikation auf einen Web Server (dem Replikationsserver) veröffentlicht werden. Für das Veröffentlichen auf einen Web Server können die SQL Server 3.5 Server Tools¹⁷⁶ verwendet werden. Die Erstellung der Serverdatenbank sowie der Publikationen müssen vor der Erstellung der mobilen Datenbank geschehen, damit die mobile Datenbank vom Schema der Publikation abgeleitet werden kann.



Abbildung 32: Anhang - Datenbank erstellen über
Serverapplikation

Damit die Server-Applikation die neu erstellte Datenbank findet, muss die Verbindungszeichenfolge angegeben werden. Für diesen Zweck existiert in der Serverapplikation die Konfigurationsdatei „*ServerSteuerung.exe.config*“, welche sich im selben Verzeichnis mit der Serversteuerung befindet. Das folgende Listing stellt einen Teil dieser Konfigurationsdatei dar. Unter der Variable „*connectionString*“ in Zeile 5 muss die neue Verbindungszeichenfolge eingetragen werden.

```

01:      ...
02:      <connectionStrings>
03:          <add name="ServerSteuerung.Properties.Settings._
04:              ServerSteuerungConnectionString"
05:                  connectionString="Data Source=<Servername>;Initial
06:                      Catalog=DBTest;User
07:                      ID=<Benutzername>;Password=<Passwort>"
08:                  providerName="System.Data.SqlClient" />
09:      </connectionStrings>
10:      ...

```

Die Web Services enthalten ebenfalls eine Konfigurationsdatei (*web.config*), innerhalb dessen die Verbindung zur Datenbank eingetragen werden muss.

Das folgende Listing zeigt, welcher Teil der *web.config* hinzugefügt werden muss. Neben einer solchen Konfiguration müssen die Web Services in eine eigene Webapplikation auf einem IIS installiert werden. Dabei muss einerseits von der

¹⁷⁵ Vergl. Microsoft (2008) oder Boddenberg, Ulrich B. (2007), S. 258ff für eine Anleitung zum Erstellen einer Publikation auf einen SQL Server 2005

¹⁷⁶ Erhältlich bei der Quelle Microsoft (2009 e) oder auf der CD zu dieser Arbeit, vergl. auch Microsoft (2008 b) oder Boddenberg, Ulrich B. (2007), S. 271ff für eine Anleitung zum Erstellen des Replikationsservers auf einem IIS.

Webapplikation aus eine Verbindung zur Datenbank möglich sein und andererseits müssen die mobile Geräte die Web Services auch von außerhalb des lokalen Netzwerks erreicht werden können, damit z. B. über GPRS eine Verbindung aufgebaut werden kann.¹⁷⁷

```
01: ...
02: <connectionStrings>
03:   <remove name="TestDatenbank" />
04:   <add name="TestDatenbank" connectionString="Data
05:       Source=<Servername>;Initial
06:       Catalog=DBTest;User
07:       ID=<Benutzername;Password=<Passwort>"
08:       providerName="System.Data.SqlClient"/>
09: </connectionStrings>
10: ...
```

Ähnlich wie die Serverseite müssen auch auf der Clientseite Einstellungen vorgenommen werden.

```
01:...
02:<Config>
03:  <Variables>
04:    <Variable name="SyncTestUrl">
05:      URL auf den Webservice für den Synchronisationstest
06:    </Variable>
07:    <Variable name="DBTestUrl">
08:      URL auf den Webservice für den Datenbanktest
09:    </Variable>
10:    <Variable name="ReplikationUrl">
11:      URL auf den Replikationsdienst
12:    </Variable>
13:    <Variable name="Publisher">
14:      Name des Servers, der die Publikation bereitstellt
15:    </Variable>
16:    <Variable name="PublisherDatabase">
17:      Name der Datenbank, auf die die Veröffentlichung basiert
18:    </Variable>
19:    <Variable name="Publication">
20:      Name der Publikation
21:    </Variable>
22:    <Variable name="Login">
23:      Benutzername, unter dem die Replikationsdienste verwendet
24:      werden
25:    </Variable>
26:    <Variable name="Password">
27:      Passwort des Benutzerkontos für die Replikationsdienste
28:    </Variable>
29:  </Variables>
30:</Config>
31:...
```

¹⁷⁷ Zur Einrichtung einer Webapplikation auf einen Windows Server 2003 mit dem IIS 6, vergl. Bünning, Uwe, Krause, Jörg (2006), S. 1048ff, für die Einrichtung auf einen Windows Server 2008 mit einem IIS 7, vergl. Bünning, Uwe, Rzepka, Dirk (2008), S. 975ff

Dafür gibt es im Anwendungsordner der mobilen Testapplikation die Datei „*Config.xml*“ im Unterordner *Config*. Dort müssen die im vorhergehenden Listing dargestellten Werte angegeben werden. Diese stellen Angaben für die Verbindungen zum Server dar.

Die Replikations-URL, der Publikationsname, Benutzername und das Passwort ergeben sich aus der Einrichtung der Publikation auf dem Webserver mithilfe der SQL Server 3.5 Server Tools, während die Web Service-URLs sich aus der Installation der Webapplikation ergeben. Der Publisher ist der Name des Servers, der die Publikation auf einem IIS bereitstellt. Nach dem Einstellen dieser Datei muss der Ordner der Clientapplikation auf dem mobilen Gerät, auf welchem der Test durchgeführt werden soll, kopiert werden. Danach ist die Testumgebung eingerichtet.

5. Datendefinition der Datenbank für die Messergebnisse

Master_Test = TestName

TestFall = TesttFallId + (Gesamtlaufzeit) + (Kommentar) + (Testschritte) +
TestfallName + TestName

Test = TestValueId + Zeit + Datenmenge

Inklusionsabhängigkeiten:

TestFall (TestName) \subseteq Master_Test (TestName)

Test (TestFallId) \subseteq TestFall (TestFallId)

6. Benutzeranleitung

Dieses Kapitel stellt die Oberfläche vor und zeigt, wie das Testsystem bedient wird.

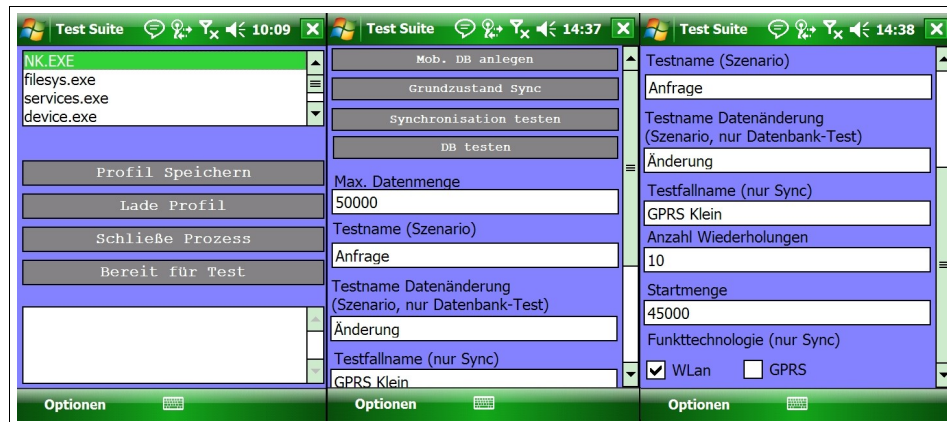


Abbildung 33: Screenshot – Mobile Teststeuerung

Abbildung 33 zeigt die mobile Teststeuerung. Der linke Teil stellt die Profilliste dar, mithilfe derer ein Testzustand hergestellt werden kann. Neben der Speicherung und dem Laden von Profilen können unnötige Prozesse beendet werden. Nach der Bestätigung, dass ein Testzustand hergestellt worden ist (*Bereit für Test*), gelangt man in die Teststeuerung. Die beiden rechten Teile stellen zusammen die Teststeuerung dar. Innerhalb der Box unter den Bedientasten der Prozessliste wird eine Zusammenfassung der Aktionen ausgegeben, welche bereits durchgeführt worden sind (z. B. welche Prozesse beendet oder gestartet worden sind).

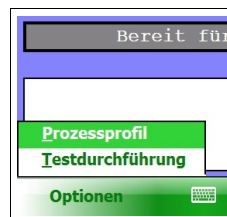


Abbildung 34: Screenshot – Optionen

Man kann auch über das Menü der Applikation (Schaltfläche *Optionen*) in die Teststeuerung kommen, ohne einen Testzustand herzustellen (Vergl. Abbildung 34). Allerdings muss spätestens bei einer Testdurchführung vom Benutzer bestätigt worden sein, dass ein Testzustand hergestellt worden ist (Vergl. Kapitel 6.8.3). Falls dies bis zur Testdurchführung nicht geschehen ist, wird die Prozessliste wieder angezeigt. In dem Fall ist das Menü unter Optionen gesperrt und man ist gezwungen, einen Testzustand herzustellen und zu bestätigen. Nach der Bestätigung wird der Test

fortgesetzt.

Über „*Mob. DB anlegen*“ kann man die mobile Datenbank anlegen, während „*Grundzustand Sync*“ eine Basisdatenmenge für den Synchronisationstest anlegt. Die beiden Schaltflächen darunter sorgen für den Start des Synchronisations- bzw. DBS-Tests. Innerhalb des DBS-Tests werden automatisch alle Testfälle des 2. und 3. Szenarios durchgeführt.

Die Eingabefelder unter den Schaltflächen dienen der Parametereingabe für die Tests. Dazu gehört die maximale Datengrenze, bis zu der getestet wird (in kB), der Name des Szenarios (Im Fall des DBS-Tests wird der Name für das 2. und 3. Szenario benötigt).

Während im DBS-Test die sechs Testfälle innerhalb eines Testdurchlaufs durchgeführt werden und somit die Namen der Testfälle klar zugeordnet werden können, benötigt der Synchronisationstest die Angabe, welcher Testfall ausgeführt werden soll. Daher gibt es eine Eingabebox für den Testfallnamen. Weiterhin kann die Anzahl der Wiederholungen angegeben werden, die pro Messreihe durchgeführt werden. Weiterhin kann, falls der Test nicht bei der niedrigsten Datenmenge anfangen soll (Vergl. Kapitel 6.5 für die jeweiligen Startmengen), eine Startmenge angegeben werden. Damit kann auch ein evtl. abgebrochener Testlauf fortgesetzt werden. Im letzteren Fall müssen die Szenario- und Testfallnamen zu denen des abgebrochenen Tests passen.

Die beiden Checkboxen am unteren Ende der Teststeuerung geben an, ob der Synchronisationstest über WLAN oder GPRS durchgeführt werden soll. Damit weiß der Synchronisationstest, mit welcher Technologie er sich wieder verbinden muss, falls einmal die Verbindung zum Netzwerk abbricht. Vor der Durchführung eines Synchronisationstestfalls muss der Tester zunächst per Hand eine Verbindung zur zu testenden Funktechnologie herstellen (Vergl. dazu das Handbuch des jeweiligen Gerätes).

Die Serverapplikation ist in drei Teile geteilt. Abbildung 35 zeigt den Teil für die Darstellung von Messergebnissen. Dabei kann das Szenario und der Testfall ausgewählt werden, darauf basierend wird der Graph des Messergebnisses in das Diagramm gezeichnet.

Die beiden Auswahlboxen unter dem Szenario bzw. dem Testfall dienen dem Umschalten der Skalierung der X- bzw. Y-Achse (Logarithmisch, linear etc.). Mithilfe

der Auswahlfeld unter der Überschrift *Zeiteinheit* (rechts neben der Testfallauswahl) kann die Größenordnung der Laufzeit für die Graphen ausgewählt werden (Millisekunden, Sekunden, Minuten), während die letzte Auswahlfeld ganz rechts die Größeneinheit der Datenmenge auswählt (kB oder MB). Die Auswahlfeld für die Größeneinheit der Datenmenge ist lediglich dann aktiv, wenn die Datenmenge nicht in Anzahl Zeilen angegeben wird (wie z. B. bei den DBS-Tests angewendet werden soll). Mithilfe des Schalters „Anzahl Zeilen/Byte“ kann man die Einheit der Datenmenge zwischen Anzahl der Zeilen und Byte umschalten.

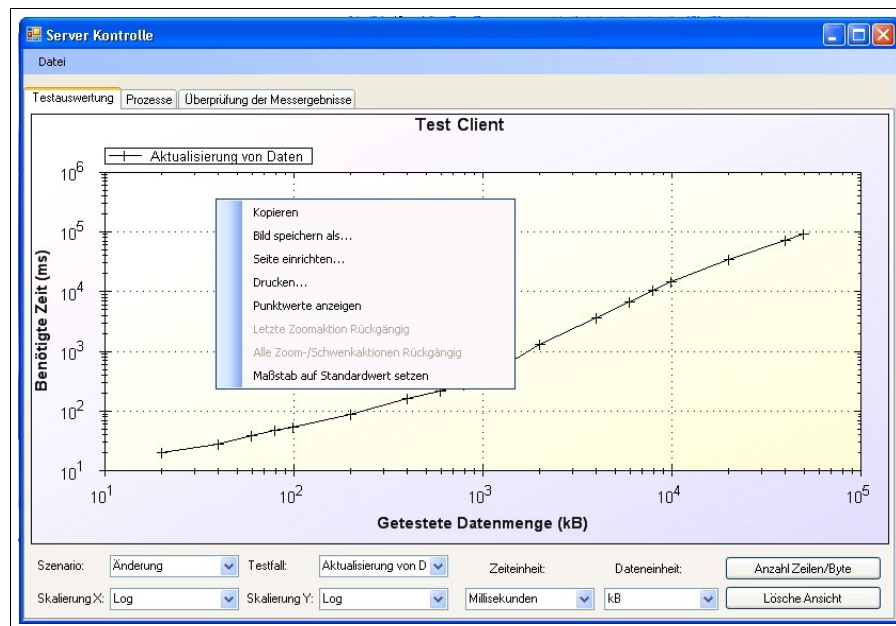


Abbildung 35: Screenshot – Darstellung der Messergebnisse

Mithilfe des Buttons „Lösche Ansicht“ stellt man den Grundzustand der Maske durch das Löschen aller Graphen her. Mithilfe eines Rechtsklick auf das Diagramm öffnet sich das im Bild dargestellte Menü. Mit deren Hilfe kann man u. a. das Diagramm als Bild abspeichern.

Abbildung 36 stellt die Prozessliste auf Serverseite dar. Die Funktionen sind dabei dieselben wie bei der mobilen Prozessliste. Es fehlt lediglich die Bestätigung, dass ein Testzustand hergestellt wurde. Das Herstellen eines Testzustands auf Serverseite wird durch eine Web Service-Methode ermöglicht. Daher muss min. ein Prozessprofil in der Webapplikation, in der die Web Services ausgeführt werden, im Unterverzeichnis „Profile“ befinden.

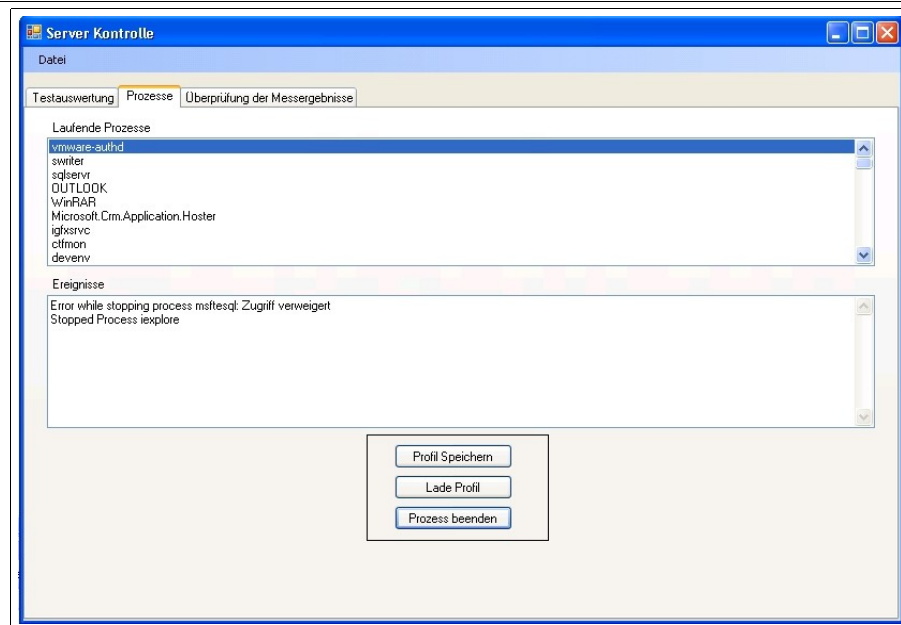


Abbildung 36: Screenshot – Prozessliste

Abbildung 37 zeigt die Maske für das Überprüfen der Messergebnisse. Hier wird ein Szenario sowie ein Testfall ausgewählt, dessen Messreihen überprüft werden sollen. Daneben kann ein Intervall und ein Quantil eingegeben werden. Das Quantil kann aber auch ausgewählt werden. Wenn über den Eingabefeldern der Punkt „*Quantileingabe per Auswahl*“ ausgewählt wird, verschwindet die Eingabebox für das Quantil und es erscheint eine Liste mit Wahrscheinlichkeiten. Durch die Auswahl der gewünschten Wahrscheinlichkeit lädt das Programm das entsprechende Quantil.

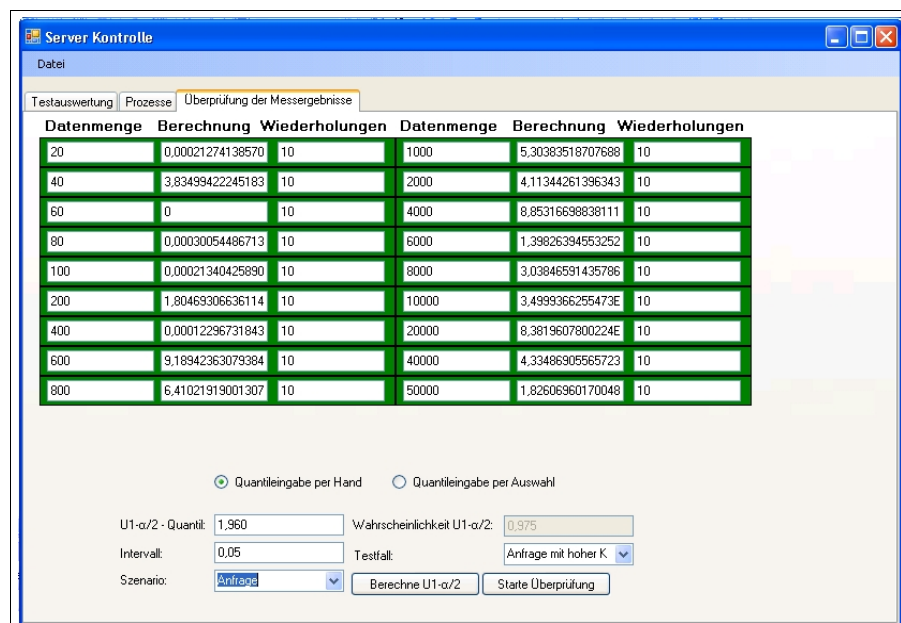


Abbildung 37: Screenshot – Überprüfung der Messergebnisse

Über die Schaltfläche „*Berechne $u_{1-\alpha/2}$* “ kann der Benutzer sich zu der Irrungswahrscheinlichkeit α die entsprechende Wahrscheinlichkeit ausrechnen, nach der er das Quantil auswählen muss.

Mithilfe der Schaltfläche „*Starte Überprüfung*“ werden die Messreihen geprüft und ausgegeben. Die grün hinterlegten Felder in der Ausgabe bedeutet, die Messreihe hat die Überprüfung bestanden, während eine rote Hinterlegung bedeutet, die Überprüfung hat ein negatives Ergebnis geliefert.

7. Detaillierte Ergebnisse der Überprüfungen der Messreihen

Innerhalb dieses Kapitels werden die detaillierten Ergebnisse der Überprüfung, ob die Messergebnisse nicht zu stark verzerrt worden sind, dargestellt. Diese sind pro Szenario tabellarisch aufgelistet.

Dabei wird pro Testfall jeweils pro Zeile die getestete Datenmenge, das Ergebnis der Berechnung sowie die Anzahl der Wiederholungen angegeben. Diese Tests wurden mit der Wahrscheinlichkeit $\alpha=5\%$ und dem Intervall $d=b \cdot (\text{Mittelwert der Messungen})$ mit $b=0,05$ durchgeführt.

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
30	1,73803595697879	10	600	1,57069336797926	10
60	1,61978467308723	10	900	2,88752876335609	10
90	3,8102621345619E	10	3000	8,55083799694549	10
300	1,51220394934679	10	5000	1,3234043179893E	10

Abbildung 38: Überprüfung Szenario 1 – Testfall „GPRS Groß“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
30	1,36850020221312	10	600	7,71726658621503	10
60	4,14307224542788	10	900	8,05530742233198	10
90	8,67573868334104	10	3000	7,90110525259493	10
300	2,64973544699122	10	5000	2,40824092870808	10

Abbildung 39: Überprüfung Szenario 1 – Testfall „GPRS Mittel“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
30	3,00831954905556	10	600	6,50566914283317	10
60	4,37236990428216	10	900	1,92034749062387	10
90	3,08545031518005	10	3000	5,5725157616542E	10
300	1,44779699335134	10	5000	3,71509534720346	10

Abbildung 40: Überprüfung Szenario 1 – Testfall „GPRS Klein“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
30	3,16468038296027	10	600	1,27407510243711	10
60	8,68612574168871	10	900	2,3839841186623E	10
90	1,93723147630505	10	3000	2,15393278398093	10
300	7,02469873566515	10	5000	3,97606703796644	10

Abbildung 41: Überprüfung Szenario 1 – Testfall „WLAN Groß“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
30	8,76615134366895	10	600	1,98341953014572	10
60	1,69102662102113	10	900	1,91438713636947	10
90	3,50045060475073	10	3000	3,06743488501296	10
300	1,85589042185285	10	5000	2,94176892240895	10

Abbildung 42: Überprüfung Szenario 1 – Testfall „WLAN Mittel“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
30	1,81265624847018	10	600	7,2250855379083E	10
60	1,20386057039384	10	900	2,24681632976675	10
90	2,08966067340328	10	3000	1,30442760057273	10
300	3,75481714897375	10	5000	7,77560841492766	10

Abbildung 43: Überprüfung Szenario 1 – Testfall „WLAN Klein“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
20	0,00021274138570	10	1000	5,30383518707688	10
40	3,83499422245183	10	2000	4,11344261396343	10
60	0,02348235823902	10	4000	8,85316698838111	10
80	0,00030054486713	10	6000	1,39826394553252	10
100	0,00021340425890	10	8000	3,03846591435786	10
200	1,80469306636114	10	10000	3,4999366255473E	10
400	0,00012296731843	10	20000	8,3819607800224E	10
600	9,18942363079384	10	40000	4,33486905565723	10
800	6,41021919001307	10	50000	1,82606960170048	10

Abbildung 44: Überprüfung Szenario 2 – Testfall „Anfrage mit hoher Komplexität“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
20	0,00348444444444	10	1000	2,73085525828629	10
40	0,00159900104058	10	2000	4,5779105626593E	10
60	0,00994118185404	10	4000	1,93720739221883	10
80	0,13234288430022	10	6000	2,87897390540199	10
100	0,00282470588235	10	8000	3,09969205824371	10
200	0,00368471432057	10	10000	4,35789953137531	10
400	0,00015402853419	10	20000	6,99671041765218	10
600	0,00155525331945	10	40000	3,37979378945934	10
800	9,42302491707696	10	50000	4,42370417977993	10

Abbildung 45: Überprüfung Szenario 2 – Testfall „Anfrage mit mittlerer Komplexität“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
20	0,2340238234	10	1000	0,00059904655825	10
40	0,0234282340882	10	2000	1,41137261539713	10
60	0,01152	10	4000	4,7900174719179E	10
80	0,00498639264467	10	6000	2,80627416008406	10
100	0,0146727777777	10	8000	1,28880740527479	10
200	0,0067228	10	10000	6,60294049269864	10
400	0,00027875555555	10	20000	1,82313246048241	10
600	0,00133019390581	10	40000	6,24058275251584	10
800	0,00039522633744	10	50000	9,83824551472209	10

Abbildung 46: Überprüfung Szenario 2 – Testfall „Anfrage mit niedriger Komplexität“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
20	9,23492845823772	10	1000	5,26863913440651	10
40	0,00109408330366	10	2000	3,34255582660912	10
60	0,02342348853934	10	4000	4,43409636919969	10
80	0,00751611871240	10	6000	3,84629309510014	10
100	0,00094070400979	10	8000	2,85812490041135	10
200	0,00051342844732	10	10000	1,44327537156085	10
400	0,00045293156027	10	20000	3,10815387414436	10
600	8,92228189867905	10	40000	1,44383377141777	10
800	3,97660576574711	10	50000	4,44399349844193	10

Abbildung 47: Überprüfung Szenario 3 – Testfall „Aktualisieren von Daten“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
20	0,004917248	10	1000	0,00822710291493	10
40	0,002348234429	10	2000	0,01040507232994	10
60	0,00282211202938	10	4000	0,00053802037743	10
80	0,0060025	10	6000	0,00154864197530	10
100	0,00531709342560	10	8000	0,00166764850458	10
200	0,00282211202938	10	10000	0,00306075871435	10
400	0,00282211202938	10	20000	4,95682327704392	10
600	0,0025088	10	40000	0,00010154836407	10
800	0,01742880907372	10	50000	0,00056512449972	10

Abbildung 48: Überprüfung Szenario 3 – Testfall „Einfügen von Daten“

Datenmenge	Berechnung	Wiederholungen	Datenmenge	Berechnung	Wiederholungen
20	0,00282211202938	10	1000	0,00021769366166	10
40	0,00166213088155	10	2000	4,72875826500806	10
60	0,0234822348298	10	4000	4,52489436103673	10
80	0,01331044214796	10	6000	2,24858981762998	10
100	0,00073572728143	10	8000	4,88217589301836	10
200	0,0034496	10	10000	5,74029656268839	10
400	0,00331291412145	10	20000	1,16194545240551	10
600	0,00308214513911	10	40000	2,68012623405172	10
800	0,00054438658022	10	50000	6,25878869890318	10

Abbildung 49: Überprüfung Szenario 3 – Testfall „Löschen von Daten“