I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

AJAX	Asynchronous JavaScript and XML
DB	Datenbank
DBMS	Datenbankmanagementsystem
JSON	JavaScript Object Notation
PHP	Hypertext Preprocessor
SQL	Structured Query Language

1 Einleitung

1.1 Die dotSource und das Magento Co-Shopping Modul

Die dotSource, eine GmbH welche sich auf E-Commerce spezialisiert hat, erarbeitete bereits eine Konzeption und Umsetzung eines Co-Shopping Moduls auf Basis von Intershop (zu sehen unter http://www.dotsource.de/co-shopping). Dieses Modul beinhaltet Funktionen wie Login per Facebook und Twitter, schreiben von Chatnachrichten sowie das Hinzufügen von Produktfavoriten zur gemeinsamen Einkaufstour. Eine weitere Implementierung soll nun auf Basis des E-Commerce Shopsystems Magento geschehen. Dabei werden die Funktionen der existierenden Bachelorarbeit von Julius Wüstefeld [Wü11] genutzt um dessen Ideen und Konzepte für Intershop auf die Magento Plattform zu übertragen.

1.2 Zielsetzung der Arbeit

Diese Arbeit wird aus zwei Teilen bestehen. Zum Einen wird über die Problematik von Datenhaltung und -verarbeitung in Multiuser Systemen gesprochen und verschiedene Verfahren miteinander verglichen. Diese werden sich stets an der Umsetzung mittels MySQL als Datenbank und PHP als Skriptsprache orientieren. Zum Anderen wird das Co-Shopping Moduls dokumentiert. Bei dessen Implementierung werden die zuvor erarbeiteten Umsetzungsstrategien zur Datenparallelität zum Einsatz kommen.

2 Mehrbenutzerhandling im E-Commerce

2.1 Nutzeranzahl und Skalierbarkeit

Internetdienste werden meist von mehreren Nutzern zugleich verwendet. Im E-Commerce beispielsweise kommt es darauf an mehrere Kunden gleichzeitig zu bedienen (z.B. Produkte ansehen). Nimmt nun die Nutzung des Shops zu, so kann es passieren, dass die aktuell verwendeten technischen Leistungen nicht mehr ausreichen und diese skaliert werden müssen. Dabei wird zwischen zwei Verfahren unterschieden[1]:

Zum Einen die vertikale Skalierung. Dies beschreibt die Verbesserung der aktuell genutzten Ressourcen: beispielsweise eine Speichervergrößerung oder aber auch den Einbau einer weiteren CPU. Als weiteres Verfahren dient die horizontale Skalierung. Es beschreibt das Hinzufügen von weiteren Ressourcen wie Servern und gemeinsame Nutzung der Daten innerhalb des Serververbundes. Im Folgenden wird auf die horizontale Skalierung, sprich die Verteilung und Synchronisation der Daten näher eingegangen.

2.2 Verfahren der Synchronisierung von Daten

Die Datensynchronisation geschieht mittels spezieller Tools. Dabei sollte unterschieden werden, welche Art von Daten auf mehreren Systemen vorhanden sein sollen. Soll eine Datenbank auf mehreren Servern die gleiche Datenbasis beinhalten, so bieten die Datenbankmanagementsysteme (DBMS) meist verschiedene Replikationstools an. Für MySQL wird beispielsweise dafür das Master – Slave Verfahren angewandt[2]: Dabei stellen Slave-Server Lesezugriff auf Daten und Master den Schreibzugriff bereit. Vorteil ist, dass Leseanfragen auf die Slaves aufgeteilt werden kann. Nachteil, dass die Daten nach Schreiben auf einen Master anschließend sämtlichen Slaves mitgeteilt werden müssen. Sollen Daten (z.B. Sessions) vom Dateisystem auf mehreren Servern geteilt werden, so

[1] [Ro07], S. 1
[2] [Prö11], S. 346 ff.

muss ebenfalls eine Software zur Synchronisation zum Einsatz kommen (beispielsweise rsync bei Linux).

Bei der Arbeit mit zu skalierenden Daten bietet es sich an, wenn bei Bedarf eine einfache Lösung zur Vergrößerung der nutzbaren Server existiert. Aufgrund der integrierten Replikationslogik von MySQL wird die Datenhaltung in einer Datenbank in dieser Eigenschaft bevorzugt. Für die Realisierung einer MySQL Replikation sei auf [Prö11] verwiesen.

2.3 Transaktionen und Isolationslevel in Datenbanken

Im Zuge von Änderungsoperationen an einer Datenbasis kann es unter Umständen passieren, dass zusammenhängende Daten in parallel ablaufenden Aktionen geschrieben und im Fehlerfalle komplett auf ihren Ursprungswert gesetzt werden sollen. Zu diesem Zweck gibt es im DBMS sogenannte Transaktionen. Diese bilden eine logische Einheit von mehreren Operationen.[3] Begonnen werden Sie in MySQL mit *BEGIN TRANSACTION* und beendet mit *COMMIT* oder *ROLLBACK*. Der erste Befehl zur Beendigung der Transaktion schreibt die geänderten Daten persistent in die Datenbank. Letzterer dient dem Rücksetzen der geänderten Daten auf den Wert vor der Transaktion.

Da eine Datenbank mehrbenutzerfähig ist, zielt der Einsatz auf das gleichzeitige Abarbeiten mehrerer Nutzeranfragen ab. Damit in diesem Falle Transaktionen nicht hintereinander (seriell) abgearbeitet werden müssen stehen verschiedene Isolationslevel zur Verfügung. Im Folgenden werden 4 Arten genannt, die auch in MySQL zum Einsatz kommen. Die Symbole werden dabei mittels Abbildungen erklärt. Die zugehörige Legende kann der nachfolgenden Tabelle entnommen werden.

[3] [Prö11], S. 180

Tabelle 1: Transaktionslevel - Symbole

Symbol	Bedeutung
R	Lesen von Daten (Read)
W	Schreiben von Daten (Write)
D	Löschen von Daten (Delete)
x	Zustand 1
y	Zustand 2
START TRANSACTION	Transaktion beginnen
COMMIT	Transaktion beenden und Daten konsistent speichern
ROLLBACK	Transaktion beenden und geänderte Daten innerhalb Transaktion zurücksetzen

READ UNCOMMITTED

Dabei ist es anderen Prozessen erlaubt alle Daten von noch nicht bestätigten Transaktionen zu lesen. Dieses Isolationslevel stellt die beste Nebenläufigkeit dar. Sie wird allerdings unter der Einschränkung erzeugt, dass die gelesenen Daten eventuell falsch werden können. Die folgende Abbildung zeigt den Sachverhalt auf.

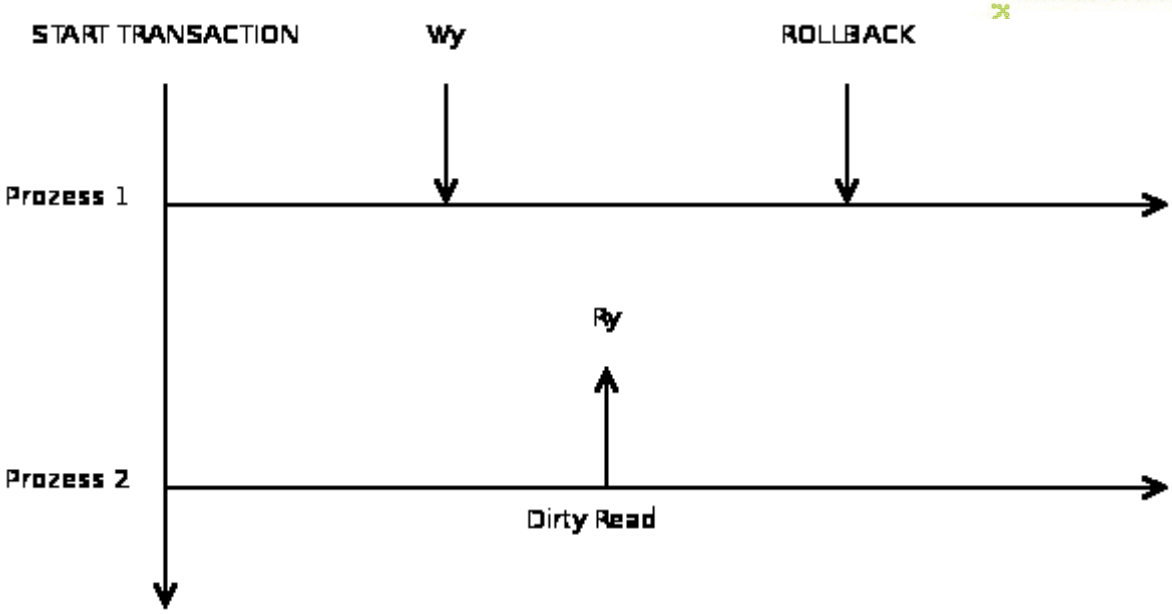


Abbildung 1: DB - Read Uncommitted

Wie in der Abbildung zu sehen liest Prozess 2 Daten bevor diese persistent in die Datenbank geschrieben wurden. Prozess 2 wird nun unabhängig vom Rollback des ersten Prozesses mit den zuvor gelesenen Daten weiterarbeiten, obwohl diese nicht mehr in der Datenbank stehen. Diese Problematik wird in der Literatur als *Dirty Read* bezeichnet.[4]

READ COMMITTED

Bei diesem Verfahren dürfen Prozesse neue Daten erst nutzen, sobald diese mittels COMMIT in die Datenbank geschrieben wurden. Da nebenläufige Prozesse allerdings vor und nach Commits eines schreibenden Prozesses arbeiten, kann es vorkommen, dass sie innerhalb der eigenen Transaktion mit mehreren unterschiedlichen Ergebnissen arbeiten. Dies wird als *Unrepeatable Read* bezeichnet. Die folgende Abbildung stellt diese Problematik dar. Dabei liest Prozess 2 einmal die ursprünglichen und ein anderes Mal die neuen Daten innerhalb einer Transaktion.

[4][Prö11], S. 187

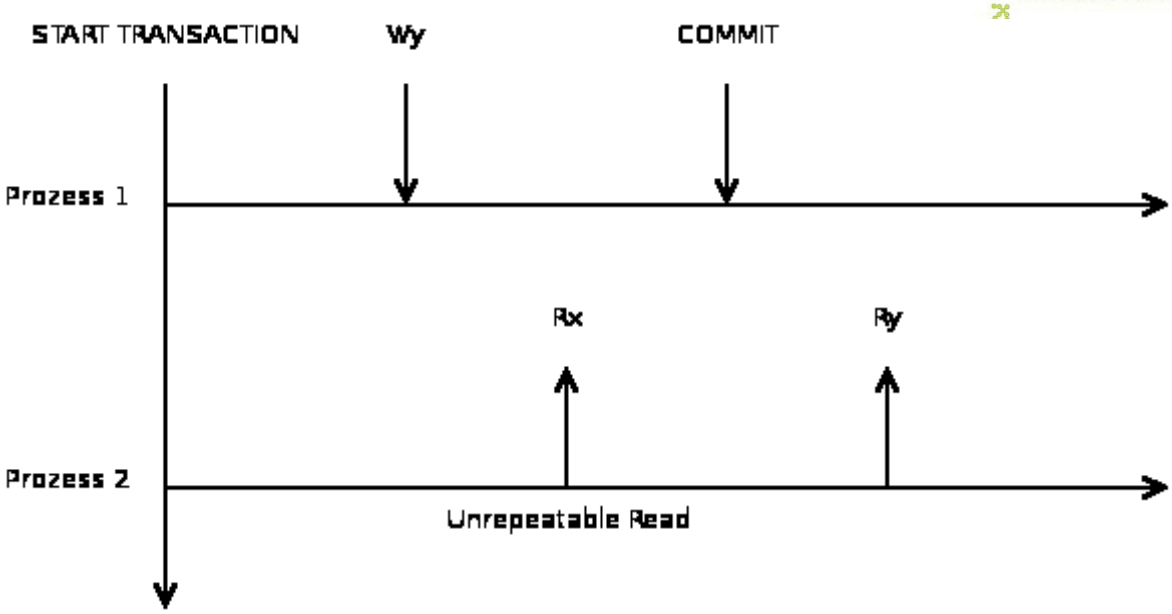


Abbildung 2: DB - Read Committed

REPEATABLE READ

Dieses Vefahren löst die Problematik des *Unrepeatable Reads* der vorhergehenden Verfahren. Hierbei bleiben einmal gelesene Daten bis Ende der Transaktion erhalten. Dies wird erreicht, indem Änderungen am Datenbestand nach erstmaligem Lesen bis Ende der eigenen Transaktion nicht mitgeteilt werden. Es kann also innerhalb des logischen Blocks stets auf die gleichen Daten zugegriffen werden. Leider entsteht bei diesem Verfahren auch ein Fehler: Da einmal gelesene Daten nicht wieder neu geladen werden, kann es passieren, dass ein anderer Prozess diese bereits gelöscht hat, aber der eigene Prozess weiterhin mit den nun nicht mehr existenten Daten arbeitet. Diese Problematik nennt sich *Phantom Reads*. Im Folgenden wird dieses Isolationslevel abermals anhand von zwei parallel arbeitenden Prozessen dargestellt.

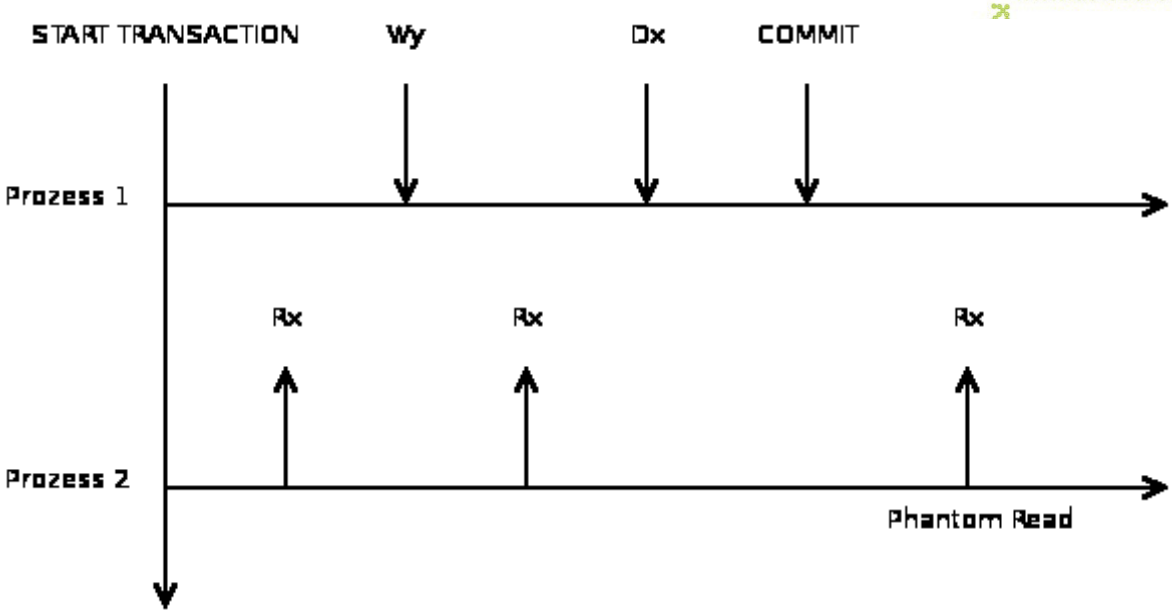


Abbildung 3: DB - Repeatable Read

SERIALIZABLE

Das letzte und sicherste Verfahren (im Bezug auf datenkonsistentes arbeiten) bietet das Isolationslevel Serializable. Dieses verhindert Schreibarbeiten am Datenbestand, solange mindestens ein Prozess mit diesen Daten lesend arbeitet. Der Schreibprozess muss dabei warten, bis alle Lesetransaktionen beendet sind und die Daten zum Schreiben freigeben. Es wird nun ein solcher Wartevorgang dargestellt.

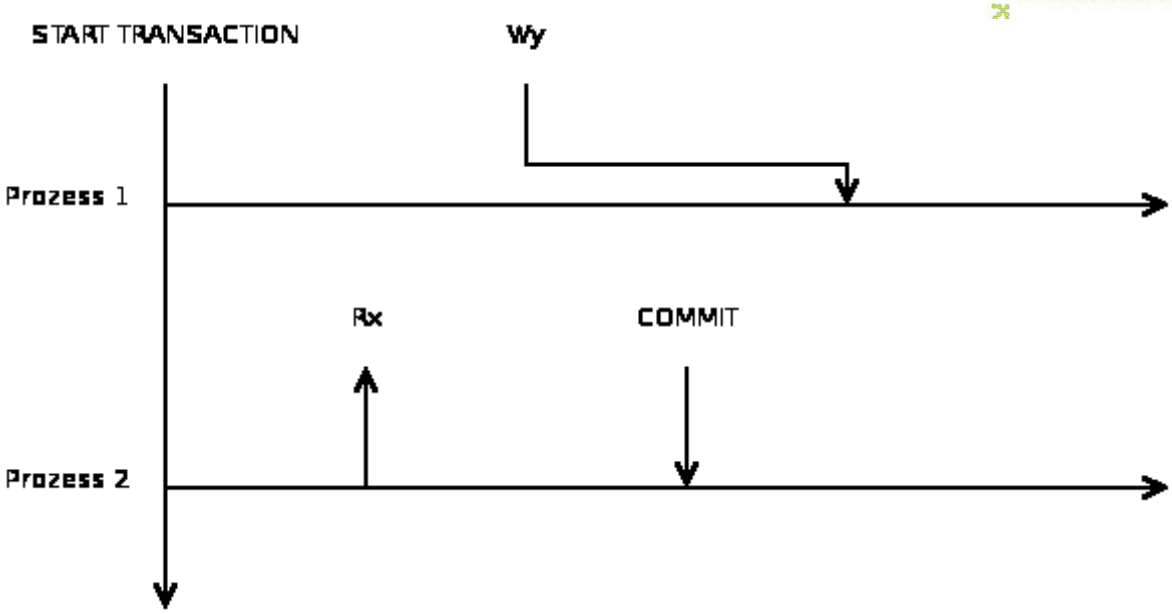


Abbildung 4: DB - Serializable

Falls nicht explizit angegeben wird *Repeatable Read* in MySQL als Standardeinstellung genutzt.[5] Des Weiteren kann der Isolationslevel in MySQL global oder pro Sitzung eingestellt werden:

SET [GLOBAL|SESSION] TRANSACTION ISOLATION LEVEL [ISOLEVEL]

Die eben genannten Isolationslevel sind nun im Bezug auf die Co-Shopping Komponente entsprechend anzuwenden. Es ist dabei die Parallelität mit der Datenkonsistenz abzuwiegen. Da ein Chat von kurzweiligen, dafür in schneller Folge geschriebenen, Texten abhängt, ist es von Vorteil den Transaktionslevel Read Uncommitted zu einzustellen. Es überwiegt somit der Wunsch einer schnellen Verarbeitung von Anfragen (lesend und schreibend) gegen ein konsistentes Lesen.

[5][Prö11], S. 189

2.4 File-Locking auf dem Dateisystem

Eine Alternative zur Datenhaltung in einer Datenbank stellt die Abspeicherung auf dem Dateisystem dar. Hierbei kann in PHP auf verschieden Funktionen zurückgegriffen werden. Zum Lesen und Schreiben von Dateien kann man sich der Dateisystem Operationen *fopen*, *fread*, *fwrite* und *fclose* bedienen. Mithilfe dieser wird von einem Filehandle gelesen oder darauf geschrieben.

Damit dieses Verfahren Multiuser fähig wird muss die zu schreibende Datei gesperrt werden. Dies kann mit der Operation *flock* geschehen. Dabei wird versucht auf die zu öffnende Datei einen exklusiven Zugriff anzufordern. Alle anderen Prozesse, welche innerhalb der Sperrzeit versuchen einen weiteren flock anzufordern werden warten, bis die erste Sperre aufgehoben ist. Es folgt ein Verwendungsbeispiel.

```php
<?php
$fileName = 'chat.txt';
$fileHandle = fopen($fileName, "w");

// Exklusiven Schreibzugriff setzen
$lockErhalten = flock($fileHandle, LOCK_EX);

if (!$lockErhalten) {
    // Behandlung falls kein Lock erfolgte.
    // ...
    return;
}


// Arbeit mit den Daten
// ...

// Freigabe der Datei
flock($fileHandle, LOCK_UN);

// Schließen der Datei
fclose($fileHandle);
```

Abbildung 5: PHP - Dateisperre

Da innerhalb der Chat Datei viele Daten liegen würden (Nachrichten und Nutzer aller aktuellen Chats), ist es von Vorteil die Daten zu strukturieren. PHP bietet dazu die Möglichkeit Objekte zu serialisieren, d.h. strukturierte Daten in eine Zeichenkettenrepräsentation umzuwandeln, damit diese in eine Datei geschrieben werden können. Dazu wird der zuvor gezeigte Quelltext um die folgenden Anweisungen ergänzt:

```php
<?php
// Datei öffenen und Dateisperre mittels flock ausführen
// ...


// Datei vollständig lesen
$fileData = fread($fileHandle, filesize($fileName));

// Dateidaten in ein Objekt überführen
$chatObjekt = unserialize($fileData);


// Arbeit mit den Daten
// ...


// Daten in einen String überführen
$fileData = serialize($chatObjekt);

// Daten schreiben
fwrite($fileData);


// Sperre aufheben und schließen der Datei
// ...
```

Abbildung 6: PHP - Dateisperre und Datenserialisierung

Jedoch ergeben sich aus diesem Verfahren mit Dateiarbeit einige Problematiken, welche nun kurz erwähnt werden:

- Die Dateisperrung mittels *flock*() basiert auf einer freiwilligen Beachtung

der Sperre.[6] Prozesse können diese Sperre umgehen, indem sie die Datei öffnen ohne anschließend nach einer Sperre zu fragen.

- Um aus einer Datei ein Objekt zu erzeugen muss diese komplett gelesen und in ein Objekt überführt werden. Dies würde zur Folge haben, dass eine Struktur benötigt wird um die Leseoperation gering zu halten. Beispielsweise durch Speicherung eines jeden Chats in eine separate Datei.
- Ebenso wie ein komplettes Lesen einer Datei muss bei einer Änderung der Daten auch die gesamte Datei neu auf das Dateisystem geschrieben werden.

2.5 Auswahl eines Verfahren zur Datenhaltung

Aufgrund der in den vorhergehenden Kapiteln genannten Vor- und Nachteile wird die Nutzung einer Datenbank angestrebt. Die zuvor aufgeworfenen Fragen und Probleme werden in der nun folgenden Tabelle nochmals anhand der beiden Datenhaltungsverfahren verglichen.

Tabelle 2: Datennutzung - Vergleich mit DB und Dateisystem

Problematik	Datenbank	Dateisystem
Paralleler Zugriff auf Daten	Dank unterschiedlichen Isolationslevels wird nebenläufiger Zugriff bei Lese- und Schreibzugriffen ermöglicht.	Nebenläufigkeit nur im Lesezugriff möglich.
Synchronisierung von	Ebenfalls durch direkte	Bedingt möglich durch

[6] [Skl09], S. 731

Problematik	Datenbank	Dateisystem
Daten auf Prozessebene	Auswahl eines Isolationslevels.	Nutzung von Dateiblockierung (flock()).
Synchronisierung von Daten auf Serverebene	Replikationslogik ist integriert.	Spiegelung von Ordnern des Dateisystems.
Menge an Lesedaten	Direkte Auswahl von Abfragedaten in SELECT-Queries.	Auslesen einer kompletten Datei.
Menge an Schreibdaten	DBMS kümmert sich um optimierte Datenhaltung.	Bei Schreibänderungen muss Datei komplett neu erstellt und befüllt werden.

Im Vergleich zur Umsetzung mittels Dateisystem überwiegen mehrere Lösungen mittels Datenbank. Da hier zum Einen viele Problematiken bereits im DBMS selbst optimiert werden und einige Verfahren wie die Replikation auf mehreren Servern bereits als aktivierbare Funktionen integriert sind.

2.6 Partitionierung von Tabellen

Eine weitere Möglichkeit von MySQL Datenhaltung performant zu gestalten ist es, sogenannte partitionierte Tabellen anzulegen. Diese stellen auf der Anwendungsebene eine einfache Tabelle dar. Intern ist jedoch die Datenhaltung mittels einer Verteilungsfunktion in kleinere Teiltabellen getrennt. Vorteile sind dabei, dass

- Bei Änderungen nicht die ganze Tabelle gesperrt wird, sondern nur die aktuelle Teilpartition

- Partitionen können physisch auf das Dateisystem angepasst werden[7]
- Bei SELECT Anfragen muss mitunter nur in einer Partition gesucht werden[8] (Abhängig von der Partitionierungsfunktion)

Anwendung finden könnte diese Partitionierung bei den gespeicherten Chatnachrichten. Befinden sich sehr viele Nachrichten in der Datenbank, so kann es durchaus sinnvoll sein, wenn man die Daten mittels einer Partition innerhalb verschiedener, physisch voneinander getrennter, Datenbestände hält. Ein Beispiel zur Implementierung kann den nachfolgenden SQL-Queries entnommen werden. Dabei wurde die Verteilungsfunktion mit Modulo 4 gewählt, was eine Trennung der Daten in vier Teilpartitionen bewirkt.

```
-- Anlegen der Tabelle
CREATE TABLE `message` (
    `message_id` INT(10) UNSIGNED NOT NULL AUTO_INCREMENT,
    `chat_id` INT(10) UNSIGNED NOT NULL,
    `text` VARCHAR(255) NOT NULL,
    PRIMARY KEY(`message_id`)
);

-- Erstellen der Partitionen
ALTER TABLE `message`
PARTITION BY RANGE(`chat_id` MOD 4) (
    PARTITION p_0 VALUES IN (0),
    PARTITION p_1 VALUES IN (1),
    PARTITION p_2 VALUES IN (2),
    PARTITION p_3 VALUES IN (3)
);
```

Abbildung 7: SQL - Partitionierung von Tabellen

Partitionierung in MySQL steht ab Version 5.1 zur Verfügung. Aufgrund von fehlenden Statistiken in Bezug auf das Nutzungsvolumen des aktuell zu entwickelnden Chats sei das soeben genannte Verfahren der Partitionierung als ein weiteres, bei Bedarf einzusetzendes Verfahren anzusehen.

[7] [Sch09], S. 278
[8] [Prö11], S. 427 f.

3 Dokumentation des Co-Shopping Moduls

3.1 Funktionsanforderungen, Abgrenzung und Funktionskomponenten

Die nachfolgende Konzeptionsdokumentation bezieht sich auf Funktionen, welche in [Wüs11] genannt und bereits für Enfinity umgesetzt wurden. Diese Dokumentation befasst sich ausschließlich mit Umsetzungsgedanken des ScCoShopping-Modules für Magento.

Bei der Entwicklung werden die folgenden Funktionen als Teilkomponenten des gesamten Chats betrachtet. Dabei wird die Unterteilung innerhalb der Funktionslogik, wie auch der Datenhaltung geschehen.

Tabelle 3: Teilkomponenten

Teilkomponente	Kurzbeschreibung Funktionen	Kurzbeschreibung Daten
Chat	Anlegen, editieren und löschen eines Chats.	Einstellungen des aktuellen Chats, dient als Zuordnung für andere Chatdaten wie z.B. Nutzer.
Nutzer	Einem Chat zuordnen und Statusänderungen tätigen (z.B. als Administrator markieren).	Nutzerdaten wie Name, Bild und aktuell angesehenes Produkt.
Nachricht	Speichern Neuer und löschen alter Nachrichten.	Daten wie Text und zugeordneter Nutzer.
Favorit	Anlegen und löschen von Produkt-Favoriten zu einem	ProduktcacheId und NutzerId.

	Nutzer.	
Produkt	Produktdaten von Magento laden und in Datenbank abspeichern. Storeabhängige Speicherung. Bei Produktänderungen aktualisieren.	Daten wie Name, Preis und Produkt URL.
Chatfenster	Setzen und lesen der aktuellen Fensterposition und geöffneter Tabs.	Fensterposition und geöffnete Tabs

3.2 Anforderungen an Magento

Zur Entwicklung verwendete Magentoversion ist: CE 1.4.2.0
Als Orientierung dienten zur Versionsauswahl die aktuell verwendeten Projektversionen innerhalb der dotSource.

Die Umsetzung soll unabhängig zur jeweilig verwendeten Magentoinstallation implementiert werden, sodass das Modul später in andere Magentos ohne viel Installationsaufwand eingebaut werden kann.

3.3 Caching von Produktdaten

Viele Funktionen des Moduls nutzen eine Anzeige von Produkten. Damit nicht bei jedem Request ein Magento Produkt-Model instanziiert werden muss, werden die Produkte in der Chat Datenbank zwischengespeichert. Es werden dabei alle benötigten Produktdaten *storespezifisch* abgelegt. D.h. es wird pro *StoreView* ein Produkt mit dessen Attributen in der Chat-DB gecached. Demzufolge wird (z.B. wenn ein Produkt einer Nachricht zugeordnet ist) die Produktdaten des StoreViews genutzt, von welchem aus die Verlinkung stattfand.

Zusätzlich ist zu beachten, dass Produktbilder über Magento generiert:

```
Mage_Catalog_Helper_Image::init($product)->resize($width)->__toString();
```

und in der Chat-DB gecached werden. Bei jeder Verwendung der gecachten URLs sollte deshalb im Filesystem geprüft werden, ob die gecachten Produktbilder noch existieren, oder sie müssen per Magento Funktion neu angelegt und wieder in der Chat-DB abgelegt werden.

3.4 Datenhaltung mittels MySQL

Bei der Speicherung der Chatdaten wird eine MySQL Datenbank verwendet. Das DB-Schema kann der nachfolgenden Abbildung entnommen werden.

Abbildung 8: DB-Schema

Um eine Datenkonsistenz zu errreichen werden Foreign Keys verwendet. Es ist im Laufenden Betrieb abzuwägen, ob diese zu löschen sind um Performance-Gewinne zu erzielen. Jedoch müssen diese dann mittels Triggern oder Cron-Funktionen ersetzt werden, damit keine ungültigen Daten erhalten bleiben.

Das Passwort für den Chat wird mittels einfacher md5 Funktion generiert.

```
return md5(uniqid('sccoshopping_chat_'));
```

Da md5 nicht automatisch unique ist, wird beim Einloggen in einen existierenden Chat das Passwort und die Chat ID benötigt.

3.5 Nutzung von Transaktionen

Bei der Speicherung der Daten können Transaktionen verwendet werden. Diese werden eingesetzt bei:

1. Hinzufügen eines Chats mit gleichzeitigem Anlegen des Initiators
2. Hinzufügen eines Produkt-Caches mit gleichzeitigem Anlegen des zugehörigen:
    a. Favoriten
    b. Links in einer Message
    c. Hinzufügen eines Users mit aktuell angesehenem Produkt
    d. Hinzufügen des aktuell angesehen Produktes zu einem existierenden Nutzer

3.6 Nutzung von Magento Events

Zum Setzen von Chatdaten werden die folgenden Magento-Events genutzt. Diese müssen bei jedem assoziierten Request gesetzt werden, damit stets das richtige zugeordnete Produkt angezeigt wird.

Tabelle 4: Eventnutzung Magento

Event	Nutzung
catalog_product_save_after • product	Cache der Produktdaten aktualisieren
controller_action_predispatch	Prüfen, ob die letzte Seite eine Produktdetailseite war. Falls ja, dann die Änderungen zum jetzigen Request (keine Produktdetailseite/andere Produktdetailseite) in die Datenbank schreiben.

3.7 Löschen von alten/unerwünschten Chats via Cron

Es kann passieren, dass ein Chat von den Usern nicht richtig geschlossen wird und in der DB verbleibt. Eine Überprüfung nach Aktualität bei jeder Änderung wäre im Laufenden Betrieb unpraktisch. Darum wird der Magento eigene Cron Mechanismus verwendet, welche beispielsweise aller 30 Minuten lange nicht mehr verwendete Chats löscht.

3.8 Überblick Datenbankoperationen

Im Folgenden werden alle Datenbankanfragen des Chats aufgelistet. Die Tabellen werden dabei nach Schreib- und Lesezugriffen geordnet.

Tabelle 5: DB Schreibzugriffe

Funktion	Teilkomponente
Chat hinzufügen	Chat
Chat entfernen	Chat
Nutzer zu Chat hinzufügen	Nutzer
Nutzer von Chat entfernen	Nutzer
Nutzer erlauben	Nutzer
Nutzer zu Operator machen	Nutzer
Aktuell angesehenes Produkt am Nutzer setzen	Nutzer
Chatnachricht schreiben (+ mit Produktverlinkung)	Nachricht
Letzte Chatnachrichten löschen	Nachricht
Favorit hinzufügen	Favorit
Favorit löschen	Favorit
Änderung des Produktdatencaches (Preis, Name, Produkt-URL, Bild-URL)	Produkt

Tabelle 6: DB Lesezugriffe

Funktion	Teilkomponente

Hole alle Nutzer des Chats (+ aktuell angesehene Produkte)	Nutzer
Lade alle letzten X Nachrichten	Nachricht
Zeige alle Favoriten • mit Anzahl der User, welche als Fav markiert haben • mit Hervorhebung der eigenen als Fav markierten Einträge	Favorit
Lade Produkt-Cachedaten bei: • Nutzeransicht des Produktes • Nachrichtenverlinkung • Favoritenverlinkung	Produkt

3.9 Requestlogik an Server

Es müssen die folgenden Punkte beachtet werden:

- Anfragen sollten möglichst in Echtzeit gestellt werden, d.h. relativ oft sollte per AJAX ein Polling an den Server stattfinden.
- Alternativ kann auch die Logik wie bei meebo.com zu sehen genutzt werden. Diese funktioniert nach folgender Logik:
  - Es wird ständig eine AJAX Anfrage an den Server gestartet. Diese hat einen Timeout von 30 Sekunden
  - Der Server bearbeitet in der Zeit die Anfrage und
    - wartet (z.B. **sleep(1)**),
    - prüft anschließend wieder nach neuen Ereignissen und
    - gibt bei Änderungen die neuen Daten zurück.
    - Ansonstent Verfahren durchziehen, bis Anfrage nach 30 Sekunden vom User gecancelled wird (**ignore_user_abort(false)**)
  - Zu beachten ist bei diesem Verfahren,

  - wie viel Ressourcen pro wartenden Chat-Thread alloziert bleiben
  - Wie hoch die Standard Threadanzahl eines PHP Servers ist
  - Ob eine stets aktive Verbindung via AJAX-HTTP auch andere Anfragen an den Server beeinflussen
- Oder Implementierung von beiden Verfahren. Der Mehraufwand betrifft ausschließlich eine IF-Condition im JS und PHP. Dadurch wird dem Shopbetreiber das Verfahren zur Auswahl im BO überlassen und über die Zeit werden best-cases generiert.

Für die aktuelle Implementierung wird nur das erste Verfahren, sprich das Polling via AJAX, verwendet, da letzteres einer genaueren Analyse bedarf.

3.10 Feinspezifikation von Magento Requesthandler Funktionen (AJAX<>PHP)

Die Übergabe von Daten zwischen Client und Server geschieht mittels JSON Format. Sämtliche Serverfunktionen sind zu finden unter: **Dotsource_Sccoshopping_Model_Request_*** in diesen Klassen finden sich auch die Konstanten zur Zuordnung der Anfrage Parameter mit den entsprechenden Funktionen. Die nachfolgende Tabelle stellt die Namen der Funktionen, dessen zugeordneten Parameterkey und die benötigten Parameter dar. Funktionen ohne Parameterkey werden serverseitig verwendet und von JavaScript nicht direkt angesprochen.

Tabelle 7: Requesthandler Funktionen

Funktion	Parameterkey für Requesthandler: REQUEST_KEY_*	Parameter
addChat	ADD_CHAT	allowedUserCount: int isOperatorConfirmationRequired: bool
removeChat	REMOVE_CHAT	

deleteAllInactiveChats		
addUser	ADD_USER	chatId: int chatPassword: string name: string imageUrl: string
removeUser	REMOVE_USER	userId: int
makeUserOperator	MAKE_USER_OPERATOR	userId: int
setViewedProduct		productId: int
unsetViewedProduct		
getAllActiveChatUser	GET_ALL_ACTIVE_CHAT_USER	
getAllWaitingChatUser	GET_ALL_WAITING_CHAT_USER	
addMessage	ADD_MESSAGE	
truncateMessageStack		
getLatestMessages	GET_LATEST_MESSAGES	$firstMessageIdToLoad: int
addFavourites	ADD_FAVOURITES	productIds: array
removeFavourites	REMOVE_FAVOURITES	productIds: array
getFavourites	GET_FAVOURITES	
refreshProductCacheStoreData		productId: int
getProductCache		productId: int
getProductCacheId		productId: int
removeProductCache		productId: int storeId: null|int
setIsWindowOpen	SET_IS_WINDOW_OPEN	$open: bool
isWindowOpen		
saveWindowPosition	SAVE_WINDOW_POSITION	$top: int $left: int
getWindowPosition		

setOpenTabIds	SET_OPEN_TAB_IDS	$tabIds: array
getOpenTabIds		

Die Parameter: **chatId**, **userId** und **storeId** werden vom AJAX-Request bei den meisten Funktionen nicht benötigt. Sie werden automatisch vor Aufruf der entsprechenden Funktionen gesetzt. Alle Funktionen, welche die drei Standardparameter nicht als separaten Parameter erlauben, nutzen standardmäßig die Daten aus der Session. Falls doch für einige Funktionen andere chat-, user- und store Parameter benötigt werden, so können diese bei Instanziierung oder später durch eine entsprechende Getter-Funktion geändert werden.

Die übergebenen Anfragen werden anhand des Parameterkeys auf die entsprechenden Funktionen gemappt. Zurückgegeben wird stets der gleiche Key mit den neuen Daten. Diese sind eines der folgenden drei Werte:

- Die zu erwartenden Daten (string|array)
- Einen leeren String im Erfolgsfall, wenn keine Daten erwartet werden
- Den folgenden Eintrag, falls in der Funktion ein Fehler aufgetreten ist: array(‚exception‘ => ‚Error String‘)

Die Rückgabedaten bei einer lesenden Anfrage (z.B. gib die letzten Nachtichten zurück), beinhalten jeweils die kompletten Daten der jeweiligen Tabelle. Im Folgenden sind alle aktuell verwendeten Anfragen und deren zugehörige Komponenten genannt. Innerhalb des aufrufenden Funktion (PHP, AJAX) kann anschließend eine Filterung nach nicht benötigten Daten erfolgen.

Tabelle 8: Rückgabekomponenten der Requesthandler Funktionen

Funktion	Rückgabedaten
getAllChatUser	Nutzer Aktuell angesehenes Produkt
getLatestMessages	Nachricht Schreibender Nutzer

	Zugeordnetes Produkt
getFavourites	Favorit Produkt Nutzer

Es existiert eine Ausnahme zur oben genannten Logik: Wenn bei einem AJAX-Request kein Chat oder kein ChatUser zugeordnet ist und auch keine Anfrage an **addUser()** geschieht, so wird angenommen, dass

a.) Der User einen falschen Request aufruft oder

b.) Der User in einem anderen Tab deaktiviert wurde

Dabei wird ein gesondertes Objekt zurückgegeben, welches folgenden Aufbau hat: array(‚exeption' => ‚User not Found Error String').

Im Folgenden wird aufgezeigt, wie ein Ajax-Request in PHP abgefangen und bearbeitet wird. Dabei kommen die zuvor genannten Funktionen und Parameterkeys zum Einsatz, damit eine Zuordnung zwischen Anfrage und Bearbeiter getroffen werden kann. Die Liste der RequestHandler wird dabei über die config.xml gesetzt. Durch hinzufügen neuer Einträge können auch andere Module Daten erhalten und verarbeiten. Die XML Struktur sieht wie folgt aus.

```
<config>
  <global>
    <sccoshopping>
      <request_model>
        <chat> sccoshopping/request_chat</chat>
        <user> sccoshopping/request_user</user>
        <message> sccoshopping/request_message</message>
      </request_model>
    </sccoshopping>
  </global>
</config>
```

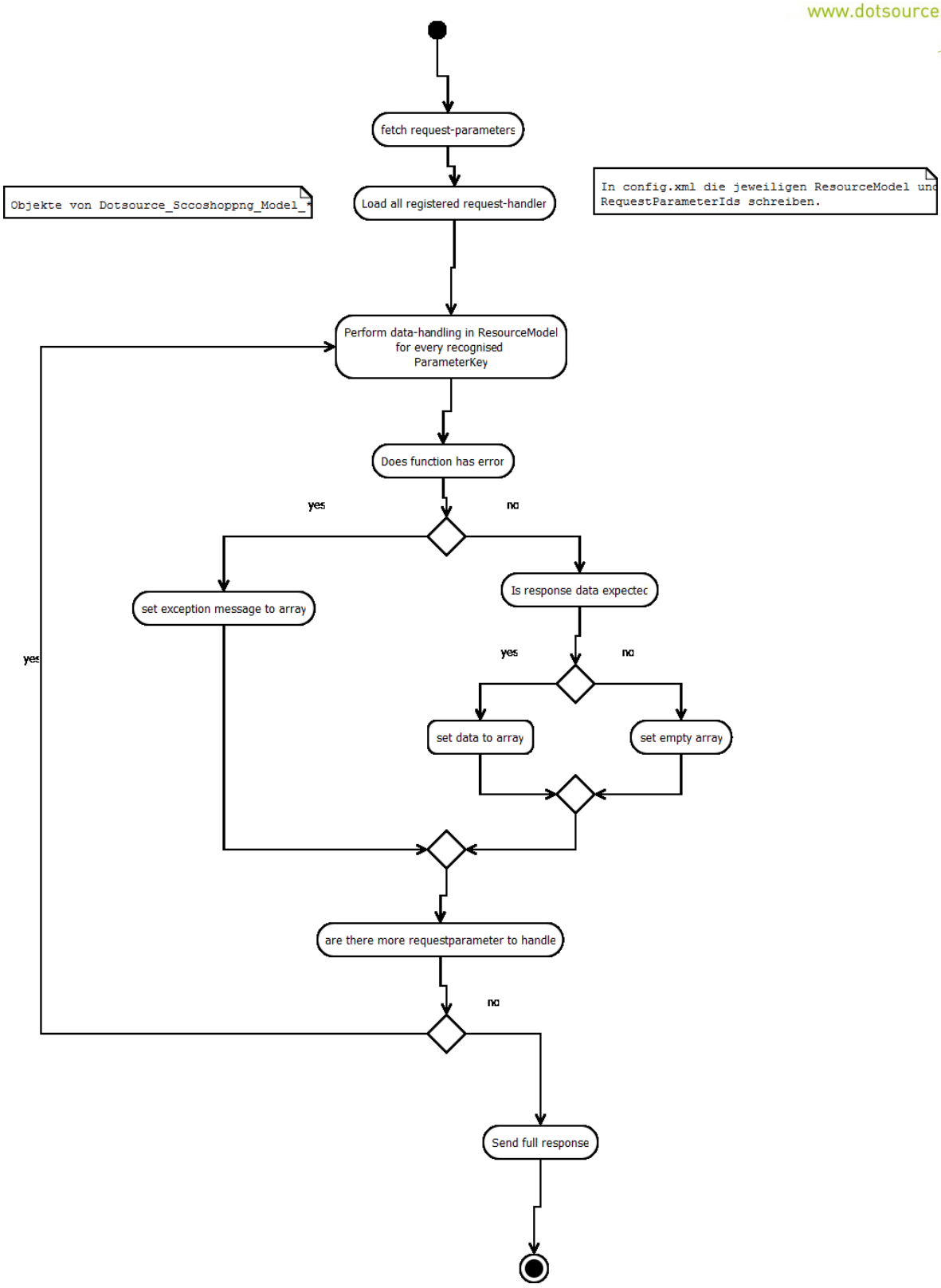


Abbildung 9: Requestlogik AJAX<>PHP

3.11 Klassendiagramme

Die aus den zuvor genannten Funktionen erzeugten Klassendiagramme können in den Anhängen *Anlage C: Klassendiagramm der Chat-Funktionen* sowie *Anlage D: Klassendiagramm der Sessionnutzung, Controlleranfrag und Blockverwendung (Abstrakt)* entnommen werden.

3.12 Nutzung der Magento Session

In Magento wird die Session zur Speicherung von:

- der aktuellen **chat_id** und **user_id**
- Fensterposition des Chats
- Geöffnet/Geschlossen Status
- Zuletzt geöffnete Tabs

genutzt. Zum Einen sind diese Daten nur beim eigenen Nutzer von Wichtigkeit (z.B. Zuletzt geöffnete Tabs), zum Anderen müssen diese eine Zuordnung zum Chat und User zwischen Seitenaufrufen treffen.

Eine Zuordnung: Nutzer => Chat geschieht demzufolge über eine Magento Session. Nun kann es allerdings passieren, dass der gleiche Nutzer über zwei Browser auf dem Shop ist. Magento legt dabei eine weitere Customer Session an. In der Implementierung wird dieser Fall nicht betrachtet. Der Customer kann demzufolge über unterschiedliche Magento Sessions in mehreren Chats unterwegs sein. Eine genauere Zuordnung kann nicht getroffen werden, da Gastkunden selbst in Magento keine Zuordnung zu anderen Sessions besitzen (Achtung: Es ist die Rede von **Customer_Model_Session**, nicht dem zugeordneten Mage_Customer_Model_Customer).

Eine weitere Frage stellt sich bei der Magento Einstellung Kunden *global* oder *pro Webseite* zu verteilen. Dabei ist zu beachten, dass der Chat nur auf einer Webseite aktiv ist. Folgendes Beispiel soll dies verdeutlichen:

- Kunde1 von Webseite1 gibt link an Kunde2 von Webseite2.

- Kunde2 klickt auf Link
- Er wird zwar nun als Chat-User eingetragen, allerdings nur auf Webseite1.
- Wenn er nun normal auf seiner Webseite2 chatten möchte, so ist dies nicht möglich (da Magento eine eigene Session pro Webseite initiiert.)

Eine Zuordnung für beide Webseiten ist hierbei ebenfalls nicht möglich, da selbst Magento nicht weiß, dass ein und dieselbe Person auf 2 Webseiten aktiv ist. Ein Problem sollte dies jedoch trotzdem nicht darstellen, da die Logik von Verteilung pro Webseite auch auf getrennte Nutzerkonten und Produktdaten abzielt und im Normalfall keine Verbindung unter den Webseiten schaffen soll.

3.13 Bekannte Probleme bei der Entwicklung

Bei AJAX Requests ausgehend von einer HTTP Seite auf eine SSL Verschlüsselte oder umgekehrt, wird dieser Request nicht ausgeführt. Es muss demzufolge eine Abfrage im Template geschehen, ob alle die aktuelle Anfrage per HTTP(S) ausgeführt werden soll.

4 Fazit

Die Bearbeitung von Problematiken der schnellen Datenverarbeitung in einem Chat-System zeigten viele Möglichkeiten der effektiven Verarbeitung innerhalb der Verwendung einer Datenbank und Programmiersprache auf. Bei der Konzeption kamen diese Ideen wie Transaktionslevel anschließend zum Einsatz. Nach Implementierung des Chats können diese später überprüft und auf weitere Optimierungsmaßnahmen hin untersucht werden. Die dabei genannten Ideen wie Replikationen sollen dabei als Möglichkeiten der Verbesserung dienen. Auch kann bei einer Optimierung die Clientseite mit einbezogen werden. Beispielsweise bei der Abarbeitung von Daten und der Anfrageintervallen an den Server. Dafür sei hier jedoch auf weiterführende Arbeiten verwiesen.

Anlagenverzeichnis

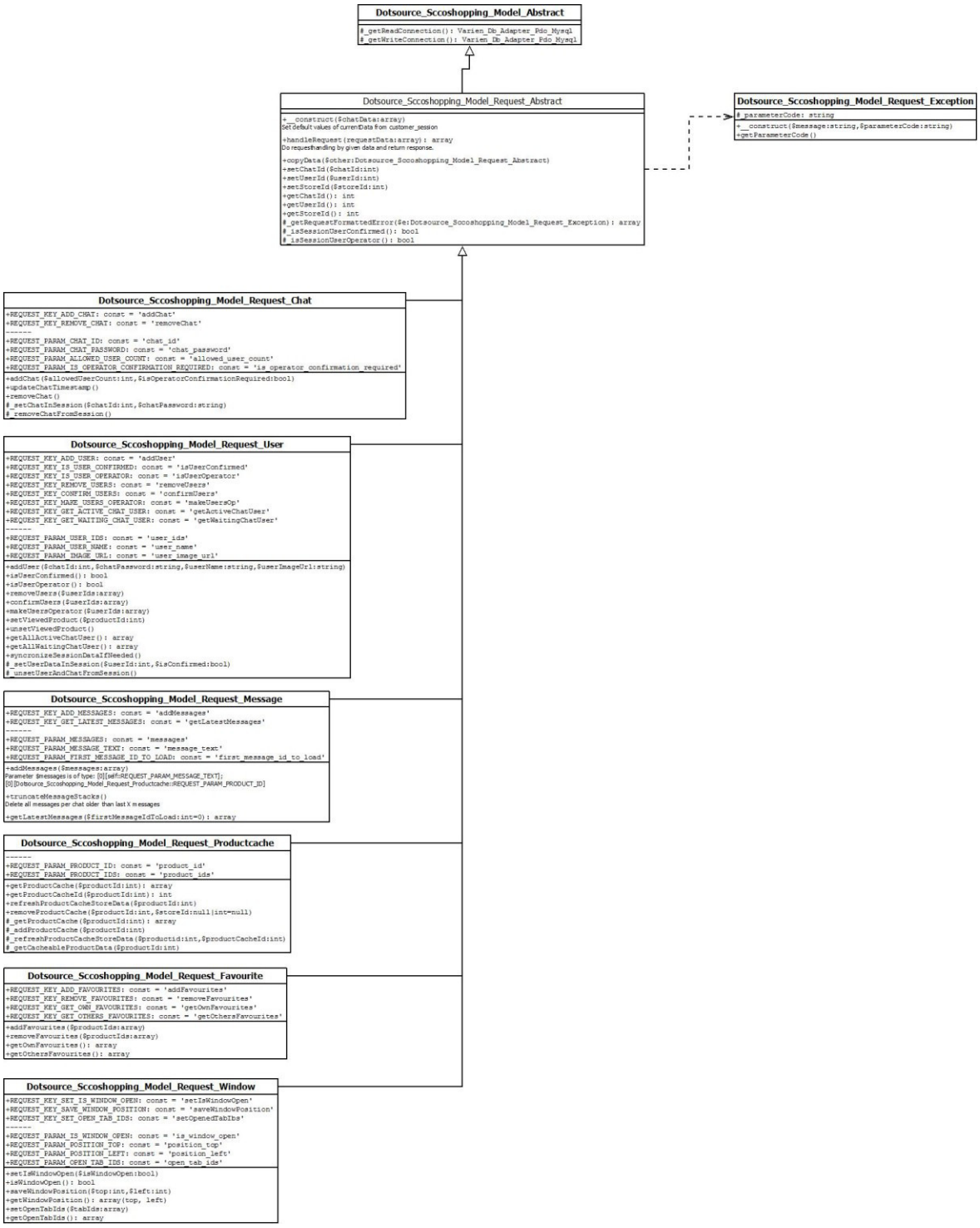
Anlage A: Link und Kurzbeschreibung zu wichtigen genannten PHP Datei- und Serialisierungsfunktionen

PHP Funktion	Kurzbeschreibung	Link
fclose	Schließen einer Dateiressource	http://de.php.net/manual/de/function.fclose.php
fread	Lesen von Daten einer Datei	http://de.php.net/manual/de/function.fread.php
flock	Sperren und Freigeben einer Dateiressource	http://de.php.net/manual/de/function.flock.php
fopen	Öffnen einer Datei	http://de.php.net/manual/de/function.fopen.php
fwrite	Schreiben in eine Datei	http://de.php.net/manual/de/function.fwrite.php
serialize	Überführen einer Datenstruktur in einen String	http://de.php.net/manual/de/function.serialize.php
unserialize	Datenstrukturen aus einem String extrahieren	http://de.php.net/manual/de/function.unserialize.php

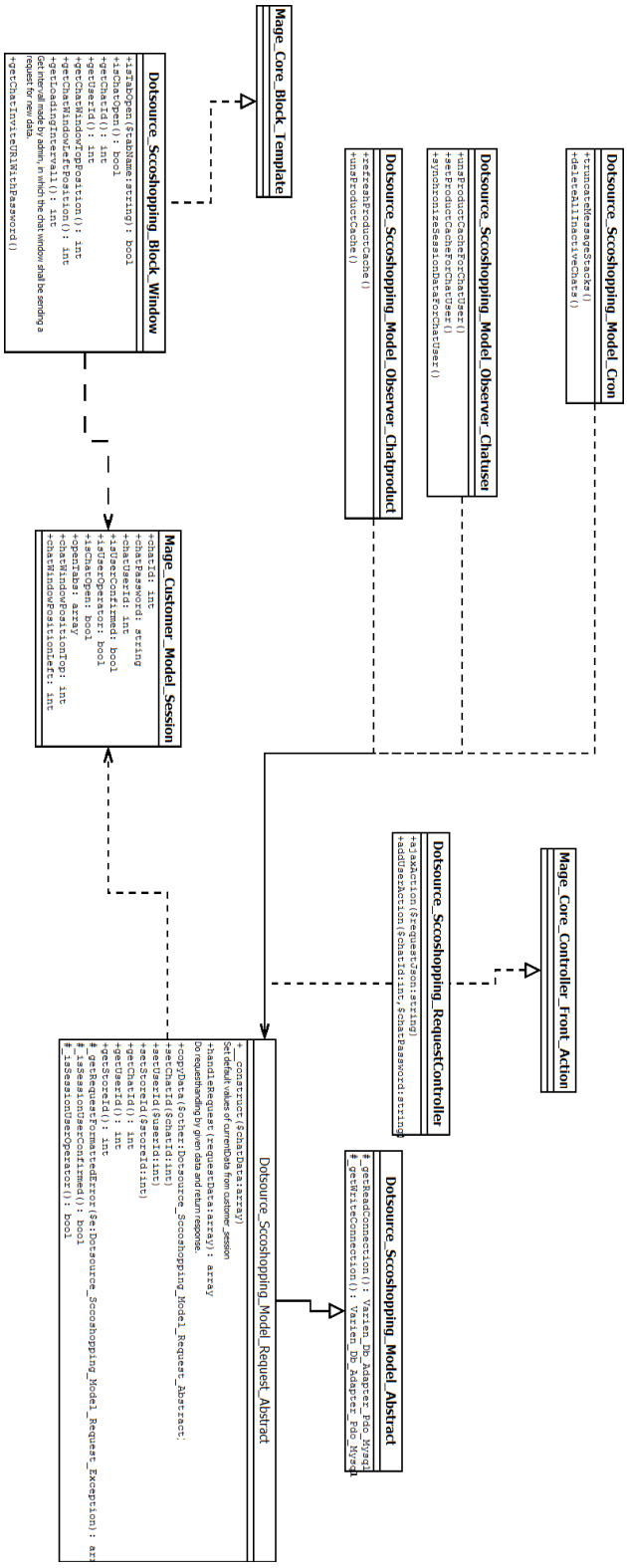
Anlage B: Links zu MySQL Dokumentationen der genannten Funktionen

Funktion	Kurzbeschreibung	Link
Partitionierung	Physische Aufteilung von Tabellen mittels Verteilungsfunktion	http://dev.mysql.com/doc/refman/5.1/de/partitioning.html
Replikation	Horizontale Skalierung durch Maste-Slave Server Systeme	http://dev.mysql.com/doc/refman/5.1/de/replication.html
Transaktionslevel	Steuerung des parallenen Zuriffs auf Daten	http://dev.mysql.com/doc/refman/5.1/de/innodb-transaction-isolation.html

Anlage C: Klassendiagramm der Chat-Funktionen



Anlage D: Klassendiagramm der Sessionnutzung, Controlleranfrage und Blockverwendung (Abstrakt)



Literaturverzeichnis

[Prö11] Pröll, S., Zangerle, E., Gassler, W.: "MySQL – Das Handbuch für Administratoren", 1. Auflage, Bonn, Galileo Press, 2011

[Sch09] Schwarzt, B., Zaitsev, P., Tkachenko, V., Zawodny, J., Lentz, A., Balling, D.: "High Performance MySQL – Optimierung, Datensicherung & Lastverteilung", 2. Auflage, Köln, O'Reilly Verlag GmbH & Co. KG, 2009

[Skl09] Sklar, D., Trachtenberg, A., Lcuke, C., Brusdeylins, M., Speidel, U., Schmidt, S.: "PHP5 Kochbuch", 3. Auflage, Köln, O'Reilly Verlag GmbH & Co. KG, 2009

[Wü11] Julius Wüstefeld: "Entwicklung eines Co-Shopping-Features für eine

Standard-Shopsoftware unter Einbeziehung

sozialer Netzwerke.", Fachhochschule Jena

Fachbereich Wirtschaftsingenieurwesen, dotSource GmbH, Oktober 2011

[Ro07] Royans K Tharakan: "Scalable web architectures", 2007, http://www.royans.net/arch/what-is-scalability/

Abruf: 07.01.2012

Ehrenwörtliche Erklärung

Ich erkläre hiermit ehrenwörtlich,

1. dass ich meine Studienarbeit mit dem Thema:

   Co-Shopping in Magento - Dokumentation des Social-Commerce Features unter Beachtung von Multiuser Problematiken

   ohne fremde Hilfe angefertigt habe,
2. dass ich die Übernahme wörtlicher Zitate aus der Literatur sowie die Verwendung der Gedanken anderer Autoren an den entsprechenden Stellen innerhalb der Arbeit gekennzeichnet habe und
3. dass ich meine Studienarbeit bei keiner anderen Prüfung vorgelegt habe.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben wird.

_____ _____

Ort, Datum Unterschrift