So fing alles an

Am Anfang setzten wir uns hin und überlegten welche „Probleme“ es bei unserem Projekt geben könnten und sind auf die „Inserts“ gekommen, da wenn man alle händisch schreiben würde, bei mehreren Tabellen und paar hunderttausend Einträgen sehr lange daran sitzen würden.

Deswegen entschlossen wir uns, ein Programm zu schreiben, das uns die Arbeit abnimmt und für uns die Inserts generiert Wir verwendeten eine einfache Schleife mit einem Counter, und nannten die Inserts immer so, wie auch das „Attribut“ heiß und stellten hintennach noch eine Zahl damit sie unterschiedlich sind und verwendeten dafür gleich den Counter (da der ja nach jedem Schleifen Durchgang sich erhöht und somit immer verändert!)

Wir sind aber schnell drauf gekommen, als wir dann von der direkten Eingabe über SSH hin zu der PHP Programmierung gegangen sind, das man hier ja nicht nur die reinen Inserts hat, sondern auch Query Code für PHP. Somit musste das Programm angepasst werden, wobei uns dann das ständig Copy und Paste auf die nerven ging und uns die Idee kam, dass wenn wir eh ein Programm programmiert haben, das für uns die Inserts macht, und der restliche PHP Code eh „starr“ und immer gleich ist (egal ob man jetzt 1 Insert hat oder Millionen, ob man eine Tabelle erzeugt oder löscht,…) hatten wir vor ein Java Programm zu entwickeln, das alle Anforderungen erfüllt und das wir dann anpassen könnten. Da wir mit dem Filewriter arbeitet konnten wir auch Datenströme direkt in Dateien schreiben und konnten somit direkt .php Dateien aus unserem Java Programm erzeugen lassen. Wir bauten nach und nach die Funktionen ein die wir brauchten (z.b. eine Stoppuhr) und splitterten dann am Ende das Programm in die einzelnen Bereiche damit die einzelnen Programme nur die Funktionen haben die wir brauchen und de Quellcode somit übersichtlich machen. Das erzeugen des PHP Codes aus dem Java Programm hat unter anderem diese Vorteile:

- Direkt aus dem Programm heraus
- Nicht fehleranfällig da man nicht 100% richtig Copy und Paste macht
- Zeitersparnis da man nach ausführen der Java Datei eine lauffähige PHP Version hat
- Man kann über Kommandozeile sagen z.b. wie viele Inserts das Programm machen soll
- Man kann das Programm sehr leicht an die einzelnen Datenbanken anpassen
- Ohne zu wissen wie PHP geht kann man mithilfe des Java Programms funktionierende Dateien machen/ erzeugen
- Wenn sich was ändert, ändert man das an einer Stelle im Programm und muss nicht den gesamten PHP Code verändern
- Da man nur das Java Programm braucht, kann man auch als Laie PHP Dateien erzeugen und Performancetest durchführen => einfach Programm starten, sagen was man will, und das dann verwenden!

MS SQL oder PostgresSQL ?

Da ich nicht wusste ob ich MS SQL oder PostgresSQL verwenden würde, stützte ich mich auf das Programmieren während meine Kollegin sich auf die konkrete Datenbank stürzte. Diese

Aufteilung klappte recht gut, da jeder seinen Bereich hatte und als ich dann von ihr die „Eigenschaften“ der Datenbank bekam mein Grundgerüst vom Programm fertig war und wir es dann implementieren konnten. Als dann alles funktionierte und die Funktionen eingebaut waren, passten wir dann das Programm der 3 Datenbanken an.

Pro/ Contra MS SQL

Das Problem war das ich sehr gerne MS SQL nehmen wollte, da mich die Performance sehr interessieren würde und ich schon immer wissen wollte, wie MS mit der Konkurrenz mithalten kann bei Datenbanken und wie sehr die von Haus aus „optimiert“ sind bzw. sich optimieren lassen. Vor allem da Microsoft ja auf ASP setzt und nicht auf PHP fand ich es interessant, ob es überhaupt eine „PHP Anbindung“ gebe und wenn ja, wie gut die ist, da Microsoft ja wie gesagt nichts mit PHP zu tun haben will (da es ja eine Konkurrenz-Programmiersprache ist). Da im Internet aber leider kaum etwas zu finden war, schaut ich in Bibliotheken, Büchergeschäfte, Tauschbörsen,… aber leider konnte man kaum etwas über die MS SQ/ PHP Schnittstelle finden (da Microsoft sichtlich nicht will das man diese nützt sondern lieber auf ASP setzt). Daher hatte ich dann vor auf Grund der mangelnden Informationen PostgresSQL zu verwenden.

Pro/ Contra PostgresSQL

PostgresSQL hat den Vorteil, dass es darüber im Internet ähnlich wie bei MySQl sehr viele Informationen gibt. Vor allem da PostgresSQL den Anschein hat, sehr aktiv Benutzer andere Datenbanken abzuwerben, findet man im Internet dutzende Tutorials, Bücher, Berichte, Erfahrungsberichte, Tests,… wo einem gezeigt wird wie man von anderen Datenbanken ohne größeren Aufwand und Probleme auf PostgresSQL umsteigen kann. Vor allem da wir doch 3 Datenbanken hatten, und wir uns mit Oracle aus der UE auskennen, und mit MySQL dank vieler Büchern, Lernvideo DVDs, Internet,… war nur noch die 3te Datenbank die wir wählen würden die Frage und da wir schon für die anderen Datenbanken ein „Grundgerüst“ hatten würde somit ein „umstieg“ (in unserem Fall hätten wir dann schon die 3te Datenbank) auf PostgresSQL die einfachere Version gewesen, da es wie gesagt sehr viele Tutorials und infos gibt, wie man von anderen Datenbanken auf PostgresSQL umsteigen kann und man sich somit die arbeit der Informations- Suche für MS SQL, das „Neulernen“ und viel Probleme sich erspart hätten.

Die Entscheidung für MS SQL

Doch dann fand ich nach 3 wöchentlicher suche endlich ein gutes Buch über MS SQL, das sehr gut, aber auch unnötig kompliziert war. Trotzdem entschloss ich mich, weil ich wie gesagt unbedingt mal MS SQL Benchmarken wollte, MS SQL zu verwenden. Da ich dann auch von der Tutorin ein Mail mit vielen Infos bekam, die alle sehr nützlich waren, war die Entscheidung endgültig gefallen das wir bei der 3ten Datenbank auf MS SQL setzen werden.

Probleme über Probleme (bei MS SQL)

MS SQL an sich unterscheidet sich ein wenig von SQL, Oracle, MySQL und hat ein paar Eigenheiten wie das es int als int nicht gibt, das man wenn man über Kommandozeile eine MS SQL Datenbank bedient, man ständig go eingeben muss (was bei vielen Eingaben echt sehr nerven kann!), und auch manchmal ; dazu, manchmal auch nicht, was auch SEHR zu Frustration führte da ich das Gefühl hatte das das im Gegensatz zu Programmiersprachen wie Java oder C++, wo man weiß ob man jetzt ein ; braucht, () oder {} oder keine Klammern, mir einfach alles sehr willkürlich vorkam. Vor allem nahm er dann viele Inputs nicht an, und oft wusste ich nicht ob es am go lag, am ; oder daran das ich nicht Enter drückte, oder drückte obwohl ich es nicht sollte, oder ob meine Eingabe schlicht falsch war. Von daher war es sehr ansträngend, Zeitraubend und auch Psychisch „belastend" da ich mir immer dann dachte „siehst, hättest doch PostgresSQL nehmen sollen". Leider kam noch dazu, dass dann oft mich der Server „disconnected" hat, einfach die Eingaben ignorierte, auf „stur" machte oder mich einfach raus warf und mich neu einloggen lies. Das war auch sehr nervend da das viel Zeit kostete, wenn man eh schon unsicher ist einem noch mehr unsicher macht und vor allem da man nicht weiß wo der Fehler liegt und dann das nächste mal das beachten kann und somit den „Fehler verhindern kann. Was komisch war ist, dass wenn man eine PHP Seite macht dann wiederum dann das alles wie go, ; und ähnliches wieder nicht braucht. Von daher war das alles ein wenig verwirrend da in SSH auf alles penibel genau geachtet wird und dann bei den PHP Abfragen das alles wurscht ist.

Leider stellte sich auch bald heraus das viele Infos wie z.b. von MS selbst zu vergessen sind. Auf der Webseite findet man eigentlich keine Basics und die „Onlinehilfe" die es nicht online sondern nur zum runter laden gibt und auch mit mehr als hundert MB pro Version ordentlich ins Gewicht fällt, enttäuschte sehr, da sie sehr unübersichtlich aufgebaut ist, einem immer nur weiter leitet und auch so für die Basics zum vergessen ist!

Darüber hinaus war am Anfand des Testen (in SSH) auch das Problem das ich nicht die normalen MS SQL Abfragen verwendete sonder TSQL da das in den ganzen Büchern drinnen stand (und sonst nichts) die ich am Anfang gefunden hatte und ich so gut wie nichts machen konnte, da ich keine Rechte dafür hatte. Wenn man eh schon sooo viele Probleme mit der Datenbank hat, und dann kaum Bücher hat, und dann die ganzen Programme die da geschrieben werden nicht gehen, dann ist das auch recht deprimierend. Aber da ich dann wie schon gesagt nach 3 Woche endlich ein brauchbares Buch fand, konnte ich endlich gut loslegen.

Auch sehr komisch war die Abfrage welche MS SQL Version am Server installiert ist. Lauf php info sollte 2005 installiert sein, aber die manuelle Abfrage über SSH meinte das 200 installiert ist. Die Tutorien, die ich um Rat bat meinte auch das sie es nicht weiß, da sie auch diese Merkwürdigkeit feststellte

Probleme über Probleme (bei PHP)

Als komisch erwies sich auch ein kleiner Programmierfehler in meinem Java Programm, der dazu führte, dass in allen PHP Programmen das PHP Tag am Schluss nicht geschlossen wurde. Das komische ist, dass obwohl das in allen PHP Dateien falsch war (da eben die PHP Eingabe nicht „geschlossen" wurde) viele Dateien richtig gingen, andere hingegen überhaupt

nicht. Warum das so ist, das er bei vielen dann große Probleme hatte, bei anderen hingegen überhaupt nicht meckerte und alles einwandfrei laufen lies, konnte ich bis dato nicht erorieren (würde mich aber auch sehr interessieren ob PHP bei bestimmten Operationen verlangt das das PHP Tag geschlossen wird und bei anderen darüber hinweg sieht, das „Zufall“ war oder an was anderem liegt)

Mehrbenutzertest

Als ich mich mit meiner Kollegin traf und wir über Datenbanken redeten kamen wir schnell drauf, das normale Datenbanken in Firmen ganz andere Eigenschaften erfüllen müssen, als wir testen! Klar muss sie schnell Daten Inserten, Tables erzeugen und Updates vornehmen, aber in Firmen ist vor allem der „Mehrbenutzer“ das Problem , da ja doch meistens nicht einer, sondern sehr viele auf eine Datenbank zugreifen, und dann andere „Gesetzte“ und „Benchmarks“ gelten als wenn nur ein Benutzer angemeldet ist. Da man aber nur sehr schwer einschätzen kann, ob eine Datenbank die sich im Einbenutzerbetrieb sehr gut durchschlägt gegen die Konkurrenz auch im Mehrbenutzerbetrieb bestehen kann, entschlossen wir uns, auch einen Mehrbenutzertest einzuführen. Das Problem bestand dann sehr schnell darin, wie man am besten einen fairen und objektiven Test machen kann.

Mehrbenutzertest- Wie objektiv machen?

Das Problem ist, das wirklich 2 Benutzer zur selben Zeit testen müssen, wobei das Problem darin besteht, das wenn einer früher oder später testet, alles nicht mehr stimmt, und somit man wirklich zeitgleich den Test machen muss! Am Anfang überlegten wir uns, dass jeder auf seinen Account zu genau der selben Zeit die PHP Datei startet, und da die Zeitmessung auch Serverseitig ist und nicht unschön über JavaScript oder ähnliches Clientseitiges gelöst ist sollte das auch objektiv sein. Zur Sicherheit fragten wir die Tutorin, die meinte das das so funktionieren sollte, und auch objektiv ist, aber dann kam mir die Überlegung das auch wenn man nebeneinander sitzt und zeitgleich den Test startet, er sehr lange dauern muss, da man sonst wenn z.b. der Test 0,05 Sekunden dauert, sehr wahrscheinlich nicht zeitgleich den Test startet und die Programme hintereinander ausgeführt werden auf dem Server, was wir ja nicht wollten. Folglich beschlossen wir uns, einen Test zu machen, bei der es auf ein paar Zehntel nicht ankommt. Z.b. Wenn man paar hunderttausend Inserts hat, dann würde es auf 0,5 Sekunden nicht ankommen

Benutzer1:

START----------ENDE
START----------ENDE

Bernutzer2:

Man nimmt also die Zeit von START bis ENDE. Wenn man den oberen „Test“ sich ansieht sieht man, das die Test annähernd gleich lange dauern, nur da man nicht weiß, wie viel auf dem Server vor Start von Benutzer 2 und nach Ende von Benutzer 1 los ist, könnte es z.b. sein das vor dem Start des Benutzers2 extrem viel los ist am Server, und deswegen die ganzen Operationen von Benutzer 1 nicht so schnell abgearbeitet werden können bzw. umgekehrt bei

Ende des Benutzers1 extrem viel los sein könnte am Server, und deswegen die ganzen Operationen von Benutzer 2 nicht so schnell abgearbeitet werden können und deswegen es länger dauert

Mehrbenutzertest- Die Lösung (Ansatz)

Trotzdem ist dieser Test nicht 100% sauber und objektiv da z.b. wenn einer 1 Sekunde später den Mehrbenutzertest startet, und der Test z.b. 50 Sekunden dauert, es trotzdem Schwankungen gibt und auch sonst es nicht ganz schön ist. Deswegen überlegten wir weiter wie wir das Problem des „nicht gleichzeitig starten“ des Testes umgehen können. Die Tutorin meinte dann, dass man ja mehrer Test hintereinander machen kann und die alle Benchmarkt, da dann die Ungenauigkeit um vieles geringer ist und eigentlich schon marginal sein sollte.

Benutzer1:

START----START2----START3----START4----START5----START6----ENDE
START----START2----START3----START4----START5----START6----ENDE

Bernutzer2:

Man nimmt also die Zeit von START bis ENDE. Es ist eine Verbesserung zur vorigen Variante da die einzelnen Tests nicht so stark ins Gewicht fallen, aber trotzdem gibt es noch immer das Problem das der Anfang und das Ende nicht 100% objektiv genommen wird, da es da zu Serverschwankungen kommen könnte, und wir diese aber unbedingt komplett weghaben wollten, um einen 100% objektiven Test zu haben und nicht einen ~98% fairen!

Mehrbenutzertest- Die Lösung

Doch diese Idee brachte mich noch auf eine „Steigerung“ des Perfektionismus nämlich, dass man mehrere Test hintereinander macht wie vorher schon erklärt, damit der Server mit einem Client sehr lange beschäftigt ist, und man dann nicht den ersten, den letzten, oder alle Tests „stoppt“, sondern den mittleren, da zu der Zeit auch der andere Benutzer schon das System korrekt austestet und auslasten sollte

Benutzer1:

START----START2 ----Zeitnehmung Beginn----Ende der Zeitnehmung----ENDE
START----START2 ----Zeitnehmung Beginn----Ende der Zeitnehmung----ENDE

Bernutzer2:

Man nimmt also die Zeit von Zeitnehmung Beginn **bis** Ende der Zeitnehmung.

Mehrbenutzertest- Die Programmierung (Grundüberlegung)

Die Frage war, wie man am besten diesen Test programmieren kann. Man könnte die fertigen PHP Datein her nehmen und dann mit Copy und Paste 4 Test machen und nur für den 3ten Test die Zeit nehmen und ausgeben, oder für alle 4 und dann nur den 3ten anzeigen lassen oder alle und einfach nur die 3te Zeit verwenden. Da wir mit einem Java Programm die PHP Dateien erzeugten und somit den Test in ein Java Programm einbauen musste, eröffnete sich das Problem, wie, wann und vor allem für was wir diesen Test machen werden. Wir entschlossen uns, einfach alles so fertig zu programmieren, das alles 100% ging (die Datenbanken an sich) und dann erst den Code zu machen für den Mehrbenutzertest, da jede Änderung im normalen Code dann im Code für den Mehrbenutzertest auch eine Code-Veränderung erzwingt und das im Code des Mehrbenutzertest ein extrem großer Aufwand wäre, dort den Code anzupassen, dass alles wieder passt.

Deswegen überlegten wir uns einmal, wie man am besten programmiertechnisch so etwas umsetzt, damit er mehrer Inserts für Tests macht, und erst beim 3ten Test die Zeit nimmt. Man muss bedenken, wir wollten alles vollautomatisch machen, da wir aus der Java Programm heraus alles generieren lassen wollten, es auch für alle Tests (inserts, update, …) verwenden und für alle Datenbankentypen und Stellungen, und von daher musste es einwandfrei überall gehen, und vor allem auch fair, sicher und von der Lösung her gut programmiert sein!

Mehrbenutzertest- Die Programmierung (Umsetzungsüberlegung)

Als Ideen kam uns, dass man entweder dem Java Programm durch eine Schleife sagt, das er z.b. 5 Test machen soll, und beim 3ten die Zeitnehmung machen soll, also würde man eine „Schleife in der Schleife“ Konstruktion haben bei der man für den normalen Test der gebenchmarkt wird, eine Abzweigung hat, in der dem FileWriter gesagt wird, das er den Code für die Zeitmessung auch einfügen soll.

Der andere Ansatz der viel einfacher wäre, aber viel länger (vom Quelltext), unsauberer und vor allem extrem problematisch bei Änderungen des Quelltextes sein würde ist, das man einfach den normalen Quelltext für die einzelnen Test her nimmt, den kopiert, entsprechend oft der Anzahl der Test einfügt und dann überall die Befehle für den FileWriter das „Zeitnehmen und Stoppen“ Konstrukt zu löschen oder die Ergebnisse dann einfach am Ende nicht zu beachten. Das Problem ist aber, wenn man etwas falsch hat bei einem Test, muss man das dann an allen stellen löschen, und wenn man z.b. 4 Fehler hat und 5 Durchgänge hat für den Test, dann hätte man schon 6*4, also 24 Stellen zu ändern.

Mehrbenutzertest- Die Programmierung (Umsetzung)

Die Umsetzung an sich war nicht so schwer. Wie nahmen einfach die schon programmierten Programme her, bauten eine Schleife ein, und machten das Programm um einiges übersichtlicher. Dann kamen wir darauf, dass es doch besser ist alle Programm die man zum Inserten braucht (Table droppen ,Table createn, Isnerten, Select,… waren alle eigenständige

„Programme“ bei uns!!!) doch in einem Programm zusammen zu fassen und auch grafisch aufzuarbeiten und einige Features einzubauen.

Java Programme

Da aus den Java Programmen der PHP Code gemacht wird, und die ganze Arbeit in diesen Programmen stecken und es somit sehr wichtig ist für das Projekt, folgen nun die Programme, was sie tun, wie sie aufgebaut sind und die Programmcodes

Basisprogramm

Das Programm ist das Ursprungsprogramm aus den wir dann die einzelnen, nun folgenden “Teilprogramme” machten. Da bei vielen Programmen es einen ähnlichen Aufbau gibt, ähnliche Funktionen (File Writer, Inputs lesen, Zeit stoppen…),.. haben wir eben am Anfang ein „großes“ Programm gemacht und es dann eben erst in die einzelnen Teile gegliedert, und den einzelnen Anforderungen angepasst.

Vor allem am Anfang war es sehr praktisch alle Funktionen in einer Datei zu haben, da man so schnell wusste was geht, und was nicht, und dann an die Probleme ran ging die nicht gingen. Man sah mit nur einem ausführen gleich, ob der die Tables löschen, erzeugen, die Inserts erstellen kann, die Inserts anzeigen kann und auch die Zeit korrekt stoppe konnte. Da dann das Programm einwandfrei funktionierte, zerlegten wir es da wir wussten es geht und jede Funktion erfüllt seinen Zweck! Wie entwickelten dann die Teile einzeln weiter und führten sie am Ende dann wieder zusammen!

```
import java.io.*;

public class basisprogramm
{
	public static void main (String[]args) throws IOException
	{
		//Deklaration von den Variablen
		int count = 0;  //Beginnt mit 0 damit die Gesamtzahlt stimmt
(Endelement:Anzahl -1=>Beginng = o)!
		double eingabe = 0;

		//Ausgabe um zu erfragen wieviele Inserts notwendig sind
		System.out.print ("Wieviele Inserts sollen es sein?:");

		//Inputs werden eingelesen und unter dem String eingabe gespeichert
		BufferedReader tastatur = new BufferedReader (new
InputStreamReader(System.in));
		String eingabetemp = tastatur.readLine();
		eingabe = Double.parseDouble(eingabetemp);

		//Hier wird festegelet unter welchem Namen und Endung die Daten die erzeugt
werden gespeichert werden
```

```
                String ausgabename = "ms_sql_insert" +eingabetemp;  //Fixer Name +
Insertanzahl
                ausgabename = ausgabename + ".php"; //Soll als PHP Datei abgespeichert
werden

                //Alles vorbereiten damit man in Datei schreiben kann
                FileWriter dateiStream = new FileWriter(ausgabename);
                PrintWriter ausgabe = new PrintWriter(dateiStream);


        //#########################################################################
#########################
                //## Hier ist der Abschnitt in dem die Daten für die SQL "Richtigkeit"
zusammen gesetzt werden #

        //#########################################################################
#########################

                //Ausgabe der "Header" Daten
                ausgabe.println ("<?php");
                ausgabe.println ("$verbindung = mssql_connect('MyServer2k', 'a0402913',
'dbs2005');");
                ausgabe.println ("$msdb_selected = mssql_select_db(\"labor\",$verbindung);");
                ausgabe.println ("$zeitmessung1=microtime(); ");
                ausgabe.println ("$zeittemp=explode(\" \",$zeitmessung1); ");
                ausgabe.println ("$zeitmessung1=$zeittemp[0]+$zeittemp[1];");

                ausgabe.println ("mssql_query(\"DROP TABLE arbeitetan\"); ");
                ausgabe.println ("mssql_query(\"DROP TABLE besitzt\"); ");
                ausgabe.println ("mssql_query(\"DROP TABLE buch\"); ");
                ausgabe.println ("mssql_query(\"DROP TABLE computer\"); ");
                ausgabe.println ("mssql_query(\"DROP TABLE hat\"); ");
                ausgabe.println ("mssql_query(\"DROP TABLE person\"); ");
                ausgabe.println ("mssql_query(\"DROP TABLE projekt\"); ");

                ausgabe.println ("mssql_query(\"CREATE TABLE arbeitetan(SerienNr
char(11) NOT NULL,ID char(11) NOT NULL)\"); ");
                ausgabe.println ("mssql_query(\"CREATE TABLE besitzt(SVNR char(11)
NOT NULL,SerienNr char(11) NOT NULL)\"); ");
                ausgabe.println ("mssql_query(\"CREATE TABLE buch(ISBNNR char(11)
NOT NULL,Titel varchar(100) NOT NULL,PRIMARY KEY  (ISBNNR))\"); ");
                ausgabe.println ("mssql_query(\"CREATE TABLE computer(SerienNr
char(11) NOT NULL,Hersteller varchar(100) NOT NULL,PRIMARY KEY  (SerienNr))\");
");
                ausgabe.println ("mssql_query(\"CREATE TABLE hat(SVNR char(11) NOT
NULL,ISBNR char(11) NOT NULL)\"); ");
                ausgabe.println ("mssql_query(\"CREATE TABLE person(SVNR char(11)
NOT NULL,Name varchar(100) NOT NULL,PRIMARY KEY  (SVNR))\"); ");
                ausgabe.println ("mssql_query(\"CREATE TABLE projekt(ID char(11) NOT
NULL,Projektname varchar(100) NOT NULL,PRIMARY KEY  (ID))\"); ");
```

```
                //Jetzt kommen Schleifen damit nicht immer die selben Datensaetze entstehen

                //arbeitetan Tables werden erzeugt
                while (count < eingabe)
                {
                        ausgabe.print ("mssql_query(\"INSERT INTO arbeitetan (SerienNr, ID)
VALUES ('SerienNr" +count);
                        ausgabe.print ("', 'ID" +count);
                        ausgabe.println ("')\"); ");
                        count++;
                }



                //SQL Anweisungen das die Tables ausgelesen werden
                ausgabe.println("");
                ausgabe.println ("$result = mssql_query(\"SELECT * FROM arbeitetan\"); ");
                ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
                ausgabe.println("{echo \"$msrow[0]  <br>\"; echo \"$msrow[1]  <br>\"; }");
                ausgabe.println("?>");

                //Stoppen der Zeit
                ausgabe.println ("$zeitmessung2=microtime(); ");
                ausgabe.println ("$zeittemp=explode(\" \",$zeitmessung2); ");
                ausgabe.println ("$zeitmessung2=$zeittemp[0]+$zeittemp[1]; ");
                ausgabe.println ("$zeitmessung=$zeitmessung2-$zeitmessung1;  ");
                ausgabe.println ("$zeitmessung=substr($zeitmessung,0,8); ");
                ausgabe.println ("print(\"Die Funktion dauert: $zeitmessung Sekunden.\"); ");
                ausgabe.println("?>");

                //Hier wird die Ausgabe geschlossen damit es sauber ist!
                ausgabe.close();
        }
}
```

create_inserts_mod

Das Programm erzeugt die Inserts. Die Datei wird als fertige PHP Datei abgespeichert wobei das Programm so „intelligent“ ist und der Dateiname automatisch angepasst wird. Wenn man z.b. in der Eingabeaufforderung des Programms 1000 eingibt, das somit das Programm pro Table 1000 Inserts erzeugt, so nimmt er den Input und speichert ihn in den Namen der Datei mit ein, also würde dann die Datei bei 1000 Inserts create_inserts1000.php heißen, während es bei 50 Inserts folglich create_inserts50.php heißen würde. Das hat 4 Vorteile:

- Am Dateinamen erkennt man, was es macht, und wie viel Inserts es gibt
- Die Namensgebung ist genauso wie das erzeugen automatisch und somit kostet es keine Zeit wenn man hunderte Dateien erzeugen würde, diese passend zu benennen.

- Wenn meine Kollegin das selbe Programm verwendet und es für sich und ihre Datenbanken anpasst, habe wir bei den Java und den PHP Programmen automatisch die selbe Namensvergabe und somit keine Probleme da man dann nicht weiß, was das Programm des anderen macht, oder es sonstige Probleme gibt die man ansonsten recht schnell haben kann!
- Wenn man die Datei Uploaded wird die alte automatisch überschrieben, die genau die selben Eigenschaften hat, da die Namensvergabe automatisch ist, und somit man nicht groß überlegen muss, was noch mal was macht, wie welche Datei heißt, was neu und was alt ist….

```
import java.io.*;

public class create_inserts_mod_tatjana
{ //ISBNR oder ISBNNR oder ISBNNE???????
      public static void main (String[]args) throws IOException
      {
            //Deklaration von den Variablen
            int count = 0;    //Beginnt mit 0 damit die Gesamtzahlt stimmt
(Endelement:Anzahl -1=>Beginng = o)!
            int count2 = 0;
            double eingabe = 0;

            //Ausgabe um zu erfragen wieviele Inserts notwendig sind
            System.out.print ("Wieviele Inserts sollen es sein?:");

            //Inputs werden eingelesen und unter dem String eingabe
gespeichert
            BufferedReader tastatur = new BufferedReader (new
InputStreamReader(System.in));
            String eingabetemp = tastatur.readLine();
            eingabe = Double.parseDouble(eingabetemp);

            //Hier wird festegelet unter welchem Namen und Endung die Daten
die erzeugt werden gespeichert werden
            String ausgabename = "create_inserts_mod_tatjana" +eingabetemp;
//Fixer Name + Insertanzahl
            ausgabename = ausgabename + ".php"; //Soll als PHP Datei
abgespeichert werden

            //Alles vorbereiten damit man in Datei schreiben kann
            FileWriter dateiStream = new FileWriter(ausgabename);
            PrintWriter ausgabe = new PrintWriter(dateiStream);


      //#####################################################################
############################
            //## Hier ist der Abschnitt in dem die Daten für die SQL
"Richtigkeit" zusammen gesetzt werden #

      //#####################################################################
############################

            //Ausgabe der "Header" Daten
            ausgabe.println ("<?php");
            ausgabe.println ("$verbindung = mssql_connect('MyServer2k',
'a0402913', 'dbs2005');");
            ausgabe.println ("$msdb_selected =
mssql_select_db(\"labor\",$verbindung);");
            ausgabe.println ("$zeitmessung1=microtime(); ");
            ausgabe.println ("$zeittemp=explode(\" \",$zeitmessung1); ");
```

```
            ausgabe.println ("$zeitmessung1=$zeittemp[0]+$zeittemp[1];");

            //Jetzt kommen Schleifen damit nicht immer die selben
Datensaetze entstehen

            //person Inserts werden erzeugt
            count = 0;
            while (count < eingabe*5)
            {
                  ausgabe.print ("mssql_query(\"INSERT INTO person (SVNR,
Name) VALUES ('SVNR" +count);
                  ausgabe.print ("', 'Name" +count);
                  ausgabe.println ("')\"); ");
                  count++;
            }


            //arbeitetan Inserts werden erzeugt
            count = 0;
            while (count < eingabe)
            {
                  ausgabe.print ("mssql_query(\"INSERT INTO arbeitetan
(SerienNr, ID) VALUES ('SerienNr" +count);
                  ausgabe.print ("', 'ID" +count);
                  ausgabe.println ("')\"); ");
                  count++;
            }


            //buch Inserts werden erzeugt
            //Counter damit man gerade Zahlen (ISBNNR) hat
            count = 0;
            count2 = 0;
            while (count < eingabe*2)
            {
                  ausgabe.print ("mssql_query(\"INSERT INTO buch (SVNR,
ISBNNR, Titel) VALUES ('SVNR" +count);
                  ausgabe.print ("', 'ISBNNR" +count2);
                  ausgabe.print ("', 'Titel" +count2);
                  ausgabe.println ("')\"); ");
                  count++;
                  count2 = count2 + 2;
            }


            //Counter damit man ungerade Zahlen (ISBNNR) hat
            count = 0;
            count2 = 1;
            while (count < eingabe*2+1)
            {
                  ausgabe.print ("mssql_query(\"INSERT INTO buch (SVNR,
ISBNNR, Titel) VALUES ('SVNR" +count);
                  ausgabe.print ("', 'ISBNNR" +count2);
                  ausgabe.print ("', 'Titel" +count2);
                  ausgabe.println ("')\"); ");
                  count++;
                  count2 = count2 + 2;
            }


            //computer Inserts werden erzeugt
            // Counter damit man gerade Zahlen (Seriennummern) hat
```

```
            count = 0;
            count2 = 0;
            while (count < eingabe*2)
            {
                  ausgabe.print ("mssql_query(\"INSERT INTO computer (SVNR,
SerienNr, Hersteller) VALUES ('SVNR" +count);
                  ausgabe.print ("', 'SerienNr" +count2);
                  ausgabe.print ("', 'Hersteller" +count2);
                  ausgabe.println ("')\"); ");
                  count++;
                  count2 = count2 + 2;
            }


            //Counter damit man ungerade Zahlen (Seriennummern) hat
            count = 0;
            count2 = 1;
            while (count < eingabe*2+1)
            {
                  ausgabe.print ("mssql_query(\"INSERT INTO computer (SVNR,
SerienNr, Hersteller) VALUES ('SVNR" +count);
                  ausgabe.print ("', 'SerienNr" +count2);
                  ausgabe.print ("', 'Hersteller" +count2);
                  ausgabe.println ("')\"); ");
                  count++;
                  count2 = count2 + 2;
            }


            //projekt Inserts werden erzeugt
            count = 0;
            while (count < eingabe)
            {
                  ausgabe.print ("mssql_query(\"INSERT INTO projekt (ID,
Projektname) VALUES ('ID" +count);
                  ausgabe.print ("', 'Projektname" +count);
                  ausgabe.println ("')\"); ");
                  count++;
            }


            //Stoppen der Zeit
            ausgabe.println ("$zeitmessung2=microtime(); ");
            ausgabe.println ("$zeittemp=explode(\" \",$zeitmessung2); ");
            ausgabe.println ("$zeitmessung2=$zeittemp[0]+$zeittemp[1]; ");
            ausgabe.println ("$zeitmessung=$zeitmessung2-$zeitmessung1;
");
            ausgabe.println ("$zeitmessung=substr($zeitmessung,0,8); ");
            ausgabe.println ("print(\"Die Funktion dauert: $zeitmessung
Sekunden.\"); ");
            ausgabe.println("?>");

            //Hier wird die Ausgabe geschlossen damit es sauber ist!
            ausgabe.close();
      }
}
```

create_table

Die Datei erzeugt die Tables und hat sich als sehr Praktisch erwiesen, da man sehr schnell Änderungen vornehmen kann und man die Datei durch minimale Anpassungen auch für die anderen Datenbanken verwenden kann.

```
import java.io.*;

public class create_table_mod_tatjana
{
	public static void main (String[]args) throws IOException
	{
		//Alles vorbereiten damit man in Datei schreiben kann
		FileWriter dateiStream = new FileWriter("create_table.php");
		PrintWriter ausgabe = new PrintWriter(dateiStream);


	//#########################################################################
#########################
		//## Hier ist der Abschnitt in dem die Daten für die SQL "Richtigkeit"
zusammen gesetzt werden #

	//#########################################################################
#########################

		//Ausgabe der "Header" Daten
		ausgabe.println ("<?php");
		ausgabe.println ("$verbindung = mssql_connect('MyServer2k', 'a0402913',
'dbs2005');");
		ausgabe.println ("$msdb_selected = mssql_select_db(\"labor\",$verbindung);");
		ausgabe.println ("");

		ausgabe.println ("mssql_query(\"CREATE TABLE person(SVNR char(11)
NOT NULL,Name varchar(100) NOT NULL,PRIMARY KEY (SVNR))\"); ");
		ausgabe.println ("mssql_query(\"CREATE TABLE arbeitetan(SerienNr
char(11) NOT NULL,ID char(11) NOT NULL, PRIMARY KEY (SerienNr,ID))\"); ");
		ausgabe.println ("mssql_query(\"CREATE TABLE buch(SVNR char(11) NOT
NULL, ISBNNR char(11) NOT NULL,Titel varchar(100) NOT NULL,PRIMARY KEY
(ISBNNR), Foreign Key (SVNR) references person (SVNR))\"); ");
		ausgabe.println ("mssql_query(\"CREATE TABLE computer(SVNR char(11)
NOT NULL, SerienNr char(11) NOT NULL,Hersteller varchar(100) NOT NULL,PRIMARY
KEY(SerienNr), Foreign Key (SVNR) references person (SVNR))\"); ");
		ausgabe.println ("mssql_query(\"CREATE TABLE projekt(ID char(11) NOT
NULL,Projektname varchar(100) NOT NULL,PRIMARY KEY (ID))\"); ");
		ausgabe.println("?>");

		//Hier wird die Ausgabe geschlossen damit es sauber ist!
		ausgabe.close();
	}
}
```

drop_all

Die Datei dropt alle Tables die es gibt.

```
import java.io.*;

public class drop_all
{
	public static void main (String[]args) throws IOException
	{
		//Alles vorbereiten damit man in Datei schreiben kann
		FileWriter dateiStream = new FileWriter("drop_all.php");
		PrintWriter ausgabe = new PrintWriter(dateiStream);


	//##########################################################################
#########################
		//## Hier ist der Abschnitt in dem die Daten für die SQL "Richtigkeit"
zusammen gesetzt werden #

	//##########################################################################
#########################

		//Ausgabe der "Header" Daten
		ausgabe.println ("<?php");
		ausgabe.println ("$verbindung = mssql_connect('MyServer2k', 'a0402913',
'dbs2005');");
		ausgabe.println ("$msdb_selected = mssql_select_db(\"labor\",$verbindung);");

		ausgabe.println ("mssql_query(\"DROP TABLE arbeitetan\"); ");
		ausgabe.println ("mssql_query(\"DROP TABLE buch\"); ");
		ausgabe.println ("mssql_query(\"DROP TABLE computer\"); ");
		ausgabe.println ("mssql_query(\"DROP TABLE person\"); ");
		ausgabe.println ("mssql_query(\"DROP TABLE projekt\"); ");
		ausgabe.println("?>");

		//Hier wird die Ausgabe geschlossen damit es sauber ist!
		ausgabe.close();
	}
}
```

select_all

Mit der Datei sieht man welche Inserts es in den Tables gibt.

```
package Funktioniert;
import java.io.*;
```

```
public class select_all
{
	public static void main (String[]args) throws IOException
	{
		//Alles vorbereiten damit man in Datei schreiben kann
		FileWriter dateiStream = new FileWriter("select_all.php");
		PrintWriter ausgabe = new PrintWriter(dateiStream);


	//#########################################################################
##########################
		//## Hier ist der Abschnitt in dem die Daten für die SQL "Richtigkeit"
zusammen gesetzt werden #

	//#########################################################################
##########################

		//Ausgabe der "Header" Daten
		ausgabe.println ("<?php");
		ausgabe.println ("$verbindung = mssql_connect('MyServer2k', 'a0402913',
'dbs2005');");
		ausgabe.println ("$msdb_selected = mssql_select_db(\"labor\",$verbindung);");

		//SQL Anweisungen das die Tables ausgelesen werden
		ausgabe.println ("$result = mssql_query(\"SELECT * FROM arbeitetan\"); ");
		ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
		ausgabe.println("{echo \"$msrow[0]  \"; echo \"$msrow[1]  <br>\"; }");

		/* ausgabe.println ("$result = mssql_query(\"SELECT * FROM besitzt\"); ");
		ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
		ausgabe.println("{echo \"$msrow[0]  \"; echo \"$msrow[1]  <br>\"; }"); */

		ausgabe.println ("$result = mssql_query(\"SELECT * FROM buch\"); ");
		ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
		ausgabe.println("{echo \"$msrow[0] \"; echo \"$msrow[1]  <br>\"; }");

		ausgabe.println ("$result = mssql_query(\"SELECT * FROM computer\"); ");
		ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
		ausgabe.println("{echo \"$msrow[0] \"; echo \"$msrow[1]  <br>\"; }");

	/*	ausgabe.println ("$result = mssql_query(\"SELECT * FROM hat\"); ");
		ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
		ausgabe.println("{echo \"$msrow[0]\"; echo \"$msrow[1]  <br>\"; }"); */

		ausgabe.println ("$result = mssql_query(\"SELECT * FROM person\"); ");
		ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
		ausgabe.println("{echo \"$msrow[0]\"; echo \"$msrow[1]  <br>\"; }");

		ausgabe.println ("$result = mssql_query(\"SELECT * FROM projekt\"); ");
		ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
		ausgabe.println("{echo \"$msrow[0]\"; echo \"$msrow[1]  <br>\"; }");
```

```
                ausgabe.println("?>");

                //Hier wird die Ausgabe geschlossen damit es sauber ist!
                ausgabe.close();
        }
}
```

Mehrbenutzertests

Wie schon vorher gesagt nahmen wir dann aber alle Programme zusammen und machten sie "Mehrbenutzertestfähig". Die „Mehrbenutzertest"- Programm sind all die Programme, die wir auch verwendeten um alle .php Dateien die wir für das Projekt brauchten zu erzeugen!

create_inserts_mehrbenutzer.java

Mit diesem Programm kann man beliebig viele Inserts erzeugen und auch dem Programm sagen, wie viele Schleifen/ Wiederholungen es geben soll. Das ist deswegen von großem Vorteil, damit man einen Mehrbenutzertest beinahe 100% fair ablaufen lassen kann und da man beim Benchmarken nicht 10 Tests machen muss damit Serverschwankungen ausgeschlossen werden, sondern nur einen, und man alle nötigen Infos auch noch erhält (Durchschnittszeit, Gesamtzeit, Durchläufe, Inserts anzeigen,…)

```
import java.io.*;

public class create_inserts_mehrbenutzer
{
        public static void main (String[]args) throws IOException
        {
                //Deklaration von den Variablen
                int count = 0;  //Beginnt mit 0 damit die Gesamtzahlt stimmt
(Endelement:Anzahl -1=>Beginn = o)!
                int count2 = 0; //Zusatzcounter für speziellere Schleifen
                int count3 = 0; //wird verwendet um zu schauen ob das Ende der
"Wiederholungen" erreicht ist
                double eingabe = 0;
                double eingabe2 = 0;

                //Ausgabe um zu erfragen wieviele Inserts notwendig sind
                System.out.println ("Wieviele Inserts sollen es sein?:");

                //Inputs werden eingelesen und unter dem String eingabe gespeichert
                BufferedReader tastatur = new BufferedReader (new
InputStreamReader(System.in));
                String eingabetemp = tastatur.readLine();
                eingabe = Double.parseDouble(eingabetemp);
```

```
                //Ausgabe um zu erfragen wieviele Wiederholungen notwendig sind
                System.out.println ("Wieviele Wiederholungen sollen es sein?:");

                //Inputs werden eingelesen und unter dem String eingabe gespeichert
                BufferedReader tastatur2 = new BufferedReader (new
InputStreamReader(System.in));
                String eingabetemp2 = tastatur2.readLine();
                eingabe2 = Double.parseDouble(eingabetemp2);


                //Hier wird festegelet unter welchem Namen und Endung die Daten die erzeugt
werden gespeichert werden
                String ausgabename = "create_ " +eingabetemp;  //Fixer Name + Insertanzahl
                ausgabename = ausgabename + "inserts_"; //
                ausgabename = ausgabename + eingabetemp2; //
                ausgabename = ausgabename + "schleifen.php"; //Soll als PHP Datei
abgespeichert werden

                //Alles vorbereiten damit man in Datei schreiben kann
                FileWriter dateiStream = new FileWriter(ausgabename);
                PrintWriter ausgabe = new PrintWriter(dateiStream);


        //#########################################################################
##########################
                //## Hier ist der Abschnitt in dem die Daten für die SQL "Richtigkeit"
zusammen gesetzt werden #

        //#########################################################################
##########################

                //Ausgabe der "Header" Daten
                ausgabe.println ("<?php");
                ausgabe.println ("$verbindung = mssql_connect('MyServer2k', 'a0402913',
'dbs2005');");
                ausgabe.println ("$msdb_selected = mssql_select_db(\"labor\",$verbindung);");

                //Initialisierung der Variablen die in PHP gebraucht werden und "überregional
sind" und somit außerhalb der Schleife sein müssen!
                ausgabe.println ("$gesammtzeit = 0; "); //Initialisierung von der Variable
Gesamtzeit
                ausgabe.println ("$durchlaufzaehler =0; "); //Initialisierung von der Variable
die die Durchlaufe zeaehlt
                ausgabe.println ("$durchschnitt_ergebnis = 0; ");

                while (count3 < eingabe2)
                {

                        //Alle Tables werden gelöscht
                        ausgabe.println ("mssql_query(\"DROP TABLE arbeitetan\"); ");
                        ausgabe.println ("mssql_query(\"DROP TABLE buch\"); ");
```

```
                        ausgabe.println ("mssql_query(\"DROP TABLE computer\"); ");
                        ausgabe.println ("mssql_query(\"DROP TABLE person\"); ");
                        ausgabe.println ("mssql_query(\"DROP TABLE projekt\"); ");

                        //Alle Tables werden erzeugt
                        ausgabe.println ("mssql_query(\"CREATE TABLE person(SVNR
char(100) NOT NULL,Name varchar(100) NOT NULL,PRIMARY KEY (SVNR))\"); ");
                        ausgabe.println ("mssql_query(\"CREATE TABLE arbeitetan(SerienNr
char(100) NOT NULL,ID char(100) NOT NULL, PRIMARY KEY (SerienNr,ID))\"); ");
                        ausgabe.println ("mssql_query(\"CREATE TABLE buch(SVNR
char(100) NOT NULL, ISBNNR char(100) NOT NULL,Titel varchar(100) NOT
NULL,PRIMARY KEY (ISBNNR), Foreign Key (SVNR) references person (SVNR))\"); ");
                        ausgabe.println ("mssql_query(\"CREATE TABLE computer(SVNR
char(100) NOT NULL, SerienNr char(100) NOT NULL,Hersteller varchar(100) NOT
NULL,PRIMARY KEY(SerienNr), Foreign Key (SVNR) references person (SVNR))\"); ");
                        ausgabe.println ("mssql_query(\"CREATE TABLE projekt(ID
char(100) NOT NULL,Projektname varchar(100) NOT NULL,PRIMARY KEY (ID))\"); ");



                        //Stoppen der Einzelzeit beginnt
                        ausgabe.println ("$zeitmessung1=microtime(); ");
                        ausgabe.println ("$zeittemp=explode(\" \",$zeitmessung1); ");
                        ausgabe.println ("$zeitmessung1=$zeittemp[0]+$zeittemp[1];");



                        //Jetzt kommen Schleifen damit nicht immer die selben Datensaetze
entstehen

                        //person Inserts werden erzeugt
                        count = 0;
                        while (count < eingabe*2+1)
                        {
                                ausgabe.print ("mssql_query(\"INSERT INTO person (SVNR,
Name) VALUES ('SVNR" +count);
                                ausgabe.print ("', 'Name" +count);
                                ausgabe.println ("')\"); ");
                                count++;
                        }



                        //arbeitetan Inserts werden erzeugt
                        count = 0;
                        while (count < eingabe)
                        {
                                ausgabe.print ("mssql_query(\"INSERT INTO arbeitetan
(SerienNr, ID) VALUES ('SerienNr" +count);
                                ausgabe.print ("', 'ID" +count);
                                ausgabe.println ("')\"); ");
                                count++;
                        }
```

```
                        //buch Inserts werden erzeugt
                        //Counter damit man gerade Zahlen (ISBNNR) hat
                        count = 0;
                        count2 = 0;
                        while (count < eingabe*2)
                        {
                                ausgabe.print ("mssql_query(\"INSERT INTO buch (SVNR,
ISBNNR, Titel) VALUES ('SVNR" +count);
                                ausgabe.print ("', 'ISBNNR" +count2);
                                ausgabe.print ("', 'Titel" +count2);
                                ausgabe.println ("')\"); ");
                                count++;
                                count2 = count2 + 2;
                        }


                        //Counter damit man ungerade Zahlen (ISBNNR) hat
                        count = 0;
                        count2 = 1;
                        while (count < eingabe*2+1)
                        {
                                ausgabe.print ("mssql_query(\"INSERT INTO buch (SVNR,
ISBNNR, Titel) VALUES ('SVNR" +count);
                                ausgabe.print ("', 'ISBNNR" +count2);
                                ausgabe.print ("', 'Titel" +count2);
                                ausgabe.println ("')\"); ");
                                count++;
                                count2 = count2 + 2;
                        }


                        //computer Inserts werden erzeugt
                        // Counter damit man gerade Zahlen (Seriennummern) hat
                        count = 0;
                        count2 = 0;
                        while (count < eingabe*2)
                        {
                                ausgabe.print ("mssql_query(\"INSERT INTO computer
(SVNR, SerienNr, Hersteller) VALUES ('SVNR" +count);
                                ausgabe.print ("', 'SerienNr" +count2);
                                ausgabe.print ("', 'Hersteller" +count2);
                                ausgabe.println ("')\"); ");
                                count++;
                                count2 = count2 + 2;
                        }


                        //Counter damit man ungerade Zahlen (Seriennummern) hat
                        count = 0;
                        count2 = 1;
```

```
                        while (count < eingabe*2+1)
                        {
                                ausgabe.print ("mssql_query(\"INSERT INTO computer
(SVNR, SerienNr, Hersteller) VALUES ('SVNR" +count);
                                ausgabe.print ("', 'SerienNr" +count2);
                                ausgabe.print ("', 'Hersteller" +count2);
                                ausgabe.println ("')\"); ");
                                count++;
                                count2 = count2 + 2;
                        }


                        //projekt Inserts werden erzeugt
                        count = 0;
                        while (count < eingabe)
                        {
                                ausgabe.print ("mssql_query(\"INSERT INTO projekt (ID,
Projektname) VALUES ('ID" +count);
                                ausgabe.print ("', 'Projektname" +count);
                                ausgabe.println ("')\"); ");
                                count++;
                        }

                        //Stoppen der Zeit
                        ausgabe.println ("$zeitmessung2 = microtime(); ");
                        ausgabe.println ("$zeittemp = explode(\" \",$zeitmessung2); ");
                        ausgabe.println ("$zeitmessung2 = $zeittemp[0]+$zeittemp[1]; ");
                        ausgabe.println ("$zeitmessung = $zeitmessung2-$zeitmessung1;  ");
                        ausgabe.println ("$zeitmessung = substr($zeitmessung,0,8); ");
                        ausgabe.println ("$gesammtzeit = $gesammtzeit + $zeitmessung; ");
                        ausgabe.println ("print(\" Die Funktion dauert: <strong> $zeitmessung
</strong> Sekunden. <br> \"); ");


                        // Erhöhung des Counter der Java Schleife und der für PHP
                        ausgabe.println ("$durchlaufzaehler = $durchlaufzaehler + 1; "); //PHP
Counter
                        count3++; //Java Counter
                }

                ausgabe.println ("  print  (\"----------------------------------------------------------
---" +"<br> \"); ");
                ausgabe.println ("$durchschnitt_ergebnis = $gesammtzeit /$durchlaufzaehler;
");
                ausgabe.println ("  print  (\"Die Funktionen dauerten im Durchschnitt :<strong>
$durchschnitt_ergebnis </strong> Sekunden.<br><br> \"); ");
                ausgabe.println("print  (\"Zusatzinfo: es gab bei dem Test <strong>
$durchlaufzaehler </strong>Durchläufe und er dauerte insgesamt <strong> $gesammtzeit
</strong> Sekunden. <br><br> \"); ");
                ausgabe.println("print  (\"Und so sehen die Inserts in der Tabelle aus:
<br><br>\"); ");
```

```
            //SQL Anweisungen das die Tables ausgelesen werden
            ausgabe.println(" print  (\" <strong>Inserts in arbeitetan: </strong><br> \") ; ");
            ausgabe.println ("$result = mssql_query(\"SELECT * FROM arbeitetan\"); ");
            ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
            ausgabe.println("{echo \"$msrow[0]  \"; echo \"$msrow[1] </br>\"; }");

            ausgabe.println("print  (\" <strong>Inserts in buch: </strong><br> \") ; ");
            ausgabe.println ("$result = mssql_query(\"SELECT * FROM buch\"); ");
            ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
            ausgabe.println("{echo \"$msrow[0] \"; echo \"$msrow[1]  <br>\"; }");

            ausgabe.println(" print  (\" <strong>Inserts in computer: </strong><br> \") ; ");
            ausgabe.println ("$result = mssql_query(\"SELECT * FROM computer\"); ");
            ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
            ausgabe.println("{echo \"$msrow[0] \"; echo \"$msrow[1]  <br>\"; }");

            ausgabe.println(" print  (\" <strong>Inserts in person: </strong><br> \") ; ");
            ausgabe.println ("$result = mssql_query(\"SELECT * FROM person\"); ");
            ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
            ausgabe.println("{echo \"$msrow[0]\"; echo \"$msrow[1]  <br>\"; }");

            ausgabe.println(" print  (\" <strong>Inserts in projekt: </strong><br> \") ; ");
            ausgabe.println ("$result = mssql_query(\"SELECT * FROM projekt\"); ");
            ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
            ausgabe.println("{echo \"$msrow[0]\"; echo \"$msrow[1]  <br>\"; }");

            ausgabe.println("?>");          //Schließen des PHP Tags
            ausgabe.close(); //Hier wird die Ausgabe geschlossen damit es sauber ist!
        }
}
```

update_mehrbenutzer.java

Mit diesem Programm werden die Update- PHP Dateien erzeugt, wobei immer die Inserts mit den gleichen Updates überschrieben werden! Der Titel ’Titel0’ wird z.B. auf ’Titel0’ upgedatet, und der Hersteller ’Hersteller8’ auf ’Hersteller8’. Wir haben dies deswegen so gemacht, da wir mit extrem großen Inserts/ Updates arbeiten, die PHP Dateien so gut wie alle um die 100-200MB hatten und wir sie nicht zusätzlich noch durch lange Insertnamen noch größer machen wollten bzw. es, wenn wir kleine Namen verwendet hätten, es Probleme mit ähnlichen Schlüsseln in der Datenbank gegeben hätte!

```
import java.io.*;

public class update_mehrbenutzer
{
        public static void main (String[]args) throws IOException
        {
                //Deklaration von den Variablen
```

```
		int count = 0;  //Beginnt mit 0 damit die Gesamtzahlt stimmt (Endelement:Anzahl -1=>Beginn = o)!
		int count2 = 0; //Zusatzcounter für speziellere Schleifen
		int count3 = 0; //wird verwendet um zu schauen ob das Ende der "Wiederholungen" erreicht ist
		double eingabe = 0;
		double eingabe2 = 0;

		//Ausgabe um zu erfragen wieviele Inserts notwendig sind
		System.out.println ("Wieviele Updates sollen es sein?:");

		//Inputs werden eingelesen und unter dem String eingabe gespeichert
		BufferedReader tastatur = new BufferedReader (new InputStreamReader(System.in));
		String eingabetemp = tastatur.readLine();
		eingabe = Double.parseDouble(eingabetemp);


		//Ausgabe um zu erfragen wieviele Wiederholungen notwendig sind
		System.out.println ("Wieviele Wiederholungen sollen es sein?:");

		//Inputs werden eingelesen und unter dem String eingabe gespeichert
		BufferedReader tastatur2 = new BufferedReader (new InputStreamReader(System.in));
		String eingabetemp2 = tastatur2.readLine();
		eingabe2 = Double.parseDouble(eingabetemp2);


		//Hier wird festegelet unter welchem Namen und Endung die Daten die erzeugt werden gespeichert werden
		String ausgabename = eingabetemp;  //Fixer Name + Insertanzahl
		ausgabename = ausgabename + "updates_"; //
		ausgabename = ausgabename + eingabetemp2; //
		ausgabename = ausgabename + "schleifen.php"; //Soll als PHP Datei abgespeichert werden

		//Alles vorbereiten damit man in Datei schreiben kann
		FileWriter dateiStream = new FileWriter(ausgabename);
		PrintWriter ausgabe = new PrintWriter(dateiStream);


	//#########################################################################################################
		//## Hier ist der Abschnitt in dem die Daten für die SQL "Richtigkeit" zusammen gesetzt werden #

	//#########################################################################################################

		//Ausgabe der "Header" Daten
		ausgabe.println ("<?php");
```

```
		ausgabe.println ("$verbindung = mssql_connect('MyServer2k', 'a0402913', 'dbs2005');");
		ausgabe.println ("$msdb_selected = mssql_select_db(\"labor\",$verbindung);");

		//Initialisierung der Variablen die in PHP gebraucht werden und "überregional sind" und somit außerhalb der Schleife sein müssen!
		ausgabe.println ("$gesammtzeit = 0; "); //Initialisierung von der Variable Gesamtzeit
		ausgabe.println ("$durchlaufzaehler =0; "); //Initialisierung von der Variable die die Durchlaufe zeaehlt
		ausgabe.println ("$durchschnitt_ergebnis = 0; ");

		while (count3 < eingabe2)
		{

			//Stoppen der Einzelzeit beginnt
			ausgabe.println ("$zeitmessung1=microtime(); ");
			ausgabe.println ("$zeittemp=explode(\" \",$zeitmessung1); ");
			ausgabe.println ("$zeitmessung1=$zeittemp[0]+$zeittemp[1];");

			//ausgabe.println ("mssql_query(\"update person\"); ");
			//ausgabe.println ("mssql_query(\"set SVNR = SVNR\"); ");


			//Jetzt kommen Schleifen damit nicht immer die selben Datensaetze entstehen

			//person Updates werden erzeugt
			count = 0;
			while (count < eingabe*2+1)
			{
				ausgabe.print ("mssql_query(\"update person set Name = 'Name" +count);
				ausgabe.print ("' where SVNR ='SVNR" +count);
				ausgabe.println ("'\"); ");
				count++;
			}


			//arbeitetan Updates werden erzeugt
			count = 0;
			while (count < eingabe)
			{
				ausgabe.print ("mssql_query(\"update arbeitetan set ID = 'ID" +count);
				ausgabe.print ("' where SerienNr ='SerienNr" +count);
				ausgabe.println ("'\"); ");
				count++;
			}
```

```
                        //buch Updates werden erzeugt
                        //Counter damit man gerade Zahlen (ISBNNR) hat
                        count = 0;
                        count2 = 0;
                        while (count < eingabe*2)
                        {
                                ausgabe.print ("mssql_query(\"update buch set Titel = 'Titel"
+count2);
                                ausgabe.print ("' where SVNR ='SVNR" +count);
                                ausgabe.println ("'\"); ");
                                count++;
                                count2 = count2 + 2;
                        }


                        //Counter damit man ungerade Zahlen (ISBNNR) hat
                        count = 0;
                        count2 = 1;
                        while (count < eingabe*2+1)
                        {
                                ausgabe.print ("mssql_query(\"update buch set Titel = 'Titel"
+count2);
                                ausgabe.print ("' where SVNR ='SVNR" +count);
                                ausgabe.println ("'\"); ");
                                count++;
                                count2 = count2 + 2;
                        }


                        //computer Updates werden erzeugt
                        // Counter damit man gerade Zahlen (Seriennummern) hat
                        count = 0;
                        count2 = 0;
                        while (count < eingabe*2)
                        {
                                ausgabe.print ("mssql_query(\"update computer set Hersteller =
'Hersteller" +count2);
                                ausgabe.print ("' where SVNR ='SVNR" +count);
                                ausgabe.println ("'\"); ");
                                count++;
                                count2 = count2 + 2;
                        }


                        //Counter damit man ungerade Zahlen (Seriennummern) hat
                        count = 0;
                        count2 = 1;
                        while (count < eingabe*2+1)
                        {
                                ausgabe.print ("mssql_query(\"update computer set Hersteller =
'Hersteller" +count2);
```

```
                                ausgabe.print ('" where SVNR ='SVNR" +count);
                                ausgabe.println ('"\"); ");
                                count++;
                                count2 = count2 + 2;
                        }


                        //projekt Updates werden erzeugt
                        count = 0;
                        while (count < eingabe)
                        {
                                ausgabe.print ("mssql_query(\"update projekt set Projektname =
'Projektname" +count);
                                ausgabe.print ('" where ID ='ID" +count);
                                ausgabe.println ('"\"); ");
                                count++;
                        }


                        //Stoppen der Zeit
                        ausgabe.println ("$zeitmessung2 = microtime(); ");
                        ausgabe.println ("$zeittemp = explode(\" \",$zeitmessung2); ");
                        ausgabe.println ("$zeitmessung2 = $zeittemp[0]+$zeittemp[1]; ");
                        ausgabe.println ("$zeitmessung = $zeitmessung2-$zeitmessung1;  ");
                        ausgabe.println ("$zeitmessung = substr($zeitmessung,0,8); ");
                        ausgabe.println ("$gesammtzeit = $gesammtzeit + $zeitmessung; ");
                        ausgabe.println ("print(\" Die Funktion dauert: <strong> $zeitmessung
</strong> Sekunden. <br> \"); ");


                        // Erhöhung des Counter der Java Schleife und der für PHP
                        ausgabe.println ("$durchlaufzaehler = $durchlaufzaehler + 1; "); //PHP
Counter
                        count3++; //Java Counter
                }

                ausgabe.println ("  print  (\"-----------------------------------------------------------
---" +"<br> \"); ");
                ausgabe.println ("$durchschnitt_ergebnis = $gesammtzeit /$durchlaufzaehler;
");
                ausgabe.println ("  print  (\"Die Funktionen dauerten im Durchschnitt :<strong>
$durchschnitt_ergebnis </strong> Sekunden.<br><br> \"); ");
                ausgabe.println("print  (\"Zusatzinfo: es gab bei dem Test <strong>
$durchlaufzaehler </strong>Durchläufe und er dauerte insgesamt <strong> $gesammtzeit
</strong> Sekunden. <br><br> \"); ");
        /*      ausgabe.println("print  (\"Und so sehen die Updates in der Tabelle aus:
<br><br>\"); ");


                //SQL Anweisungen das die Tables ausgelesen werden
```

```
                ausgabe.println(" print  (\" <strong>Updates in arbeitetan: </strong><br> \") ;
");
                ausgabe.println ("$result = mssql_query(\"SELECT * FROM arbeitetan\"); ");
                ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
                ausgabe.println("{echo \"$msrow[0]  \"; echo \"$msrow[1] </br>\"; }");

                ausgabe.println("print  (\" <strong>Updates in buch: </strong><br> \") ; ");
                ausgabe.println ("$result = mssql_query(\"SELECT * FROM buch\"); ");
                ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
                ausgabe.println("{echo \"$msrow[0] \"; echo \"$msrow[1]  <br>\"; }");

                ausgabe.println(" print  (\" <strong>Updates in computer: </strong><br> \") ;
");
                ausgabe.println ("$result = mssql_query(\"SELECT * FROM computer\"); ");
                ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
                ausgabe.println("{echo \"$msrow[0] \"; echo \"$msrow[1]  <br>\"; }");

                ausgabe.println(" print  (\" <strong>Updates in person: </strong><br> \") ; ");
                ausgabe.println ("$result = mssql_query(\"SELECT * FROM person\"); ");
                ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
                ausgabe.println("{echo \"$msrow[0]\"; echo \"$msrow[1]  <br>\"; }");

                ausgabe.println(" print  (\" <strong>Updates in projekt: </strong><br> \") ; ");
                ausgabe.println ("$result = mssql_query(\"SELECT * FROM projekt\"); ");
                ausgabe.println ("while ($msrow = mssql_fetch_row($result))");
                ausgabe.println("{echo \"$msrow[0]\"; echo \"$msrow[1]  <br>\"; }"); */

                ausgabe.println("?>");          //Schließen des PHP Tags
                ausgabe.close(); //Hier wird die Ausgabe geschlossen damit es sauber ist!
        }
}
```

Allgemeine Benchmark Informationen

Wir hatten 10 Tests mit jeweils unterschiedlichen Datensätzen.

```
1) 1.000 Inserts/ Updates
2) 5.000 Inserts / Updates
3) 10.000 Inserts/ Updates
4) 50.000 Inserts/ Updates
5) 100.000 Inserts/ Updates
6) 250.000 Inserts/ Updates
7) 500.000 Inserts/ Updates
8) 750.000 Inserts/ Updates
9) 1.000.000 Inserts/ Updates
10) 2.500.00 Inserts/ Updates
```

Es gab oft mehrere Durchläufe bei den Tests damit die Dauer, die vor allem bei kleineren Werten verhältnismäßig recht stark schwankt ausgeglichener zu machen.

```
1, 2, 3 und 4:  jeweils 5 Durchläufe
5: 3 Durchläufe
6: 2 Durchläufe
7, 8, 9 und 10: jeweils 1 Durchlauf
```

Inserts mit X Inserts

Da wir in unserem Programm auch 2 n: m Beziehungen hatten und deswegen oft mehrere Schlüssel einander zugewiesen haben, haben wir wenn wir eigentlich X Inserts anlegen, nicht X Inserts, oder 5 Mal so viele (da wir 5 Tables haben), sondern um einiges mehr. Deswegen habe ich hier die Inserts in den jeweiligen Tables aufgelistet und wie viel Inserts es dann immer Insgesamt in der Datenbank gibt!

person: **X*2 +1** Inserts
arbeitetan: **X** Inserts
buch: **X*4 +1** Inserts
computer: **X*4 +1** Inserts
projekt: **X** Inserts
Summe: (X*2+1)+ X + (X*4 +1)+ (X*4 +1) +X
=> 2X+ 1 +X+ X*4+ 1+ X*4+ 1 + X
=> **12X+3** Inserts

Wenn man also X Inserts macht, hat man eigentlich dann 12X+3 Inserts in der Datenbank, sprich wenn man z.B. 5 Inserts macht, hat man effektiv 12*5+3 Inserts=> 63 Inserts

Updates

person: **X*2 +1** Updates
arbeitetan: **X** Updates
buch: **X** Updates
computer: **X** Updates
projekt: **X** Updates
Summe: (X*2 +1) +X+ X+ X+ X =>**6X+1** Updates

Wenn man also X Updates macht, hat man eigentlich dann 6X+1 Inserts in der Datenbank, sprich wenn man z.B. 5 Inserts macht, hat man effektiv 6*5+1 Inserts=> 31 Inserts

Abschlussworte

Wir hatten mit 3 Datenbanken zu tun, programmierten PHP Seiten, programmierten Java Programme (zum ersten Mal in unserem Leben auch mit FileWriter und BufferedReader), ER Diagramme erstellten, eine Webseite machten und dort alles verknüpften, uns Datenbanken überlegten auf denen die Anforderungen zu treffen, uns die Informationen besorgen mussten

und schnell lernten, dass dies bei einigen Datenbanksystemen extrem einfach geht, bei einigen aber so gut wie unmöglich ist bzw. es große Probleme gibt an Infos ran zu kommen. Deswegen fand ich das Projekt sehr interessant, habe viel gelernt und fand es auch sehr praxisnah und von daher auch gut für meine Zukunft! Und vor allem konnte ich endlich persönlich raus finden, ob MS SQL sich gut schlagen kann gegen die Konkurrenz oder untergeht.

Mit meiner Kollegin hat auch alles bestens geklappt, da wir eine sehr gute Aufteilung hatten, sie sehr zuverlässig ist, jeder seine Sache machte/ lernte und wir mit der Zeit auch die Datenbanksysteme des anderen kennen lernten und uns deswegen auch so sehr gut gegenseitig helfen konnten, wenn der andere ein Problem hatte und nicht weiter kam. Da wir aber trotzdem ein paar mal unsicher waren ob das so stimmen kann bzw. allgemeine Fragen hatten („Geht das überhaupt?“, „Wie viel können wir auf den Server uploaden?“, „Wie könnte man das überhaupt vom Prinzip her machen und geht das überhaupt?“) war die Tutorin auch sehr hilfreich, die in der Tutoriumszeit uns gute Denkanstöße gab, uns bei Problemen weiter helfen konnte, und auch unter anderem am Wochenende auf unsere Mails oft innerhalb von paar Minuten kompetent antwortete und weiter half. Von daher war alles sehr gut!

Schade war nur ein wenig, dass es einfach so schwer war/ ist, gute MS SQL Informationen aufzutreiben.

Der Server an sich den wir zur Verfügung gestellt bekommen haben, war sehr gut, aber hatte leider einige male komische Probleme, wo PHP Dateien plötzlich nicht gingen, die vorher gingen, und paar Minuten später wieder problemlos gingen! Weiters war das Problem, dass es ab und zu große Serverschwankungen gab, wo die Testresultate recht weit auseinander waren.

Vor allem wenn man Dateien auf dem Server uploadede, konnte man in dieser Zeit ein benchmarken vergessen, da oft die Testzeiten zwischen dem 2 bis 8 fachen über den normalen Werten lagen. Auch Select Tests konnte man vergessen, da sogar die oft statt 4 Sekunden an die 20- 30 Sekunden dauerte, was sehr problematisch war, da die ganzen Tests doch SEHR lange dauerten, und das Uploaden auch, und man somit wirklich diese 2 Sachen nicht gleichzeitig machen konnte sondern nur hintereinander, und wenn der andere (Kollegin oder ich) auch noch Probleme mit seiner PHP Datei hatte und man sich die dann anschauen wollte, und zum testen auf den eigene Account geben wollte, konnte man das dann nicht machen, da man entweder wieder nen „Stundenlangen“ Benchmarktest machte, oder es wieder nen mehrstündigen PHP Dateien Upload gab und man nichts anderes Uploaden konnte bzw. testen konnte oder man dann oft die Tests/ uploaden abbrechen musste. Da wir aber die Tests so gut wie möglich machen wollten, mussten wir aber so viele Tests machen und große PHP Dateien ins Internet stellen und konnten von daher keine Einschränkungen machen!

Die Eingabe in SSH verursachte auch einige Male große Probleme am Anfang, aber diese Probleme waren zum glück recht selten.

Trotzdem war es ein schönes Projekt, bei dem man vielen Probleme kennen gelernt hat (mit Java, PHP, ER Diagramme, den unterschiedlichen Datenbanken, Server) was einem auch viel für die Zukunft bringen kann, denn in der Realität ist es nun mal so das nicht alles immer gleich geht wie man es haben will. Und vor allem weiß ich endlich, wie sich MS SQL gegen die anderen Datenbanksysteme schlägt ☺ .