DBS 2 - Stoff – Zusammenfassung (Ullrich)

MySQL - Teil

WAMP:	Windows – Apache – MySQL – PHP – Installation
Apache:	Der im Internet am weitesten verbreitete Webserver
MySQL:	Relationales Datenbanksystem, das häufig für dynamische Internetanwendungen eingesetzt wird. Open-Source-Produkt, daher lizenzfrei verfügbar
PHP:	HypertextPreprocessor, ermöglicht mittels serverseitiger Skriptsprache die Gestaltung dynamischer Webseiten mit Datenbankunterstützung; ist Open Source
Wann kann ich mit WAMP arbeiten?	• Apache muss gestartet sein • PHP-Dateien müssen in Unterordner des Document-Root abgelegt sein • In Adresszeile des Browsers muss „localhost/“ eingegeben sein
SQL benutzen	• Dialogorientiert: Befehle eingeben • Batchbetrieb: Befehle werden in Textdatei geschrieben und abgearbeitet • Hilfsprogramme
Besonderheit!:	Sollen Befehle über eine Batch-Datei abgearbeitet werden, so muss diese Datei zuerst mit ASCII-Editor erstellt werden, der Aufruf erfolgt anschließend über die cmd-shell.
Richtlinien:	Groß- und Kleinschreibung ist bei Befehlen irrelevant, bei DB-Namen, Tabellen, Attributen und Werten hingegen nicht. Alle Befehle werden durch **;** abgeschlossen Kommentare: /* Kommentar */
LITERALE: **Zeichenketten:**	werden durch einfache oder doppelte Anführungszeichen gekennzeichnet. Um Anführungszeichen in Zeichenketten verwenden zu können, setzen Sie das entgegengesetzte Anführungszeichen: *“Jetzt schlägt‘s 13 “ ‘Dies ist ein 19 “- Monitor‘* Oder setzten Sie vor das Anführungszeichen innerhalb der Zeichenkette das Escape-Zeichen (Backslash): *‘Jetzt schlägt\‘s 13‘*
Sonderzeichen:	**\n** Zeilenumbruch **\t** Tabulator **\r** Wagenrücklauf **\b** Backspace **** Backslash **_** Unterstrich (SQL-Platzhalterzeichen) **\“** Doppeltes Anführungszeichen **\‘** Einfaches Anführungszeichen **\%** Prozentzeichen (SQL-Platzhalterzeichen)

LITERALE: **Nullwerte:**	Das spezielle Schlüsselwort NULL repräsentiert nicht 0, sondern steht für undefinierte Felder oder nicht vorhandene Daten. Achtung: Die leere Zeichenkette "" entspricht nicht NULL. **ACHTUNG bei Vergleich auf den Wert NULL muss der Vergleichsoperator IS NULL verwendet werden, nicht = NULL**
Aliase:	**Aliasbezeichner in FROM-Klausel**, um Tabellen mit einem Kürzel ansprechen zu können: *SELECT k.knr, k.name k.adresse, k.mail FROM **kunden k**;* Spaltenalias zur **Benennung von Spalten**: *SELECT price **AS** netto, price*1,2 **AS** brutto FROM products;*
DATENBANK:	
Erstellen:	CREATE DATABASE *NAME;*
Zeigen:	SHOW DATABASES;
Auswählen:	USE *NAME*;
Tabellen anzeigen:	SHOW TABLES;
Zeigen aller Felder einer Tabelle:	DESCRIBE *TABELLENNAME;* oder SHOW COMUMNS FROM *TABELLENNAME;*
Löschen einer Tab.:	DROP TABLE *NAME*;
Löschen einer DB:	DROP DATABASE *DBNAME*;
Tabelle erzeugen:	CREATE TABLE *NAME* („Create-Definitionen");
Create-Definitionen:	z.B.: CREATE TABLE kategorien(kategorie_nr integer primary key, kategoriename varchar(20) not null, beschreibung text);
Tabellen ändern:	ALTER TABLE table_name alter_spezification alter_specification: • ADD PRIMARY KEY (index_spalte,...) • ADD UNIQUE (index_spalte) • DROP PRIMARY KEY • DROP column_name • RENAME new_table_name • ADD [column] create_definition • CHANGE old_column_name create_definition
Einfügen von Tupeln:	INSERT INTO personen (p_nr, name, v_name, geschlecht, gehalt) VALUES ('4', 'Huber', 'Markus', 'm',7000);
Aktualisieren:	UPDATE personen SET gehalt = gehalt * 1,1 WHERE geschlecht = 'w';
Löschen von Tupeln:	DELETE FROM *tablename* *Bsp.: DELETE FROM personen WHERE p_nr = 4*
Filtern/Auswählen:	Bsp.: SELECT ename, deptno FROM emp WHERE deptno in(10,30) ORDER BY ename;

PHP TEIL

Was ist PHP?	eine serverseitige, in HTML eingebettete Scriptsprache d.h. PHP-Scripts werden auf dem Server ausgeführt. Code wird in eine HTML-Datei geschrieben. Code erscheint nicht auf Client-Seite. HTML Code wird beim Aufruf der Website zum Client geschickt, PHP-Code aber beim Server ausgeführt, nur die Ausgabe wird an Client gesandt.
GRUNDSÄTZLICHES:	PHP–Programme können sowohl vollständig innerhalb des <head>-Tags, als auch vollständig innerhalb des <body>-Tags einer HTML-Seite untergebracht werden, sie dürfen aber nicht im <head> beginnen und im <body> enden. Es kann mehrmals zwischen HTML und PHP gewechselt werden, es genügt, ein PHP-Tag zu öffnen und danach wieder zu schließen. Soll HTML_Code innerhalb von PHP-Tags verarbeitet werden, so muss dies mit dem Befehl echo (Ausgabe) geschehen. Durch echo kommen die HTML-Befehle zu Ausführung. BSP: <?php echo „<form>“; ?>
VARIABLEN:	werden in PHP durch vorgestelltes ‚$‘ gekennzeichnet. Sie müssen nicht vorher deklariert werden, d.h. der Datentyp muss nicht durch den Programmierer festgelegt werden. Eine Variable kann ihren Datentyp innerhalb des Programms wechseln Bsp.: $var=17; $a=„hallo“; $a=$var; // $a enthält jetzt den Wert 17 müssen mit $-Zeichen beginnen, dürfen keine Leerzeichen enthalten, dürfen nur aus Buchstaben und Ziffern bestehen, das erste Zeichen soll ein Buchstabe sein, es wird zwischen Groß- und Kleinschreibung unterschieden, das einzige zulässige Sonderzeichen ist _ (Underscore), Umlaute sind nicht zulässig Will man der Variablen $a den Wert der Variablen $b zuweisen, so muss das mittels Gleichheitszeichen geschehen: $a=$b
GRUNDBEFEHLE:	
Kennzeichnung PHP:	<?php ..... ?> oder <script language=“php“>.....</script>
Ausgeben von Text:	**echo** „Hallo Leute“;

Strings verketten:	Bsp.: $a = “hallo“;
	$b = $a.“PHP“; // $b enthält den Wert: „hallo PHP“
	Bsp.: $anrede=„Herrn“;
	$vorname=„Hans“;
	$name=$anrede.“ “.$vorname.“ “;
	$nachname =„Maier“;
	$name .=$nachname;
	// oder $name = $name . $nachname;
	echo „der heißt: $name“;

VERZWEIGUNGEN:

IF:

falls $a kleiner als $b ist wird der entsprechende Text ausgegeben, ansonsten geschieht nichts.

IF ($a<$b)

{

echo „$a ist kleiner als $b“;

}

die Bedingung muss in runden Klammern stehen, nach jedem Statement folgt ein Strichpunkt, die Anweisungen, die von der Bedingung abhängig sind stehen in geschweiften Klammern.

IF – ELSE:

Bsp.: $a=10;

$b=12;

IF ($a==$b)

{ echo „$a ist gleich $b“; }

ELSE

{ echo „$a ist ungleich $b“; }

IF – ELSEIF – ELSE:

Bsp.: IF ($a==$b)

{ echo „$a ist gleich $b“; **}**

ELSEIF ($a<$b)

{ echo „$a ist kleiner $b**“; }**

ELSE

{ echo „$a ist größer $b; **}**

ACHTUNG: hinter IF und ELSEIF **muss** eine Bedingung angeführt werden, hinter ELSE darf **KEINE** Bedingung angegeben werden.

SWITCH:

Ähnlich einer Serie von IF-Anweisungen mit denselben Ausdrücken.

switch($i) {

case ($i==1): echo „sehr gut!“; break;

case ($i==2): echo „gut“; break;

....

default: echo „keine gültige Note“; }

break: Sofortiges Verlassen der Schleife

default: wird ausgeführt, wenn kein anderer Fall zuvor zutrifft.

SCHLEIFEN:

Falls sich innerhalb eines Programms einzelne Anweisungen oder Blöcke von Anweisungen wiederholen, werden Schleifen verwendet.

- **FOR**: Falls die Anzahl der Wiederholungen dem Programmierer bekannt ist, oder sich eindeutig im Verlauf des Programms vor der Schleife ergibt.
- **WHILE**: Falls die Anzahl der Wiederholungen dem Programmierer nicht bekannt ist, sondern die Schleife aufgrund einer Bedingung wiederholt bzw. abgebrochen wird.

  Bsp.:

SCHLEIFEN:
BEISPIELE:
	Es wird solange eine Zufallszahl zwischen 1 und 6 summiert, solange die Summe der Zufallszahlen kleiner 30 ist.
	srand((double)microtime()*1000000) // init von zufallszahlengenerator
	$summe=0;
	WHILE ($summe<30) {
	$zufallszahl = rand(1,6);
	$summe =$summe + $zufallszahl;
	echo „Zahl: $zufallszahl, Summe: $summe“;**}**
	
	For ($i=1;$i<=10;$i++) {
	echo $i.“
	“; }
	Alle Zahlen von 1 bis 10 werden ausgegeben.

SCHLEIFENSCHACH-TELUNGEN:	Eine Schleife befindet sich innerhalb der anderen. Bsp.: **FOR($i=1;$i<=5;$i++) {** **FOR($j=1;$j<=3;$j++) {** **echo „Zeile: $i / Spalte: $j “;}** **echo “<p>“;}** Die äußere Schleife wird 5mal durchlaufen, innerhalb dieser steht wiederum eine Schleife, die wird pro äußerem Schleifendurchlauf je 3x durchlaufen. **Output**: Zeile:1/ Spalte:1 Zeile:1/ Spalte:2 Zeile:1/ Spalte:3 Zeile:2/ Spalte:1 Zeile:2/ Spalte:2 Zeile:2/ Spalte:3
BREAK:	Mit Hilfe der Anweisung break kann die Schleife sofort beendet werden Bsp.: **if ($zaehler >9)break;**
WEITERE SCHLEIFEN-ANWEISUNGEN: **FOREACH:**	wird besonders im Zusammenhang mit Arrays verwendet. foreach([Array-Ausdruck]) { statements; }
CONTINUE:	Die Anweisung continue wird verwendet, um aufgrund einer Bedingung den Rest der Schleife zu überspringen und unmittelbar mit dem nächsten Schleifendurchlauf fortzusetzen.
INCLUDE („DATEI“):	fügt an dieser Stelle den Inhalt der Datei “dateiname“ ein. Dadurch ist es möglich, Quellcode, der an mehreren Stellen benötigt wird, zentral zu halten, sodass Änderungen einfacher werden.

FUNCTIONS: **FUNKTIONSTYPEN:** **BSP.:**	dienen zum Zusammenfassen mehrerer Befehle zu einem Aufruf, dadurch werden Programme lesbarer, weil es klar ist, wozu ein Befehlsblock dient • **Funktionen ohne Parameter** Diese Fkt. führen bei jedem Aufruf immer die gleiche Aufgabe aus. • **Funktionen mit einem oder mehreren Parametern** Diese Fkt. führen bei jedem Aufruf in Abhängigkeit von den Parametern ähnliche Aufgaben aus. • **Funktionen mit Rückgabewerten** Fkt. Führen gleiche oder ähnliche Aktionen aus und liefern ein Ergebnis an die aufrufende Stelle zurück **function** add($zahl1, $zahl2) { $summe = $zahl1+$zahl2; return $summe; }
PHP + HTML-FORMULARE: **BSP.:**	Im Formular muss als ‚action' der Dateiname der php-Datei und als method=‚post' angegeben werden <html> <body> <form action = "ausgabe.php" method = post> Feld1: <input name = „feld1"> Feld2: <input name = „feld2"> <input type= submit value =„OK"> <input type= reset value =„abbrechen"> </form> </body> </html>
ARRAYS: **EINFACHES BSP.:** **LÄNGE DES ARRAY ERMITTELN:** **ASSOZIATIVE ARRAYS:**	Variablen, die mehrere Werte gleichzeitig speichern können. *$seasons = array["Spring","Summer","Autumn","Winter"];* *$seasons = array();* *$seasons[0] = "Spring"; ...* FOR($i = 0; $i<count($seasons);$i++) { echo $seasons[$i]." "; } Elemente werden nicht über einen Index, sondern über einen Schlüssel angesprochen. (z.B. Holen von Zeilen aus Tabellen von DB's). $person = array("vname"=>"Karl", "nname"=>"Maier", "email"=>karl.maier@aon.at); Hier sind vname, nname und email die Schlüssel. Will man einen weiteren Wert hinzufügen, so kann dieser mittels: *$person["alter"] = 32;* erzeugt und gleichzeitig gesetzt werden.

ZWEIDIMENSIONALE ARRAYS:	Solche Arrays sind wie Tabellen, sie speichern Informationen an Hand von 2 Schlüsseln. Ideal zum Speichern von Info über mehrere gleich strukturierte Objekte: $personen = array(“Maier“ => array(“vname“=>“Karl“, “email“=>karl.maier@aon.at), “Huber“ => array(“vname“=>“Hans“, “email“=>hans.huber@aon.at)); Das Array $personen enthält Informationen über Personen Maier und Huber, will man eine konkrete Information, so muss man beide Schlüssel verwenden. Die email von Hans Huber könnte man dann so ausgeben: *echo $personen[“Huber“][“email“];*

MySQL – PHP API

***BEFEHLE*:**	
MYSQL_CONNECT:	öffnet eine Verbindung zum DB-Server; **int mysql_connect(string host, string user, string password);**
MYSQL_CLOSE:	explizit genau eine Verbindung (die dem link_identifier entsprechende) zum MYSQL-Server wieder geschlossen; bei keinem link_id wird die zuletzt geöffnete Verb. geschl. **int mysql_close(int link_identifier);**
MYSQL_SELECT_DB:	aktiviert die Datenbank auf dem Server zu dem die Verbindung mit der übergegebenen Verbindungs-Kennung besteht **int mysql_select_db(string dbname, int link_identifier);**
MYSQL_QUERY:	sendet eine Anfrage an die zur Zeit aktive Datenbank **int mysql_query(string query, int link_identifier);**
MYSQL_DB_QUERY:	bestimmt eine Datenbank und führtgleichzeitig eine Anfrage an diese aus **int mysql_db_query(string dbname, string query);**
MYSQL_NUM_ROWS:	Gibt die Anzahl der Zeilen der Ergebnismenge einer SELECT-Query zurück **int mysql_num_rows(int result);**
MYSQL_AFFECTEC_ ROWS:	Gibt die Anzahl der von der letzten Aktionsabfrage (update, delete, insert) betroffenen Zeilen zurück **int mysql_affected_rows(int result);**

MYSQL_FETCH_ROW:	Holt jeweils eine Zeile aus dem Ergebnis der DB-Abfrage, gibt diese als Array zurück und setzt den Result-Zeiger auf die nächste Zeile **array mysql_fetch_row(int result);**

KURZES BSP.:	<? Php
	$db =mysql_connect($host, $user, $ password);
	mysql_select_db(“databasename“, $db);
	$res = mysql_query(“select empno, ename from emp“);
	While($row = mysql_fetch_row($res))
	{
	echo $row[0]; // Wert aus Spalte empno
	echo $row[1]; // Wert aus Spalte ename
	} ?>

MYSQL_FETCH_ARRAY:	Liefert ein Array das dem aktuellen Datensatz entspricht oder **FALSE**, wenn keine weiteren Datensätze vorliegen ist eine erweiterte Version von mysql_fetch_row(). **array mysql_fetch_array(int result);**

KURZES BSP.:	<? Php
	$db =mysql_connect($host, $user, $ password);
	mysql_select_db(“databasename“, $db);
	$res = mysql_query(“select empno, ename from emp“);
	While($row = mysql_fetch_array($res))
	{
	echo $row[“empno“]; // Wert aus Spalte empno
	echo $row[“ename“]; // Wert aus Spalte ename
	} ?>

MYSQL_RESULT:	liefert den Inhalt eines Felds aus einem Anfrageergebnis und zwar den Inhalt der Spalte “spaltenname“, des $i-ten Datensatzes. D.h. genau einen Wert! **$var = mysql_result($res, $i, “spaltenname”);**
MYSQL_NUM_FIELDS:	liefert die Anzahl der Felder (Spalten) in der Ergebnismenge, die mit dem Parameter Ergebnis-Kennung angegeben wurde **array mysql_num_fields(int result);**
MYSQL_FIELD_NAME:	liefert den Namen des Feldes, der dem angegeben Feldindex entspricht **string mysql_field_name (int result, int Feldindex)**
MYSQL_FIELD_TYPE:	ähnlich wie voriges, liefert jedoch den Datentyp **string mysql_field_type (int result, int Feldoffset)**

MYSQL_FIELD_TABLE:	Liefert den Namen der gesamten Tabelle die das genannte Feld enthält **string mysql_field_table (int result, int Feldoffset)**
MYSQL_ERROR: **MYSQL_ERRNO:**	Gibt die Fehlermeldung des letzten SQL-Befehls zurück, wenn es keinen Fehler gab, wird nichts zurückgegeben; oder die Fehlernummer **string mysql_error([int link_identifier]);** **string mysql_errorno([int link_identifier]);**
MYSQL_INSERT_ID:	Gibt die Nummer zurück, die beim letzten INSERT dem Feld mit AUTO_INCREMENT zugewiesen wurde **int mysql_insert_id(int link_identifier);**
INCLUDE: **BSP.:**	fügt an dieser Stelle den Inhalt der Datei “dateiname“ ein. Dadurch ist es möglich, Quellcode, der an mehreren Stellen benötigt wird, zentral zu halten, sodass Änderungen einfacher werden *<? php include(“dbconnect.php“); ?>*

Regular Expressions:

DEFINITION:	sind ein gutes Tool zur Suche in Texten

BSP.:	Auffinden des Wortes “Test“ in einem Text:
	if (ereg(“Test“,$str))
	{
	....
	}

KOMPLEXERES BSP.:	Überprüfung einer e-mail Adresse ^[\.0-9a-z-]+@([0-9a-z][0-9a-7-]+\.)+[a-z]{2,3}$
ERLAUBTE ZEICHEN:	• \n Zeilensprung • \t Tabulator • \r Return • \f Zeilenvorschub • ^ Beginn einer Zeile • $ Ende einer Zeile • . Beliebiges druckbares Zeichen • **** schützt die eigentliche Bedeutung eines Zeichens
BSP.:	• ereg(“^Test“,$str) Prüfung, ob Test am Anfang der Zeile. • ereg(“^Test$“,$str) Ist Test das einzige in $str? • ereg(“Test(....)“,$str,$arr); Welche 4 Zeichen folgen Test? echo $arr[1];

FESTSTEHENDE ZEICHENKLASSEN:	[[:alpha:]] Groß- oder Kleinbuchstaben [[:digit:]] Ziffern(0-9) [[:space:]] alle Arten von Leerzeichen (whitespace) [[:upper:]] Großbuchstaben [[:lower:]] Kleinbuchstaben [[:punct:]] Interpunktionszeichen [[:xdigit:]] Hex-Ziffern
EIGENE ZEICHENKLASSEN:	Mit Hilfe eckiger Klammern können eigene Zeichenklassen gebildet werden: Dabei kann mit Bereichen, oder mit dem Auflisten einzelner Zeichen gearbeitet werden. [A-Za-z] oder [a-m] oder [0-4] oder aufzählen der gewünschten Menge [dog0-9]
WICHTIGSTE QUANTOREN:	*** vorhergehendes Zeichen kann gar nicht, oder beliebig oft vorkommen.** **+ vorhergehendes Zeichen muss zumindest einmal vorkommen.** **? vorhergehendes Zeichen darf höchstens einmal vorkommen.**

BSP.:	if ereg("\?(.*)$", $str, $arr)
	{
	echo "Die Parameter:", $arr[1];
	}
	Durch das Erzeugen des Teilstrings mittels (), sind alle dem ? folgenden Parameter in $arr[1] gespeichert.

Beispiel zu +:	[0-9]+ Ausdruck für die Sequenz von ein oder mehreren Ziffern Will man die Anzahl von bestimmten Zeichen festlegen, so kann man dazu {} verwenden. {3} einzelne Zahl besagt, dass genau diese Anzahl gemeint ist. Z.B. j{3} sucht nur nach jjj. {3,5} es werden minimale und maximale Grenzwerte definiert z.B. j{3,5} sucht nach jjj, jjjj, und jjjjj. {3, } es wird nur der minimale Grenzwert definiert.