

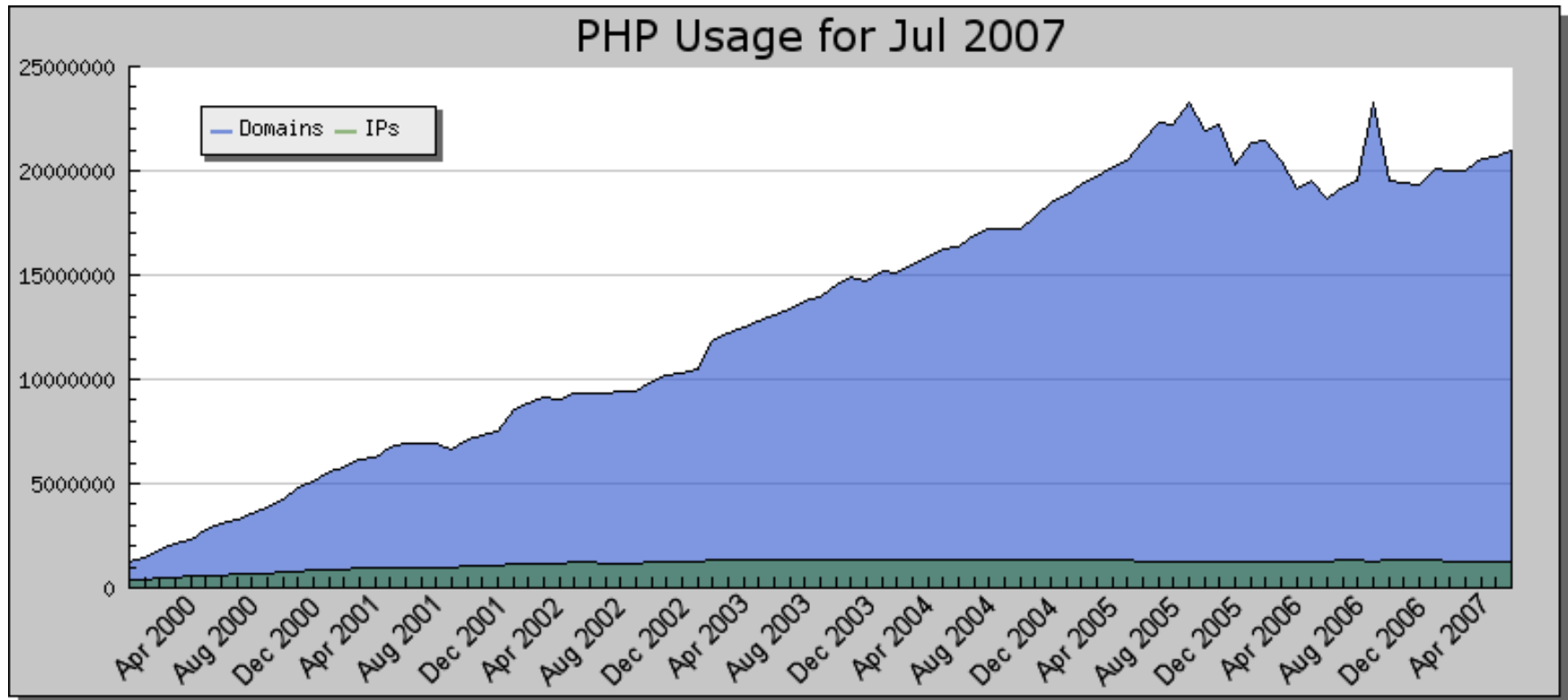
Web-Programmierung mit PHP und MySQL

**Sebastian Tramp
Januar 2012**

Vielen Dank an Andreas Thor und Sören Auer für die Überlassung der Folien

- Dynamische Webseiten
 - Entwicklung von PHP
 - PHP an Beispielen
 - Anbindung an MySQL
-
- Theoretische Grundlagen von relationalen Datenbanken
 - Erstellung einer Datenbank am Beispiel
 - Grundlagen von SQL
 - Datenbanken in der Praxis

- Personal Homepage Tools, Rasmus Lerdorf
- Erste Version 1994
 - CGI Tools
 - keine Scriptsprache
- PHP/FI 1996
 - Scriptsprache, Anbindung an Datenbanken
- PHP Version 3 1998
 - Open Source, Suraski, Gutmans
- PHP Version 4 2000
 - Redesign, Zend
- PHP Version 5 2004
 - Exceptions, Reflections API, Namespaces, Performance



Eine der verbreitetsten Web-Scripting-Umgebungen
U.a. verwendet für Wikipedia, Yahoo, Facebook, ...

Programm erzeugt HTML

```
public class myServlet
    extends HttpServlet {

    protected void doGet(...) {
        response.setContentType(
            "text/html");
        java.io.PrintWriter out =
            response.getWriter();
        out.println("<HTML>");
        ...
        out.println("Hallo");
        out.println("</HTML>");
    }
}
```

HTML enthält Programm

```
<HTML>
...
<?php
    echo "Hallo";
?>
</HTML>
```

verschiedene Implementierungen:
PHP (C-ähnlich)
JSP (JAVA)
ASP (Visual Basic (und andere)
usw.

```
<html>
```

```
...
```

```
<?php  
  echo "Hallo";  
?>
```

// XML-Stil

```
<?  
  echo "Hallo";  
?>
```

/* SGML (PI) Stil */

```
<%  
echo "Hallo";  
%>
```

ASP Stil

```
<script language="php">  
echo "Hallo";  
</script>  
</html>
```

// „korrektes“ HTML

- Lösung eines (Anwendungs-) Problems durch Computer
- Formulierung des Problems in für Computer verständlicher Form
- Hier: Algorithmen, d.h. Beschreibung der Lösung durch Abfolge von einzelnen Schritten

<Schritt 1>

<Schritt 2>

<Schritt 3>

usw.

Folge von Anweisungen, Beispiel:

echo "Hallo";

Beispiel: Lösung(en) von $ax^2+bx+c=0$

Berechne b^2-4ac

Berechne Lösung 1

Berechne Lösung 2

Problem: Ergebnis aus Schritt 1 wird später gebraucht

→ Muss „gemerkt“ werden, wie? --> Variable

Variable = Platz (im Hauptspeicher) zur Speicherung von
(Zwischen-) Ergebnissen


```
<?php
```

```
$i = 0;           // Variablen werden nicht deklariert  
echo $i;          // Name beginnt mit $  
$i= "Hallo";      // Datentyp implizit (Zuweisung)  
$i= 'Hallo';      // String durch ' oder " terminiert  
$i=0.3E-4;        // Integer, Fließkomma, String,  
$i=true;          // Boolean  
$i=5;             // Ganzzahl  
$j=2*$i;          // j==10  
$j=$j+1;          // j==11
```

```
?>
```

Soll ich oder soll ich nicht ...

Berechne Preis = preis1 + preis2 + ...

Berechne Versandkosten (5€ pro Paket, 0€ wenn Preis > 50€)

Fallunterscheidungen (Verzweigungen)

Wenn ($\text{\$Preis} \geq 50$) dann $\text{\$Versand}$ ist 0€

Sonst $\text{\$Versand}$ ist 5€

```
<?php
    $preis1 = 10;           // Initialisierung
    $preis2 = 30;
    $preis3 = 5;

    $gesamt = $preis1 + $preis2 + $preis3; // Berechne Gesamt

    if ($gesamt >= 50) {    // Blöcke durch {}
        $Versand = 0;
    } else {
        $Versand = 5;
    }
?>
```

Oops, I did it again....

Wiederholungen:

<Schritt A>

<Schritt B>

<Schritt A>

<Schritt B>

<Schritt A>

<Schritt B>

<Schritt A>

<Schritt B>

<Schritt A>

<Schritt B>

.....

Besser:

Solange (Bedingung erfuehlt)

<Schritt A>

<Schritt B>

```
$i=0;
while ($i < 10) {
    echo $i;
    $i = $i + 1;
}
```

// Zahlen von 0 bis 9, langweilig

```
$i=0;
while ($i < 10) {
    echo "PHP ist toll";
    $i = $i + 1;
}
```

// besser

```
$zahl=4711; $i=2; $istPrim=true;
while ($i < $zahl) {
    if (($zahl % $i) == 0) {
        $istPrim =false;
        $i = $i + 1;
    }
}
```

// viel besser

Viele Daten ...

Berechnung des Notendurchschnitts (Klausur)

```
$Note1=1.0;
```

```
$Note2=1.8;
```

```
$Note3=2.7;
```

```
...
```

```
$Note24=3.0;
```

```
$Durchschnitt=( $Note1 + $Note2 + ... + $Note24) / 24
```

```
→Quatsch
```

Mathematik: Vektor --- Programmiersprache: Array

```
$Noten=array(1.0, 1.8, 2.7);
```

```
$Durchschnitt = 0.0;
```

```
$i=0;
```

```
while ($i < count($Noten) ) {
```

```
    $Durchschnitt = $Durchschnitt + $Noten[i];
```

```
    $i = $i + 1;
```

```
}
```

```
$Durchschnitt = $Durchschnitt / $Anzahl;
```

```
if (1 == 1)                                // Syntax wie in JAVA
//Schleifen: for, while, do ... while, foreach
$namen = array("max","hugo","gerd");

for ( $i=0; $i<count($namen); $i++) {
    echo $namen[$i];
    while ($i < 10) {
        echo $i;
        $i++;
    }
}

// Durchlaufen von Arrays
foreach($namen as $name) {
    echo $name;
}
```

Warum alles selbst programmieren?
Mehrfach gebrauchte Funktionalitäten werden als
„Funktion“ zur Verfügung gestellt, Beispiel Datum:

```
<?php
    $datum = getdate();
    echo $datum["year"];    // assoziatives Array
?>
```

Eine Funktion liefert (meistens) einen Wert zurück,
Kann Parameter haben (in Klammern, () wenn keine)
Kann man auch selbst schreiben:

```
function sagHallo() {
    echo "Hallo";
}
```



```
/*  
 * Berechnet den Durchschnitt eines Arrays von Noten  
 */  
function durchschnitt( array $Noten ) {  
    $Summe = 0; $i=0;  
    while ($i < count($Noten) ) {  
        $Summe = $Summe + $Noten[$i];  
        $i = $i + 1;  
    }  
    return $Summe / count($Noten);  
}  
  
// Benutzung der neuen Funktion  
$Noten = array(1.0, 1.8, 2.7);  
echo 'Durchschnitt der Noten ist ' . durchschnitt($Noten);
```

Wie andere Scriptsprachen auch ist PHP ziemlich flexibel,
ein Beispiel:

```
<?php
function minus($i) {
    return $i-1;
}

function plus($i) {
    return $i+1;
}

$op = "plus";
echo $op(3);
?>
```

Ein Beispiel

```
<html>
<head><title>Ein Beispiel</title></head>
<body>
<?php
for($i=0; $i<10; $i++)
    echo "Hello world<br>";
?>
</body>
</html>
```

```
<html>
<body>
<form action="DoHelloForm.php" method="post">
Wie oft ? <input type="text" name="WieOft">
<input type="submit">
</form>
</body>
</html>
```

DoHelloForm.php

```
<html>
<body>
<?
for($i=0;$i<$WieOft;$i++) // $WieOft enthält Eingabe
    echo "Hello world<br>";
?>
</body>
</html>
```

```
<html>
<body>
<?
if (isset($WieOft)) {
    for($i=0;$i<$WieOft;$i++)
        echo "Hello world<br>";
}
else {
    ?>
    <form action="CompactHelloForm.php" method="post">
    Wie oft ? <input type="text" name="WieOft">
    <input type="submit">
    </form>
    <?
}
?>
</body>
</html>
```

- Klassen
 - Kapselung von Datenstrukturen und
 - Kombination mit Funktionen (Methoden)

```
<?php
class SimpleClass
{
    // Deklaration einer Eigenschaft
    public $var = 'ein Standardwert';
    // Deklaration einer Methode
    public function displayVar() {
        echo $this->var;
    }
}

$object = new SimpleClass();
$object->displayVar();
?>
```

Datenbankdefinition

(nach Wikipedia)

„Ein Datenbanksystem (DBS) ist ein System zur elektronischen Datenverwaltung. Die wesentliche Aufgabe eines DBS ist es, große Datenmengen **effizient, widerspruchsfrei und dauerhaft** zu speichern und benötigte Teilmengen in **unterschiedlichen, bedarfsgerechten Darstellungsformen** für Benutzer und Anwendungsprogramme bereitzustellen.“

„Ein DBS besteht aus zwei Teilen: der Verwaltungssoftware, genannt **Datenbankmanagementsystem (DBMS)** und der Menge der zu verwaltenden Daten, **der eigentlichen Datenbank (DB)**. Die Verwaltungssoftware organisiert intern die strukturierte **Speicherung** der Daten und kontrolliert alle lesenden und schreibenden **Zugriffe** auf die Datenbank. Zur Abfrage und Verwaltung der Daten bietet ein Datenbanksystem eine **Datenbanksprache** an.“

Aufgaben / Eigenschaften eines DBS

Generell: effiziente und flexible Verwaltung großer Mengen persistenter Daten (z. B. GBytes - T Bytes)

1. Zentrale Kontrolle über die operationalen Daten
2. Hoher Grad an Datenunabhängigkeit
3. Hohe Leistung und Skalierbarkeit
4. Mächtige Datenmodelle und Anfragesprachen / leichte Handhabbarkeit
5. Transaktionskonzept (ACID), Datenkontrolle
6. Ständige Betriebsbereitschaft (hohe Verfügbarkeit und Fehlertoleranz)

Quelle: Vorlesungsskript DBS1 (Prof. Rahm, Univ. Leipzig)

- Relationale Datenbanken wichtigster DB-Typ
 - andere Typen: objekt-orientiert, XML, objekt-relational, ...
- Intuition: „Relationale Datenbanken bestehen aus Tabellen, in denen Daten gespeichert sind. Die Tabellen stehen in Beziehungen zueinander, d. h. manche Felder (Tabellenspalten) sind Verweise auf Felder in anderen Tabellen.“

- Datenstruktur: Relation (Tabelle)
 - jede Zeile (Tupel) ist eindeutig und beschreibt ein „Objekt“
 - alle Informationen (ausschließlich) durch atomare Werte dargestellt
 - Ordnung der Zeilen und Spalten unerheblich
- Darstellung von Beziehungen durch Fremdschlüssel
 - Attribut, das in Bezug auf Primärschlüssel einer anderen (oder derselben) Relation definiert ist -> „Verweis“
- Operatoren auf (mehreren) Relationen
 - Vereinigung, Differenz
 - Kartesisches Produkt (Kombination der Datensätze)
 - Projektion (Auswahl von Attributen)
 - Selektion (Filterung von Datensätzen)
 - Änderungsoperationen: Einfügen, Löschen, Ändern

Relationale Datenbanken: wichtige Begriffe

Begriff	Bedeutung
Relation	Tabelle
Tupel	Eine Zeile einer Tabelle
Kardinalität	Anzahl der Zeilen einer Tabelle
Attribut	Eine Spalte (Feld) einer Tabelle
Grad	Anzahl der Spalten einer Tabelle
Primärschlüssel	Eindeutiger Bezeichner einer Zeile (Attribut oder Attribut-Kombination), Bsp: Matrikelnr.
Fremdschlüssel	in einer Tabelle verwendetes Attribut(-komb.), das auf eine andere Tabelle verweist und dort ein Primärschlüssel ist
NULL	undefinierter Attributwert („unbekannt“) (für PS nicht möglich, bei anderen möglich)

DB-Beispiel: Entwurf (1)

Allgemein: Datenbankerstellung setzt konzeptionellen Entwurf einer „Mini-Welt“ (Ausschnitt der Realwelt) voraus

- Modellierung z.B. mittels ER oder UML
- Definition der Entitäten („Objekte“) und Relationen („Beziehungen“) und ihren Eigenschaften („Attribute“)

Das Prinzip einer Relationalen Datenbank wird am Aufbau einer einfachen Bestelldatenbank für einen EDV-Zubehörhändler erläutert.

Schritt 1: Datenerfassung – welche Daten sind für jeden Bestellvorgang erforderlich?

- Kundendaten: Name, Vorname, Adresse, Telefon, Email
- Artikeldaten: Kurzbezeichnung, Beschreibung, Preis
- Bestelldaten: Anzahl, Datum, Lieferstatus

DB-Beispiel: Entwurf (2)

Schritt 2: Erster Entwurf mit einer Tabelle bestehend aus 13 Spalten

Name	Vorname	Straße	PLZ	Ort	Tel	Email	>>
Artikel- bezeichnung	Beschreibung	Preis	Anzahl	Bestelldatum	Lieferstatus		

Nachteile des Entwurfs:

- Mehrfache Speicherung von Kunden- und Artikeldaten
- Änderungsprobleme: Bei Änderungen in Kunden- und Artikeldaten müssen viele Datensätze geändert werden – Gefahr von Inkonsistenz

DB-Beispiel: Entwurf (3)

Schritt 3: Normalisierung (Datenredundanz vermeiden):
Aus einer Tabelle werden 3 gemacht

- Tabelle 'Kunden' mit Informationen über die Kunden
- Tabelle 'Artikel' mit der Beschreibung der Waren
- Tabelle 'Bestellungen'

Kunde

ID	Name	Vorname	Straße	PLZ	Ort	Tel	email
----	------	---------	--------	-----	-----	-----	-------

Artikel

ID	Hersteller	Bezeichnung	Preis	Beschreibung	Lieferfrist
----	------------	-------------	-------	--------------	-------------

Bestellung

ID	Kunden_id	Artikel_id	Anzahl	Datum	Status
----	-----------	------------	--------	-------	--------

DB-Beispiel: Beispielausprägung



Kunde

ID	Name	Vorname	Straße	PLZ	Ort	Tel	email
11	Maier	Ulrich	Rosenweg 23	70599	Stuttgart	0711- 456 896	Umaier@gmx.com
12	Braun	Martina	Gartenstr. 12	70794	Filderstadt	0711 - 705661	m.braun@web.de

Artikel

ID	Hersteller	Bezeichnung	Preis	Beschreibung	Lieferfrist
21	HP	HP CX-895	2100	Tintenstrahl-Farbdrucker	3 Tage
22	Canon	BJ 1200 C	3400	Tintenstrahl- Farbdrucker	7 Tage

Bestellung

ID	Kunden_id	Artikel_id	Anzahl	Datum	Status
31	11	22	5	22.06.2001	Erledigt
32	12	21	1	04.01.2002	In Bearbeitung

Beziehungen zwischen den Tabellen

- Konzeptionell: Ein Kunde kann mehrere Artikel bestellen; ein Artikel von mehreren Kunden bestellt werden → N:M-Beziehung
- DB-Realisierung mit „Hilfstabelle“ Bestellung
 - Kunde : Bestellung → 1:N Beziehung (realisiert durch FS)
 - Artikel : Bestellung → 1:N Beziehung

Primär- und Fremdschlüssel im Beispiel

- Die Spalte ID ist ein Primärschlüssel in jeder Tabelle. Sie dient der eindeutigen Identifizierung eines Datensatzes.
- Die Spalten Kunden_id und Artikel_id in der Tabelle 'Bestellungen' sind Sekundär- oder Fremdschlüssel. Sie stellen die Verknüpfung zu den Tabellen 'Kunden' und 'Artikel' her.

- SQL = standardisierte Datenbank-Sprache für relationale Datenbanken
- Von SQL unterstützte Operationen (Auswahl)
 - Datenmanipulation
 - SELECT: Abfragen von Datensätzen
 - INSERT: Einfügen von Datensätzen
 - UPDATE: Verändern von Datensätzen
 - DELETE: Löschen von Datensätzen
 - Datendefinition
 - CREATE TABLE: Erstellen von Tabellen (Relationen)
 - ALTER TABLE: Verändern von Tabellen
 - Datenkontrolle
 - Constraints wie z.B. PRIMARY KEY: Sicherung von Integritätsbedingungen
 - GRANT, REVOKE: Management von DB-Nutzer-Rechten

- Deskriptive Anfrage, d.h. Beschreibung der gesuchten Ergebnismenge und nicht der Vorgehensweise, wie Ergebnismenge erstellt wird
 - Alle Kunden deren Name mit „A“ beginnt
- Vereinfachte Syntax nach MySQL (<http://dev.mysql.com/doc/refman/5.1/de/select.html>)

```
SELECT [DISTINCT] select_expr, ...  
  [FROM table_references  
  [WHERE where_condition]  
  [GROUP BY {col_name | expr | position} [ASC | DESC], ...]  
  [HAVING where_condition]  
  [ORDER BY {col_name | expr | position} [ASC | DESC], ...]  
  [LIMIT {[offset,] row_count | row_count OFFSET offset}]
```

Interpretation

- SELECT: Welche Attribute (Werte) sollen ausgegeben werden?
- FROM: Welche Tabellen werden verwendet?
- WHERE: Welche Bedingungen müssen die Datensätze erfüllen?
 - Beziehungen zwischen Tabellen (Fremdschlüssel) auch als Bedingung formulierbar (Tab1.PS = Tab2.FS)
- ORDER BY: Angabe der Sortierung
- LIMIT: Begrenzung des Ergebnisses (z.B. nur 100 Datensätze)

SQL-SELECT: Beispielanfragen

Kunde

ID	Name	Vorname	Straße	PLZ	Ort	Tel	email
11	Maier	Ulrich	Rosenweg 23	70599	Stuttgart	0711- 456 896	Umaier@gmx.com
12	Braun	Martina	Gartenstr. 12	70794	Filderstadt	0711 – 705661	m.braun@web.de

SELECT Name
FROM Kunde

Name

Maier

Braun

SELECT Name, Vorname
FROM Kunde

Name

Vorname

Maier

Ulrich

Braun

Martina

SELECT Name
FROM Kunde
WHERE ID = 11

Name

Maier

SELECT Name
FROM Kunde
WHERE Name LIKE 'A%'

Name

SQL-SELECT: Beispielanfragen (Join)

Kunde

ID	Name	Vorname	Straße	PLZ	Ort	Tel	email
11	Maier	Ulrich	Rosenweg 23	70599	Stuttgart	0711- 456 896	Umaier@gmx.com
12	Braun	Martina	Gartenstr. 12	70794	Filderstadt	0711 - 705661	m.braun@web.de

Bestellung

ID	Kunden_id	Artikel_id	Anzahl	Datum	Status
31	11	22	5	22.06.2001	Erledigt
32	12	21	1	04.01.2002	In Bearbeitung

```
SELECT Kunde.Name, Bestellung.Datum
FROM Kunde, Bestellung
WHERE Kunde.ID = Bestellung.Kunden_ID
```

Kunde.Nam e	Bestellung.Datu m
Maier	22.06.2001
Braun	04.01.2002

```
SELECT Kunde.Name, Bestellung.Datum
FROM Kunde, Bestellung
WHERE Kunde.ID = Bestellung.Kunden_ID
AND Status = "Erledigt"
```

Kunde.Nam e	Bestellung.Datu m
Maier	22.06.2001

SQL-SELECT: Beispielanfragen (Aggregation)

Bestellung

ID	Kunden_id	Artikel_id	Anzahl	Datum	Status
31	11	22	5	22.06.2001	Erledigt
32	12	21	1	04.01.2002	In Bearbeitung
32	12	22	2	07.06.2003	Erledigt

```
SELECT COUNT(ID)
FROM Bestellung
```

Count(Id)

3

```
SELECT Artikel_Id, SUM(Anzahl)
FROM Bestellung
GROUP BY Artikel_Id
ORDER BY Artikel_Id
```

Artikel_id

Sum(Anzahl)

21

1

22

7

- **MySQL** ist ein Datenbank Management System für relationale Datenbanken.
 - in kompilierter Form und auch als Source-Code frei verfügbar
 - läuft auf verschiedenen OS (Windows, Linux, UNIX, ...)
 - Installation und Administration sind relativ einfach
- Programmierschnittstellen zu wichtigsten Programmiersprachen, u.a. Java, C++, PHP, Perl
- Unterstützt grundlegende Sicherheitsmerkmale
 - Nutzer (authentifiziert durch Name und Passwort) haben definierte Rechte an DB-Objekten und können Rechte weitergeben
 - DB-Objekte: Datenbank, Tabelle, Attribut, Nutzer, ...
 - Rechte: Lesen, Verändern, Löschen, ...

PhpMyAdmin: Admintool für MySQL



- PhpMyAdmin ist ein in PHP programmiertes Administrationstool für MySQL.
- Mit PhpMyAdmin kann man alle wichtigen Arbeitsgänge durchführen:
 - Datenbanken und Tabellen erstellen und modifizieren
 - Datensätze eingeben, modifizieren und auswählen
 - Rechteverwaltung
 - ...
- XAMPP = komplette Serversoftware mit MySQL (inkl. phpMyAdmin), Apache, PHP und Perl
 - <http://www.apachefriends.org/xampp.html>

- Definition mit CREATE TABLE-Statement
 - Tabellenname und Schema
- Angabe der Attribute mit
 - Namen
 - Datentyp
 - numerisch (INTEGER, TINYINT, FLOAT, ...)
 - String (VARCHAR, TEXT, BLOB, ...)
 - Datum/Zeit (DATE, TIME, DATETIME, ...)
 - Constraints
 - PRIMARY KEY, FOREIGN KEY
 - Wertebereichseinschränkungen (z.B. $0 \leq \text{Alter} \leq 120$)
 - Index
- Technische Angaben
 - z.B. StorageEngine (MyISAM, InnoDB, ...)

- Indizes können für eine oder mehrere Tabellenspalten angelegt werden.
- Indizes sind sinnvoll bei Spalten, die oft als Suchkriterium benutzt werden.
- Sie beschleunigen Suchabfragen.
- Indizes benötigen Speicherplatz und verlangsamen das Einfügen von neuen Datensätzen.

DB-Anbindung an Programmiersprachen

- Open DataBase Connectivity (ODBC) = standardisierte Datenbankschnittstelle mit Anfragesprache SQL
- ODBC hat Programmierschnittstellen (APIs) für verschiedene Sprachen, u.a., Java, Visual Basic, C++
- Ermöglicht Absetzen von Datenbankabfragen aus einem Programm
- Kernproblem bei SQL-Einbettung in konventionelle Programmiersprachen: Abbildung von Tupelmengen auf die Variablen der Programmiersprache (Impedance Mismatch)
- Lösungsansatz: Cursor-Konzept
 - satzweise Abarbeitung von DBS-Ergebnismengen
- Unterstützung durch Objekt-Relationale Frameworks
 - Beispiel: Hibernate