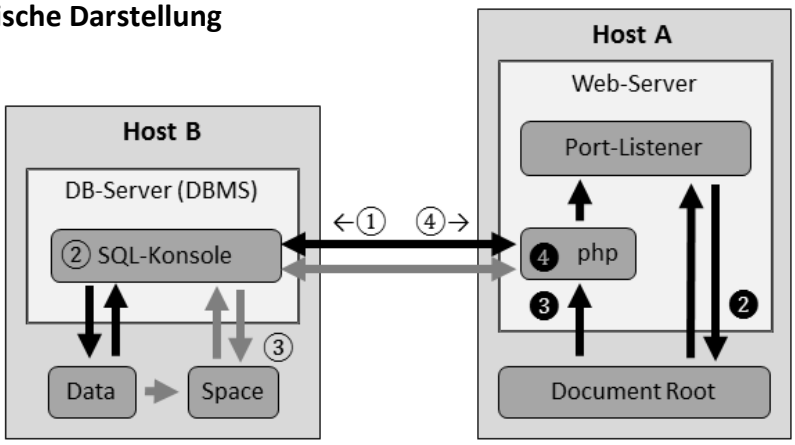
1. (L)AMP – Umgebung / schematische Darstellung

Voraussetzung ist die korrekte Installation eines Web-Servers mit PHP (Host A) und eines MySQL-Servers (Host B). Beide Dienste müssen nicht notwendiger Weise auf einem Computer installiert (aber gegenseitig erreichbar) sein. Initialisiert wird ein Datenbankzugriff durch ein Client-Request einer php-Ressource (siehe Umdruck „php-Grundlagen“) .

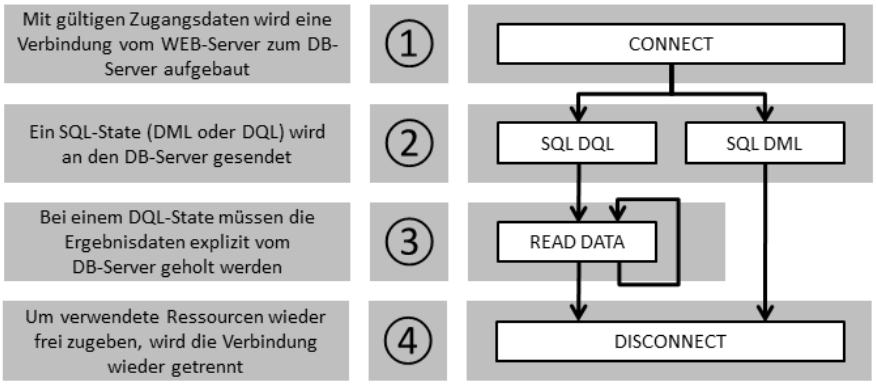


2. Kommunikations-Schnittstellen

Für den scriptgesteuerten Datenbankzugriff stand in php die (historisch ältere) **mysql-Schnittstelle** zur Verfügung. Dabei wurden alle Aufgaben durch Funktionen realisiert. Seit der php-Version 5.xx ist die **mysqli-Schnittstelle** (improved Extension) hinzugekommen, die wesentliche Erweiterungen (z.B. Unterstützung der OOP in php) beinhaltet. Welche Schnittstelle genutzt werden kann, hängt von der Konfiguration des WEB-Servers (implementierte Schnittstelle) ab und kann mit dem php - Befehl `phpinfo();` ermittelt werden.
Da die **mysql-Schnitstelle** (ab php Version 5.3) als deprecated gekennzeichnet wurde, sollte nach Möglichkeit **mysqli** verwendet werden.

3. Grundlegender Ablauf

Mit dieser Schnittstelle ergeben sich verschiedene Zugriffsarten (z.B.: mysqli-prozedural , mysqli-objektorientiert) die nachstehend weiter beschrieben werden. Die Ablaufschritte beim Datenbankzugriff sind bei allen Arten ähnlich. Es kann auch ein mehrfacher Zugriff (sequentiell und parallel) in einem Script realisiert werden.



4. Programmtechnische Realisierung

Grundlage für alle Zugriffsarten sind die drei Klassen deren Objektinstanzen meist automatisch erzeugt werden.

- **MySQLi** - allgemeine Aufgaben wie Verbindungsauf- und abbau und Datenbasisauswahl (O_{CON}).
- **MySQL_Result** - repräsentiert das Ergebnis von Abfragen und realisiert die Auswertung (O_{RES}).
- **MySQLi_STMT** - ist für die Verarbeitung von prepared Statments zuständig (O_{STM}).

Ausgangspunkt für die schematischen Syntax - Darstellungen sind die korrekten Zugangsdaten und SQL-States, die hier exemplarisch mittels php-Variablen übergeben werden.

Copyright by A.Rimbakowsky© (www.rimbakowsky.de)

`$host` = "hostname oder IP";	`$data` = "Datenbasis";
`$user` = "Benutzername";	`$sqlQ` = "SELECT * FROM tab";
`$pass` = "Password";	`$sqlM` = "UPDATE tab SET col = value WHERE …";

Bei allen Arten ist die Rückgabe der Funktion/Methode "SQL-State senden" in Abhängigkeit der Art des States (DQL oder DML) gleich. DML-States liefern nur einen Bool zurück, der Aufschluss über den Erfolg der Aktion (nicht die Anzahl der betroffenen Datensätze) gibt. DML-States liefern (im Erfolgsfall) eine Ressource -Adresse zurück, unter der auf das eigentliche Abfrage-Ergebnis (auch mehrfach) zugegriffen werden kann, sonst FALSE.

4.1. MySQLi Unterschiede der prozeduralen und OOP Schreibweise

MySQLi wurde entwickelt damit es sich in das objektorientierte Umfeld von PHP 5 einreiht. Die prozeduralen Funktionen wurden aus Gründen der Kompatibilität und eines einfachen Umstiegs beibehalten. Eine große Schwäche (Fehlerquelle) von MySQLi ist, dass man beides miteinander kombinieren kann. Das liegt daran, dass immer ein Objekt zurückgegeben wird.

Die meisten Funktionen beginnen mit dem Präfix **`mysqli_`** gefolgt von dem Bezeichner. Die Methoden beim OOP sind (meistens) ebenso eindeutig. So wird aus dem Präfix der **`Objektname`** mit dem **`->`** Operator.

prozedurale Funktionen	**objektorientierte Methoden**
`$result = mysqli_query( $con , $sql );`	`$result = $con -> query( $sql );`
Einige prozedurale Funktionen haben keine OOP Methode, sondern stehen als Eigenschaft zu Verfügung	
`$anzahl = mysqli_num_rows( $result );`	`$anzahl = $result -> num_rows;`

4.2. Prozeduraler Zugriff mit mysqli – Funktionen

Phasen	Syntax	Rückgabetyp
①	`$O_CON = mysqli_connect($host , $user , $pass [,$data] );` `[ mysqli_select_db( $O_CON , $data ); ]`	Object / [false] Boolean
②	`$O_RES = mysqli_query( $O_CON , $sqlQ | $sqlM);`	Object / [false]
③	`$DS = mysqli_fetch_array( $O_RES [, Result Typ] );`	Array
④	`mysqli_close( $O_CON );`	Boolean

4.3. Objektorientierter Zugriff mit mysqli - Methoden

Phasen	Syntax	Rückgabetyp
①	`$O_CON = new mysqli($host , $user , $pass [,$data] );` `[ $O_CON -> select_db( $data ); }`	Object / [false] Boolean
②	`$O_RES = $O_CON -> query( $sqlQ | $sqlM );`	Object / [false]
③	`$DS = $O_RES -> fetch_array( [Result Typ] )`	Array
④	`$O_CON -> close();`	Boolean

Result Typ	Beschreibung des fetch_array(Result Typ) Ergebnis
MYSQLI_ASSOC	Rückgabewert ist ein assoziativ indiziertes Array / z.B.: $feld["col_name"]
MYSQLI_NUM	Rückgabewert ist ein numerisch indiziertes Array / z.B.: $feld[3]
MYSQLI_BOTH	Rückgabewert ist ein assoziativ und numerisch indiziertes Array (Standard)

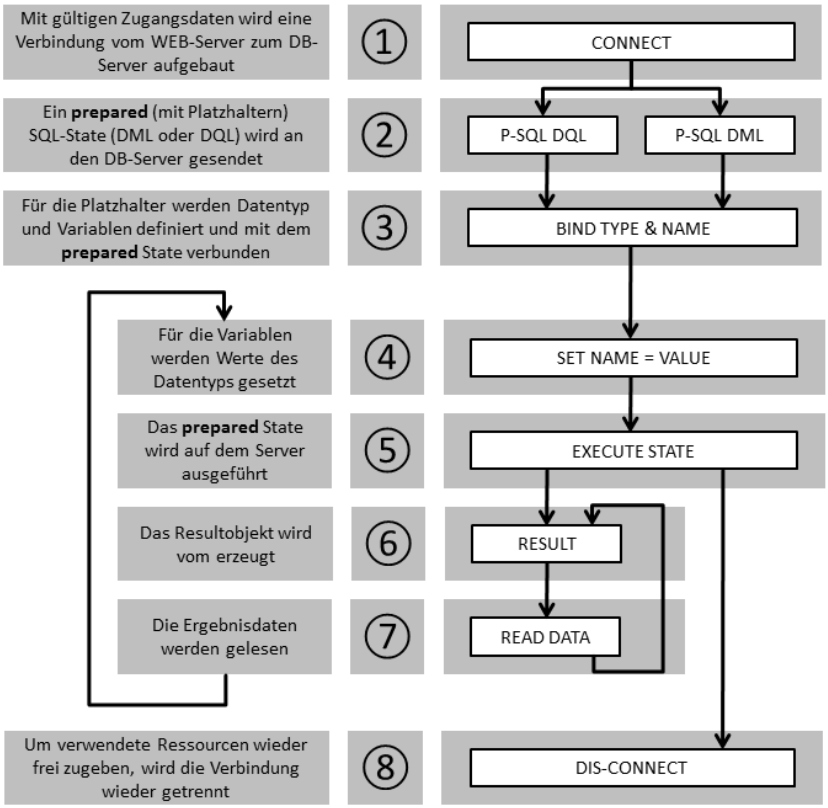
Copyright by A.Rimbakowsky© (www.rimbakowsky.de)

Prepared States

Bei der Verarbeitung von prepared States wird das auszuführende SQL-State in zwei Schritten an den Server übertragen. Im ersten Schritt wird das SQL-State mit Platzhaltern (**?**) für Parameter gesendet. Später werden dann nur die Parameter-Platzhalter durch typdefinierte Werte ersetzt.

Durch das Trennen von SQL-State und Parameter (meist Anwendereingabe) **ist eine SQL-Injection auf diesem Weg nicht mehr möglich!**

Diese States können folgendes Aussehen haben:



DML oder DQL prepared States	Datentypdefinition
`$PsqlM = "INSERT INTO tab(col1,col2) VALUES (?,?) ";`	**i** - Integer **s** - String
`$PsqlQ = "SELECT * FROM tab WHERE col = ? ";`	**d** - Double **b** - Blob

4.4. prozedurale Zugriff mit mysqli - Funktionen

Phasen	Syntax (prozedural)	Rückgabetyp
①	`$O_CON = mysqli_connect($host , $user , $pass , $data );`	Object / [false]
②	`$O_STM = mysqli_prepare($O_CON , $PsqlQ | $PsqlM );`	Object / [false]
③	`mysqli_stmt_bind_param ($O_STM,"Typ[en]",$Variable[n] );`	Boolean
④	`$Variable = Wert [, $Variablen = Wert ];`	
⑤	`mysqli_stmt_execute( $O_STM );`	Boolean
⑥	`$O_RES = mysqli_stmt_get_result($O_STM);`	Object / [false]
⑦	`$DS = mysqli_fetch_array( $O_RES , [Result Typ] ) ;`	Array
⑧	`mysqli_stmt_close($O_STM);` `mysqli_close($O_CON);`	Boolean

4.5. Objektorientierter Zugriff mit mysqli - Methoden

Phasen	Syntax (objektorientiert)	Rückgabetyp
①	`$O_CON = new mysqli($host , $user , $pass , $data );`	Object / [false]
②	`$O_STM = $O_CON -> prepare( $PsqlQ | $PsqlM );`	Object / [false]
③	`$O_STM -> bind_param( "Typ[en]" , $Variable[n] );`	Boolean
④	`$Variable = Wert [, $Variablen = Wert ];`	
⑤	`$O_STM -> execute();`	Boolean
⑥	`$O_RES = $O_STM -> get_result();`	Object / [false]
⑦	`$DS = $O_RES -> fetch_array( [Result Typ] )`	Array
⑧	`$O_CON -> close();`	Boolean

Copyright by A.Rimbakowsky© (www.rimbakowsky.de)