

Bem.: Das vorliegende Skript ist eine Mischung aus Vorlesung, Projektarbeiten und anderen Skripten.

1. Installation der benötigten Software

1.1 Apache (1.3.x) - Webserver (Win32)

Apache-Distribution herunterladen (<http://www.apache.org>) und in ein Verzeichnis installieren (z.B.: c:\apache)

Die Datei „*http.conf*“ (z.B.: c:\apache\conf\http.conf) wie folgt anpassen:

```
ServerName localhost
ScriptAlias /php3/ "c:/php3/"
DirectoryIndex index.html index.php3
AddType application/x-httpd-php3 .php3
AddType application/x-httpd-php3 .phtml (Dieser Eintrag hat bei mir Probleme gemacht!)
Action application/x-httpd-php3 /php3/php.exe
```

Bei Apache für Windows müssen alle Backslashes in einer Verzeichnis-Anweisung in Forward-Slashes (normaler Slash) konvertiert werden!

Anschließend den Apache als Dienst installieren (NT) bzw. in Autostart aufrufen lassen (95/98).

Nun sollte man unter „*http://localhost/*“ den Webserver erreichen.

1.2 PHP 4.0 (Win32)

1.2.1 Standardinstallation

PHP-Distribution herunterladen (<http://www.php.net>) und in ein Verzeichnis (z.B. c:\php3) entpacken.

Die Datei „*php3.ini-dist*“ ins Windows-Verzeichnis kopieren und in „*php3.ini*“ umbenennen und dann wie folgt anpassen:

```
extension_dir=c:\php3
doc_root=<htdocs-Verzeichnis der Apache-Installation>

mysql.default_port = 3306
mysql.default_host = localhost
mysql.default_user = root

extension=php3_mysql.dll (Wegen integriertem MySQL-Support eigentlich nicht nötig!)
```

Jetzt evtl. noch ein kurzes Testskript erstellen, z.B.:

```
<?php phpinfo()?>
```

und in einer Datei mit Namen „*test.php3*“ im Verzeichnis „*htdocs*“ der Apache-Installation abspeichern und dann über „*http://localhost/test.php3*“ aufrufen

1.2.2 PHP Base Library

Für datenbankbasierte Anwendungen mit Session-Tracking über Cookies bzw. URL-Codierung sowie serverseitigem Warenkorb empfiehlt sich die PHP Base Library.

Von <http://phplib.netuse.de/> downloaden (Version 6.1), entpacken, das `/php` Verzeichnis z.B. nach `/Apache/` verschieben und folgende Einstellungen in der `php3.ini` vornehmen:

```
auto_prepend_file = /Apache/php/prepend.php3
include_path      = ".;/Apache/php/"
```

Ohne den 1. Teil der Pathangabe (`.`) funktioniert phpMyAdmin nicht mehr!

Anschließend sollte man eine neue Datenbank für das gewünschte Projekt mit phpMyAdmin anlegen und das SQL-Skript `create_database.mysql` ausführen (ich habe das aus `/unsup` benutzt), wozu man sich ebenfalls phpMyAdmin bedient.

Für diese Datenbank, in der phplib die Session- und Userinformationen verwaltet, ist abschließend dann noch die `/Apache/php/local.inc` anzupassen. In der Klasse `DB_Poe` muß eingetragen werden:

```
class DB_Poe extends DB_Sql {
    var $Host      = "localhost";
    var $Database = "projekt";
    var $User      = "root";
    var $Password = "";
}
```

Sinnvollerweise verwendet man hier gleich einen anderen User als root. Zum Testen das mitgelieferte `/pages/` Verzeichnis samt Inhalt nach `/apache/htdocs/` kopieren und im Browser aufrufen ...

Autor für Abschnitt 1.2.2: thomas@schulz.net - Vielen Dank!

1.3 MySQL (Win32)

MySQL Version für Win32 herunterladen (<http://www.tcx.se>) und installieren (z.B.: `c:\mysql`).

Die Windows-Version ist Shareware, deshalb bitte die Lizenzbedingungen beachten! Die Funktionen sollten jedoch die gleichen wie unter Linux sein.

Anschließend laut Anleitung als Dienst unter NT einrichten: „`c:\mysql\bin\mysqld --install`“ oder unter 95/98 `mysqld-shareware.exe` in der Autostart aufrufen.

1.4 phpMyAdmin

Aktuelle Version unter <http://www.phpwizard.net/phpMyAdmin/> herunterladen und in ein Verzeichnis unterhalb von `htdocs` entpacken (z.B. `phpMyAdmin`).

phpMyAdmin über <http://localhost/phpMyAdmin/> aufrufen. Eventuelle Anpassungen können in der `config.inc.php3` erfolgen.

Auch hier Danke für den Tipp an: thomas@schulz.net

2. MySQL

2.1 Datenbank ist vorhanden - Quickstart

Die wichtigsten Programme des Packets:

mysql	Der Datenbankserver
mysqladmin	Admin-Tool
mysqldump	Ausgeben von Datenbankinhalten

2.2 Die wichtigsten Befehle im Überblick

2.2.1 Einfach anmelden

```
c:\> mysql -h<hostname> -u<user> -p<password> --database=<datenbank>
```

2.2.2 Textdatei mit SQL-Befehlen in MySQL übertragen

```
c:\> mysql -h<hostname> -u<user> -p<password> --database=<datenbank> <<datei.sql>
```

2.2.3 Datenbank auswählen bzw. wechseln

```
mysql> use <datenbank>;
```

2.2.4 Alle Tabellen in der Datenbank anzeigen

```
mysql> show tables;
```

2.2.5 Wie ist eine bestimmte Tabelle aufgebaut?

```
mysql> show columns from <tabelle>;
```

2.2.5 Wie lege ich eine neue Datenbank an?

```
c:\> mysqladmin create <datenbank>
```

Bem.: Alle Befehle in der Kommandozeile unter Windows können mit gleichem Syntax unter Linux in der Shell verwendet werden!

2.3 SQL-Befehle - eine kurze Übersicht der wichtigsten Befehle (In MySQL nicht vorhandene Befehle sind mit einem * gekennzeichnet.)

```
ALTER DATABASE <Datenbankname>;
```

Ändert die Größe bzw. Einstellungen an der Datenbank (Syntax ist je nach Datenbanksystem unterschiedlich).

```
ALTER USER <Benutzername>;
```

Ändert Systemeinstellungen (z.B.: Passwort) des Benutzers.

```
BEGIN TRANSACTION <Transaktionsname>; (*)
```

Markiert den Beginn einer Transaktion - die Transaktion ist beendet wenn sie beendet (`COMMIT TRANSACTION`) oder abgebrochen (`ROLLBACK TRANSACTION`) wird.

```
CLOSE CURSOR; (*)
```

Schließt einen Cursor und löscht dessen Daten - mit `DEALLOCATE CURSOR` wird der Cursor vollständig entfernt.

```
COMMIT TRANSACTION; (*)
```

Führt die Transaktion aus - alle seit `BEGIN TRANSACTION` ausgeführten Operationen werden auf der Datenbank ausgeführt.

```
CREATE DATABASE <Datenbankname>
```

Erstellt eine neue Datenbank.

```
CREATE TABLE <Tabellenname>( <Feld1> <Datentyp> [NOTNULL],  
                             <Feldn> <Datentypn> [NOT NULL] );
```

Erstellt eine neue Tabelle in einer Datenbank.

```
CREATE USER <Benutzername>
```

Erstellt ein neues Benutzerkonto.

```
CREATE VIEW <Sichtname> [ (Spalte1, Spalte2, ...) ] AS  
SELECT      <Tabellenname>.Spaltennamen  
FROM        <Tabellenname>;
```

Erzeugt eine Sicht (virtuelle Tabelle) - die Sicht kann nach der Erzeugung wie eine normale Tabelle gehandhabt werden.

```
DEALLOCATE CURSOR <Cursorname>; (*)
```

Entfernt den Cursor vollständig aus dem Speicher und gibt seinen Namen wieder frei. Bevor ein Cursor entfernt wird, sollte er mit `CLOSE CURSOR` geschlossen werden.

```
DECLARE <Cursorname> CURSOR FOR <SelectAnweisung>; (*)
```

Erzeugt aus der Select-Abfrage einen neuen Cursor. Mit `FETCH` kann durch die Daten gescrollt werden, bis die Variablen geladen sind. Der Cursor schaltet dann zum nächsten Datensatz weiter.

```
DROP DATABASE <Datenbankname>;
```

Löscht eine Datenbank (incl. aller Daten und physikalischer Struktur)

```
DROP INDEX <Indexname>;
```

Entfernt einen Index von einer Tabelle.

```
DROP PROCEDURE <Prozedurname>; (*)
```

Löscht eine gespeicherte Prozedur aus der Datenbank.

```
DROP TABLE <Tabellenname>;
```

Löscht eine Tabelle aus der Datenbank.

```
DROP TRIGGER <Triggernamen>; (*)
```

Entfernt einen trigger aus der Datenbank.

```
DROP VIEW <Sichtname>; (*)
```

Entfernt eine Sicht aus der Datenbank;

```
EXECUTE [@Rückgabestatus = ] <Prozedurname>  
[[@Parametername = ] <Wert> |  
[@Parametername = ] @ Variable [OUTPUT]...] (*)
```

Führt eine gespeicherte Prozedur aus.
Parameterübergabe ist möglich - wird das Schlüsselwort `OUTPUT` angege

```
FETCH <Cursorname> [into fetch_target_list]; (*)
```

Lädt den Inhalt der Cursordaten in die bereitgestellten Programmvariablen. Nach dem laden der Variablen, scrollt der Cursor zum nächsten Datensatz.

```
FROM <Tabellenreferenz> [, <Tabellenreferenz> ...];
```

Legt fest, welche Tabellen verwendet und/oder verknüpft werden.

```
GRANT <Rolle> TO <Benutzer>;  
GRANT <Systemprivileg> TO <Benutzer | Rolle | PUBLIC>;
```

Gewährt ein Privilig oder eine Rolle, die mit `CREATE USER` erzeugt wurde, an einen Benutzer.

```
GROUP BY <Gruppenliste>
```

Gruppirt Zeilen mit einem bestimmten , gleichen Spaltenwert

```
HAVING <Bedingung>
```

Schränkt die Auswahl auf Gruppen ein, die der Suchbedingung genügen (nur mit `GROUP BY` gültig!)

```
INTERSECT
```

Zurückgeben aller gemeinsamen Elemente aus den Ergebnismengen zweiter `SELECT`-Anweisungen.

```
ORDER BY <Sortierliste>;
```

Ordnet die zurückgelieferten Werte nach der/den angegebenen Spalten.

```
ROLLBACK TRANSACTION; (*)
```

Bricht alle Aktionen ab, welche innerhalb einer Transaktion (seit `BEGIN TRANSACTION`) ausgeführt wurden.

```
REVOKE <Rolle> FROM Benutzer;  
REVOKE {Objektprivileg | ALL [Privileges]}  
      [, {Objektprivileg | ALL | [Privileges]} ] ...  
ON    [Schema.]Objekt  
FROM  {Benutzer | Rolle | PUBLIC}  
      [, {Benutzer | Rolle | PUBLIC}] ...
```

Entfernt einen Datenbankprivileg, einen Systemprivileg oder eine Rolle von einem Benutzer.

```
SELECT [DISTINCT ALL]
```

Leitet alle Operationen zum Abrufen von Daten ein

`DISTINCT` spezifiziert eindeutige Werte und verhindert Duplikate.

`ALL` erlaubt Duplikate (Vorgabewert!)

```
SET TRANSAKTION (READ ONLY | USE SEGEMNT); (*)
```

Erlaubt dem Benutzer den Beginn einer Transaktion festzulegen. Die Option `READ ONLY` sperrt eine Gruppe von Datensätzen bis zum Abschluß der Transaktion, um zu verhindern, daß die Daten nicht geändert werden.

```
UNION
```

Gibt alle Elemente aus zwei `SELECT`-Anweisungen zurück.

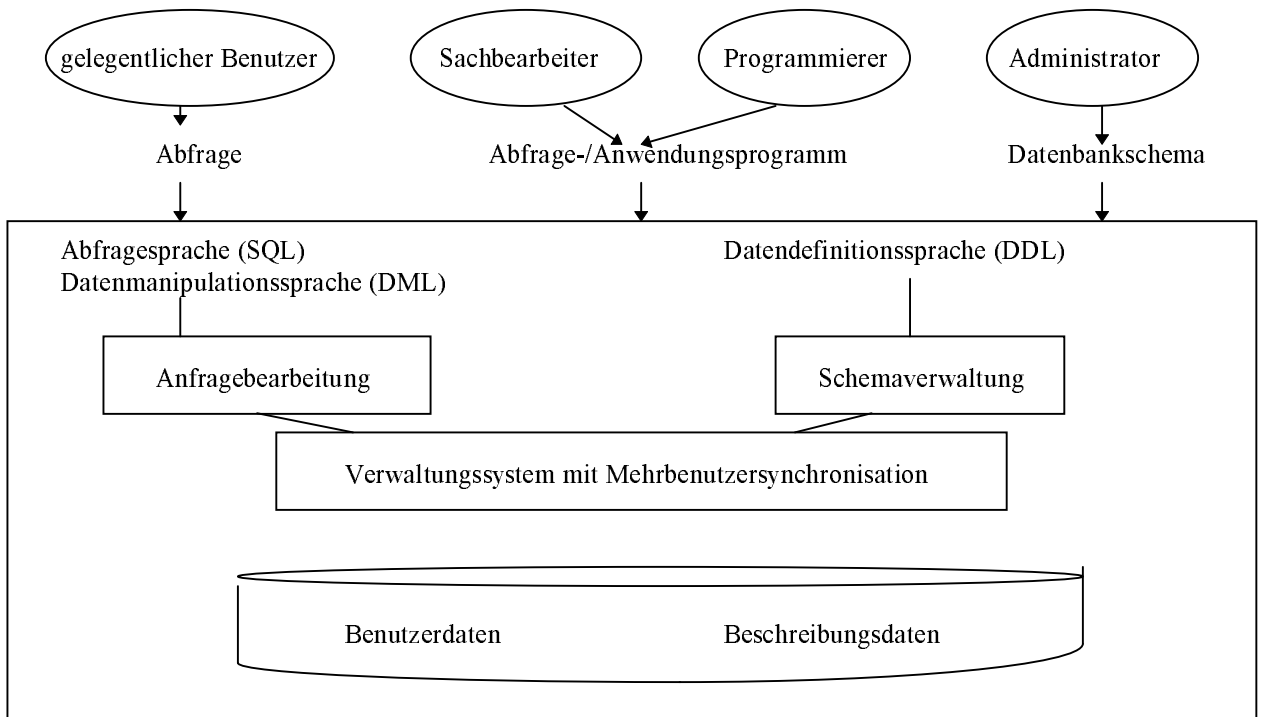
```
WHERE <Suchbedingung>;
```

Schränkt eine Abfrage auf die Werte ein, welche der Suchbedingung genügen.

3. Datenbanksysteme - Theorie

3.1 Grundsätzliches zum Aufbau eines Datenbanksystems

- Speicherung von Daten und Metadaten (Strukturen, Beziehungen, Zugriffsvariablen)
- Sprache zur Definition (DDL = Data Definition Language), zur Manipulation (DML) und Abfrage (SQL, API)



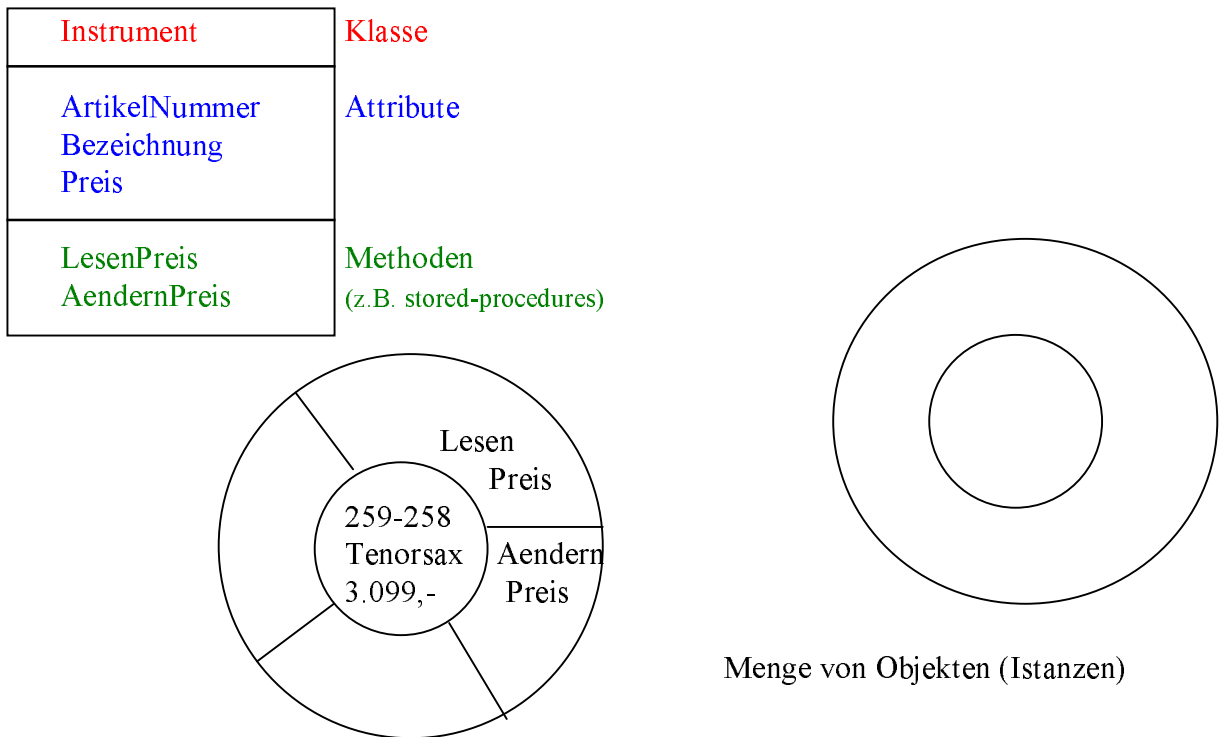
- Schemaverwaltung (-> Methoden)
Mit Hilfe der DDL wird pro Anwendungsbereich ein DB-Schema definiert.
- Abfragebearbeitung
Alle Anfragen werden zunächst optimiert und anschließend als DB-Aufrufe abgearbeitet.
- Mehrbenutzersynchronisation
Gleichzeitiger Zugriff mehrerer Prozesse im Schreibmodus ist zu synchronisieren
- Datenbanksysteme

HDBS	hierarchische DBS	z.B.: DNS-Service
NDBS	Netzwerk DBS	
RDMS	relationale DBS	(Nutzen zur Verwaltung ausschließlich Tabellen und DDL / DML in Verbindung mit SQL '92)
ORDBS	objektrationales DBS	(SQL '99 - kennt einige Elemente der Objektorientierung -> OO Schnittstellen für relationale DBS)
OODBS	objektorientiertes DBS	(Alle Daten werden in Objektform verwaltet. Die Abfrage ist objektorientiert)

3.2 Was sind Objekte?

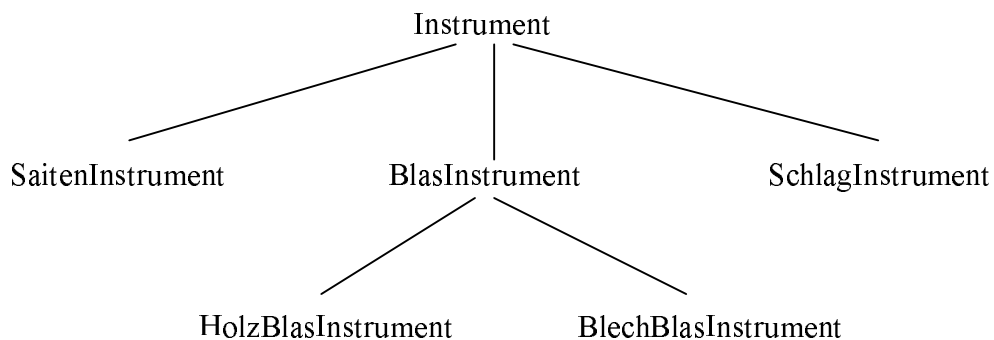
- Grundbausteine, aus welchen OO-Anwendungssysteme aufgebaut sind
- Eigenständige Einheiten, die über Struktur (Daten, Attribute) und Verhalten (Methoden) verfügen
- Objekte mit gleicher Struktur und gleichem Verhalten bilden Klassen

<Klasse als Abstraktion von Objekten>



- Kapselung - Methoden bilden die sichtbare Schnittstelle für gekapselte Daten
 - unzulässige Zustandsänderungen werden unterbunden
- Vererbung - Grundprinzip bzgl. einer Klassenhierarchie

<Vererbungseigenschaften bei Klassenhierarchien>

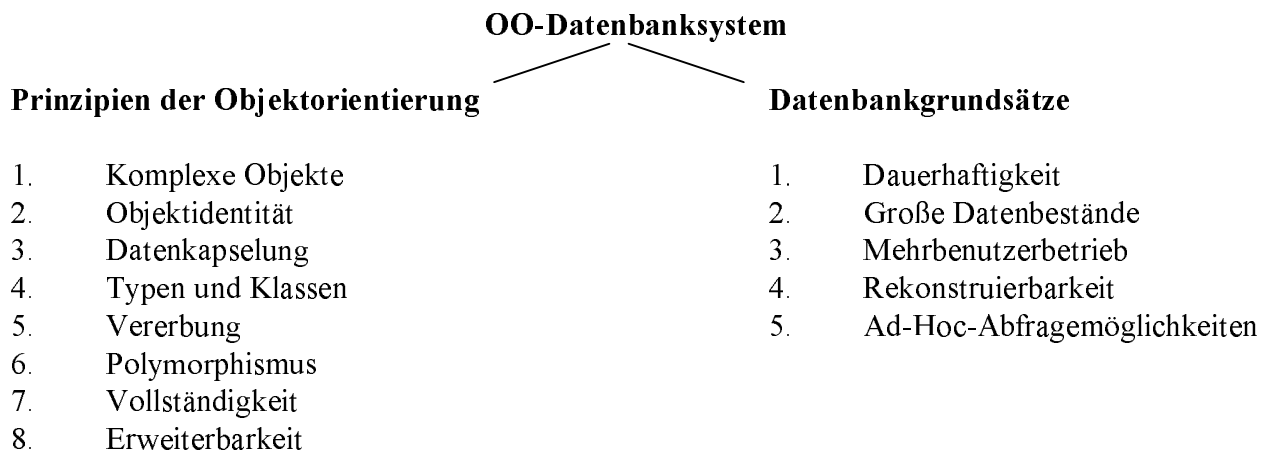


z.B.: Die Klasse „Blasinstrument“ erbt alle Attribute und Methoden der Superklasse Instrument und definiert weitere Attribute und Methoden

3.3 Eigenschaften eines OO-DBS

Welche Eigenschaften muß ein OO-DBS zwingend unterstützen?

Definition: Unter einem OO-DBS versteht man ein Softwaresystem, das die folgenden 8 Regeln der Objektorientierung und 5 Prinzipien der Datenhaltung erfüllt.



Regel 1: komplexe Objekte

„Attribute können auch Objekte sein“

„OODBS unterstützt komplexe Objekte, deren Attribute selbst auch Objekte sein können. Atomare Attributwerte können elementare Datentypen (int, char, boolean) oder Mengen, Listen oder ganze Objekte umfassen.“

Regel 2: Objektidentität

„Jedes Objekt trägt eine eindeutige Signatur“

„Die Identifikation muss mind. datenbankweit oder sogar systemweit eindeutig und unveränderbar sein. Sie muss unabhängig vom Speicherort sein!“

Regel 3: Datenkapselung

„Verbergen von Implementierungsdetails“ (nach aussen)

„Abschirmung der Attribute durch Methoden“

Regel 4: Typen und Klassen

Typen: klassische Datentypen und Binärobjekte

Klassen: Objekte gleicher Struktur und gleichen Verhaltens

Die Menge der vorhandenen Instanzen kann folglich variieren, nicht so der Wertebereich eines Typs zur Programmlaufzeit.

Regel 5: Vererbung

„Vererbung von Struktur und Verhalten **muss möglich sein**“

„Hinzufügen von Attributen und Methoden **in der Unterklasse möglich**“

„Subklasse (Unterklasse) kann Attribute und Methoden von übergeordneten Klassen Oberklassen erben“

Regel 6: Polymorphismus

„Gleichnamige Methoden unterschiedlicher Klassen mit unterschiedlichen Implementierungen“

„Methoden in unterschiedlichen Klassen können auf Objekte derjenigen Klasse unterschiedlich (weil abweichend implementiert) angewendet werden - Methoden mit dem selben Namen können überschrieben werden“

Regel 7: Vollständigkeit

„Vollständigkeit des OO-DBS“

„d.h. die zugehörige SB-Sprache ist berechnungsvollständig“

„d.h. jede DB-Operation kann durch die DB-Sprache ausgedrückt werden“

Regel 8: Erweiterbarkeit

„Benutzerdefinierte Klassen und Typen **müssen möglich sein (Unterscheidung nach aussen zwischen System- und Benutzerklassen/typen ist nicht möglich)**“

Prinzip 1: Dauerhaftigkeit, Persistenz

„Der DB-Zustand muß solange gewährleistet sein, bis Daten vom Anwender bewußt geändert werden“

Prinzip 2: grosse Datenbestände

„Sekundärspeicher, Bufferpools, Indexmechanismen und Datenclustering müssen bereitgestellt werden, **um grosse Datenbestände zu bearbeiten**“

Prinzip 3: Mehrbenutzerbetrieb

„Konkurrierende Prozesse müssen korrekt abgearbeitet werden“

„Transaktionskonzept **ist notwendig (Konsistenz muss garantiert werden!)**“

Prinzip 4: Rekonstruierbarkeit

„Wiederanlaufverfahren zur DB-Restaurierung im Fehlerfall werden bereitgestellt“

Prinzip 5: ad-hoc-Abfrage

„Bereitstellung deklarativer Sprachkonstrukte zur ad-hoc (sofortigen; nicht zeitversetzten) Auswertung der Datenbestände“

„Unterstützung aller Eigenschaften des Datenbankmodells (Mächtigkeit)“

Zwischenkapitel: Nutzung in der Praxis

Bei der OO-Entwicklung werden 3 Ebenen unterschieden:

- **View:** Mensch-Maschine Schnittstelle (Präsentation von Objekten - grafische Benutzerschnittstelle)
- **Control:** Entwicklungsumgebung (Bearbeitung von Objekten mit Programmiersprache)
- **Model:** Datenhaltung (längerfristige Aufbewahrung von Objekten)

(näheres siehe Literaturverzeichnis!)

3.4 Objektorientierte Modellierung

Analyse und Entwurfsmethoden sind für die Gestaltung von OO-Informationssystemen unumgänglich.

Analyse: ist die nachvollziehbare und widerspruchsfreie Beschreibung des Problembereichs.
(reine Anforderungen)

Entwurf: ist die nachvollziehbare und widerspruchsfreie Beschreibung des informationstechnischen Lösungsbereichs.

OO-Analyse- und Entwurfsmethoden modellieren - statische Struktureigenschaften
- dynamische Zustandsänderungen

„Objektstruktur und Objektverhalten“

Die Grundelemente des Objektmodells sind:

Objekte sind individuelle Ausprägungen (Instanzen) ganzheitlicher Dinge (Entity) oder Konzepte des Problembereichs

- Objektidentifikator
- Attributwerte
- zugehörige Methoden

Klassen - Intension (Beschreibung) (Attribute und Methoden)
- Extension (Menge der Ausprägungen)

Assoziationen sind Beziehungen zwischen Klassen

- Generalisierung (Verallgemeinerung)
- Aggregation (Zusammensetzung)

Vererbung wird durch die explizite Modellierung der Generalisierung ermöglicht
d.h. Attribute und Methoden können in anderen Klassen genutzt (vererbt) werden

Nachrichten können zwischen den Objekten ausgetauscht werden.
Erhält ein Objekt eine Nachricht, so führt es eigene Methoden aus und/oder verschickt neue Nachrichten.
Durch Nachrichten modelliert man das dynamische Verhalten der Klasse/Objekte.

3.5 Objektindikatoren

Eindeutige Identifikatoren (OID) werden mit Hilfe von Surrogaten (Stellvertretern) gewährleistet.

Surrogate sind unabhängig

- von der Struktur der Attributwerte
- vom aktuellen Originalzustand
- von der Speicherungsform

Das DBS garantiert automatisch für die eindeutige Identifizierbarkeit eines neuen Objekts (egal wie das Objekt aussieht und wo es erzeugt wird) durch eine Art eindeutige (systemweit und u.U. sogar weltweit) Objekt ID (Surrogat). Oder anders: systemweiter, eindeutiger Objektschlüssel.

allgemeine Eigenschaften:

- Eindeutigkeit
- Unveränderbarkeit
- Strukturunabhängigkeit
- Ortsunabhängigkeit

Surrogate sind für den referenzbasierten Aufbau von Beziehungen nutzbar.

Fragestellung: „Was ist besser?“ - „Surrogat oder benutzerdefinierter Schlüssel?“

Benutzerdefinierte Schlüssel:

- basieren auf Attributen (ArtikelNr, KundenNr, usw.)
- dienen zum Auffinden spezieller Objekte in einer Objektmenge
- stehen unter Benutzerkontrolle
- stehen unter Benutzerverantwortung
- werden von OO-Datenbanksystemen als Objekteigenschaft interpretiert

Referenzielle Integrität wird bei Nutzung von OID's (Surrogaten) automatisch gewährleistet.

3.6 Klasseigenschaften

Es gibt Klassenattribute (CA) und Klassenmethoden (CM)

Beispiel:	Name:	Instrument
	Attribute:	artikelNummer bezeichnung preis anzahl_instrumente (CA)
	Methoden:	lesenPreis aendernPreis erstellenListe (CM) anpassenZaehler (CM)

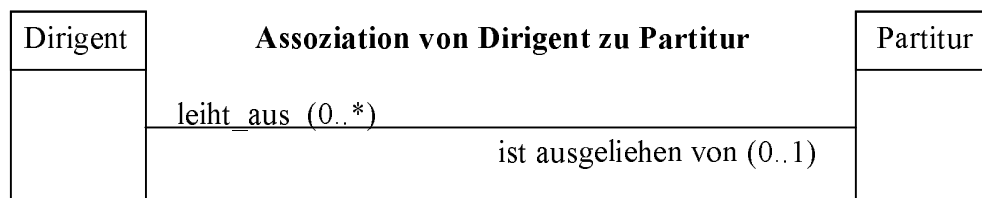
Vordefinierte Attribute (z.B.: OID) und Methoden (z.B.: Konstruktor, Destruktor) werden im Objektmodell nicht explizit angegeben.

3.7 Beziehungskonzepte

3.7.1 Assoziationen

Beziehungen zwischen Objekten/Klassen werden charakterisiert durch: - Semantik (**Beschreibung**)
- Mächtigkeit

<Bild: Zwei Assoziationen bilden eine Beziehung>



Assoziationstypen:	einfach:	1	(entspricht 1:1 bei relationalen DB)
	konditionell einfach:	0..1	
	mehrfach:	1..*	
	konditionell mehrfach:	0..*	

Alle Assoziationstypen werden zur Beschreibung von Beziehungen genutzt.

<Bild: einfach-einfach Beziehung>



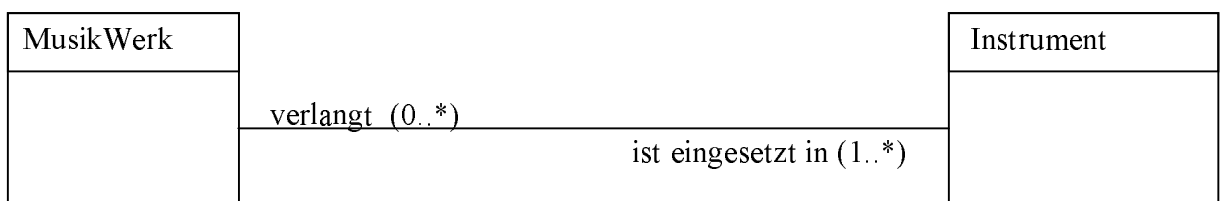
Ein Musikfestspiel verpflichtet höchstens in Festspielorchester.
 Ein Festspielorchester spielt exklusiv für genau ein Musikfestspiel.

<Bild: komplex-einfache Beziehung>



Ein Musikhaus hat mehrere Musikabteilungen.
 Eine bestimmte Musikabteilung gehört zu genau einem Musikhaus.

<Bild: komplex-komplexe Beziehung>



Ein Musikwerk verlangt kein, ein oder mehrere Instrumente.
 Ein Instrument ist eingesetzt in mehreren Musikwerken.

Beziehungstypen: einfach-einfach
 komplex-einfach
 komplex-komplex

wobei: einfach $\hat{=}$ 1; 0..1 und komplex $\hat{=}$ 1..*; 0..*

3.7.2 Beziehungsklassen

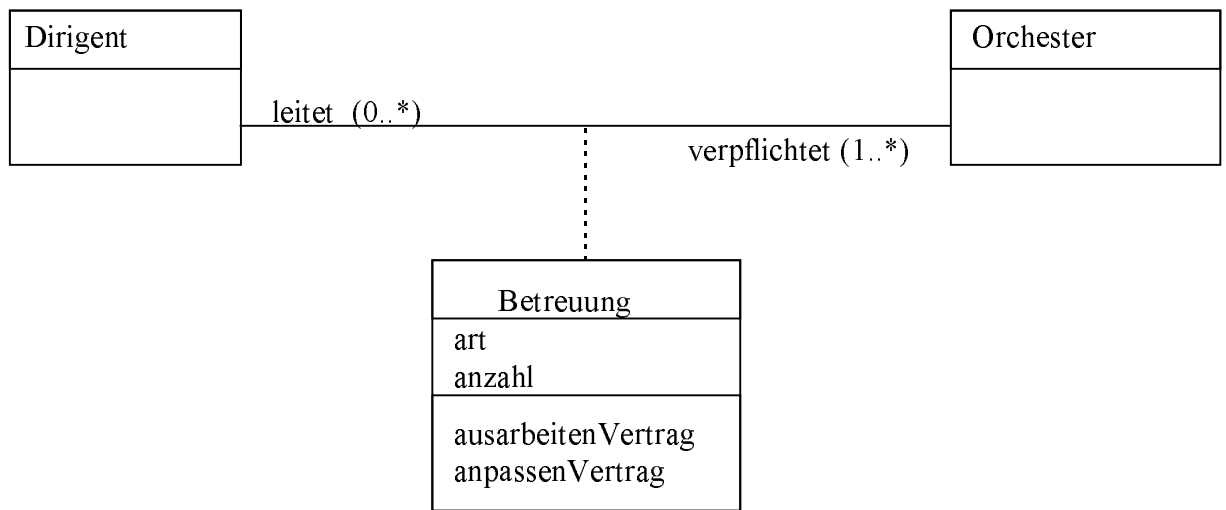
Beziehungen können über beschreibende Attribute und Methoden verfügen.

-> explizite Beziehungsklassen

Beziehungsklassen

- sind von mehr als einer Klasse abhängig
- verfügen über Beziehungsmethoden zum Zugriff auf Beziehungsattribute
- werden durch gestrichelte Linien an der entsprechenden Beziehung verankert

<Bild 1: Beziehungsklasse>



Die Beziehungsklasse „Betreuung“ definiert die Beziehung zwischen den Klassen Dirigent und Orchester

Beziehungsattribute	art:	{ Hausdirigent, Gastdirigent }
	anzahl:	Anzahl der Verpflichtungen pro Spielzeit

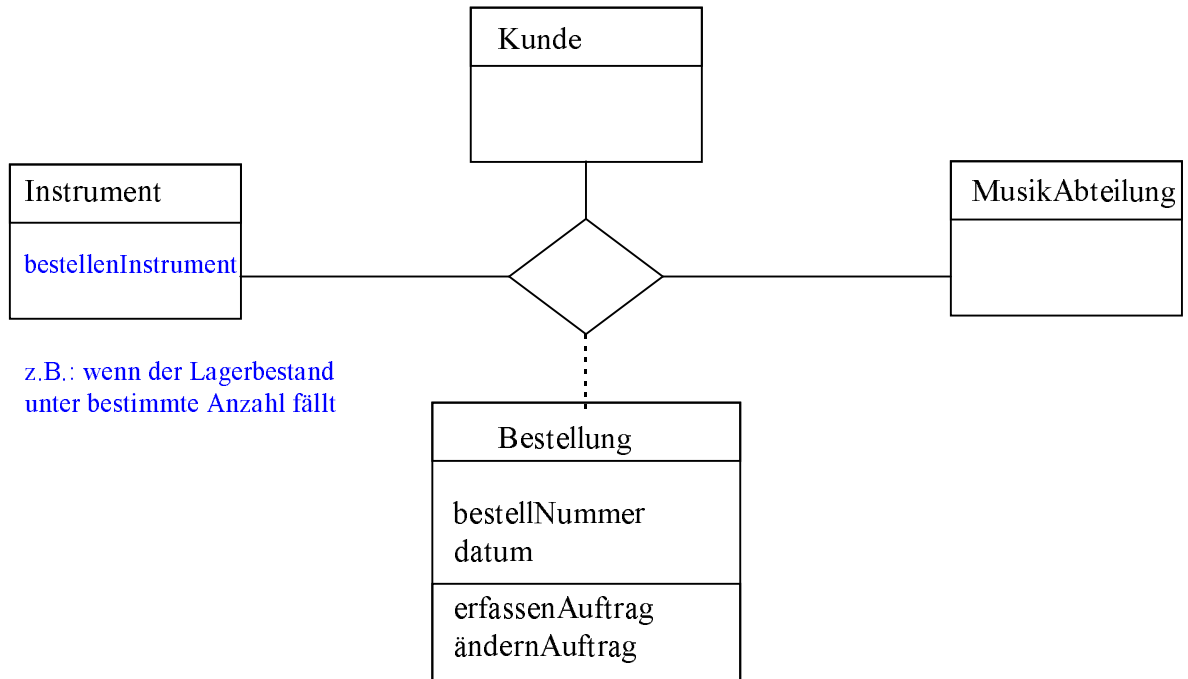
Beide Attribute können weder der Klasse „Dirigent“, noch der Klasse „Orchester“ zugeordnet werden!

Beziehungsmethoden (Regel: alle Methoden beginnen mit einem Verb)

ausarbeitenVertrag: legt fest, welcher Dirigent mit welchem Orchester eine Verpflichtung eingeht.
(erstellenVertrag)

anpassenVertrag: dient zum Ändern von Verträgen

<Bild 2: Beziehungsklassen zwischen mehr als 2 Klassen>

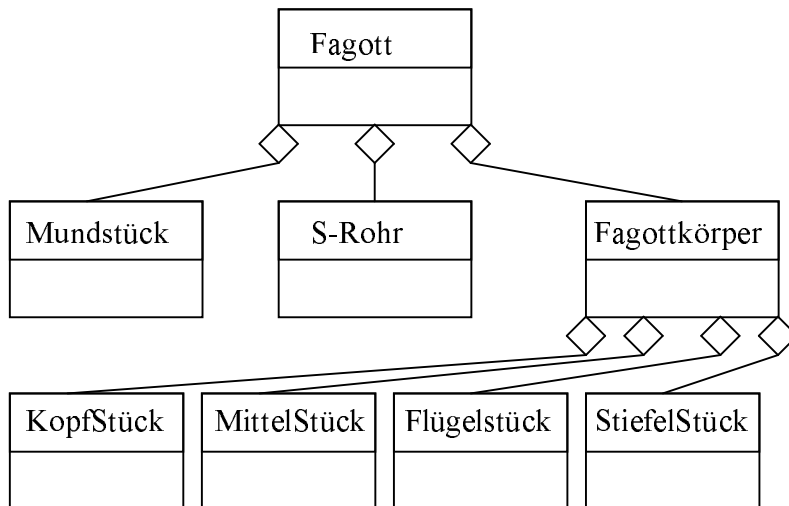


Beziehungsklassen können zwischen mehr als zwei Klassen eine Beziehung beschreiben.
(Eine Beziehung besteht aus mehreren (mindestens jedoch zwei) Assoziationen)

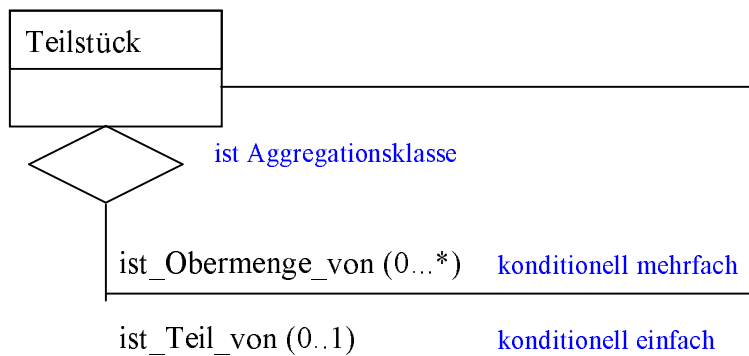
3.7.3 Aggregation

Eine Aggregationsklasse wird aus unterschiedlichen Komponentenklassen gebildet. Grafisch wird die Aggregationsbeziehung durch eine Raute visualisiert.

<Bild 3: „Ist Teil von Beziehung“>



<Bild 4: Rekursive Definition von Stücklisten>



Aggregation ermöglicht das kombinieren von Teilobjekten zu zusammengesetzten Objekten.

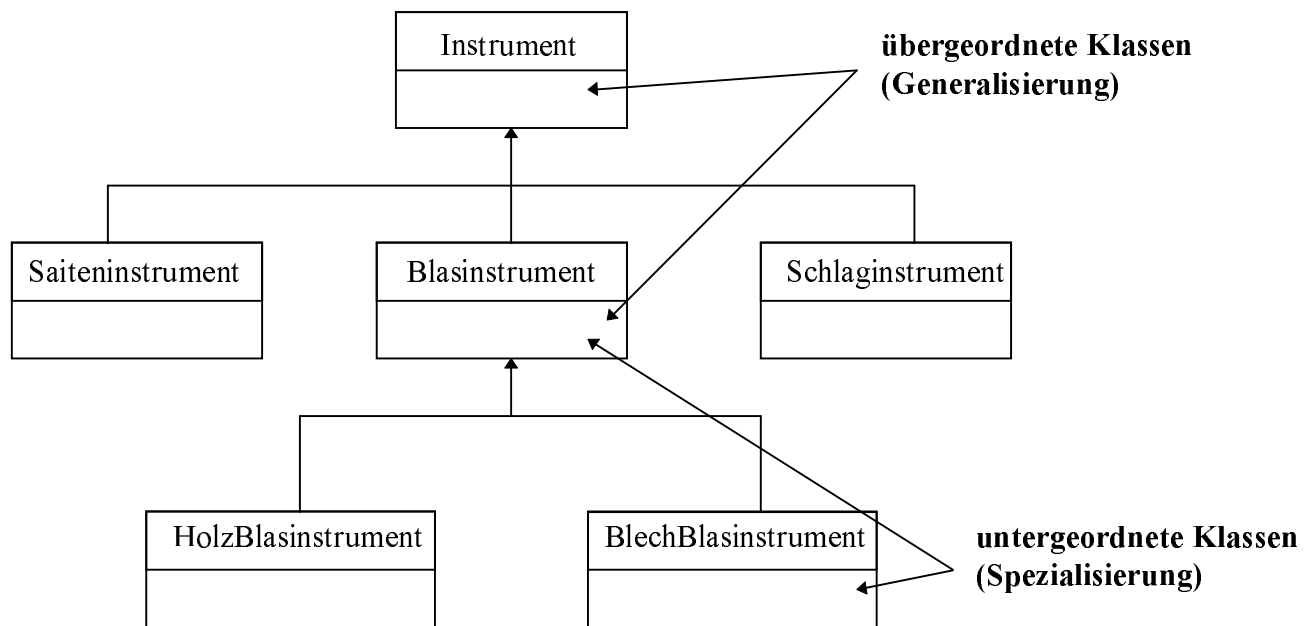
Es gibt folgende Aggregationsbeziehungen: `ist_Teil_von`
`ist_Obermenge_von`

Eine Stückliste ist nichts anderes als eine rekursive Aggregationsklasse. Jedem Teilstück ist ein Teilstück übergeordnet (Ausnahme: oberstes Teilstück)

3.8 Vererbung

3.8.1 Generalisierung

- Verallgemeinerung von Klassen in einer hierarchischen Beziehungsstruktur



- Verallgemeinerung erfolgt in übergeordnete Klassen, Spezialisierung in untergeordnete Klassen

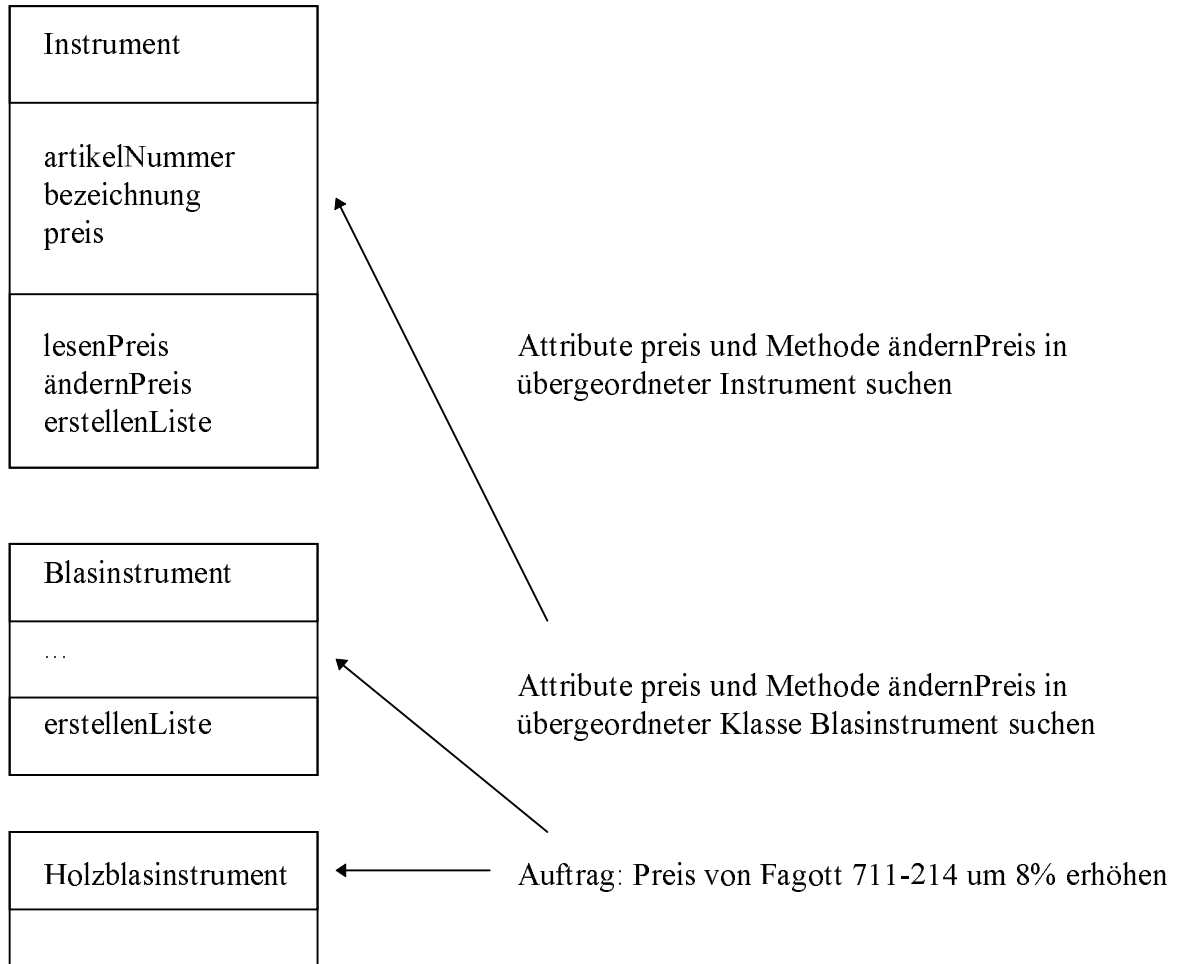
- Eine Generalisierungsbeziehung zwischen zwei Klassen besteht, wenn die übergeordnete Klasse gemeinsame Attribute und Methoden besitzt.

- Realisierung durch Vererbung

3.8.2 Vererbung

Unter Vererbung (inheritance) versteht man die Tatsache, dass Attribute und Methoden einer übergeordneten Klasse in einer untergeordneten Klasse verfügbar sind.

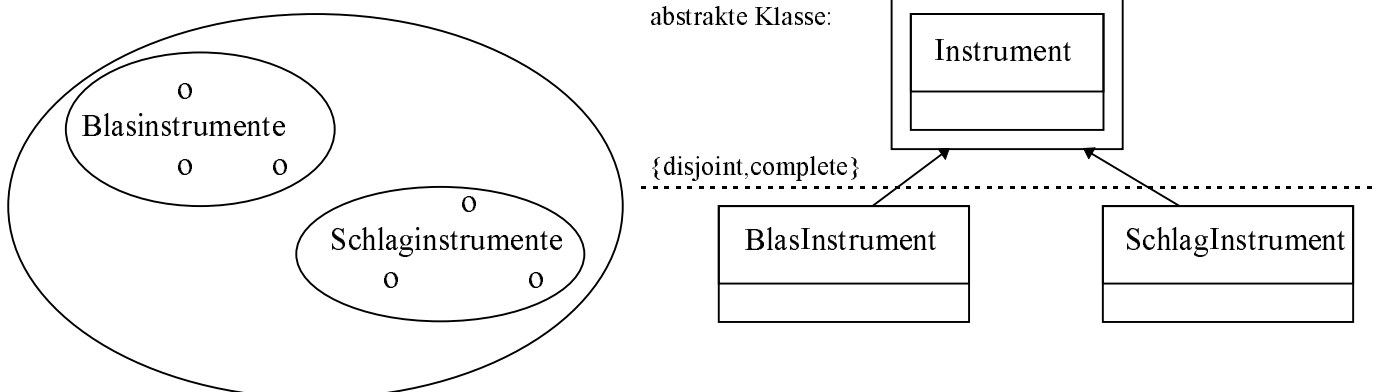
=> Wiederverwendbarkeit von Datenstrukturen und Methoden mit der Möglichkeit des Überschreibens.



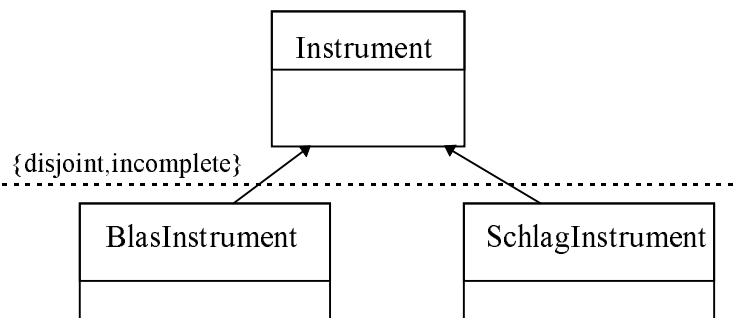
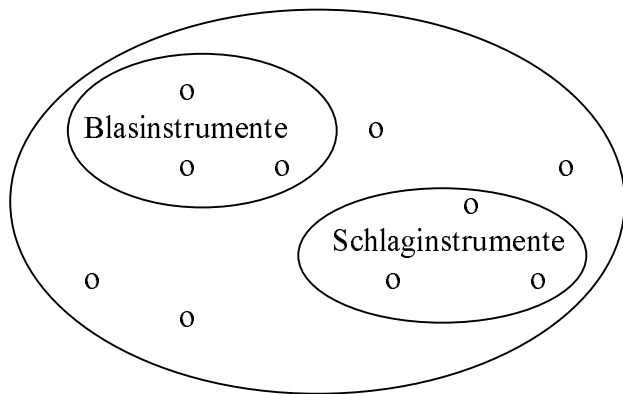
- einfache Vererbung: es gibt nur eine übergeordnete Klasse
- mehrfache Vererbung: es gibt mehrere übergeordnete Klassen

3.8.3 Vererbungsstrukturen

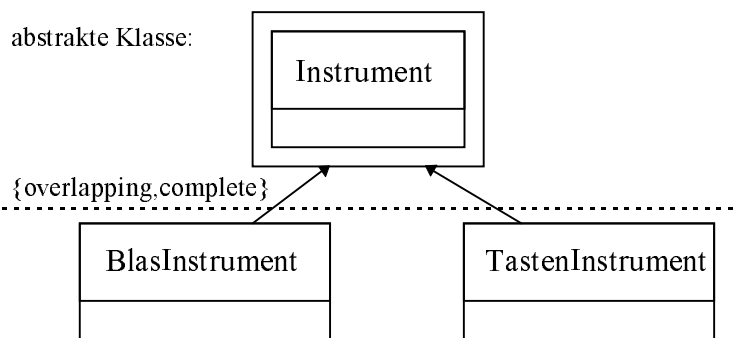
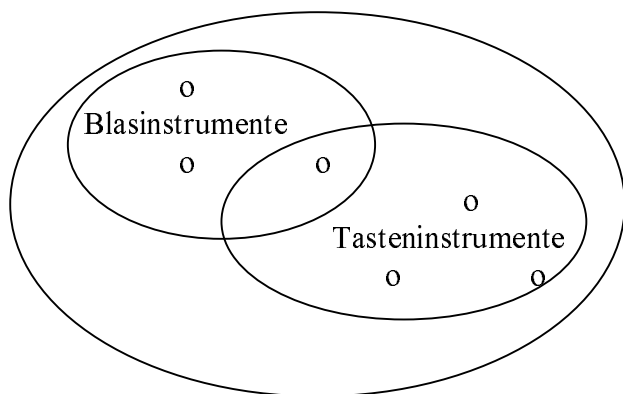
- **disjunkte und vollständige Überdeckung** (der Spezialisierungsmengen)



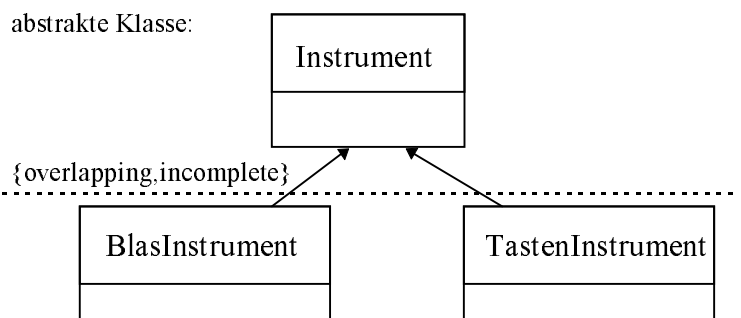
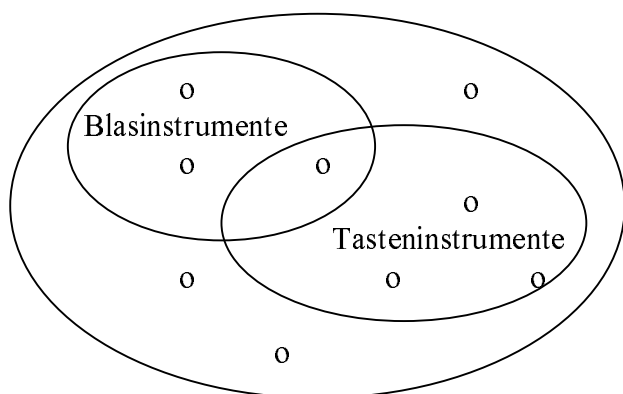
- disjunkte, aber unvollständige Überdeckung (der Spezialisierungsmengen)



- überlappende und vollständige Überdeckung (der Spezialisierungsmengen)



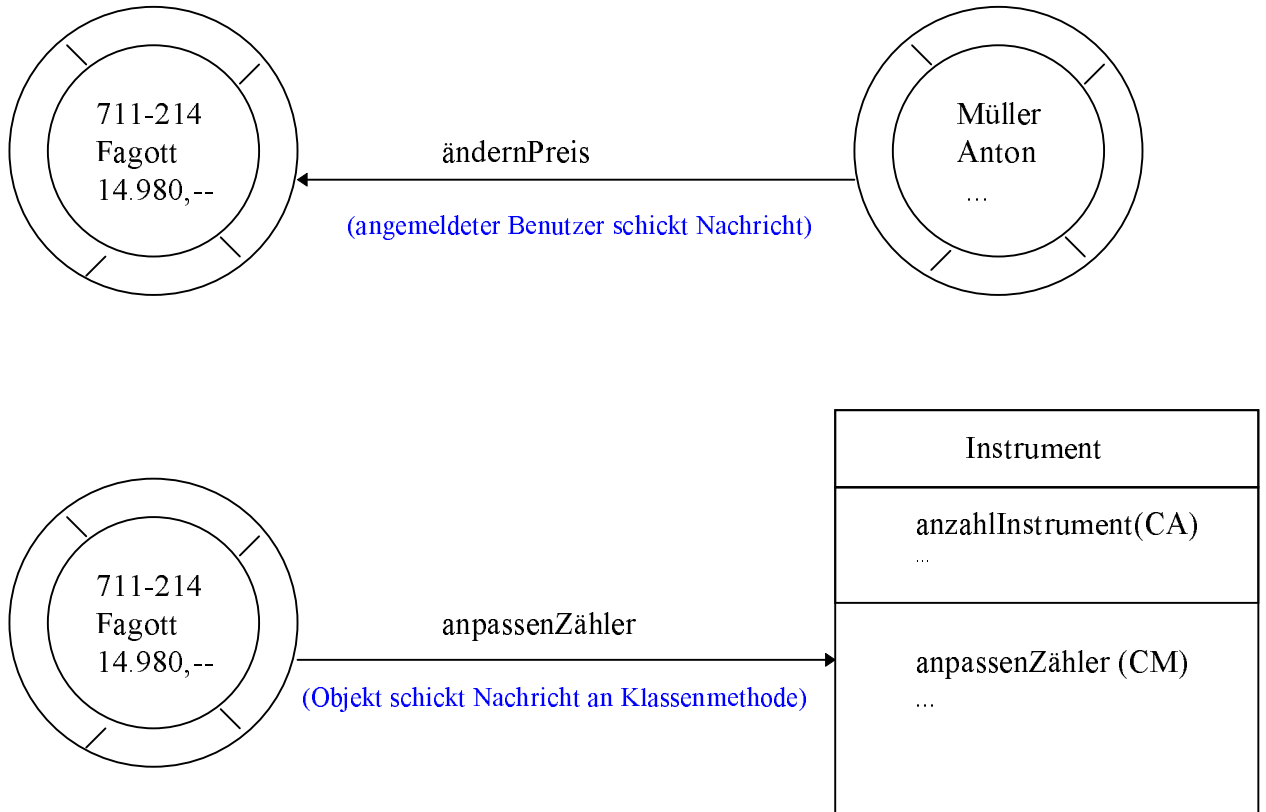
- überlappende und unvollständige Überdeckung (der Spezialisierungsmengen)



3.9 Dynamisches Verhalten

3.9.1 Nachrichten

Unter einer Nachricht versteht man die Aufforderung eines Objektes an ein anderes Objekt, eine bestimmte Methode auszuführen.



- Die Menge aller öffentlichen Nachrichten, die einer Klasse zur Verfügung stehen, bezeichnet man als Protokoll.

- Die Modellierung von Nachrichten erfolgt als gerichteter Pfeil (Sender $\xrightarrow{\text{Methode}}$ Empfänger)

- dynamisches Verhalten

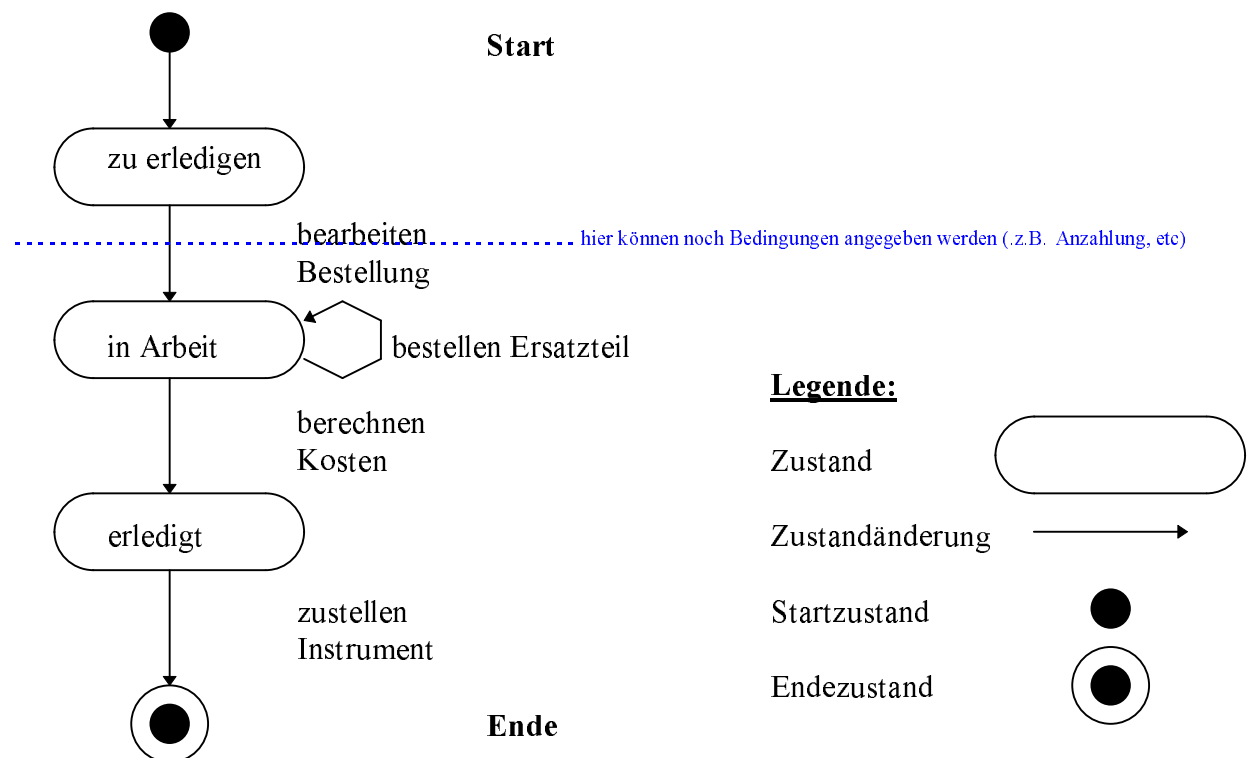
- Reihenfolge der Nachrichten

- Zeitverhalten (Ausführungszeiten, Wartezeiten, Timeout)

3.9.2 Zustandsübergangsdiagramme

- state transition diagramm
- Zustandsübergangsdiagramme zeigen die Ablauffolge der Zustände in einem System
- Zustandsänderungen werden durch Nachrichten aufgelöst (Ereignisse)

<Bild Beispiel Reparaturauftrag>



- Es können Bedingungen an Zustandsänderungen geknüpft werden (z.B. Anzahlung d. durch Kunden)

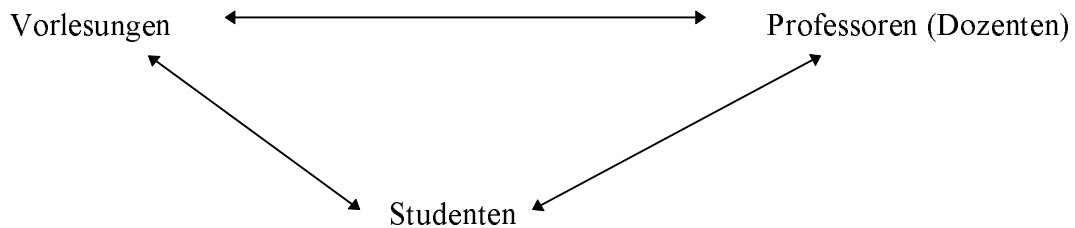
4. Beispiele zur Vorlesung

4.1 Gruppenarbeit Hochschuldatenbank

4.1.1 Aufgabe

Entwerfen Sie ein Objektmodell;

Vorlesungen werden von Dozenten gehalten und von Studenten besucht



4.1.2 Stoffsammlung

- Eine Vorlesung wird von einem Dozent gehalten (Assoziation)
- Ein Dozent hält Null oder mehrerer Vorlesungen (Assoziation)
- Null oder mehrerer Studenten besuchen eine Vorlesung (Assoziation)
- Dozent - Student => Person (Generalisierung)
- Terminvereinbarung (Dozent, Student)
(Beziehungsklasse: Thema, Teilnehmer, Datum, Uhrzeit) ... Methoden?)
- Ein Student besucht Null oder mehrere Vorlesungen (Assoziation)
- Prüfungsanforderungen einer Veranstaltung (Objektattribut)
- Klasse Vorlesung (Bezeichnung, Kürzel, Studiengang, Semester, ...)
- Klasse Student (Methoden: Immatrikulation, Exmatrikulation)
- Klasse Person (Generalisierung)
- Klasse Dozent (Sprechzeiten, Raum, Dienstart)
- Vorlesung - Dozent (Beziehungsklasse)
- Vorlesung - Student (Beziehungsklasse)

4.1.3 Klassenausarbeitung

5. OO Konzepte und Begriffe

5.1 Allgemein

Kapselung	Attribute und Methoden (Eigenschaften) sind nach außen sichtbar - die Implementierung der Methoden ist nach außen nicht sichtbar.
Konstruktor	Erzeugt und Initialisiert Objekte Wird beim Erzeugen des Objekts aufgerufen.
Destruktor	Löscht Objekte (Speicherberwaltung) Wird beim löschen des Objekts aufgerufen.
Polymorphismus	Methoden können unabhängig von der Art der Parameter definiert werden.

6. Einführung PHP

6.1 Einführung

PHP = **P**HP **H**ypertext **P**rozessor

PHP ist eine serverseitige, in HTML eingebettete Skriptsprache zur Erzeugung dynamischer Webseiten, d.h. der PHP Code wird vom Server interpretiert und es werden reine HTML-Seiten zurück zum Browser geliefert.

Was kann PHP?

- Formulardaten verarbeiten
- dynamische Seiten erzeugen
- Cookies senden/empfangen
- Datenbanken komfortabel ansprechen
- mit verschiedenen Netzdiensten zusammenarbeiten (IMAP, POP3, SNMP,...)

Voraussetzungen zum Betrieb von PHP:

- Webserver (ideal: Apache)
- PHP4-Distribution (www.php.org)
- Datenbanksystem (z.B. MySQL)
- Texteditor

6.2 Erste Schritte

6.2.1 „Hallo Welt“

```
<html>
<body>
  <?PHP
    echo „Hallo Welt“;
  ?>
</body>
</html>
```

Die Notation `<?PHP ... ?>` entspricht dem XML-Standard.

Als Dateinamenendung sind `.php` | `.php3` | `.php4` | `.phtml` Standard. (Unter Windows ist `.php3` zu empfehlen!)

6.2.2 Textausgabe

a) `echo (<String1>, <String2>, ...);` mehrere Zeichenketten
`echo <String1>`

z.B.: `echo („Hallo Welt“);`
`echo „Hallo Welt“;`

b) `print (<String1>);` eine Zeichenkette

c) `printf (<String Format>, <Argumente>)` formatierte Ausgabe

6.3 Die Sprache PHP

6.3.1 Variablen und Datentypen

- Alle Variablen habe das \$-Zeichen vorangestellt

- Wertzuweisung: \$a = 50;
 \$a = „Hallo Welt“;

- Datentypen: Ganze Zahlen
 Fließkommazahlen
 Zeichenketten

- Expansion von Variablen:

```
$a = 5;  
echo „Wert = $a“;            Ausgabe: Wert = 5
```

- Explizite Ausgabe:

```
echo ‘Wert = $a’;            Ausgabe: Wert = $a
```

- Verschachtelung möglich

z.B.: echo „Bitte geben Sie ‘ja’ ein“;

- Aneinanderhängen von Zeichenketten

```
echo $a.“ ist eine Zahl“     Ausgabe: 5 ist eine Zahl
```

```
$x = „Hallo“;  
$y = $x.“ Welt“;  
echo $y;                    Ausgabe: Hallo Welt
```

- Felder (Arrays) werden dynamisch erzeugt

```
$farbe=(„rot“, „grün“, „blau“);  
$farbe[5] = „orange“;  
echo $farbe[2];
```

Bemerkung: \$farbe[3] enthält „
 \$farbe[4] enthält „
 \$farbe[5] enthält „orange

- Assoziative Felder (Hash-Arrays)

```
$color[„rot“] = „0xff0000“;
```

Bemerkung: Hash-Arrays entsprechen Key-Value-Paaren; „rot“ = Key ; „0xff0000“ = Value

- Typumwandlung erfolgt implizit

6.3.2 Operatoren und Ausdrücke

- Logische Ausdrücke

Vergleichsoperatoren:	>, >=, ==, <, <=, !=
Verknüpfungsoperatoren:	&& (AND) (OR) ! (NOT)

- Arithmetische Ausdrücke

Operatoren:	+, #, *, /, %
Bitoperatoren:	& (AND) (OR) ^ (XOR) ~ (NOT)
Shiftoperatoren:	>>, <<
Zugriffsoperatoren:	=, +=, -=, *=, /=, %=, &=, =, ^=, .=

6.3.3 Kontrollstrukturen

- Kommentare

#	einzeilig
//	einzeilig
/* ... */	mehrzeilig

- Bedingte Verzweigung

```
if (<Ausdruck>
    <Anweisung1>;
else
    <Anweisung2>;
```

- Fallunterscheidung

```
switch (<Ausdruck>){  
    case <A1>: <Anweisung1>;  
    case <A2>: <Anweisung2>;  
    ...  
    default: <Anweisung>;  
}
```

```
z.B.: switch ($action){  
    case „Add“: AddData(); break;  
    case „Delete“: DelData(); break;  
}
```

- Schleifen

```
while (<Ausdruck>)  
    <Anweisung>;
```

```
do  
    <Anweisung>;  
while (<Ausdruck>);
```

```
for (<Ausdruck1>; <Ausdruck2>; <Ausdruck3>)  
    <Anweisung>;
```

```
z.B.: $max=100;  
for ($i=1; $sum=0; $i<=100; $sum += (i++) );
```

6.3.4 Funktionen

```
function <Name> (plist)  
{  
    <Anweisungen>;  
}
```

Falls eine Funktion Übergabeargumente verändern soll benötigt man eine Referenz (Zeiger) auf das Argument.

```
z.B.: function f(&$x)  
{  
    $x++;  
}
```

- Default Parameter

```
function t_html($title="Titel"){ ...}
```

- Return Anweisung erlaubt einen Rückgabewert

```
function gauss ($n)
{
    for ($i=1; $sum=0; $i<=$n; $i++)
        $sum += $i;
    return $sum;
}
```

```
...
$summe = gauss (112);
...
```

6.3.5 MySQL-PHP-Funktionen

- Verbindung zu einem MySQL-Server aufnehmen (`_pconnect` = persistent = dauerhaft)

```
$cid=mysql_pconnect($host, $user, $password);
```

- Datenbank unter Nutzung der bestehenden Verbindung auswählen

```
$success=mysql_select_db($database, $cid);
```

- Senden einer SQL-Anfrage zum DBMS

```
$result=mysql_query($sqlstring, $cid);
```

- Abfrage der Anzahl der Datensätze in der SQL-Ergebnisrelation

```
$anzahl=mysql_num_rows($result);
```

- Abfragen bzw. Zugriff auf die Datensätze in der SQL-Ergebnisrelation

```
$dsubject=mysql_fetch_object($result);                    Felder als Eigenschaften
```

oder: `$dsarray=mysql_fetch_row($result);` indiziertes, numerisches Array

oder: `$dsarray=mysql_fetch_array($result, <arraytype>)` Array, vom Typ

<code>mysql_assoc</code>	assoziatives Array mit Feldnamen als Indizes und Feldinhalt als Wert
<code>mysql_num</code>	numerisches Array mit von 0 beginnend durchnummerierten Feldern
<code>mysql_both</code>	assoziatives Array mit Feldnamen und numerischen Werten als Indizes

(Der Zugriff über Array ist schneller; Beim Zugriff über Object kann ich direkt auf die Attribute zugreifen - wird nachträglich eine Spalte in die Datenbank eingefügt kommt das Programm nicht durcheinander!)

- Abfragen eines Feldwertes bei Zugriff über `mysql_fetch_object()`

```
$wert = dsobject->Feldname;
```

- Abfragen der Fehlermeldungen der zuletzt ausgeführten MySQL-Operation

```
$err=mysql_error($cid);
```

- Abfragen der ID des zuletzt eingefügten Datensatzes (INSERT mit `auto_inc`)

```
$lid=mysql_insert_id($cid);
```

- Schließen einer offenen Verbindung

```
mysql_close($cid);
```

Bemerkung: MySQL ist nicht für Transaktionen ausgelegt!

6.3.6 Übersicht der MySQL Funktionen in PHP

<code>mysql_affected_rows()</code>	Anzahl der von der letzten Operation betroffenen Zeilen zurückgeben.
<code>mysql_close()</code>	Verbindung schliessen
<code>mysql_connect()</code>	Verbindung zum Server öffnen
<code>mysql_create_db()</code>	Datenbank erzeugen
<code>mysql_data_seek()</code>	Internen Zeiger auf Ergebnisdatensätze setzen
<code>mysql_db_query()</code>	SQL-Abfrage an die Datenbank senden
<code>mysql_drop_db()</code>	Datenbank löschen
<code>mysql_errno()</code>	Fehlernummer der letzten Datenbankoperation zurückgeben
<code>mysql_error()</code>	Fehlertext der letzten Datenbankoperation zurückgeben
<code>mysql_fetch_array()</code>	Aktuellen Ergebnisdatensatz in eine Array übertragen

<code>mysql_fetch_field()</code>	Spalteninformation eines Ergebnisdatensatzes holen und als Objekt übergeben
<code>mysql_fetch_lengths()</code>	Länge der Ausgabe eines Ergebnisses ausgeben
<code>mysql_fetch_object()</code>	Aktuellen Ergebnisdatensatz in ein Objekt übertragen
<code>mysql_fetch_row()</code>	Aktuellen Ergebnisdatensatz in ein nummeriertes Array übertragen
<code>mysql_field_name()</code>	Namen eines Feldes im aktuellen Ergebnisdatensatz zurückgeben
<code>mysql_field_seek()</code>	Ergebniszeiger auf spezielles Feld setzen
<code>mysql_field_table()</code>	Namen einer Tabelle zurückgeben, welche ein bestimmtes Feld enthält
<code>mysql_field_type()</code>	Typ eines Feldes im Ergebnisdatensatz zuurückgeben
<code>mysql_field_flags()</code>	Parameter eines Feldes im Ergebnisdatensatz zurückgeben
<code>mysql_field_len()</code>	Länge des Feldes zurückgeben
<code>mysql_free_result()</code>	Den von der Ergebnisliste benötigten Speicher löschen und wieder freigeben
<code>mysql_insert_id()</code>	ID des Autoincrement-Feldes eines vorangegangenen INSERT Befehls zurückgeben
<code>mysql_list_fields()</code>	Alle Ergebnisfelder ausgeben
<code>mysql_list_dbs()</code>	Alle Datenbanken ausgeben
<code>mysql_list_tables()</code>	Alle Tabellen einer Datenbank anzeigen
<code>mysql_num_fields()</code>	Anzahl der Felder (Spalten) einer Ergebnisliste ausgeben
<code>mysql_num_rows()</code>	Anzahl der Datensätze (Zeilen) einer Ergebnisliste ausgeben
<code>mysql_pconnect()</code>	Eine ständige (p ersistent) Verbindung zur Datenbank öffnen
<code>mysql_query()</code>	Eine SQL-Abfrage senden
<code>mysql_result()</code>	Ergebnisliste der SQL-Abfrage einlesen
<code>mysql_select_db()</code>	Standarddatenbank auswählen
<code>mysql_tablename()</code>	Namen der Tabellen zu einem Feldnamen ermitteln

6.3.7 Feldinformationen bei `mysql_fetch_fields()`

7. Eigene Beispielskripte

Zwei kleine Beispielskripte:

Filename: <showtable.php3>

```
<html>
<?php

$cid=mysql_pconnect("ls102-02","ag_16","00mmt5");
mysql_select_db("DB2_16", $cid);
$query_string="select * from $utable";
$result=mysql_query($query_string,$cid);

$fieldCount = mysql_num_fields( $result );

echo "FieldCount:". $fieldCount. "<br>";

echo "<table border=1><tr>";
for ($i=0; $i< $fieldCount; $i++) {
    echo "<td>".mysql_field_name( $result, $i )."</td>\n";
}
echo "</tr>";

while ($row = mysql_fetch_array( $result ) ) {
    echo "<tr>\n";
    for ($i = 0; $i < $fieldCount; $i++) {
        echo "<td>". $row[$i]."</td>\n";
    }
    echo "</tr>\n";
}

echo "</table>";
?>

</html>
```

Filename: <test.htm>

```
<html>

<title>Formular für Tabellennamen</title>

<body>
<form action="showtable.php3">
    <input type="TEXT" name="utable" SIZE=30>
    <input type="SUBMIT" value="Ausgabe">
</form>
</body>

</html>
```

Literaturverzeichnis:

Dokumentationen der Programmpackete	http://www.tcx.se	MySQL
	http://www.php.net	PHP
	http://www.apache.org	Apache

Vorlesung Prof. Dr. Heym im WS 2000/01 (Fachhochschule Hof - jhey@fh-hof.de)

PHP4 - Grundlagen und Profiwissen, Krause, Hanser-Verlag, ISBN 3-446-21546-8, DM 98,--

Objektorientierte und objektrelationale DB, Meier/Wüst, dpunkt.verlag, ISBN 3-932588-68-1, DM 44,--

WAMP(WindowsApacheMySQLPHP)-Dokumentation (thomas@schulz.net)

Die jeweils aktuelle Version dieses Dokuments ist unter <http://www.tobiasott.de> zu finden.

Stichwortregister

A

Assoziationstypen 10

B

Beziehungstypen 11

Inhaltsverzeichnis

- 1. Installation der benötigten Software**
 - 1.1 Apache (1.3.x) - Webserver (Win32)**
 - 1.2 PHP 4.0 (Win32)**
 - 1.2.1 Standardinstallation**
 - 1.2.2 PHP Base Library**
 - 1.3 MySQL (Win32)**
 - 1.4 phpMyAdmin**
- 2. MySQL**
 - 2.1 Datenbank ist vorhanden - Quickstart**
 - 2.2 Die wichtigsten Befehle im Überblick**
 - 2.2.1 Einfach anmelden**
 - 2.2.2 Textdatei mit SQL-Befehlen in MySQL übertragen**
 - 2.2.3 Datenbank auswählen bzw. wechseln**
 - 2.2.4 Alle Tabellen in der Datenbank anzeigen**
 - 2.2.5 Wie ist eine bestimmte Tabelle aufgebaut?**
 - 2.3 SQL-Befehle - eine kurze Übersicht der wichtigsten Befehle**
- 3. Datenbanksysteme - Theorie**
 - 3.1 Grundsätzliches zum Aufbau eines Datenbanksystems**
 - 3.2 Was sind Objekte?**
 - 3.3 Eigenschaften eines OO-DBS**
 - 3.4 Objektorientierte Modellierung**
 - 3.5 Objektindikatoren**
 - 3.6 Klasseneigenschaften**
 - 3.7 Beziehungskonzepte**
 - 3.7.1 Assoziationen**
 - 3.7.2 Beziehungsklassen**
 - 3.7.3 Aggregation**

3.8 Vererbung

3.8.1 Generalisierung

3.8.2 Vererbung

3.8.3 Vererbungsstrukturen

3.9 Dynamisches Verhalten

3.9.1 Nachrichten

3.9.2 Zustandsübergangsdigramme

4. Beispiele zur Vorlesung

4.1 Gruppenarbeit Hochschuldatenbank

4.1.1 Aufgabe

4.1.2 Stoffsammlung

4.1.3 Klassenausarbeitung

5. OO Konzepte und Begriffe

5.1 Allgemein

6. Einführung PHP

6.1 Einführung

6.2 Erste Schritte

6.2.1 "Hallo Welt"

6.2.2 Textausgabe

6.3 Die Sprache PHP

6.3.1 Variablen und Datentypen

6.3.2 Operatoren und Ausdrücke

6.3.3 Kontrollstrukturen

6.3.4 Funktionen

6.3.5 MySQL-PHP-Funktionen

6.3.6 Übersicht der MySQL Funktionen in PHP

6.3.7 Feldinformationen bei `mysql_fetch_fields()`

7. Eigene Beispielskripte

Literaturverzeichnis

Stichwortregister

Inhaltsverzeichnis