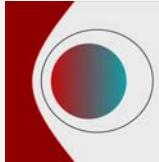




Einführung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Lehrveranstaltung

- Titel: Geoinformatik 2
- Art: Konstruktionsübung
- Semesterwochenstunden: 2
- ECTS: 3
- Betreuung
 - Institut für Geoinformation
- Anmeldung über TUGonline bis 10.10.2011



Ziel der LV

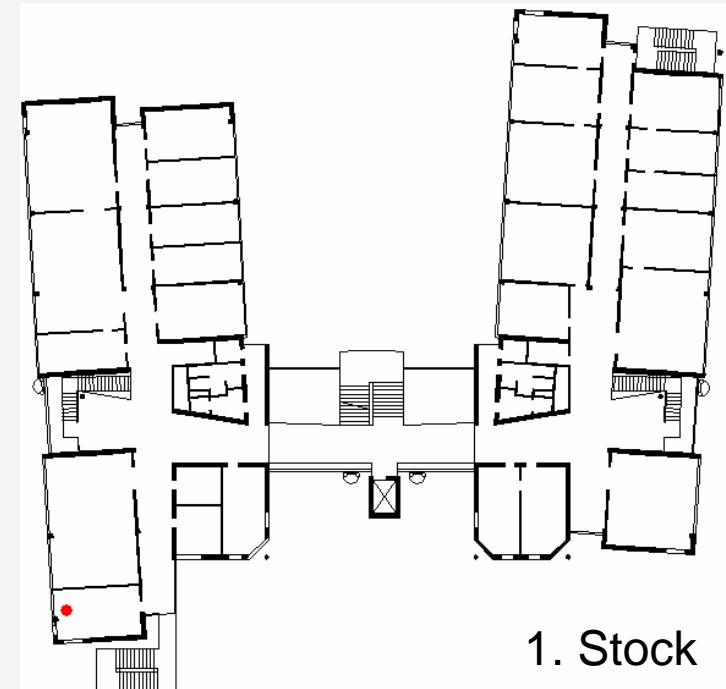
- Nach erfolgreicher Absolvierung der Lehrveranstaltung sind die Studierenden imstande, **Informationen** in Datenbankumgebungen zu **organisieren** und zu **verwalten**. Weiters sind sie mit den Besonderheiten von **ortsgebundenen Daten** in solchen Umgebungen vertraut.

TUG online



Clemens Strauß

- Büro: A114 (•)
- Kontakt
 - clemens.strauss@tugraz.at
 - 0316 / 873 – 6846
- Sprechstunde
 - Jederzeit

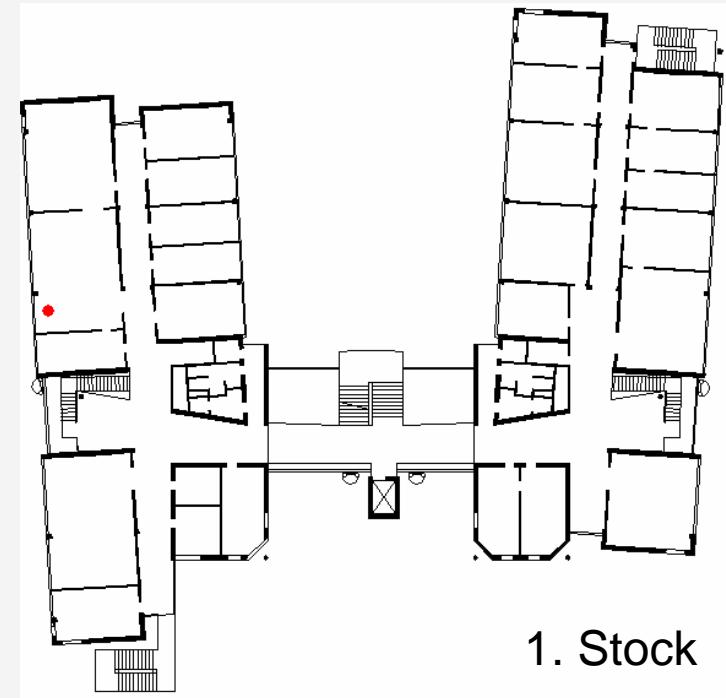


1. Stock



LV Abhaltung

- Ort: AE01 bzw. A109 (•)
- Zeiten:
 - Montags 16:15 – 17:45



1. Stock



Voraussetzung

- Keine Prüfungsvoraussetzung
- Empfohlene Lehrveranstaltungen
 - Vorangegangene Semester
 - Grundlagen der Geoinformation
 - Geodatenquellen
 - Geoinformatik 1
 - Geomathematik 1 (geodätische Hauptaufgaben)
 - Geosoftware-Applikationen 1
 - Gleiches Semester
 - Geoinformatik 2 VO
 - Bezugssysteme (Koordinatensysteme)



- Modellierung von Daten
- Normalformen
- Structured Query Language (SQL)
 - PostgreSQL Datenbankmanagementsystem
- spatialSQL
 - räumliche Erweiterung von PostgreSQL
- Quantum GIS
- PHP
 - serverseitige dynamische Webseitengestaltung
 - Externer Zugriff auf PostGIS DBMS

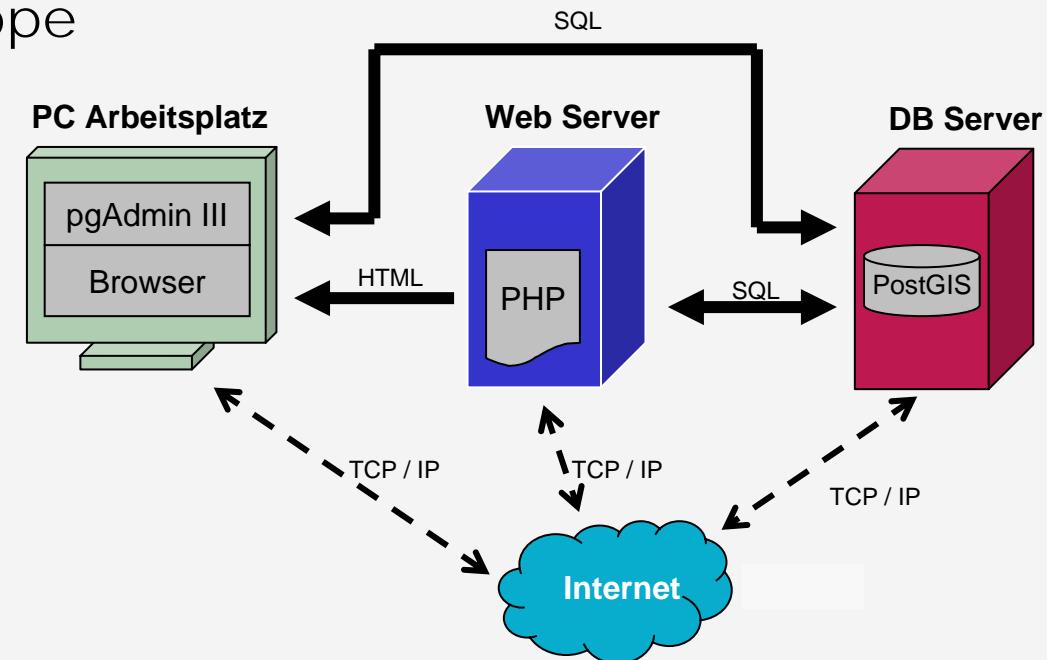


Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



- Drei Übungsprogramme
- Übungsprogramme sind aufbauend
- Gruppenarbeit
 - 2-3 Personen / Gruppe





Beurteilung

- Konventionelle Beurteilung
 - Schulnoten (1-5)
- Beurteilungsgrundlage
 - Mitarbeitsüberprüfung
 - Übungsprogramme
 - Klausur
- Anwesenheit
 - Min. 65 % (9 Einheiten)
 - Ausnahmen in begründeten Fällen



- LV Administration via *TeachCenter*
 - Kommunikation
 - Präsentationen (PDF)
 - Handouts (PDF)
 - Internet Links
- Begleitende Literatur:
 - Yeung, A., Hall, G., (2007). Spatial Database Systems, Dordrecht: Springer
 - Diverse Internettutorien (PostGIS, HTML, PHP, KML)



Allfälliges

- 17.10.2011: AE01.
- Fragen ?



Einführung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

Datenbankdesign

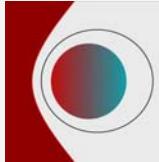
Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



Einleitung





Einleitung

Definition

DB Modelle

Datenmodellierung

- Definition und Konzepte.
- Gängige Datenbankmodelle.
- Prinzipien und Techniken der Datenmodellierung.



Definition und Konzepte



- Beschreibung der Datenbank.
- Besteht aus
 - Konzepten
 - Beschreibungen
 - Skizzen
 - ... um
 - Struktur
 - Datenverarbeitung
 - ... einer Datenbank zu beschreiben.
- Beinhaltet nicht die Methoden wie eine DB realisiert wird.



Datenbank Modell

Einleitung

Definition

DB Modelle

Datenmodellierung



Abstraktion



Datenmodellierungsteam

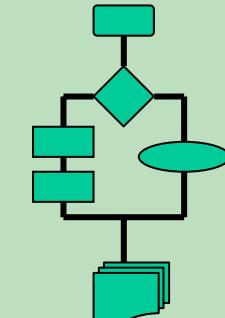
Konzeption

Datenbank Modell

Konzepte
über die
reale Welt

Dokumentation

Graphische Notation
des Datenbankmodells





- Schema
 - Repräsentiert eine bestimmte Menge der realen Welt.
 - Statisch.
 - Zeitinvariant.
 - Beinhaltet eine Sammlung von Beschreibungen und Skizzen einer Datenstruktur.
- Instanz
 - Erscheinungsbild von Objekten in einer Datenbank.
 - Dynamisch.
 - Variiert mit der Zeit.



Schema und Instanzen

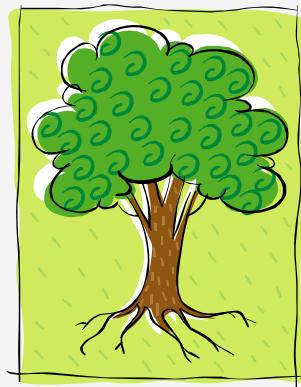
Einleitung

Definition

DB Modelle

Datenmodellierung

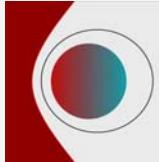
- Schema „Baum“



- Wurzeln.
- Stamm.
- Äste.
- Blätter.

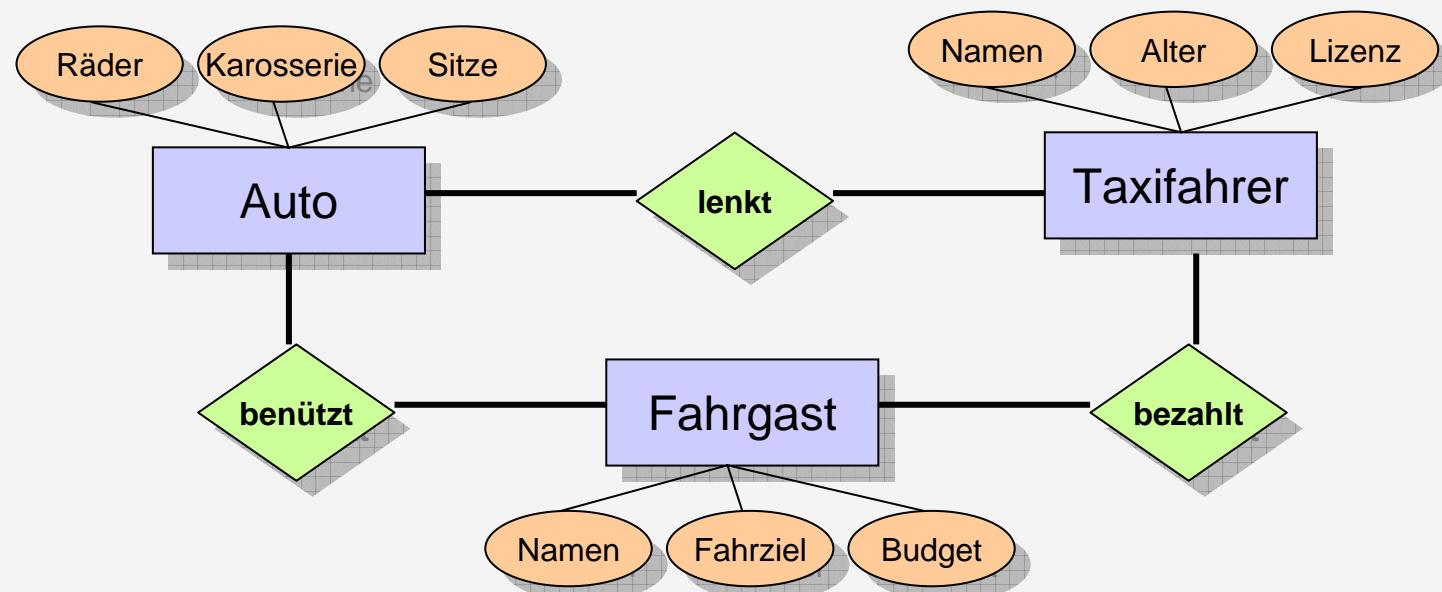
- Instanzen

- Fichte.
- Tanne.
- Buche.
- Linde.
- Kastanie.
- Palme.
- Maibaum.
- Weihnachtsbaum.
- Schlagbaum.



- Konzeptionelle Modellierung
 - Mentaler Prozess.
 - Hoher Level an Abstraktion
 - Hardwareunabhängig.
 - Softwareunabhängig.
 - Abstraktionsansätze
 1. **Klassifikation**: Definition von Klassen mit entsprechenden Eigenschaften.
(Auto: Räder, Karosserie, Sitze)
 2. **Aggregation**: Definition einer neuen Klasse unter der Verwendung bereits bestehender Klassen.
(Taxi: Auto, Taxifahrer, Fahrgast)
 3. **Generalisierung**: Definition einer Beziehung zwischen mindestens zwei Klassen.
(Kraftfahrzeuge: Auto, Lastwagen, Motorrad)

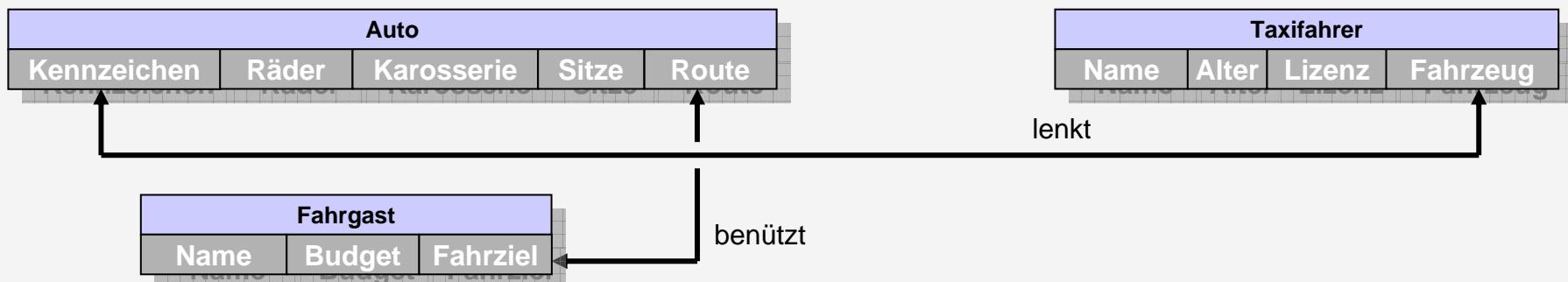
- Ergebnis der konzeptionellen Modellierung
 - Diagramm.
 - Verbale Beschreibung.
 - Ein Kombination daraus.





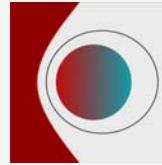
• Logische Modellierung

- Berücksichtigung der Software.
- Übertragung des konzeptionellen Modells betreffend der Syntax und der diagrammatischen Notation für ein ausgewähltes DBMS:
 - Relationale DBMS.
 - Objektrelationale DBMS.
 - Objektorientierte DBMS.





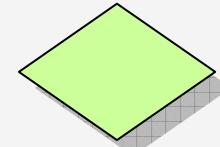
- Technische Modellierung
 - Berücksichtigung des DBMS und der Hardware.
 - Inhalte der technischen Modellierung (Auszug)
 - Datentypen (Zahlen, Zeichenketten, Boolesche Variable,...).
 - Wertebereiche (0..n, 0/1, a..z, (1:10), +/- 1000,...).
 - Speicherplatzbedarf.
 - Art des Attributs
 - Schlüsselement.
 - Optionale Werteingabe.
 - Einzigartigkeit des Wertes (z.B. für Identifikation).
 - Detaillierte verbale Beschreibung der Attribute.



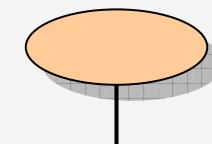
Gängige Datenbankmodelle



- Einsatz in der konzeptionellen Modellierung.
- Der erste Schritt im Datenbankdesign.
- Elemente
 - Datensatz (Entity).
 - Beziehung (Relationship).
 - Verbindung zwischen Objekten.
 - Attribute.
 - Komplexität einer Beziehung.



—

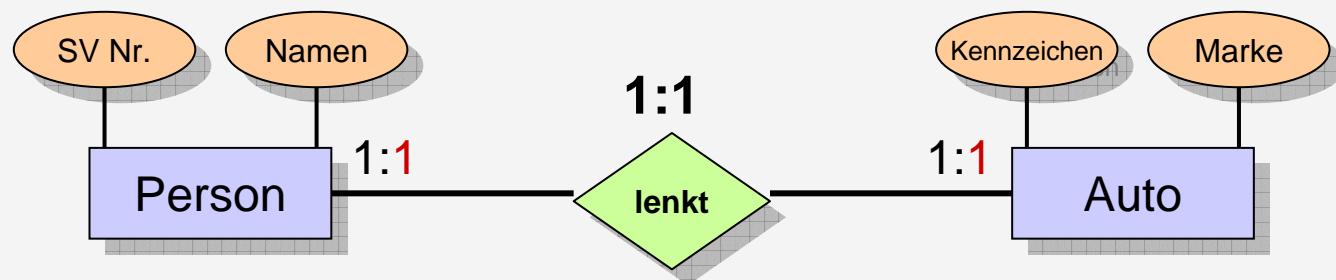


1:1 / 1:n / n:m

Entity-Relationship Modell

Einleitung Definition DB Modelle Datenmodellierung

- Einfache Beziehungen (1:1).

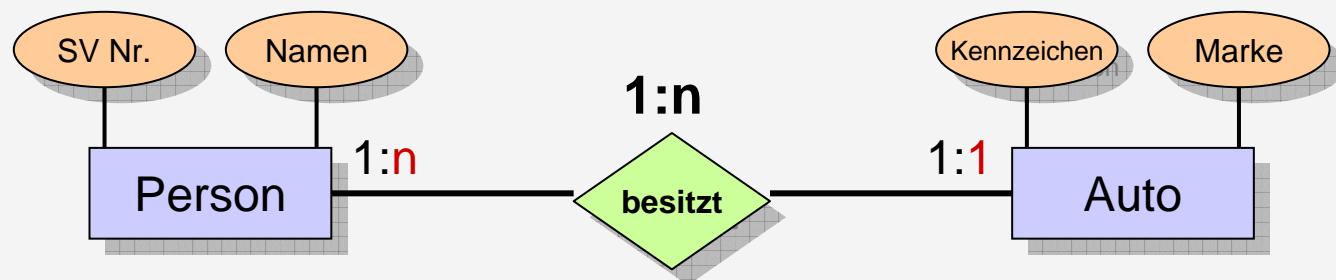


- Komplexität
 - „Eine Person lenkt ein Auto.“ (plausibel)
 - „Eine Person lenkt mehrere Autos.“ (geht nicht)
 - „Ein Auto wird von einer Person gelenkt.“ (plausibel)
 - „Ein Auto wird von mehreren Personen gelenkt.“ (geht nicht)

Entity-Relationship Modell

Einleitung Definition DB Modelle Datenmodellierung

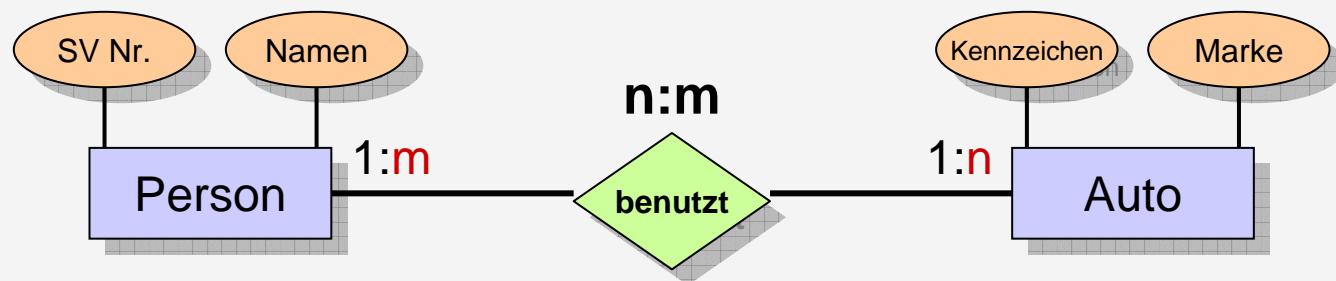
- Einfache Beziehungen (1:n).



- Komplexität
 - „Eine Person besitzt ein Auto.“ (?)
 - „Eine Person besitzt mehrere Autos.“ (plausibel)
 - „Ein Auto wird von einer Person besessen.“ (plausibel)
 - „Ein Auto wird von mehreren Personen besessen.“ (geht nicht)

Entity-Relationship Modell

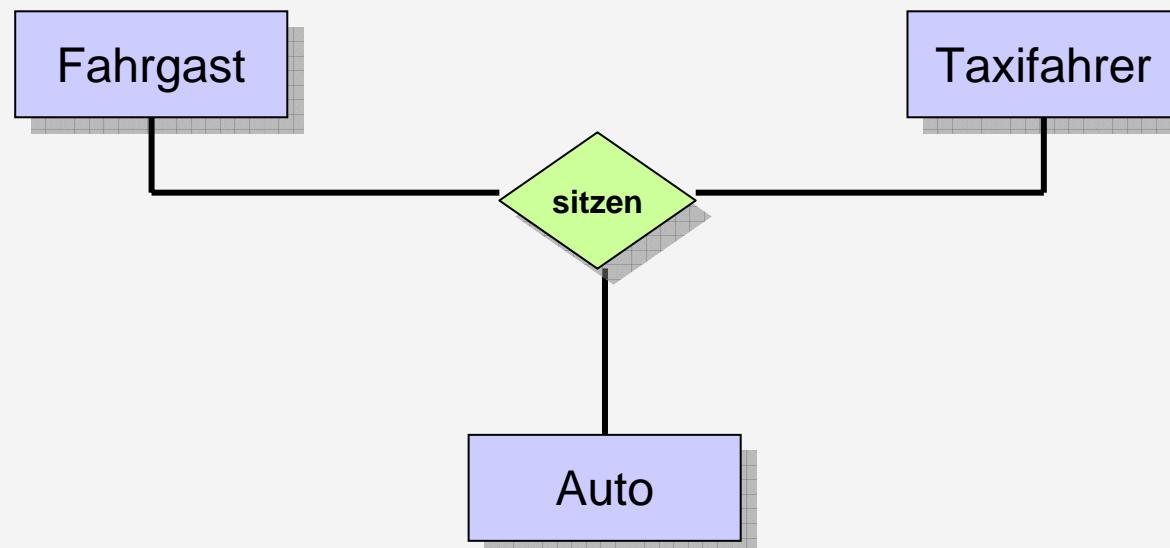
- Einfache Beziehungen (n:m).



- Komplexität
 - „Eine Person benutzt ein Auto.“ (?)
 - „Eine Person benutz mehrer Autos.“ (plausibel)
 - „Ein Auto wird von einer Person benutzt.“ (?)
 - „Ein Auto wird von mehreren Personen benutzt.“ (plausibel)



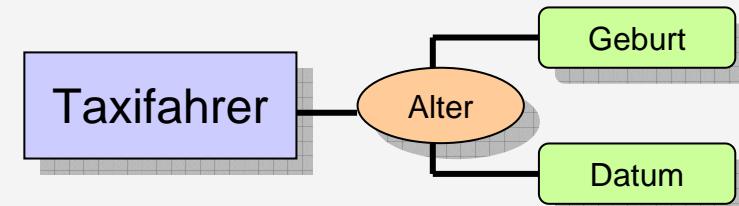
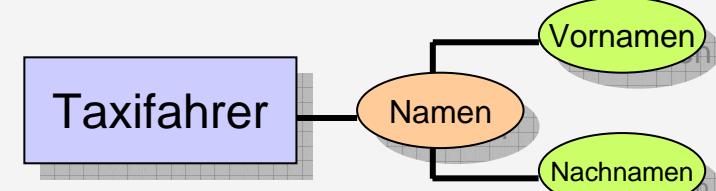
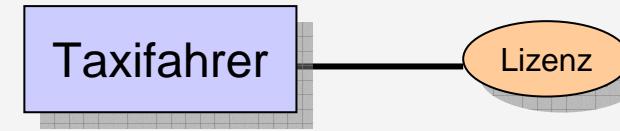
- Tenäre Beziehung
 - Beziehung zwischen drei Datensätzen.



Entity-Relationship Modell

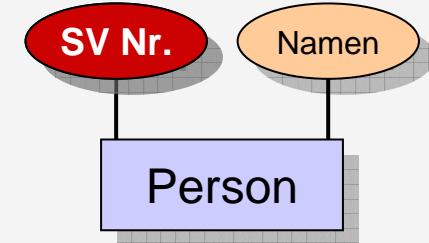
Einleitung Definition DB Modelle Datenmodellierung

- Attribute
 - Einfaches Attribut.
 - Zusammengesetztes Attribut.
 - Abgeleitetes Attribut.

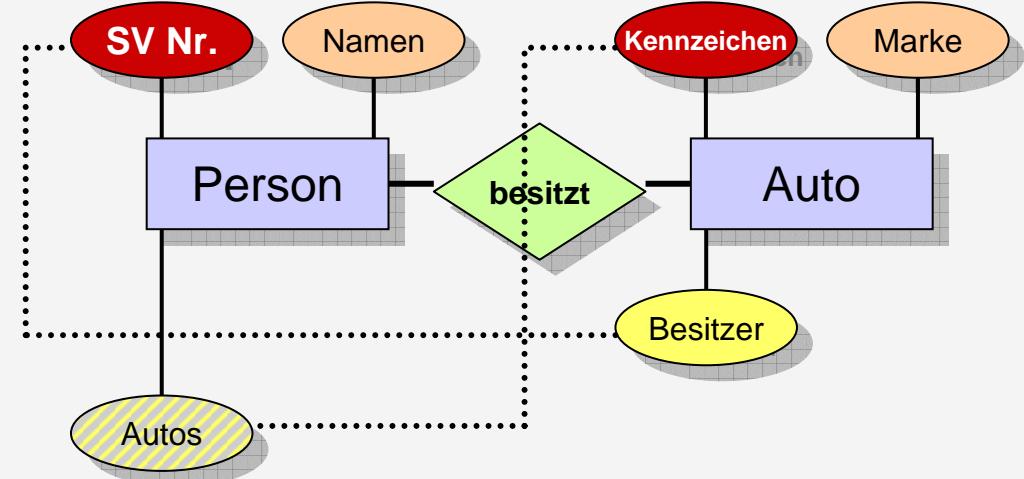




- Primärschlüssel
 - Element zur eindeutigen Identifizierung eines Datensatzes.



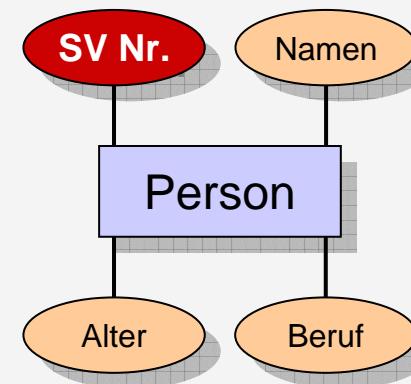
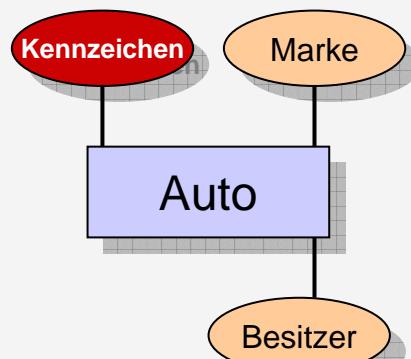
- Fremdschlüssel
 - Bindeglied zwischen zwei Relationen.



Relationales Modell

[Einleitung](#)[Definition](#)[DB Modelle](#)[Datenmodellierung](#)

- Logische Strukturierung der Daten.
- Verwendung von Relationen (Tabellen).
- Grundlage bildet ein E/R Modell.

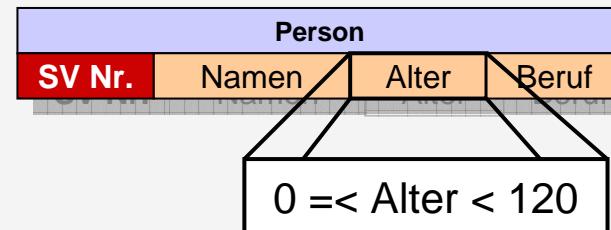


Auto		
Kennzeichen	Marke	Besitzer
Kennzeichen	Marke	Besitzer

Person			
SV Nr.	Namen	Alter	Beruf
SV Nr.	Namen	Alter	Beruf

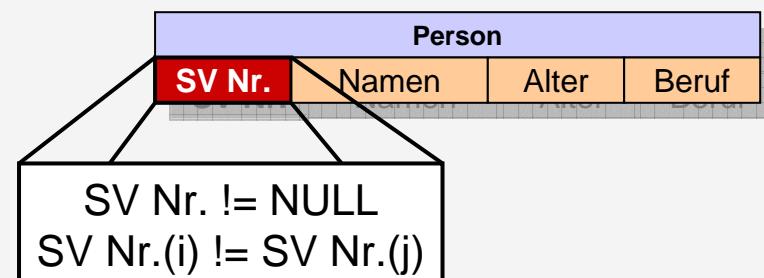


- Einsatz von Bedingungen
 - 1. Definitionsbereich der Attributwerte
 - Werte in Attributfelder müssen plausibel sein.



2. Inhalte eines Datensatzes

- Einsatz eines Primärschlüssels
 - Ungleich NULL.
 - Eindeutig.



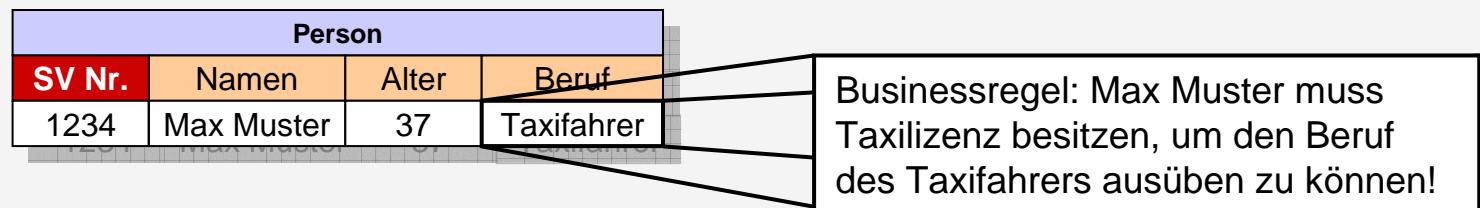
3. Referenzierung

- Wert eines Fremdschlüssels muss mit einem **bestehenden** Primärschlüssel übereinstimmen.



4. Businessregeln

- Z.B. Einhaltung von Verwaltungsabläufen.



5. Normalformen (1.-5.)

- Aufbau des Relationsschemas. (Inhalt der Übung am 18.10.2010)



- Objektorientierte Modelle
 - Konzept und Methoden entstammen der objektorientierten Programmierung.
 - Bestandteile von Objekten
 - Name.
 - Abstrakte Datentypen für Attribute.
 - Methoden.
 - ...
 - Zusammenfassung von gleichartigen Objekten in einer Klasse.
- Objekt-relationale Modelle
 - Erweiterung einer Relationalen DB durch einige objektorientierte Funktionen.



Prinzipien und Techniken der Datenmodellierung



Drei Prinzipien

Einleitung

Definition

DB Modelle

Datenmodellierung

1. Die Wahl des Modells hat tief greifenden Einfluss auf den Problemzugang und den Lösungsansatz.
2. Jedes Modell kann einen unterschiedlichen Detailgrad aufweisen.
3. Das beste Modell entspricht der realen Welt.

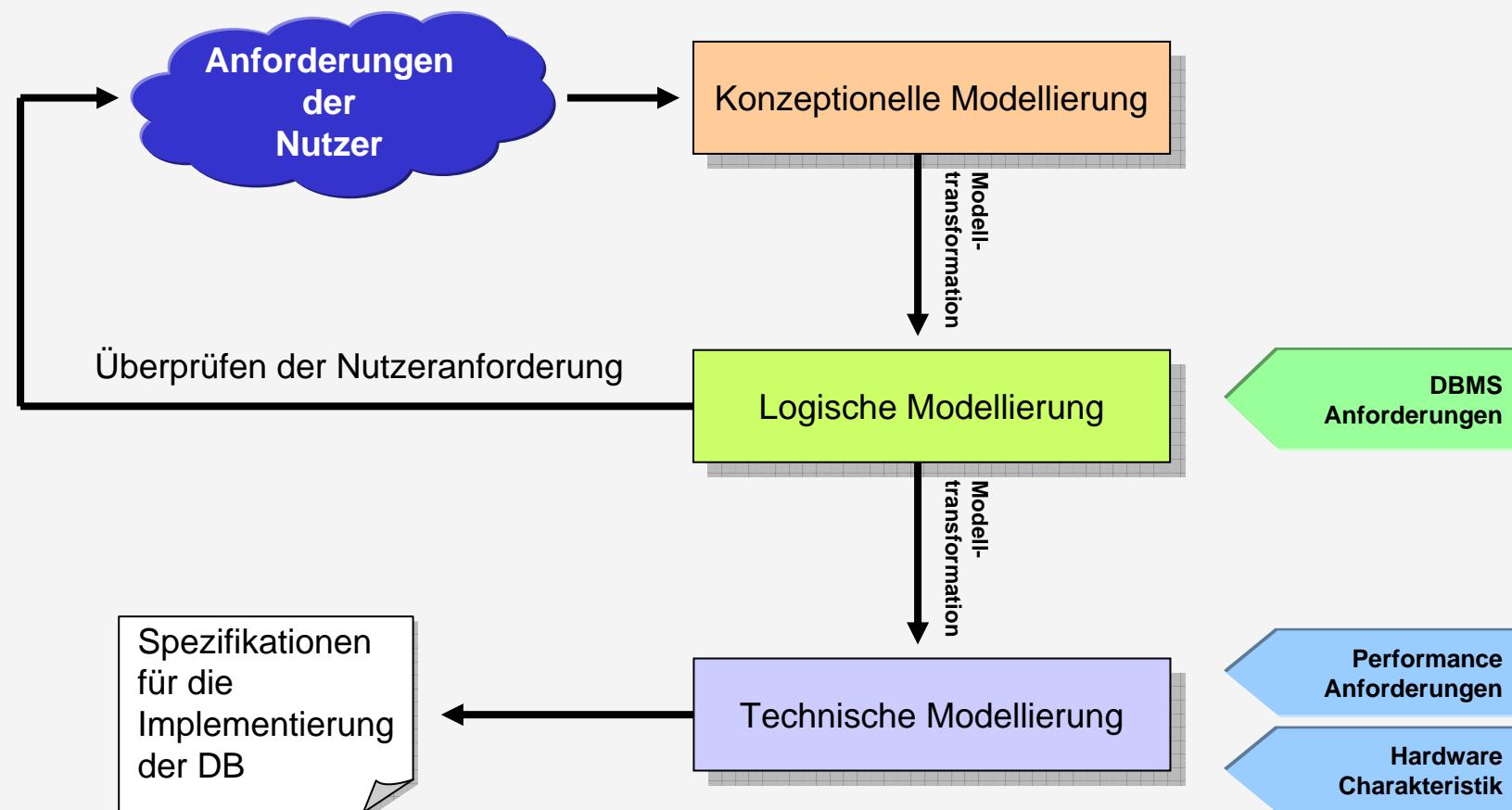
DB Modeling Life Cycle

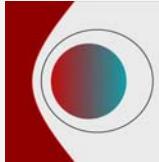
Einleitung

Definition

DB Modelle

Datenmodellierung





- **Aussagekraft**

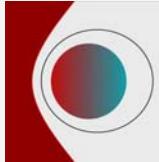
- Verwendung entsprechender Konzepte für eine verständliche Repräsentation der realen Welt.

- **Schlichtheit**

- Einfach verständlich für alle Beteiligten der DB-Entwicklung.
(Konfliktpotential zwischen Aussagekraft und Schlichtheit!)

- **Formalität**

- Alle Konzepte eines Modells sind...
 - eindeutig
 - präzis
 - gut definiert
 - ... von den Beteiligten interpretierbar.



- Ausgabe 1. Übungsprogramm
 - Gruppeneinteilung erfolgt im TeachCenter
- 24.10.2011: AE01
- Fragen ?

Datenbankdesign

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

Normalformen

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



Einleitung

tblBodenaufnahme : Tabelle													
	AufgabefID	Probenummer	Betriebsname	Vulgename	Parzellennumm	Nutzungsform	Bezeichnung	Bodengründigkeit	Hangneigung	GeometriefID	ExplfID	MesofID	MifID
▶	+	1 2004-03-18/02	rtu	eur	546	eriu			6	2	5	4	
▶	+	2 2004-05-16/03	Sepp	Jussuf	41	Gack			6	3	5	4	
▶	+	3 1558-10-25/32	Hans	Hiasl	12	Weid			1	4	2	1	
▶	+	4 5455-11-26/1	Josef	Sepp	55	Wies			1	4	2	1	
▶	+	5 9991-10-33/12	Huber	Sackbauer	01	Korn			1	4	2	1	
▶	+	10 20			10	Minut			1	4	2	1	
*	0												

tblBodenart : Tabelle

BodenartID Beschreibung

1 Sand

2 schluffiger Sand

3 lehmiger Sand

4 toniger Sand

5 sandiger Schluff

6 Schluff

7 lehmiger Schluff

tblPunkte : Tabelle

PunktID X Y Z

1 560970 353894 0

2 560166 354133 0

3 560866 353704 0

4 561050 353975 0

5 560995 3533973 0

6 561004 354079 0

7 561124 354199 0

Regeln für den Aufbau von Relationen

16 Marmor	Datensatz: 14 1 1 ► ► *	6 Mulden,Kessel	tblMikrorelief : Tabelle
17 Dolomit		7 Wanne	
18 Rauhwacke		8 Graben	
19 Phyllit		9 Oberhang	
20 Grünschiefer		10 Unterhang	
21 Tonschiefer		11 Mittelhang	
22 Glimmerschiefe		12 Hangversteilung	
23 Gneis		13 Hangverflachung	
24 Amphibolit		14 Kuppe	
25 Kalkglimmersch.		15 Rücken	
26 Serpentin		16 Riedel,Wall	
*		17 Hangfuß	
		18 Schwemmfächer, Schuttfächer	
		19 Schwemmkegel, Schuttkegel	
	Datensatz: 14 1 1 ► ► *	Datensatz: 14 1 1 ► ► * von 19	



Einleitung

Grundlegendes

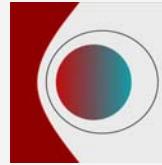
1. NF

2. NF

3. NF

Anmerkungen

- Grundlegendes
- 1. Normalform
- 2. Normalform
- 3. Normalform
- Anmerkungen



Einleitung

Grundlegendes

1. NF

2. NF

3. NF

Anmerkungen

Grundlegendes



- **Normalisierung:**

Schrittweise Zerlegung von Relationen in mehrere Relationen auf der Grundlage funktionaler Abhangigkeiten.

- Zweck

- Verhindern von Redundanzen.
- Optimieren des Speicherplatzes.
- Verkurzen von Such- und Analysealgorithmen.
- Wahren der Datenkonsistenz bei nderungen und Erweiterungen der Daten.





1. Normalform



- **Definition:**

Jedes Attribut der Relation muss einen atomaren Wertebereich haben, die Relation muss frei von Wiederholungsgruppen sein.

- Atomarer Wertebereich

- Attribut kann nicht weiter sinnvoll aufgespaltet werden.
- Bsp.: Adresse -> PLZ, Ort, Strasse, Hausnummer.

- Wiederholungsgruppen

- Attribute, die gleichartige Information enthalten, müssen in eine andere Relation ausgelagert werden.
- Bsp.: SV Nr, Name, Hauptwohnsitz, Nebenwohnsitz 1, Nebenwohnsitz 2, Nebenwohnsitz 3, ...





Beispiel (NF nicht erfüllt)

Einleitung

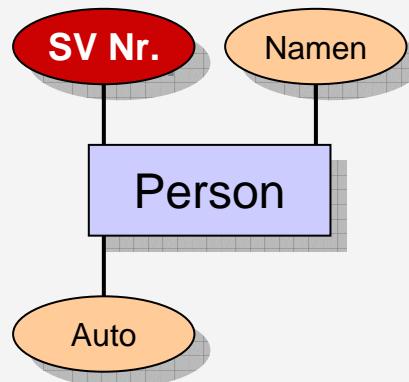
Grundlegendes

1. NF

2. NF

3. NF

Anmerkungen



Person		
SV Nr.	Namen	Auto
1234	Max Mustermann	Skoda Octavia, Skoda Roomster
1235	John Doe	Audi Q7, Skoda Superb



Vorname und Nachname.



Menge von Autos,
Marke und Typenbezeichnung.



- Aufspalten der Attribute in atomare Werte
 - Neue Attributfelder einfügen.
- Inhalte von Wiederholungsgruppen zu einzelnen Datensätzen aufteilen.
 - Relationsschema überdenken und gegebenenfalls neu planen.



Beispiel (NF erfüllt)

Einleitung

Grundlegendes

1. NF

2. NF

3. NF

Anmerkungen



Person		
SV Nr.	Namen	Auto
1234	Max Mustermann	Skoda Oktavia, Skoda Roomster
1235	John Doe	Audi Q7, Skoda Superb

Person				
SV Nr.	Vorname	Nachname	Automarke	Typenbezeichnung
1234	Max	Mustermann	Skoda	Oktavia
1234	Max	Mustermann	Skoda	Roomster
1235	John	Doe	Skoda	Superb
1235	John	Doe	Audi	Q7



Primärschlüssel ?



Beispiel (NF erfüllt)

Einleitung

Grundlegendes

1. NF

2. NF

3. NF

Anmerkungen



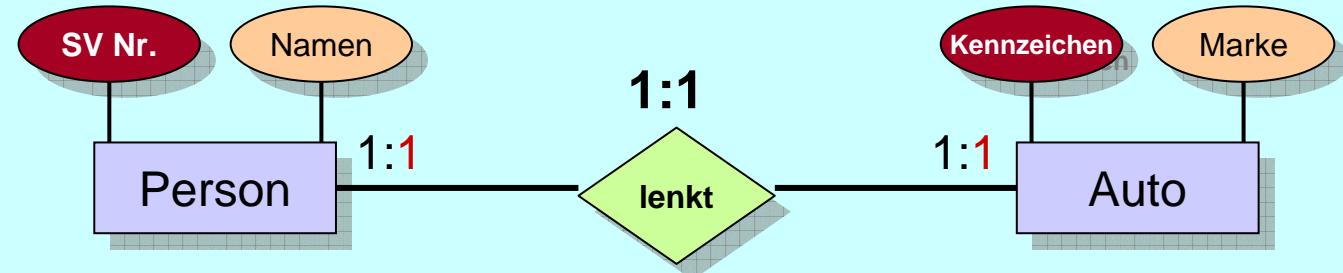
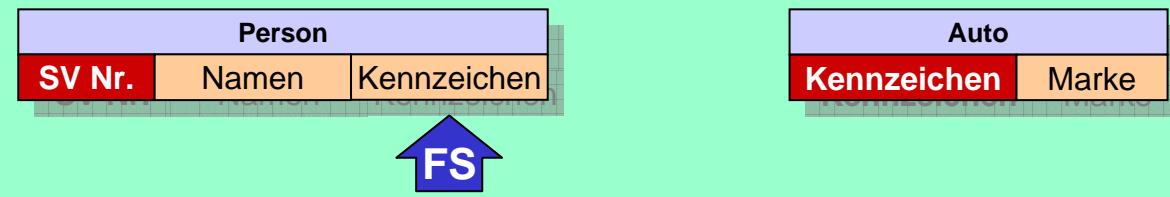
Person				
SV Nr.	Vorname	Nachnamen	Automarke	Typenbezeichnung
1234	Max	Mustermann	Skoda	Oktavia
1234	Max	Mustermann	Skoda	Roomster
1235	John	Doe	Skoda	Superb
1235	John	Doe	Audi	Q7

Auto					
Kennzeichen	Automarke	Typenbezeichnung	SV Nr.	Vorname	Nachnamen
G-123AB	Skoda	Oktavia	1234	Max	Mustermann
G-124BC	Skoda	Roomster	1234	Max	Mustermann
G-158HG	Skoda	Superb	1235	John	Doe
G-887XY	Audi	Q7	1235	John	Doe

1. NF bei 1:1 Beziehungen

[Einleitung](#)[Grundlegendes](#)[1. NF](#)[2. NF](#)[3. NF](#)[Anmerkungen](#)

Konzeptionelle Modellierung

Abgeleitete Relationen
ohne Umsetzung der BeziehungLogisches Modell
1. MöglichkeitLogisches Modell
2. Möglichkeit

1. NF bei 1:n Beziehungen

Einleitung

Grundlegendes

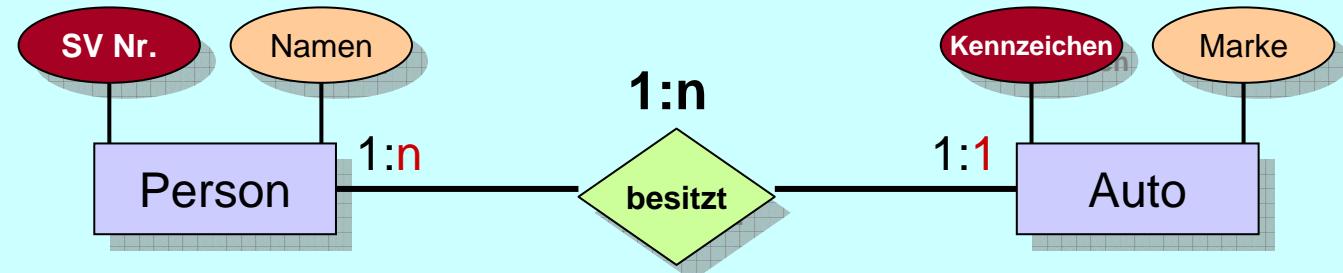
1. NF

2. NF

3. NF

Anmerkungen

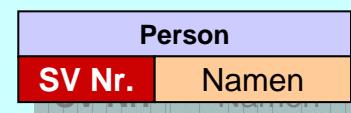
Konzeptionelle Modellierung



Abgeleitete Relationen
ohne Umsetzung der Beziehung



Logisches Modell



Beispiel mit Daten (erfüllt 1. NF)

Person	
SV Nr.	Namens
1234	Max
1235	John

Auto		
Kennzeichen	Marke	SV Nr.
G-123AB	Audi	1235
G-444ZY	VW	1235

Beispiel mit Daten (erfüllt nicht 1. NF)

Person			Auto	
SV Nr.	Namens	Kennz.	Kennzeichen	Marke
1234	Max	G-723KX, G-007	G-123AB	Audi
1235	John	G-123AB, G-444ZY	G-444ZY	VW



1. NF bei n:m Beziehungen

Einleitung

Grundlegendes

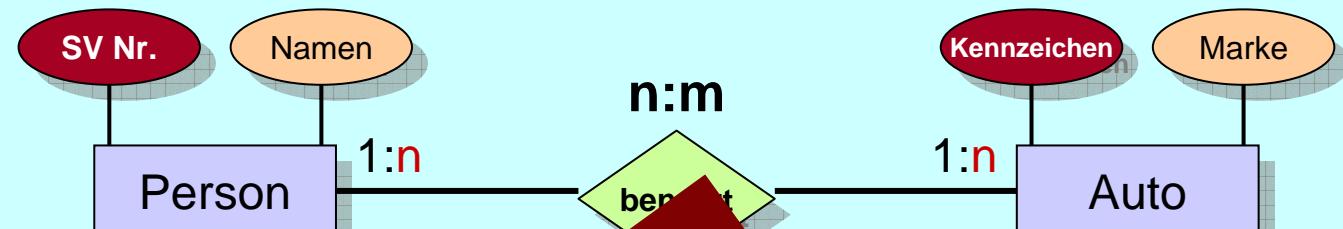
1. NF

2. NF

3. NF

Anmerkungen

Konzeptionelle Modellierung



Abgeleitete Relationen mit Daten

Person	
SV Nr.	Namen
1234	Max
1235	John

Auto	
Kennzeichen	Marke
G-123AB	Audi
G-444ZY	VW

Beispiel mit Daten

Fremdschlüssel in Auto-Tabelle
1. NF nicht erfüllt

Person	
SV Nr.	Namen
1234	Max
1235	John

Auto		
Kennzeichen	Marke	SV Nr.
G-123AB	Audi	1234, 1235
G-444ZY	VW	1234, 1235

Was tun?

Beispiel mit Daten

Fremdschlüssel in Person-Tabelle
1. NF nicht erfüllt

Person		
SV Nr.	Namen	Kennz.
1234	Max	G-123AB, G-444ZY
1235	John	G-123AB, G-444ZY

Auto	
Kennzeichen	Marke
G-123AB	Audi
G-444ZY	VW



1. NF bei n:m Beziehungen

Einleitung

Grundlegendes

1. NF

2. NF

3. NF

Anmerkungen

Logisches Modell
ohne Beziehungen
(Fremdschlüssel)

Person	
SV Nr.	Namen
1234	Max
1235	John

Auto	
Kennzeichen	Marke
G-123AB	Audi
G-444ZY	VW

Erweiterung des logischen Modells
um eine Beziehungstabelle
für die Umsetzung einer n:m Beziehung
(mit kombiniertem PS / FS,
bzw. getrennten Schlüsseln)

Benutzung	
SV Nr.	Kennzeichen
1234	G-123AB
1235	G-444ZY

Benutzung		
ID	SV Nr.	Kennzeichen
1	1234	G-123AB
2	1234	G-444ZY
3	1235	G-123AB
4	1235	G-444ZY

Das Problem

Person	
SV Nr.	Namen
1234	Max
1235	John

Auto		
Kennzeichen	Marke	SV Nr.
G-123AB	Audi	1234, 1235
G-444ZY	VW	1234, 1235

Person		
SV Nr.	Namen	Kennz.
1234	Max	G-123AB, G444ZY
1235	John	G-123AB, G-444ZY

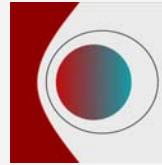
Auto	
Kennzeichen	Marke
G-123AB	Audi
G-444ZY	VW

Die Lösung

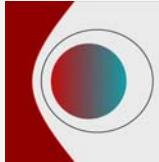
Person	
SV Nr.	Namen
1234	Max
1235	John

Benutzung		
ID	SV Nr.	Kennzeichen
1	1234	G-123AB
2	1234	G-444ZY
3	1235	G-123AB
4	1235	G-444ZY

Auto	
Kennzeichen	Marke
G-123AB	Audi
G-444ZY	VW



2. Normalform



- **Definition:**

Eine Relation ist in zweiter Normalform, wenn die erste Normalform vorliegt, und jedes Nichtschlüsselattribut von jedem Schlüsselkandidaten voll funktional abhängig ist.

- **Schlüsselkandidat**

- Primärschlüssel entspricht einem Attributfeld.
- Zusammengesetzter Primärschlüssel entspricht mehreren Attributfeldern.

Auto	
Kennzeichen	Auto
G-123AB	Skoda O
G-124BC	Skoda R
G-158HG	Skoda S
G-887XY	Audi Q7

Auto		
Motornummer	Fahrgestellnummer	Auto
125	365	Skoda O
135	987	Skoda R
957	547	Skoda S
557	450	Audi Q7





Beispiel (NF nicht erfüllt)

Einleitung

Grundlegendes

1. NF

2. NF

3. NF

Anmerkungen

Auto				
Motornummer	Kraftstoff	Fahrgestellnummer	Automarke	Typenbezeichnung
125	Benzin	365	Skoda	Oktavia
135	Diesel	987	Skoda	Roomster
957	Benzin	547	Skoda	Superb
557	Diesel	450	Audi	Q7



Funktionale nicht von allen
Schlüsselkandidaten abhängig.



- Aufteilen der Daten in mehrere Relationen.
 - Berücksichtigung der vollen funktionalen Abhängigkeit der einzelnen Attribute von den jeweiligen Schlüsselkandidaten.



Beispiel (NF erfüllt)

Einleitung

Grundlegendes

1. NF

2. NF

3. NF

Anmerkungen

Auto				
Motornummer	Kraftstoff	Fahrgestellnummer	Automarke	Typenbezeichnung
125	Benzin	365	Skoda	Oktavia
135	Diesel	987	Skoda	Roomster
957	Benzin	547	Skoda	Superb
557	Diesel	450	Audi	Q7

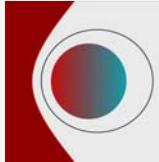


Motor	
Motornummer	Kraftstoff
125	Benzin
135	Diesel
957	Benzin
557	Diesel

Auto				
Motornummer	Fahrgestellnummer	Automarke	Typenbezeichnung	
125	365	Skoda	Oktavia	
135	987	Skoda	Roomster	
957	547	Skoda	Superb	
557	450	Audi	Q7	



3. Normalform



- **Definition:**

Die dritte Normalform ist erreicht, wenn sich das Relationsschema in 2NF befindet, und kein Nichtschlüssel von einem Schlüsselkandidaten transitiv abhängt.

- Transitive Abhängigkeit

$$P_1 \rightarrow A_1 \wedge A_1 \rightarrow A_2 \Rightarrow P_1 \rightarrow A_2$$

- Das Attribut A_1 ist direkt vom Primärschlüssel P_1 abhängig.
- Das Attribut A_2 ist direkt vom Attribut A_1 , jedoch nicht direkt vom Primärschlüssel P_1 abhängig.
- Somit ist A_2 nur transitiv von P_1 abhängig.





Beispiel (NF nicht erfüllt)

Einleitung

Grundlegendes

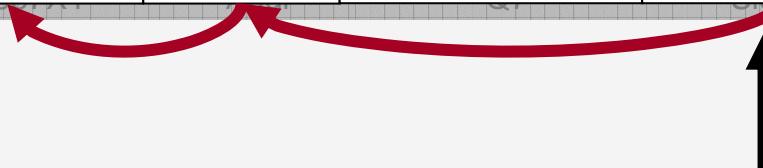
1. NF

2. NF

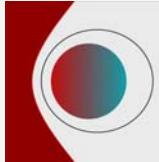
3. NF

Anmerkungen

Auto			
Kennzeichen	Automarke	Typenbezeichnung	Produktionsort
G-123AB	Skoda	Oktavia	Tschechien
G-124BC	Skoda	Roomster	Tschechien
G-158HG	Skoda	Superb	Tschechien
G-887XY	Audi	Q7	Slowakei



Transitive Abhängigkeit des Produktionsortes
von der Nummerntafel über die Automarke.



- Aufteilen der Daten in mehrere Relationen
 - Ausgliedern der transitiven Datensätze.
 - Verwendung des ehemaligen Bindegliedes (vgl. A1) zwischen transitivem Datensatz und Schlüsselkandidaten als Fremdschlüssel.



Beispiel (NF erfüllt)

Einleitung

Grundlegendes

1. NF

2. NF

3. NF

Anmerkungen

Auto			
Kennzeichen	Automarke	Typenbezeichnung	Produktionsort
G-123AB	Skoda	Oktavia	Tschechien
G-124BC	Skoda	Roomster	Tschechien
G-158HG	Skoda	Superb	Tschechien
G-887XY	Audi	Q7	Slowakei



Auto		
Kennzeichen	Automarke	Typenbezeichnung
G-123AB	Skoda	Oktavia
G-124BC	Skoda	Roomster
G-158HG	Skoda	Superb
G-887XY	Audi	Q7

Prodktion	
Automarke	Produktionsort
Skoda	Tschechien
Audi	Slowakei

Automarke als Fremdschlüssel.



Einleitung

Grundlegendes

1. NF

2. NF

3. NF

Anmerkungen

Anmerkungen



- Boyce-Codd-Normalform

Eine Relation ist in BCNF, wenn sie die Voraussetzungen der 3NF erfüllt, und jede Determinante (Attributmenge, von der andere Attribute funktional abhängen) Schlüsselkandidat ist.

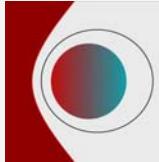
- 4. Normalform

Eine Datenbank ist dann in der 4NF, wenn sie nur noch triviale mehrwertige Abhängigkeiten enthält oder die nicht-trivialen mehrwertigen Abhängigkeiten von Schlüsselkandidaten ausgehen.

- 5. Normalform

Die 5NF vereinfacht Relationen soweit, dass durch Projektions- und Verbundsoperationen die Informationen der ursprünglichen Relation wieder hergestellt werden.





*The key, the whole key, and nothing but the key
- so help me Codd*

- Alle Werte beziehen sich auf den Schlüssel - **1NF**.
- Bei zusammengesetzten Schlüsseln beziehen sich die Informationen jeweils auf den gesamten Schlüssel - **2NF**.
- Die Werte hängen nur vom Schlüssel ab, und nicht von zusätzlichen Werten - **3NF**.





1. Ist die Tabelle in 1. Normalform und besteht der Primärschlüssel aus nur einem Attribut, so liegt automatisch die 2. Normalform vor.

2. Ist eine Tabelle in 2. Normalform und besitzt sie außer dem Primärschlüssel höchstens ein weiteres Attribut, so liegt die Tabelle in 3. Normalform vor.





Einleitung

Grundlegendes

1. NF

2. NF

3. NF

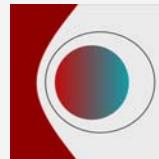
Anmerkungen

- 31.10.2011: A109
- Fragen ?

Normalformen

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Structured Query Language

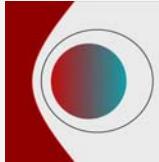
Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



- Grundlagen
- Tabelle erstellen (Primärschlüssel, Bedingungen)
- Dateneingabe (insert into, update)
- Löschen von Daten (delete)
- Datenabfrage (select)
- Ergänzung



Grundlagen

Erstellen

Eingabe

Löschen

Abfrage

Ergänzung

Grundlagen



Grundlagen

Erstellen

Eingabe

Löschen

Abfrage

Ergänzung

- Structured Query Language: SQL.
- Einfache Sprache
 - Definition von Datenbereichen.
 - Manipulation von Daten.
 - Abfrage von Daten.
- Einsatz bei Relationalen Datenbanken.
- Standardisiert durch
 - ISO (International Organisation for Standardization).
 - ANSI (American National Standard Institute).
- Semantisch an Englische Sprache angelehnt.



Geschichte

Grundlagen

Erstellen

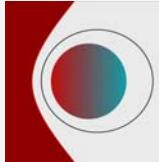
Eingabe

Löschen

Abfrage

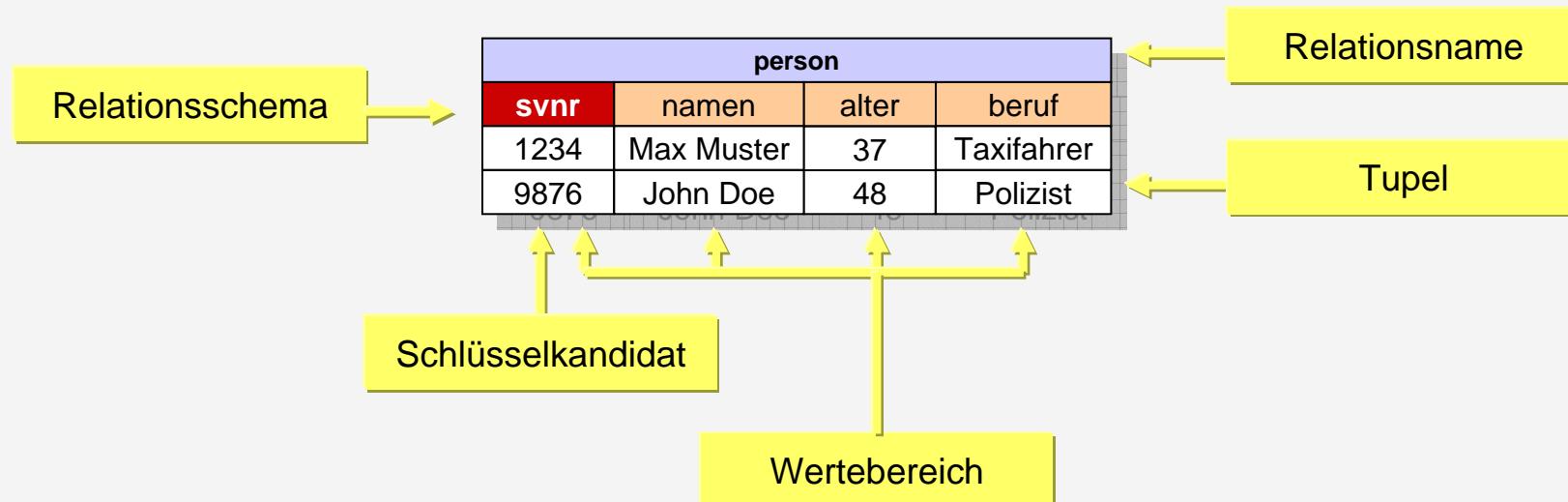
Ergänzung

- Ca. 1979: Structured English Query Language (SEQUEL).
- 1979: SQL gelangt mit Oracle V2 auf den Markt.
- 1986: SQL1 wird von ANSI als Standard verabschiedet.
- 1987: SQL1 wird von ISO als Standard verabschiedet.
- 1992: SQL2 bzw. SQL-92 wird von ISO verabschiedet.
- 1999: SQL3 bzw. SQL:1999 wird von ISO verabschiedet.
- 2003: SQL:2003 wird von ISO verabschiedet.



- Definition des Datenbankschemas
 - Data Definition Language (DDL).
 - Aufbau der Tabellen, Datentypen,....
- Datenmanipulation
 - Data Manipulation Language (DML).
 - Eingabe von Daten, Löschen von Daten,....
- Rechteverwaltung
 - Data Control Language (DCL).
 - Benutzerabhängige Datenzugriffe.

- Begrifflichkeiten in einer RDB.





Grundlagen

Erstellen

Eingabe

Löschen

Abfrage

Ergänzung

Erstellen



Grundsätzliches

Grundlagen

Erstellen

Eingabe

Löschen

Abfrage

Ergänzung

- Ziel: Erstellen einer leeren Tabelle
- Voraussetzung: keine
- Überlegung
 - Wie soll die Tabelle heißen?
 - Welche Attribute sollen gespeichert werden?
 - Welches Attribut soll als Primär-Schlüssel verwendet werden?

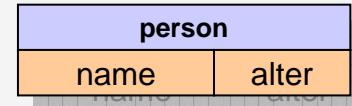


- Syntax:

`CREATE TABLE tablename (attr_1 typ_1,...);`

- Beispiel:

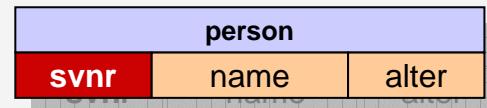
`CREATE TABLE person (name varchar, alter integer);`



- Einsatz eines Primärschlüssels

- Beispiel:

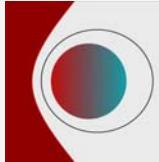
`CREATE TABLE person (svnr integer primary key, name varchar, alter integer);`





- Datentypen (Auszug)

- Serial: fortlaufender ganzzahliger Zähler (1,...,n)
- Integer: 1, 2, 3, 4, 5, ...
- Float: 1.21254, 1.999, 0.0007, ...
- Numeric: 1, 3.15, 5, 28.778, 2770, ...
- Varchar („zeichenkette“): ‚Hallo Welt!‘, ‚123vier‘, ...
- Date: 2009-02-29
- Time: 11:55:30
- Boolean: true, false



- Einzigartigkeit d. Wertes: **UNIQUE**
- Eingabe erforderlich: **NOT NULL**
- Definierter Wertebereich **CHECK(attr_i ...)**
 - **IN (val_1, ..., val_n)**
 - Aus vorgegebener Menge.
 - **BETWEEN val_1 AND val_2**
 - Zwischen den Werten **val_1** und **val_2**.

```
CREATE TABLE person (
    svnr integer PRIMARY KEY,
    name varchar NOT NULL UNIQUE,
    alter integer NOT NULL CHECK(alter BETWEEN 0 AND 100),
    beruf varchar NOT NULL);
```



Grundlagen

Erstellen

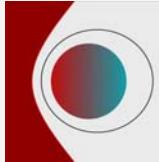
Eingabe

Löschen

Abfrage

Ergänzung

Eingabe



Grundsätzliches (a)

Grundlagen Erstellen **Eingabe** Löschen Abfrage Ergänzung

- Ziel: Hinzufügen eines Datensatzes zu einer bestehenden Tabelle.
- Voraussetzung: Tabelle muss existieren.

- Überlegung
 - Welche Tabelle möchte ich befüllen?
 - Welche Werte sollen gespeichert werden?
 - Ist mein Schlüsselkandidat einzigartig?



Grundlagen

Erstellen

Eingabe

Löschen

Abfrage

Ergänzung

- Syntax:

`INSERT INTO tablename (attr_1,...) VALUES (values_1,...);`

- Beispiel:

`INSERT INTO person (svnr,name,alter) VALUES (1234,'Max Muster',37);`

person		
svnr	name	alter
1234	Max Muster	37



- Kein Wert: `NULL`
- Boolsscher Wert: `true` / `false`
- Datum: `'yyyy-mm-dd'`
- Uhrzeit: `'hh:mm'` oder `'hh:mm:ss'` oder `'hh:mm:ss.ssss'`

```
INSERT INTO logbuch
(zeile, datum, uhrzeit, vorfall)
VALUES(332, '2007-10-24', '14:21', NULL);
```



logbuch			
zeile	datum	uhrzeit	vorfall
332	2007-10-24	14:21	NULL



- Ziel: Verändern bestehender Attributwerte.
- Voraussetzung: Tabelle mit zu verändernden Datensatz muss existieren.

- Überlegung
 - Aus welcher Tabelle möchte ich Attributwerte ändern?
 - Welche Attributwerte möchte ich ändern?
 - Zu welchem Datensatz gehören die Attributwerte?



Syntax (b)

Grundlagen Erstellen **Eingabe** Löschen Abfrage Ergänzung

- Syntax:

`UPDATE tabname SET attr = value;`

- Beispiel:

`UPDATE person SET alter = 41;`

person		
svnr	name	alter
1234	Max Muster	41
9876	John Doe	41

- Syntax:

`UPDATE tabname SET attr = value WHERE bedingung;`

- Beispiel:

`UPDATE person SET alter = 41 WHERE name = 'Max Muster';`

person		
svnr	name	alter
1234	Max Muster	41
9876	John Doe	48



Grundlagen

Erstellen

Eingabe

Löschen

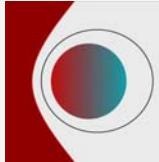
Abfrage

Ergänzung

Löschen



- Ziel: Löschen eines bestimmten Datensatzes bzw. alle bestehenden Datensätze.
- Voraussetzung: Tabelle muss existieren. Es müssen nicht zwangsläufig Datensätze vorhanden sein um ein Löschen ohne Fehlermeldung durchführen zu können.
- Überlegung
 - Aus welcher Tabelle möchte ich Daten löschen?
 - Welche Daten möchte ich löschen?
 - Möchte ich alle Daten löschen?



- Syntax:

`DELETE FROM tablename;`

- Beispiel:

`DELETE FROM person;`

person		
svnr	name	alter
1234	Max Muster	37
9876	John Doe	48

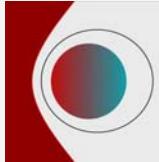
- Syntax:

`DELETE FROM tablename WHERE bedingung;`

- Beispiel:

`DELETE FROM person WHERE svnr = 9876;`

person		
svnr	name	alter
1234	Max Muster	37
9876	John Doe	48



Grundlagen

Erstellen

Eingabe

Löschen

Abfrage

Ergänzung

Abfrage



Grundsätzliches

Grundlagen Erstellen Eingabe Löschen **Abfrage** Ergänzung

- Ziel: Abfragen von Attributwerten.
- Voraussetzung: Tabelle Datensatz muss existieren.
- Überlegung
 - Aus welcher Tabelle möchte ich Attributwerte abfragen?
 - Welche Attributwerte möchte abfragen?
 - Wie möchte ich das Ergebnis einschränken?



Syntax ohne Bedingung

Grundlagen Erstellen Eingabe Löschen **Abfrage** Ergänzung

- Syntax:

`SELECT * FROM tablename;`

- Beispiel:

`SELECT * FROM person;`

person		
svnr	name	alter
1234	Max Muster	37
9876	John Doe	48



1234	Max Muster	37
9876	John Doe	48

- Syntax:

`SELECT attr_1,... FROM tablename;`

- Beispiel:

`SELECT svnr, alter FROM person;`

person		
svnr	name	alter
1234	Max Muster	37
9876	John Doe	48



1234	37
9876	48



- Syntax:

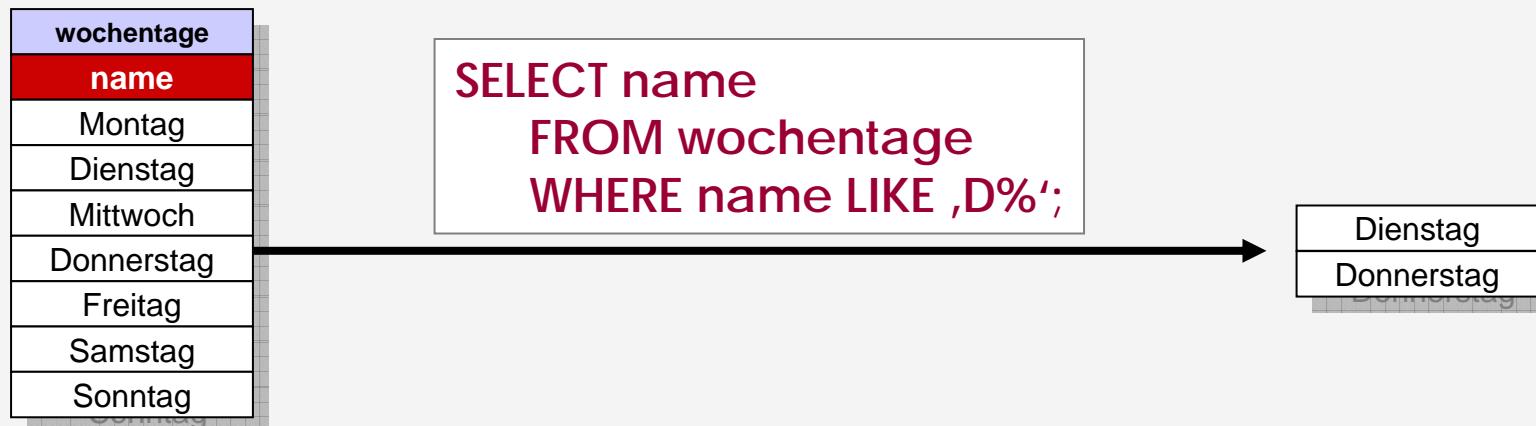
`SELECT attr_1,... FROM tablename WHERE bedingung;`

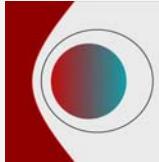
- Bedingungen

- Logische Bedingung
 - `attr [Operator] value`
 - `=, <>` bzw. `!=, <, >, <=, >=`
- BETWEEN – Bedingung
 - `attr BETWEEN value_min AND value_max`
- IN – Bedingung
 - `attr IN (value_1, ...)`



- Bedingungen
 - LIKE – Bedingung
 - attr LIKE [Muster]
 - Musterbeispiele
 - ,R%' ... Wort, das mit „R“ beginnt.
 - ,%ung' ... Wort, das auf „ung“ endet.
 - ,%ck%' ... Wort, das „ck“ beinhaltet.





Syntax Beispiele

Grundlagen

Erstellen

Eingabe

Löschen

Abfrage

Ergänzung

person		
svnr	name	alter
1234	Max Muster	37
9876	John Doe	48

SELECT * FROM person WHERE svnr = 1234;

1234	Max Muster	37
------	------------	----

SELECT * FROM person WHERE svnr <> 1234;

9876	John Doe	48
------	----------	----

SELECT * FROM person WHERE alter > 50;

--	--	--

SELECT svnr FROM person WHERE alter BETWEEN 0 AND 100;

1234
9876

SELECT svnr FROM person WHERE name IN ('Max Muster', 'Fritz');

1234

SELECT name FROM person WHERE name LIKE '%er';

Max Muster



- Syntax:

`SELECT ... FROM ... WHERE bedingung_1 [Verknpfg] bedingung_2 ;`

- Verknüpfungsoperatoren

- AND
- OR

- Beispiel:

`SELECT * FROM person WHERE alter > 20 AND alter < 50 ;`

`SELECT * FROM person WHERE alter < 20 OR alter > 50;`

- Anmerkung

- Einsatz von Klammern kann notwendig sein!

		Konjunktion		Disjunktion		Negation	
		Λ	0	1	0	1	¬
Λ		Λ	0	1	0	1	0
0	Λ	0	0	0	0	1	1
1	0	1	0	1	1	1	0



Grundlagen

Erstellen

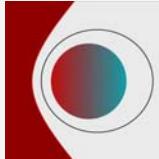
Eingabe

Löschen

Abfrage

Ergänzung

Ergänzung



- Aggregatfunktionen

- **AVG()** ... Mittelwert der Spalte.
- **COUNT()** ... Anzahl der Tupel.
- **MAX()** ... Maximaler Wert der Spalte.
- **MIN()** ... Minimaler Wert der Spalte.
- **STDDEV()** ... Standardabweichung der Spalte.
- **SUM()** ... Summe der Spalte.
- **VARIANCE()** ... Varianz der Spalte.

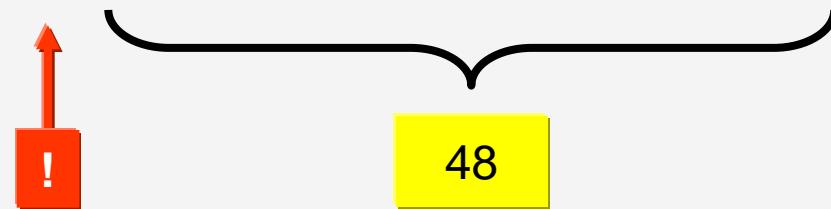
```
SELECT COUNT(*) FROM person;
```

```
SELECT MAX(alter) FROM person WHERE svnr > 5000;
```



- Bei Verschachtelung von Abfragen auf Menge der möglichen Abfrageergebnisse achten!

```
SELECT * FROM person WHERE alter = (SELECT MAX(alter) FROM person);
```



```
SELECT * FROM person WHERE name IN(SELECT name FROM person);
```



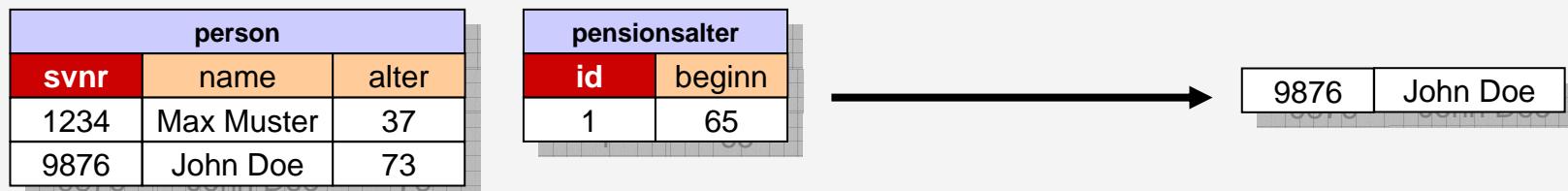


- Syntax:

```
SELECT tab_i.attr_a  
      FROM tab_i, tab_j  
     WHERE tab_i.attr_a = tab_j.attr_b;
```

- Beispiel:

```
SELECT person.svnr, person.name  
      FROM person, pensionsalter  
     WHERE person.alter >= pensionsalter.beginn;
```





Grundlagen

Erstellen

Eingabe

Löschen

Abfrage

Ergänzung

- Ziel: Ablegen eines Abfrageergebnisses zur weiteren Verwendung
- Voraussetzung: SQL-Anweisung für eine Abfrage
- Überlegung
 - Unter welchem Namen soll das Abfrageergebnis abgelegt werden?
 - Wie lautet die SQL-Anweisung?



- Syntax:

`CREATE VIEW viewname AS abfrage;`

- Beispiel:

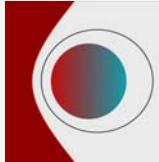
`CREATE VIEW alte_person AS SELECT * FROM person WHERE alter > 50;`

person		
svnr	name	alter
1234	Max Muster	41
9876	John Doe	53

alte_person		
svnr	name	alter
9876	John Doe	53

- Anmerkung

- Es wird eine „synthetische“ Tabelle erstellt, die physisch nicht existiert. Nicht datenverändernde Tätigkeiten (Abfragen) können wie bei einer „normalen“ Tabelle durchgeführt werden.



- Ausgabe 2. Übungsprogramm
- Softwarevorstellung
 - pgAdmin (Desktop & Web)
 - Notepad++
 - FileZilla
- 07.11.2011: A109
- Fragen ?

Structured Query Language

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

SQL Übung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Zeitplan

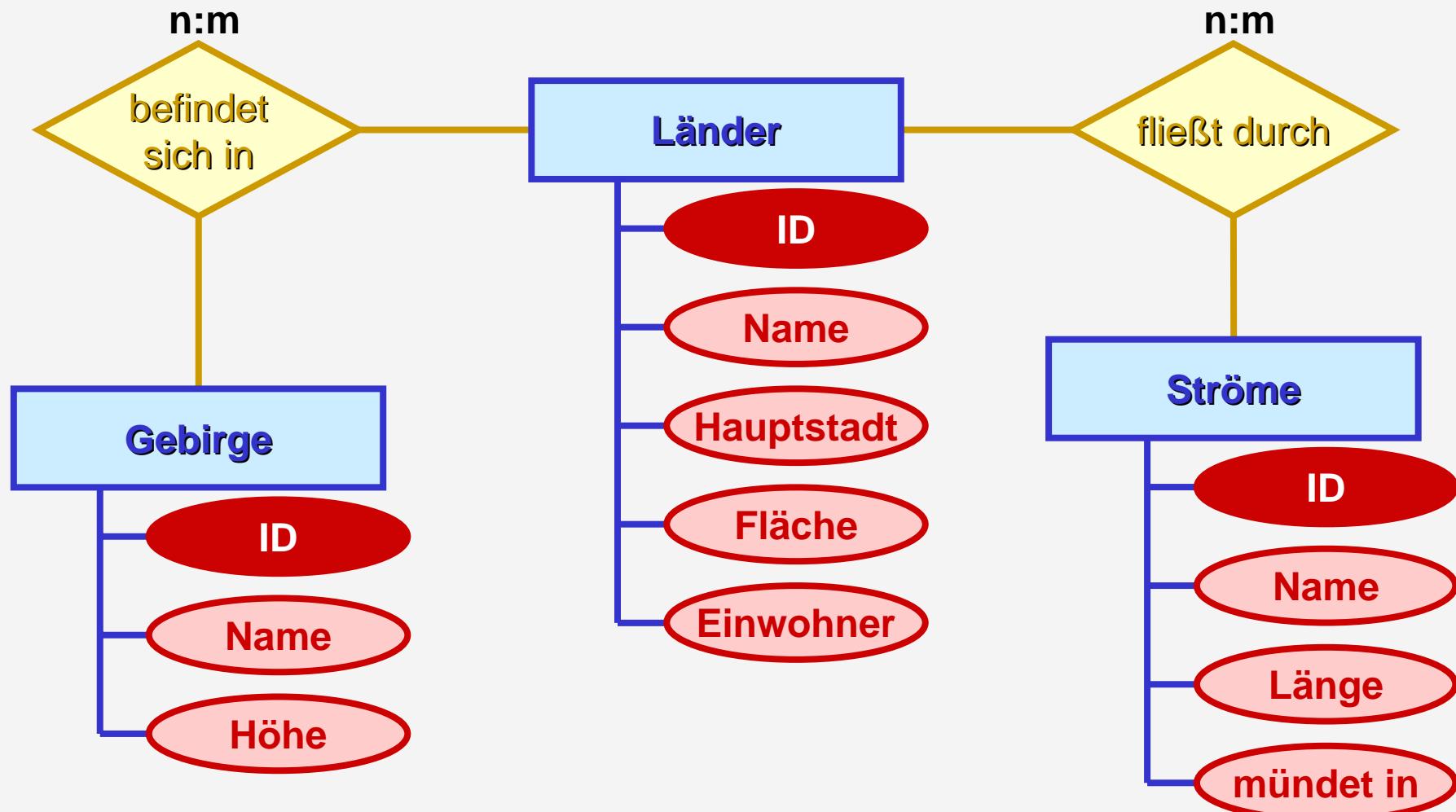
Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



- pgAdminIII
- Tabelle erstellen – *create table...*
- Dateneingabe – *insert into...*
- Datenabfrage – *select...*
- Datenänderung – *update...*
- Löschen von Daten – *delete...*
- Tabellen löschen – *drop...*



Datenstruktur





Tabellen erstellen

- Länder Europas (5, inkl. Frankreich u. Spanien)
 - ID, Name, Hauptstadt, Einwohner, Fläche in km²
- Gebirge Europas (3, inkl. Alpen)
 - ID, Name, Höhe der höchsten Erhebung
- Ströme Europas (3, inkl. Rhein & Donau)
 - ID, Name, Länge, mündet in
- Beziehungstabellen zur Realisierung der n:m Beziehungen
- **Tabellennamen und Attributnamen klein schreiben, keine Sonderzeichen und keine Leerzeichen!**



Ganz einfache Datenabfrage

- Fragen Sie **alle** Länder ab.
- Fragen Sie **nur drei** (beliebige) Länder ab.
- Fragen Sie alle Länder in **umgekehrter alphabetischer Reihenfolge** ab.
- Fragen Sie die **Anzahl** der Ströme ab.
- Fragen Sie den Wert der **mittleren Höhe** der Gebirge ab.
- Fragen Sie die **Einwohnerdichte** (EW/Fläche) aller Länder ab.



Einfache Datenabfrage

- Welche Länder sind **größer als** 100.000km²?
- Wie hoch sind die **Alpen**?
- Wie lange ist **Donau** und **Rhein** zusammen?
- Wie lautet der Name des **kürzesten** Stroms?
- Welche Ströme münden in das **Schwarze Meer**?



Komplexe Datenabfrage

- In welchen **Ländern** befinden sich die **Alpen**?
 - Erzeugen Sie für diese Antwort einen **View!**
- Wie hoch ist die **durchschnittliche** Einwohneranzahl der **Alpenländer**?
- Besitzen **Frankreich** und **Spanien** dieselbe Hauptstadt?
- Durch welche **Länder** fließt der **längste Strom**?
- **Wie viele Länder** gibt es, durch die der **längste Strom** fließt und sich gleichzeitig das **höchste Gebirge** befindet?



Datenänderung

- Ändern Sie **alle Längen** der Ströme auf 1.000km.
- Ändern Sie die Einwohnerzahlen der Länder auf 15.000.000 und **gleichzeitig** die Flächen der Länder auf 10.000km².
- Ändern Sie die Höhe der **Alpen** auf die **mittlere Höhe** der Gebirge.
- Ändern Sie den Hauptstadtnamen jener Länder auf *Paradise City*, deren ID **zwischen 5 und 15** liegt.



Löschen von Daten

- Löschen Sie **alle** Ströme.
- Löschen Sie alle Länder **mit ungerader ID**.



Tabellen löschen

- Löschen Sie den **View**.
- Löschen Sie die restlichen Tabellen **AUSSER** *geometry_columns* und *spacial_ref_sys*!

SQL Übung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

spatialSQL

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



- OGC konforme Geometrien
- Referenzsysteme
- PostGIS
- Erstellen von Geometrien (SQL)
- Eingabe von Geometrien (SQL)
- Abfrage von Geometrien (SQL)



OGC

Referenz

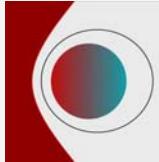
PostGIS

Erstellen

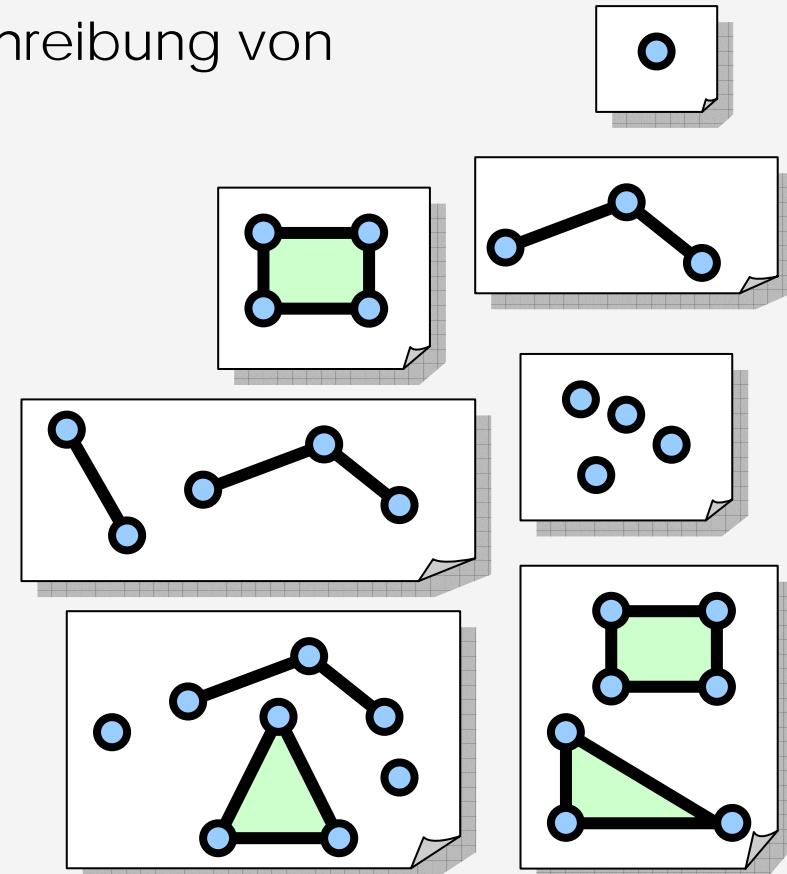
Eingabe

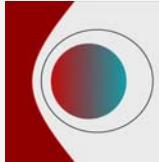
Abfrage

OGC konforme Geometrien

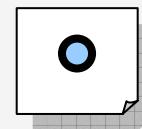


- Well Known Text (WKT)
 - ASCII basierte Geometriebeschreibung von
 - ...punktförmigen,
 - ...linienförmigen und
 - ...flächenhaften
 - ...Objekten
 - Geometriertypen
 - Point, Multipoint
 - Linestring, Multilinestring
 - Polygon, Multipolygon
 - Geometrycollection

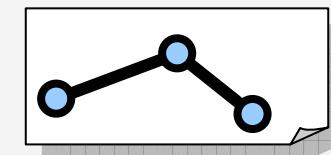




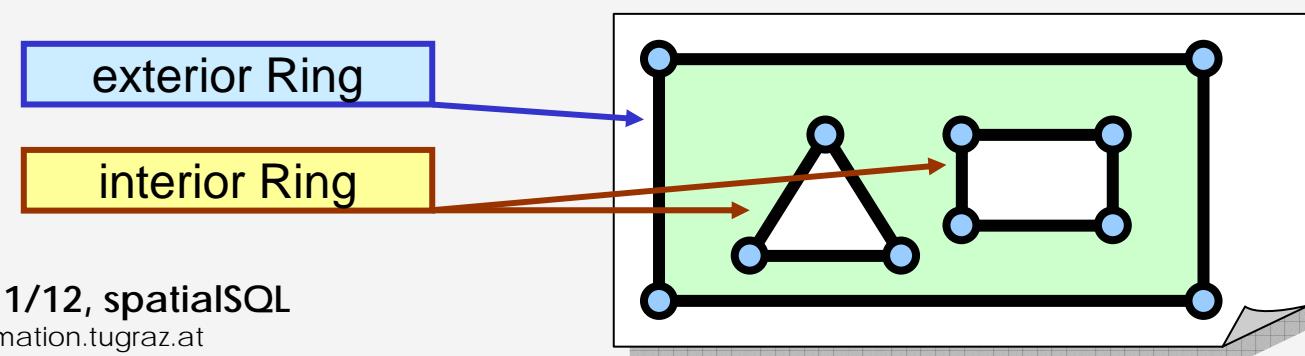
- Punkt
 - POINT(rechts hoch)
 - Bsp.: POINT(10 34)



- Linie
 - LINESTRING(r_1 h_1, r_2 h_2,...r_n h_n)
 - Bsp.: LINESTRING(0 0,10 20,43 100)



- Fläche
 - POLYGON((exteriorring),(interiorring_1),...,(interiorring_n))
 - Bsp.: POLYGON((0 0,20 0,20 20,0 20,0 0),(5 5,15 5,5 15,5 5))





Beispiel WKT

OGC

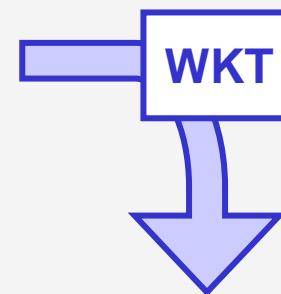
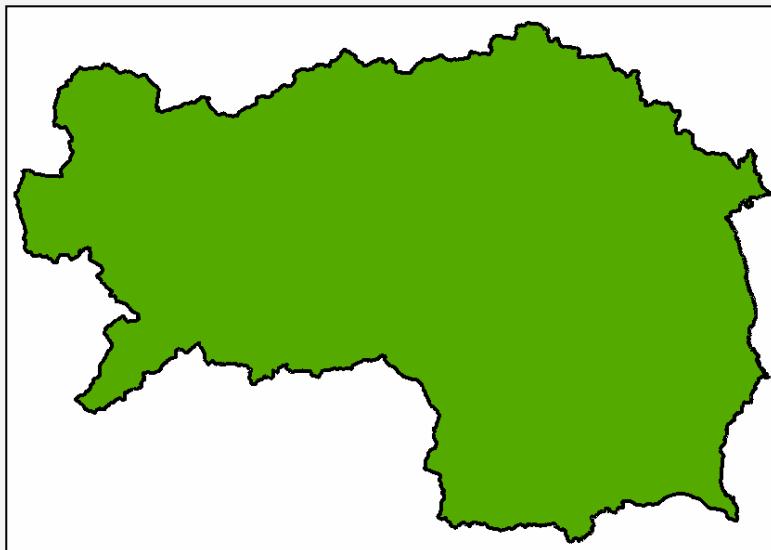
Referenz

PostGIS

Erstellen

Eingabe

Abfrage

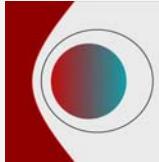


```
POLYGON((536268.447375245 5162182.36750409, 536246.281335426 5162171.68884389, 536164.109621922 5162185.30861738, 536149.137938152 5162200.82056162, 536122.076585715 5162228.86149391, 536079.65890 972799108 5162438.86908438, 535602.8 ...))
```

```
POLYGON(( 536268.447375245 5162182.36750409,  
          536246.281335426 5162171.68884389,  
          536164.109621922 5162185.30861738,  
          536149.137938152 5162200.82056162,  
          ...           ... ))
```

Rechtswert

Hochwert

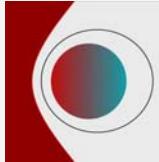


Referenzsysteme



Koordinaten

OGC	Referenz	PostGIS	Erstellen	Eingabe	Abfrage
• 15.437703		47.073486			WGS 84
• 533230.94		5213423.67	UTM 33 N		WGS 84
• -67939.84		5215318.64	GK M34		Bessel
• 559815.92		354812.54	Lambert		Bessel
• 682060.16		5215318.64	BMN M34		Bessel
			Projektion	Ellipsoid	



Das Geoid

OGC

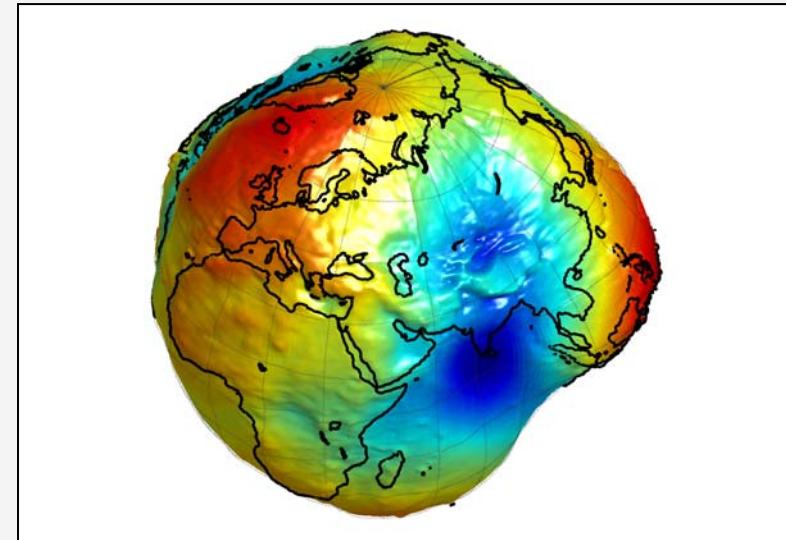
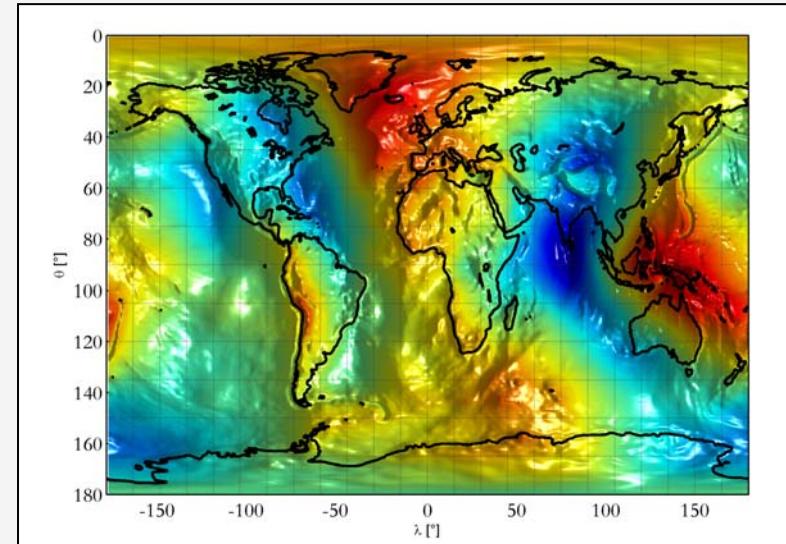
Referenz

PostGIS

Erstellen

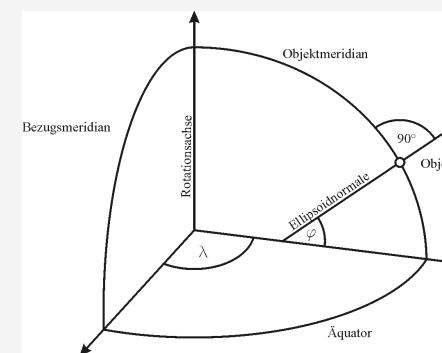
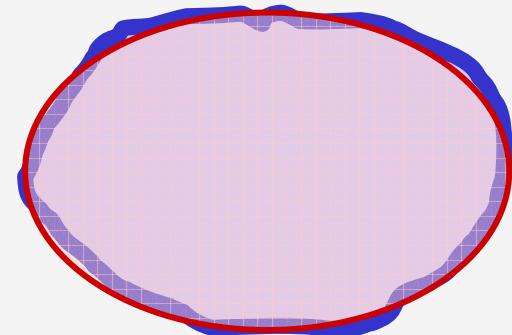
Eingabe

Abfrage





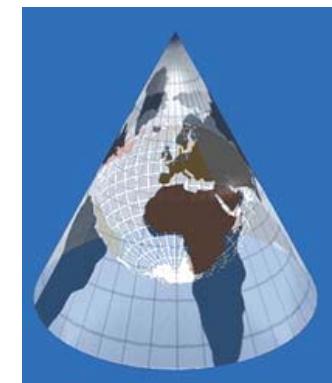
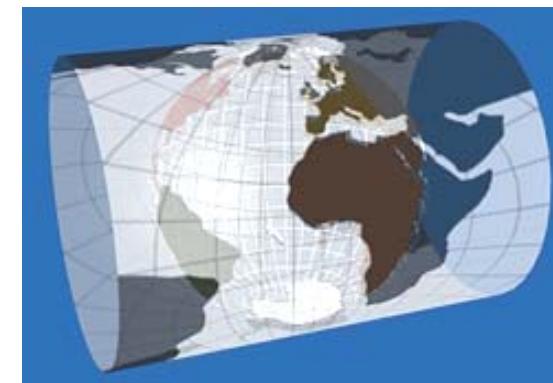
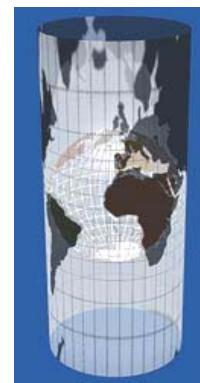
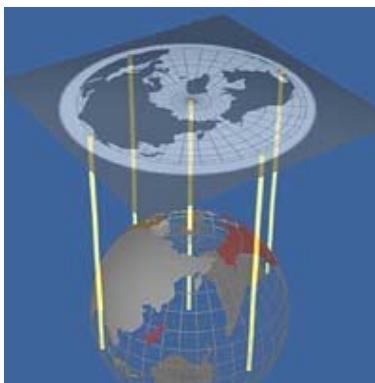
- Einfache Beschreibung des **Geoids**
- Näherungsweise **Rotationsellipsoid**
 - Berechnungen am Ellipsoid möglich
 - Globale Anpassung (z.B.: WGS 84)
 - Lokale Anpassung (z.B.: Bessel)
- Ortsbeschreibung
 - Länge: Schnittwinkel zwischen **Bezugsmeridianebene** und **Objektmeridianebene**
 - Breite: Schnittwinkel zwischen **Ellipsoidnormalen** durch Objekt und **Äquatorebene**





OGC Referenz PostGIS Erstellen Eingabe Abfrage

- Ellipsoid nicht planar darstellbar
- Umbilden auf Hilfsflächen, die planar darstellbar ist
 - Ebene (z.B.: Azimutalprojektion)
 - Zylinder (z.B.: Merkator Projektion)
 - Kegel (z.B.: Lambert)



[OGC](#)[Referenz](#)[PostGIS](#)[Erstellen](#)[Eingabe](#)[Abfrage](#)

- Eindeutige Ansprache von
 - Ellipsoid
 - und Projektionsmethode
- Zuweisung eines numerischen Codes (**SRID**)
 - Spatial Reference Identifier
 - Überblick über Codes: <http://www.spatialreference.org/>
- SRID auch als **EPSG Code** bezeichnet
 - European Petroleum Survey Group
 - <http://www.epsg.org>
- Beispiele
 - 4326: geographische Koordinaten WGS 84
 - 32633: UTM 33 N Projektion bezogen auf WGS 84



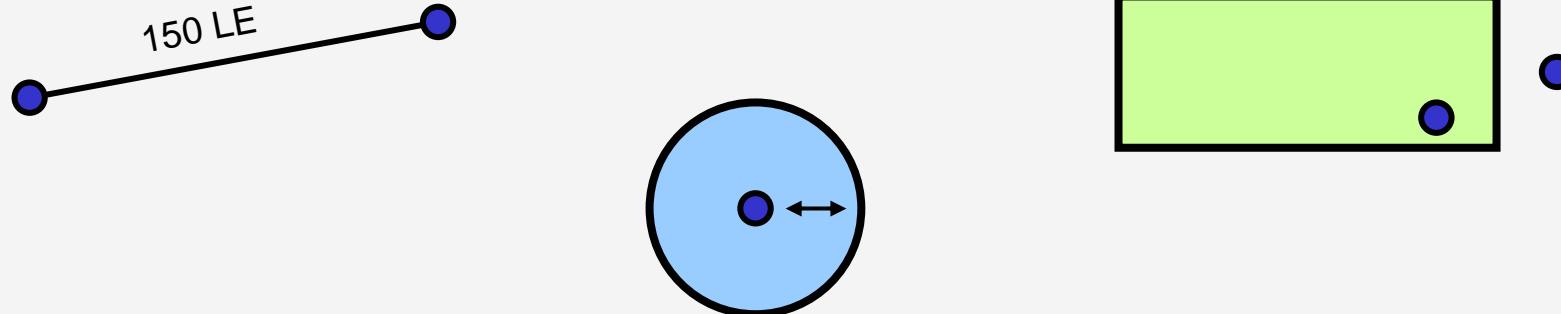
PostGIS



- Anforderung an Geo-Datenbank
 - „Verstehen“ von Geometrien
 - Form
 - Dimensionen
 - Spatial Reference System
 - Anwenden von Analysefunktionen auf Geometrien
 - Puffer, Überschneidungen, Distanz, ...
 - Import / Export von Geometrien
 - Interoperabilität
 - WKT
 - WKB
 - Schnittstelle für Programmiersprachen
 - Interaktion



- Erweiterung eines PostgreSQL DBMS.
 - Verwaltung von Geodaten.
 - Unterstützung von Simple Features gemäß OGC – Spezifikationen.
 - Import vieler GIS Formate (u.a. mittels FME).
 - Unterstützung von gängigen GIS – Analysefunktionen.





Erstellen von Geometrieneattributen



- Vorbereiten der Tabelle

- Normales Erstellen einer Tabelle (CREATE TABLE) ohne Geometriearribut!
- Hinzufügen eines Geometriearributes

- Syntax: `SELECT addgeometrycolumn(tabname,
attr,
srid,
geometriertyp,
dimension);`

- tabname Name der Tabelle als String (,')
- attr Name des Attrib. mit Geom.Information als String (,')
- srid EPSG-Code als Zahl
- geometriertyp OGC-Geometrie-Typ als String (,')
- dimension 2d bzw. 3d als Zahl



- Beispiel

- Erstellen der Grundtabelle:

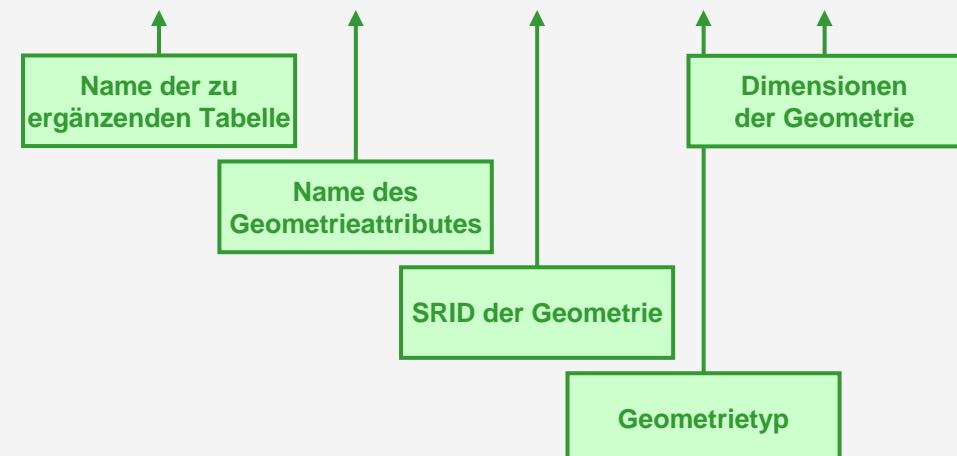
```
CREATE TABLE stadt (id integer primary key, name varchar);
```

stadt	
id	name
1	Wien

- Erweitern der Grundtabelle um eine Geometriespalte

```
SELECT addgeometrycolumn('stadt','geom',4326,'POINT',2);
```

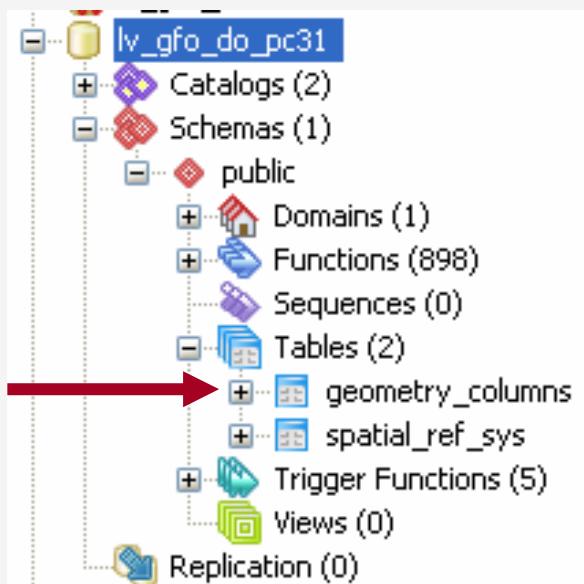
stadt		
id	name	geom
1	Wien	





addgeometrycolumn Funktion

- Erweitert eine bestehende Tabelle um ein Geometriearribut
- Fügt der Sammlung aller Geometriespalten einer Datenbank (Tabelle *geometry_columns*) einen Eintrag hinzu.



The screenshot shows the 'Edit Data' window for the 'geometry_columns' table. The table has the following structure:

oid	f_table_catalog	f_table_schema	f_table_name	f_geometry_column	coord_dimension	srid	type
*							

`select addgeometrycolumn(...);`

The screenshot shows the 'Edit Data' window again, but now it contains one row of data:

oid	f_table_catalog	f_table_schema	f_table_name	f_geometry_column	coord_dimension	srid	type	
1	4328999	"	public	stadt	geom	2	4326	POINT
*								



Eingabe von Geometrien



- „Normales“ Eingeben von Daten in eine Tabelle
 - Syntax: `INSERT INTO tabname (attr) VALUES (value);`
- Wert des Geometriearributes
 - WKT der Geometrie muss in WKB umgewandelt werden
 - Syntax: `geometryfromtext(WKT,SRID)`
 - Beispiel: `geometryfromtext('POINT(1 1)',32633)`
 - Beispiel für komplette Dateneingabe:
`INSERT INTO stadt (id, name, geom)`
`VALUES (1,'Graz', geometryfromtext('POINT(1 1)',32633));`





Eingaberesultat

OGC

Referenz

PostGIS

Erstellen

Eingabe

Abfrage

Query - lv_gfo_master on postgres@129.27.89.66:5432 *

```
File Edit Query Favourites Macros View Help
[File, Edit, Query, Favourites, Macros, View, Help icons] lv_gfo_master on p
INSERT INTO stadt (id, name, geom)
VALUES (1, 'Graz', geometryfromtext('POINT(15.43 47.07)', 4326));
```

Output pane

Data Output Explain Messages History

Query returned successfully: 1 rows affected, 16 ms execution time.

1 rows affected. Unix Ln 2 Col 65 Ch 100

Query - lv_gfo_master on postgres@129.27.89.66:5432 *

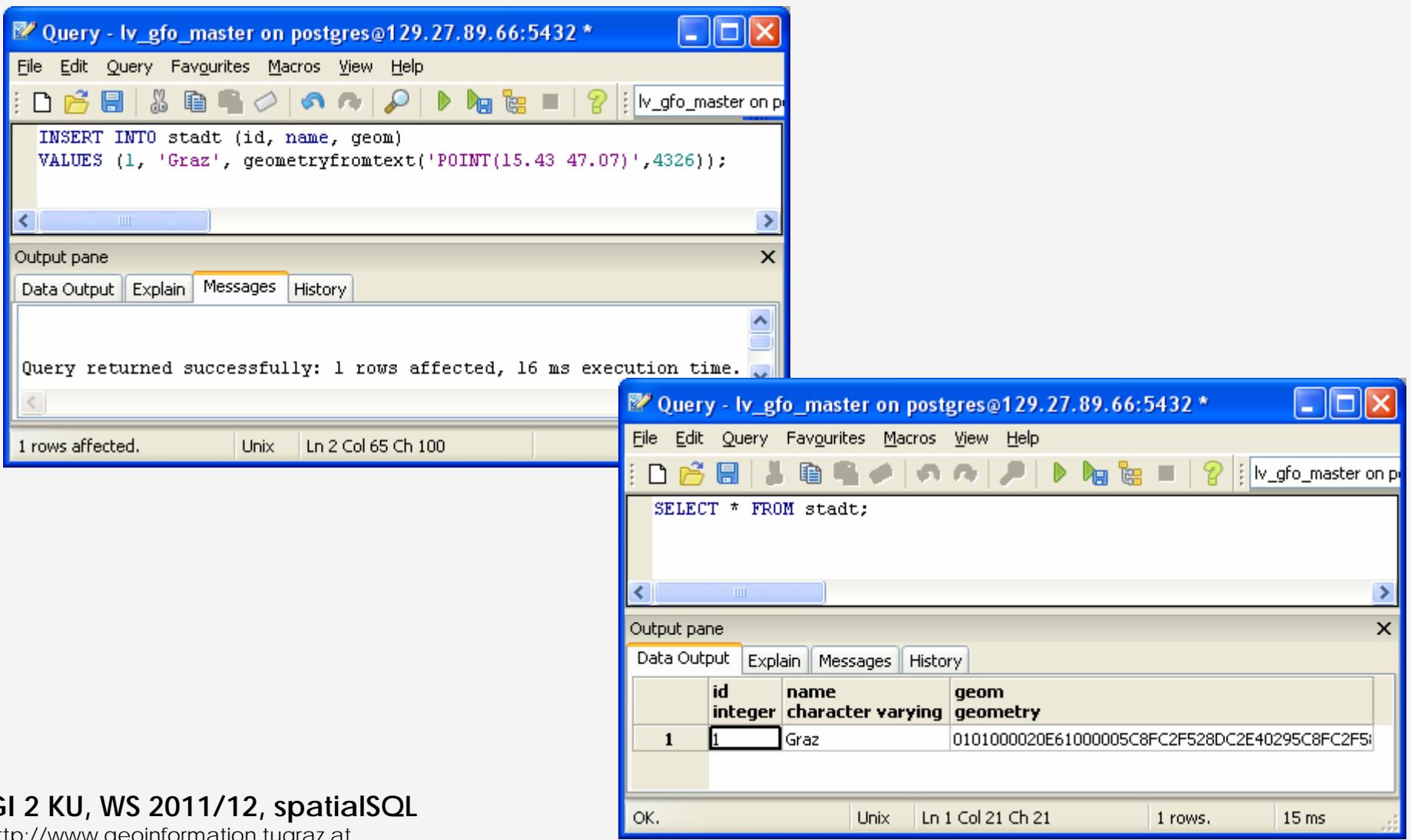
```
File Edit Query Favourites Macros View Help
[File, Edit, Query, Favourites, Macros, View, Help icons] lv_gfo_master on p
SELECT * FROM stadt;
```

Output pane

Data Output Explain Messages History

	id integer	name character varying	geom geometry
1	1	Graz	0101000020E61000005C8FC2F528DC2E40295C8FC2F5;

OK. Unix Ln 1 Col 21 Ch 21 1 rows. 15 ms





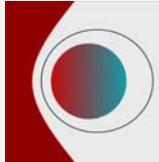
Abfrage von Geometrien



- Unaufbereitete Abfrage von Geometrien liefert ein (für uns) nicht interpretierbares Ergebnis.

The screenshot shows a PostgreSQL query window titled "Query - lv_gfo_master on postgres@129.27.89.66:5432 *". The query entered is "SELECT * FROM stadt;". In the "Output pane", the results are displayed in a table format. The table has four columns: id (integer), name (character varying), and geom (geometry). There is one row with values: 1, Graz, and a long hex string representing a geometry. The "geom" column is highlighted with a red box. The status bar at the bottom indicates "OK.", "Unix", "Ln 1 Col 21 Ch 21", "1 rows.", and "15 ms".

	id integer	name character varying	geom geometry
1	1	Graz	0101000020E61000005C8FC2F528DC2E40295C8FC2F5;



- Ausgabe von Geometrieeinformation mittels spezieller Funktionen
 - Syntax: `astext(attr_geom)`
 - Ergebnis: WKT
 - Beispiel:
`SELECT attr_1, astext(attr_geom), attr_2 FROM tablename;`
 - Syntax: `x(attr_geom)` bzw. `y(attr_geom)` nur auf Punkte
 - Ergebnis: x-Wert bzw. y-Wert (mathematisch)
 - Beispiel:
`SELECT attr_1, x(attr_geom) , x(attr_geom) FROM tablename;`



Ausgabe von Geometrien

OGC

Referenz

PostGIS

Erstellen

Eingabe

Abfrage

The screenshot shows two separate PostgreSQL query windows side-by-side.

Left Window:

- Title bar: Query - lv_gfo_master on postgres@129.27.89.66:5432 *
- Toolbar: Standard PostgreSQL icons.
- Query Editor:

```
SELECT id, name, astext(geom) FROM stadt;
```
- Output pane: Data Output tab selected. Result table:

	id integer	name character varying	astext text
1	1	Graz	POINT(15.43 47.07)

- Status bar: OK., Unix, Ln 1 Col 42 Ch 42 | 1 rows.

Right Window:

- Title bar: Query - lv_gfo_master on postgres@129.27.89.66:5432 *
- Toolbar: Standard PostgreSQL icons.
- Query Editor:

```
SELECT id, name, x(geom), y(geom) FROM stadt;
```
- Output pane: Data Output tab selected. Result table:

	id integer	name character varying	x double precision	y double precision
1	1	Graz	15.43	47.07

- Status bar: OK., Unix, Ln 1 Col 46 Ch 46 | 1 rows. | 16 ms



- Anwenden von Analysefunktionen
 - Syntax: `SELECT <Analysefunktion> FROM ... WHERE ...`
 - Syntax: `SELECT ... FROM ... WHERE <Analysefunktion>`
- Einblick in Analysefunktionen
 - `area(geometry)` ... Flächenberechnung
 - `length(geometry)` ... Längenberechnung
 - `buffer(geometry, num)` ... Puffer mit bestimmten Abstand zu einer Geometrie
 - `contains(geometry,geometry)` ... Beinhaltet eine Geometrie eine andere Geometrie
 - `centroid(geometry)` ... Schwerpunkt



Transformationen

OGC Referenz PostGIS Erstellen Eingabe Abfrage

- Syntax:

`SELECT transform(attr_geom,ZielSRID) FROM tablename;`

- Beispiel:

`SELECT astext(transform(geom,4236)) FROM bezirke`

The screenshot shows two separate pgAdmin III windows side-by-side. Both windows have a title bar labeled "Query - lv osg on postgres@129.27.89.66:5432 *". The left window contains the query `select astext(geom) from bezirke;`. The right window contains the query `select astext(transform(geom,4236)) from bezirke;`. Both windows show the results in an "Output pane" under the "Data Output" tab. The results are identical, displaying a list of MULTIPOLYGON geometry values. The left window has 8 rows, and the right window has 8 rows, both with the same content.

astext	text
1	MULTIPOLYGON(((530552.609404218 5220171.07726042,530730.686542104,530730.686542104,530552.609404218 5220171.07726042)))
2	MULTIPOLYGON(((525752.675289852 5297003.10582285,525807.744465895,525807.744465895,525752.675289852 5297003.10582285)))
3	MULTIPOLYGON(((523906.65467131 5202187.42617552,523945.492825657,523945.492825657,523906.65467131 5202187.42617552)))
4	MULTIPOLYGON(((571639.88648181 5212242.30907408,571640.689651759,571640.689651759,571639.88648181 5212242.30907408)))
5	MULTIPOLYGON(((568414.760294232 5224938.1947075,568506.180845409,568506.180845409,568414.760294232 5224938.1947075)))
6	
7	
8	MII II TTDQI VCRM//461413 428852028 5256345 20476074 461430 323567823
1	MULTIPOLYGON(((15.4076295479058 47.1313860856623,15.409971021971 47.1305605060259,15.41180211,15.4076295479058 47.1313860856623)))
2	MULTIPOLYGON(((15.348893535763 47.8228431500611,15.3496275605238 47.8226032336968,15.35070304,15.348893535763 47.8228431500611)))
3	MULTIPOLYGON(((15.3190427126118 46.9698549206879,15.3195517956187 46.9696155177124,15.3197157,15.3190427126118 46.9698549206879)))
4	MULTIPOLYGON(((15.9480136664586 47.0568471024728,15.9480142165 47.0562815594542,15.9496566147,15.9480136664586 47.0568471024728)))
5	MULTIPOLYGON(((15.9074973800762 47.1714092394477,15.9086819212141 47.1701378848814,15.90871941,15.9074973800762 47.1714092394477)))
6	
7	
8	MII II TTDQI VCRM//461413 428852028 5256345 20476074 461430 323567823



- Syntax:

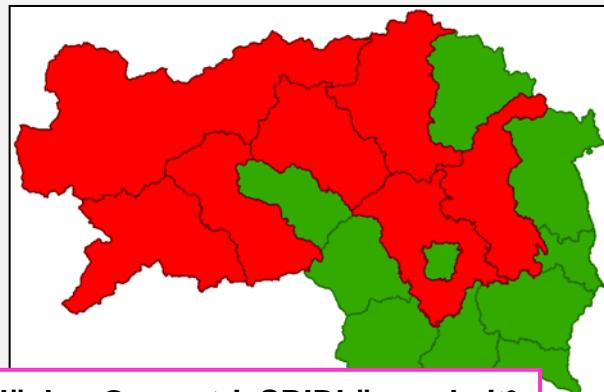
SELECT area(attr_geom) FROM tablename;

SELECT * FROM tablename WHERE area(attr_geom) [Operator] value;

- Beispiel:

SELECT * FROM bezirke

WHERE area(geom) > (SELECT avg(area(geom)) FROM bezirke);



Maßeinheit der Fläche: GeometrieSRIDLängenheit²
SRID 4326: °²
SRID 32633: m²
u. U. Transformation erforderlich!

Query - lv osg on postgres@129.27.89.66:5432 *

```
File Edit Query Favourites Macros View Help
select * from bezirke where area(geom) > (select avg(area(geom)) from bezirke)
```

Output pane

ID	Name	Fläche	Einwohner	Hauptstadt	Geom
1	Bruck an der Mur	1306096	64991	1	0106000020797
2	Graz Umgebung	1100071	137449	5	
3	Judenburg	1097036	48218	7	0106000020797
4	Leoben	1099069	67767	10	0106000020797
5	Liezen	3270037	82235	11	0106000020797
6	Murau	1384058	31472	12	0106000020797
7	Weiz	1070048	86007	16	

OK. Unix Ln 1 Col 71 Ch 71 7 rows. 109 ms



Puffer erzeugen

OGC Referenz PostGIS Erstellen Eingabe Abfrage

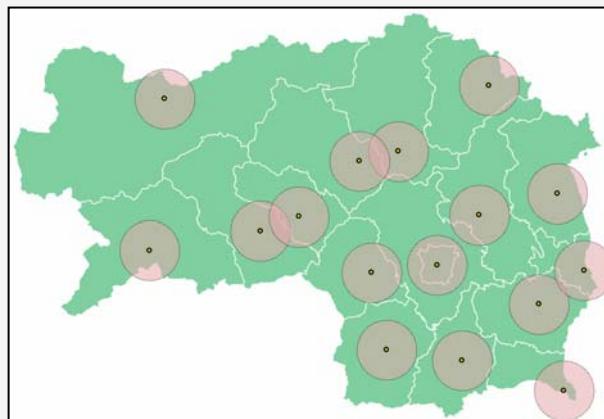
- Syntax:

`SELECT buffer(attr_geom, size) FROM tablename;`

- Beispiel:

`SELECT buffer(geom,10000) FROM stadt;`

Maßeinheit der Größe: GeometrieSRIDLängenheit²



Query - lv osg on postgres@129.27.89.66:5432 *

```
File Edit Query Favourites Macros View Help
select buffer(geom,10000) from stadt;
```

Output pane

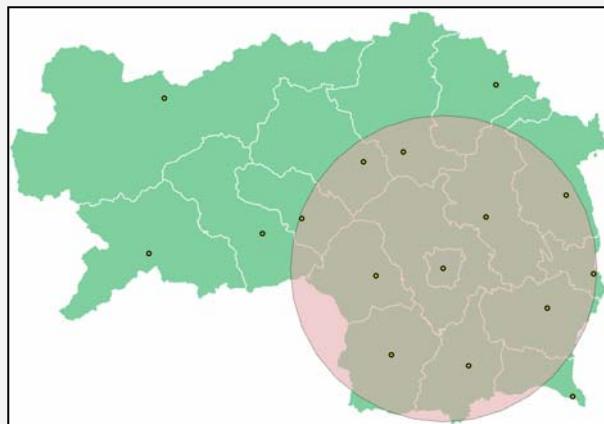
Data Output Explain Messages History

	buffer geometry
1	01030000207997F00
2	01030000207997F00
3	01030000207997F00
4	01030000207997F00
5	01030000207997F00
6	01030000207997F00
7	01030000207997F00
8	01030000207997F00

OK. Unix Ln 1 Col 32 Ch 32 16 rows. 31 ms



- OGC Referenz PostGIS Erstellen Eingabe Abfrage
- Syntax:
`SELECT contains(attr_geom, attr_geom) FROM tablename;`
`SELECT * FROM tablename WHERE contains(attr_geom, attr_geom);`
 - Beispiel:



The screenshot shows two PostgreSQL query windows side-by-side. Both windows have the title "Query - lv osg on student@129.27.89.66:5432".

The left window contains the following SQL query:

```
select name, contains(buffer, geom) from stadt
```

The right window contains the following SQL query:

```
select name from stadt where contains(buffer, geom)
```

Both windows display the same result set, which is a table with two columns: "name" (character varying) and "contains" (boolean). The results are as follows:

	name	contains
1	Bruck an der Mur	t
2	Deutschlandsberg	t
3	Feldbach	t
4	Fürstenfeld	t
5	Hartberg	t
6	Judenburg	f
7	Knittelfeld	t
8	Leibnitz	t
9	Leoben	t
10	Liezen	f
11	Murau	f
12	Mürzzuschlag	f
13	Bad Radkersburg	f
14	Voitsberg	t
15	Weiz	t
16	Graz	t



- Syntax:

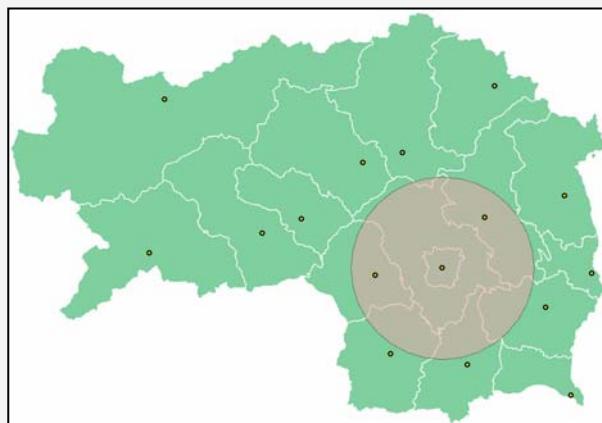
Maßeinheit der Distanz: GeometrieSRIDLängeneinheit

```
SELECT distance(attr_geom, attr_geom) FROM tablename;
```

```
SELECT * FROM tablename
```

```
    WHERE distance(attr_geom, attr_geom) = value;
```

- Beispiel:



Query - lv osg on student@129.27.89.66:5432 *

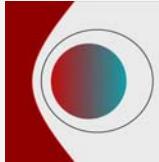
```
File Edit Query Favourites Macros View Help
SELECT name
FROM stadt
WHERE distance((SELECT geom FROM stadt WHERE name = 'Graz'), geom) < 30000;
```

Output pane

Data Output Explain Messages History

	name character var
1	Voitsberg
2	Weiz
3	Graz

OK. Unix Ln 2 Col 6 Ch 18 3 rows. 15 ms



Allfälliges

- 21.11.2011: A109
- Fragen ?

spatialSQL

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

spatialSQL Übung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Zeitplan

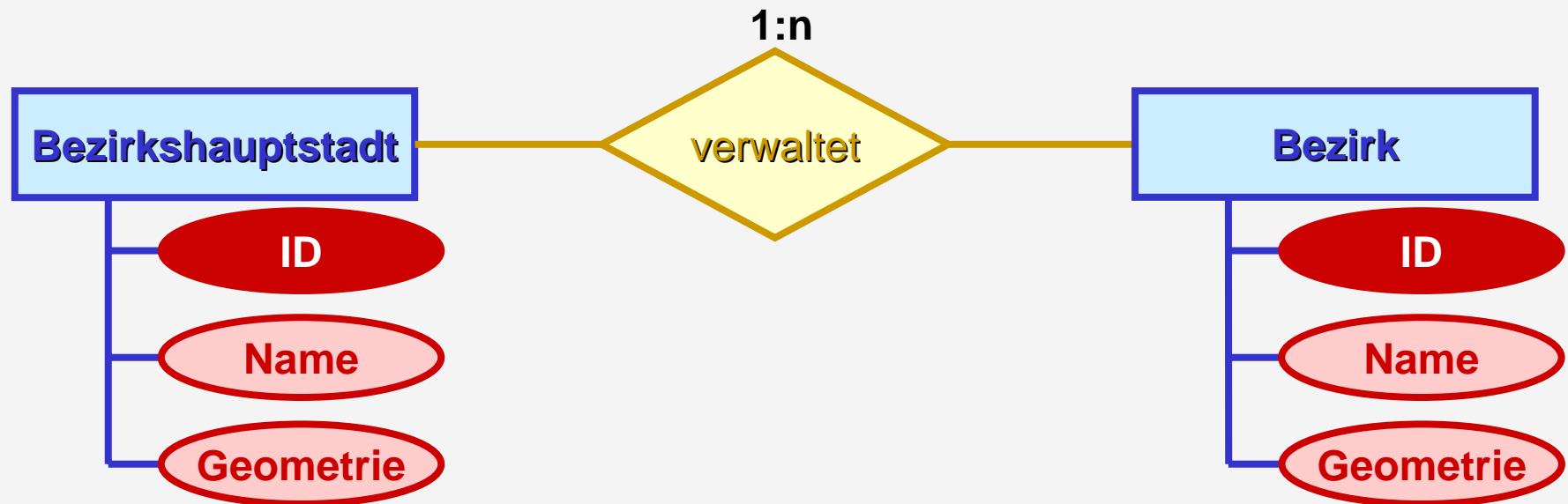
Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



- Erstellen einer Tabelle – *create table...*
- Ergänzen einer Tabelle um eine Geometriespalte – *select addgeometrycolumn(...)*
- (räumliche) Dateneingabe – *insert into...*
- (räumliche) Datenanalyse – *select...*
- Entfernen der Geometriespalte aus der Tabelle – *select dropgeometrycolumn(...)*
- Löschen der Tabelle – *drop...*



Datenstruktur





Tabellen erstellen

- **Alle** Steirische Bezirkshauptstädte
 - ID, Name, Geometrie (EPSG 4326)
- Realisierung der **Beziehung** zwischen Bezirken und Bezirkshauptstädten
- **Tabellennamen und Attributnamen klein schreiben, keine Sonderzeichen und keine Leerzeichen!**



Ganz einfache Datenabfrage

- Wie lautet der **WKT** der Stadt Leoben?
- Wie lautet die **geographische Länge** der Stadt Leoben?
- Welchem **Koordinatensystem** entsprechen die Bezirke?
- Wie lautet der **WKT** der Stadt Judenburg nach der **Transformation** in das Koordinatensystem der Bezirke?
- Welchem **Geometriertyp** entsprechen die Bezirksdaten?
- Wie **groß** (in km²) ist der Bezirk Feldbach?



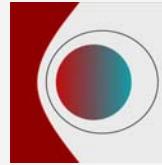
Einfache Datenabfrage

- Wie lautet der Name der **östlichsten** Bezirkshauptstadt?
- Wie lautet der Name des **größten** Bezirks?
- Wie lautet der Name des Bezirks mit der **kürzesten Grenze**?
- **Ändern** Sie kurzfristig das **Koordinatensystem** der Bezirkshauptstädte auf MGI (Ferro).



Komplexe Datenabfrage

- Wie lauten die **Nachbarbezirke** des Bezirks Fürstenfeld?
- Wie weit ist Liezen von Deutschlandsberg **entfernt** (in km)?
- Wie **groß** (in km^2) ist ein **Puffer** mit Zentrum in Graz, der die Stadt Feldbach berührt?
- Welche Bezirkshauptstädte **beinhaltet** dieser Puffer?
- Welche Bezirkshauptstadt ist **am weitesten** vom **Schwerpunkt** des zu verwaltenden Bezirks **entfernt**.



Geometriespalte löschen

- Löschen Sie die Geometriespalten der Bezirkshauptstadttabelle.



Tabellen löschen

- Löschen Sie die restlichen Tabellen **AUSSER** *geometry_columns* und *spacial_ref_sys*!

spatialSQL Übung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

Quantum GIS

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



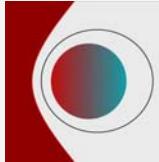
Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



Übersicht Hinzufügen Visualisierung Plugins

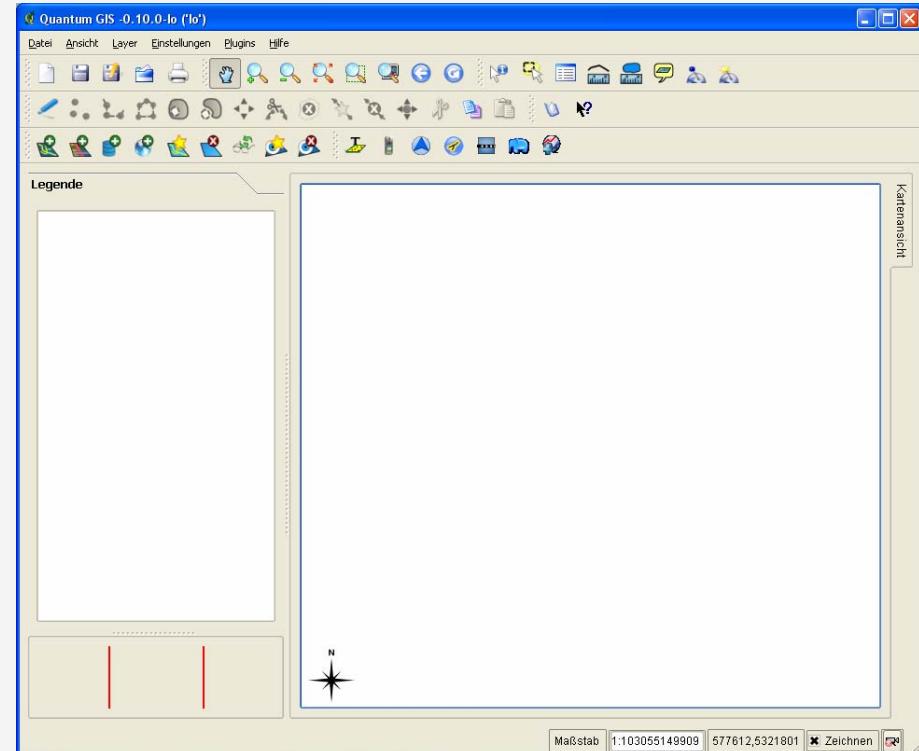
- Programmübersicht
- Hinzufügen von Daten
- Visualisieren von Daten
- Plugins
 - SPIT



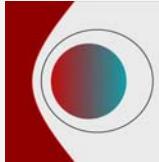
Programmübersicht

Übersicht Hinzufügen Visualisierung Plugins

- Quantum GIS
 - freies GIS
 - div. Geodaten
 - Raster
 - Vektor
 - DB
 - Text
 - kart. Darstellung
- Hier:
 - Visualisierung von PostGIS Analysen



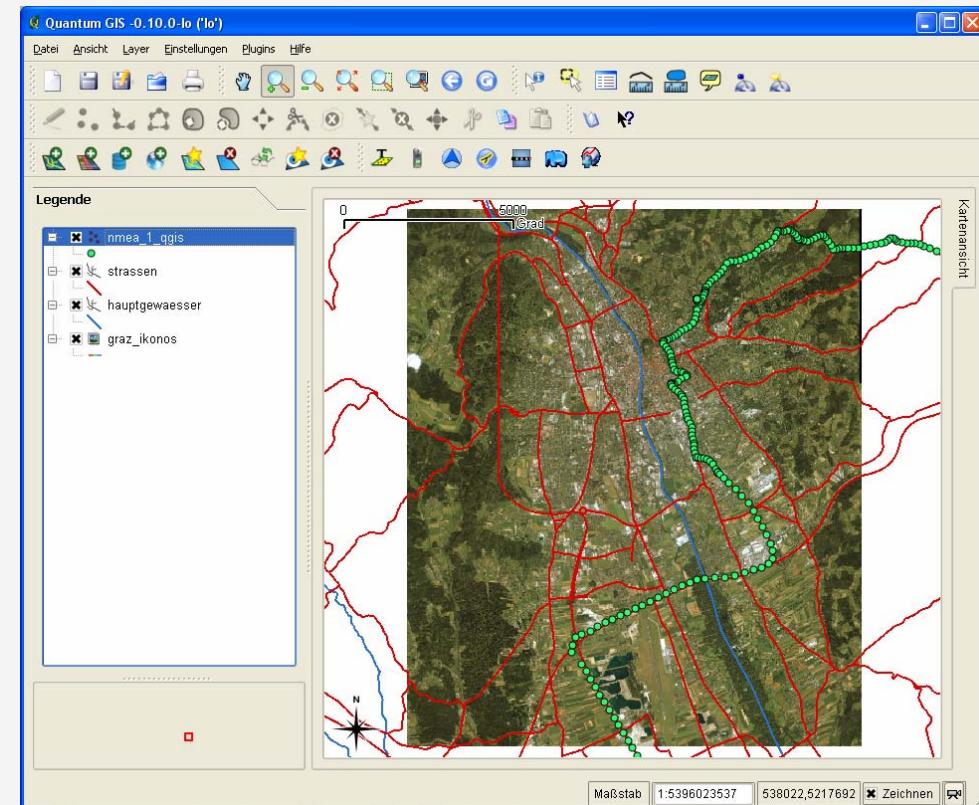
www.qgis.org



Hinzufügen von Daten

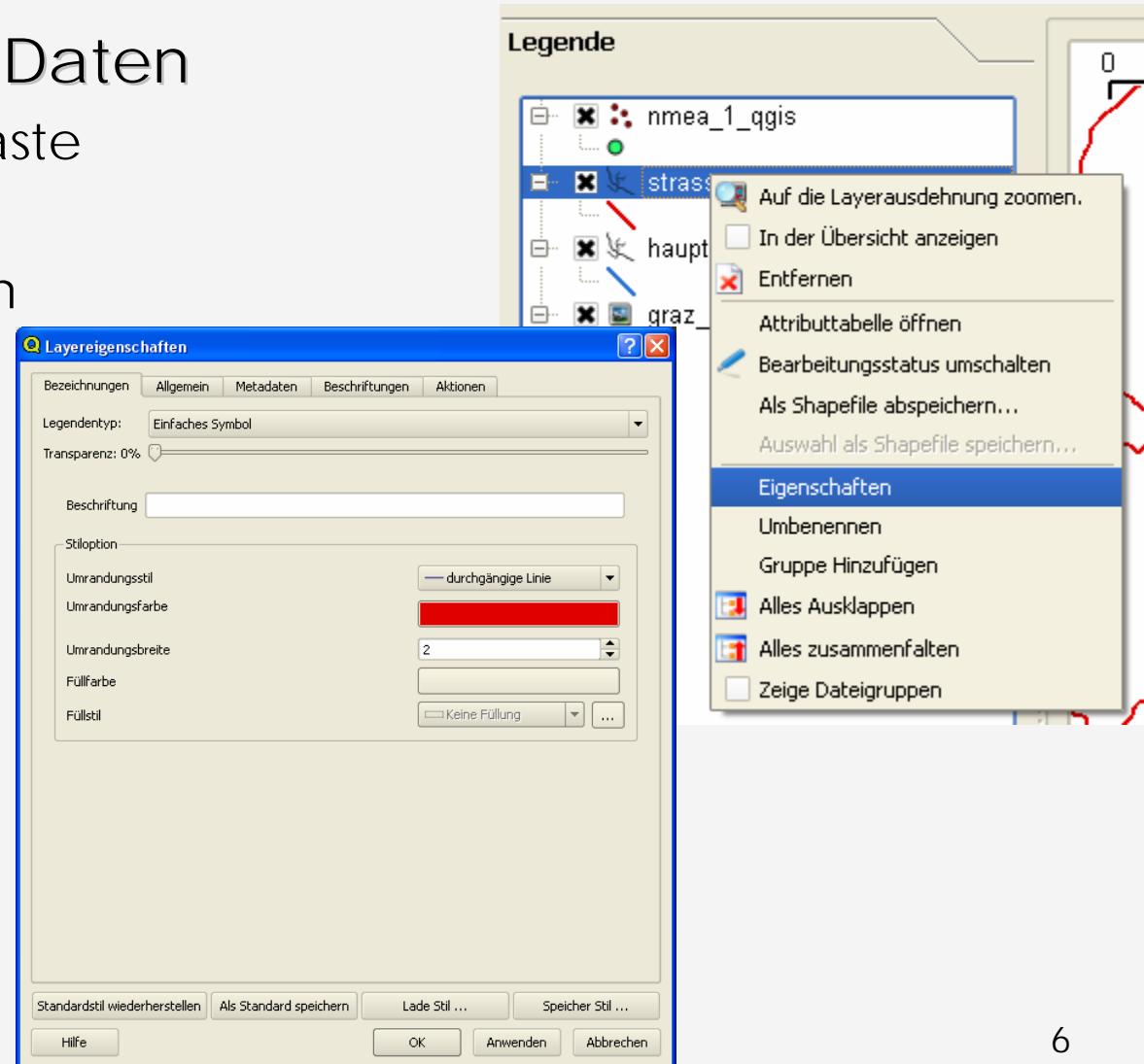
Übersicht Hinzufügen Visualisierung Plugins

- Vektordaten
 - ESRI Shape
- Rasterdaten
- PostGIS Datenbank
- WMS
 - (Web Map Service)





- Parameter der Daten
 - Rechte Maustaste
 - Eigenschaften
 - Bezeichnungen
 - Allgemein
 - Metadaten
 - Beschriftung
 - Aktionen

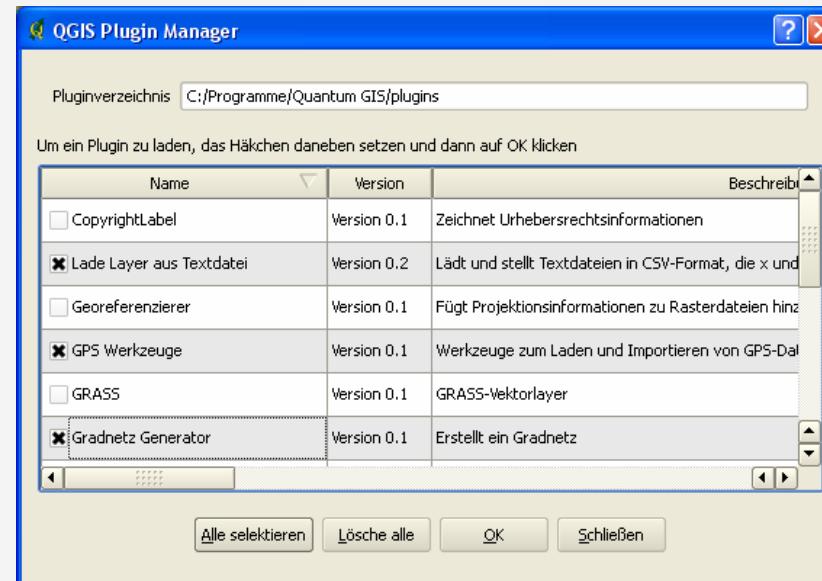




- Einstellungen/Projekteinstellungen...
 - Allgemein
 - Karteneinheiten (Meter, Fuß, DD)
 - Genauigkeit
 - Digitalisierung
 - Kartenerscheinung
 - Projektion
 - Projektion zur Laufzeit einschalten (**WICHTIG!**)
 - Suchen
 - PostGIS SRID
 - z.B.: 32633 (UTM 33 N)

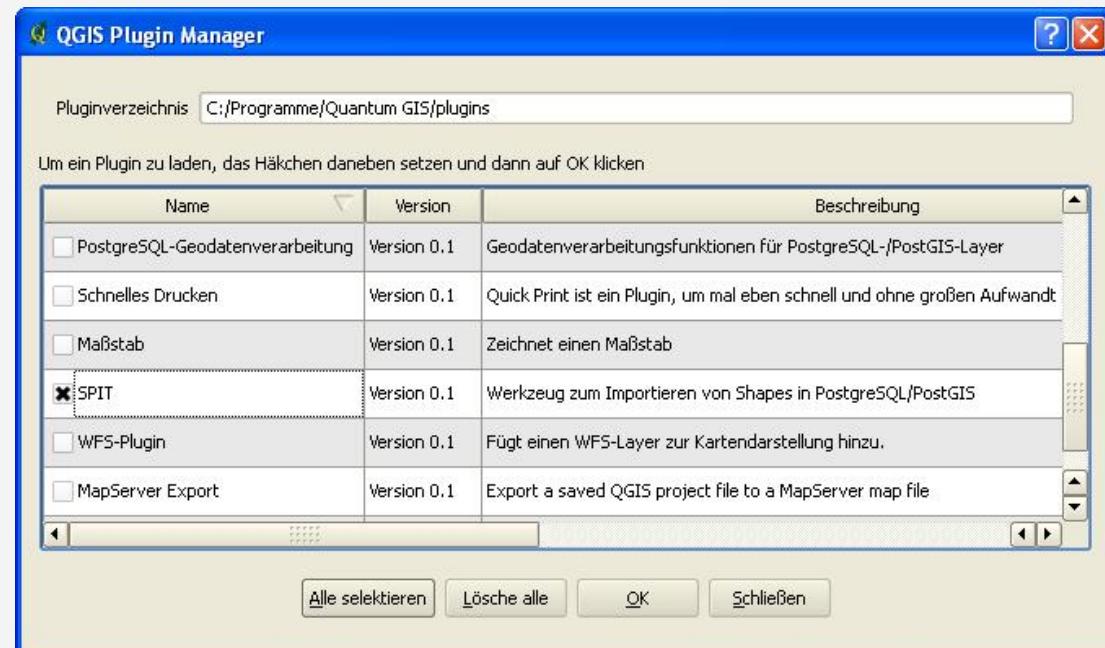


- Plugins/Plugin Manager...
 - Einige nützliche Plugins
 - Lade Layer aus Textdatei
 - GPS Werkzeug (GPX Daten)
 - PostgreSQL-Geodatenverarbeitung
 - ...





- Import von Shape Dateien in PostGIS DB mittels QGIS Plug-In **SPIT** (*Shapefile in PostGIS Import Tool*)
- Aktivieren des Plugins im Plugin Manager





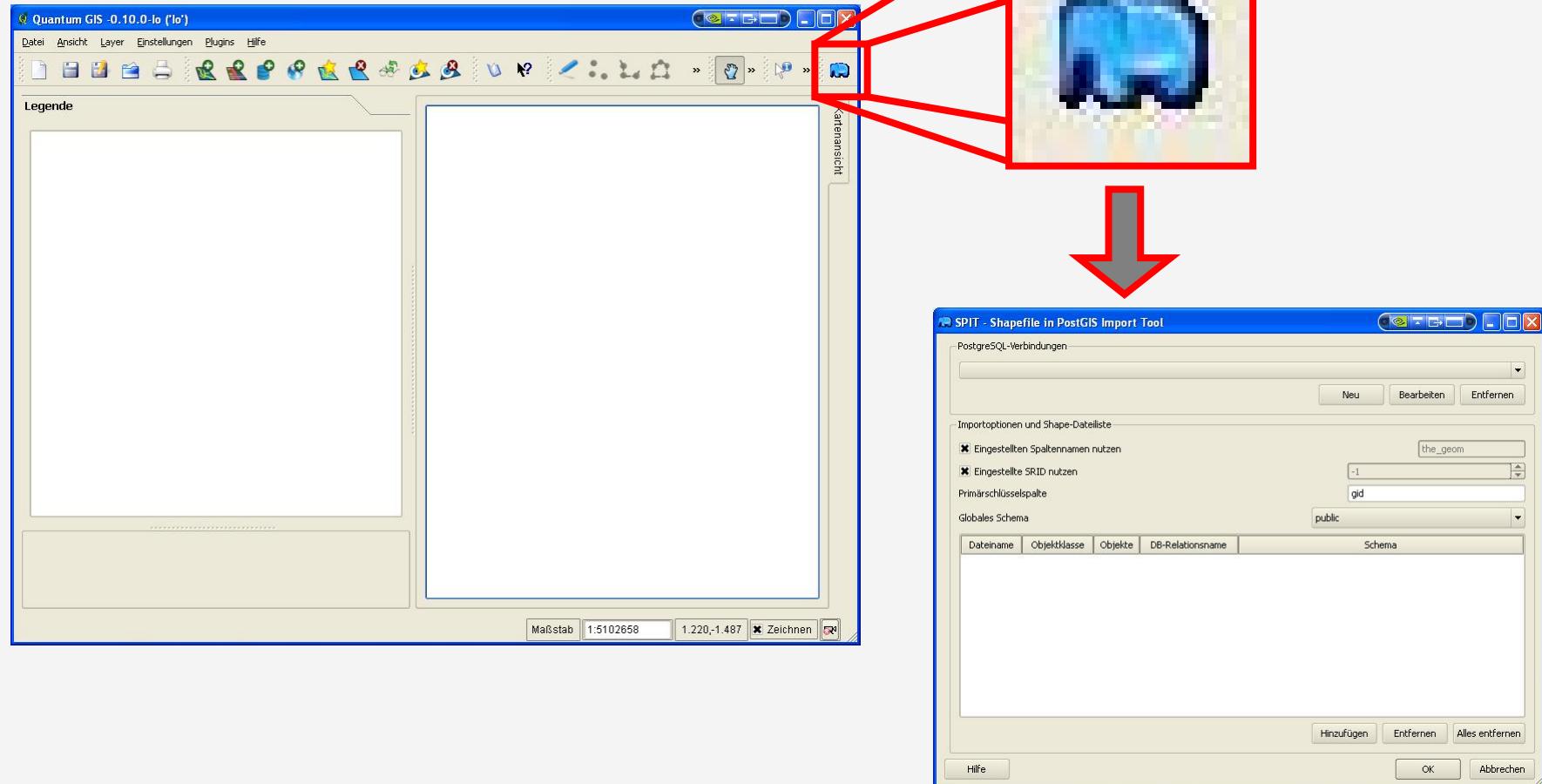
Offnen des SPIT Formulars

Übersicht

Hinzufügen

Visualisierung

Plugins



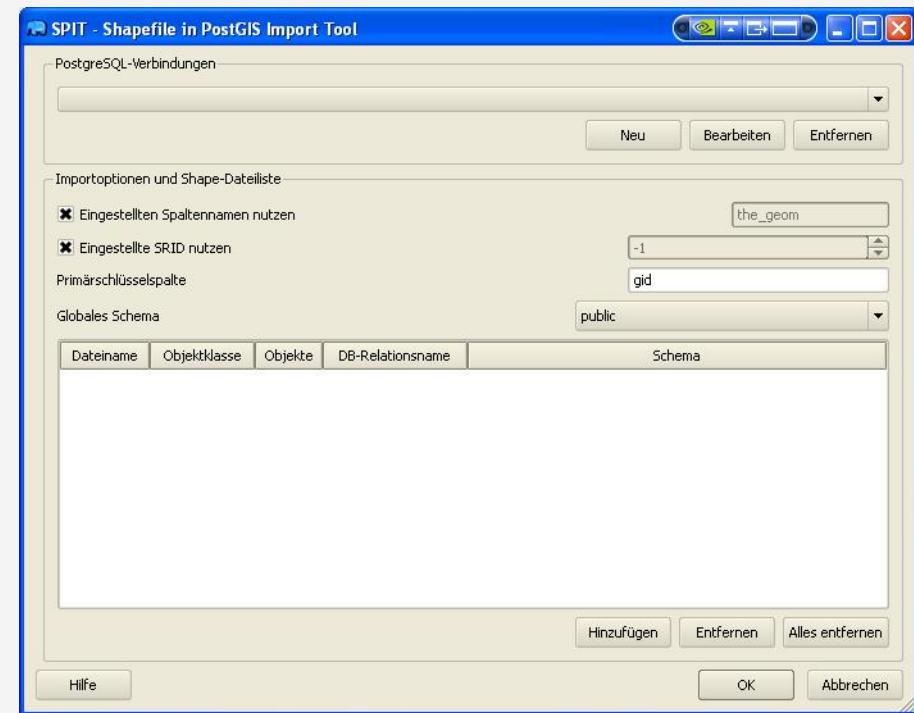


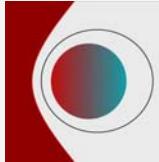
Importprozess

Übersicht Hinzufügen Visualisierung

Plugins

- Parameter
 - Ziel Datenbank
 - Name d. Geometriespalte
 - SRID
 - PK Spalte
- Starten des Importprozesses mit OK
- Keine Mischgeometrien
 - z.B.: Line & Multi Line





- Praktische Einführung QGIS
 - DB Inhalte
 - Geoimage WMS
- 05.12.2011: A109
- Fragen ?

Quantum GIS

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

HTML Übung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



HTTP/FTP Verbindung

- Öffnen Sie einen **Internetbrowser**.
- Navigieren Sie zum **geoFOSS Webspace**.

- Öffnen Sie ein **FTP Programm**.
- Navigieren Si zum **geoFOSS Webspace**.
- **Erzeugen** Sie einen **Unterordner „rechner“**.
- **Erstellen** Sie eine **HTML Datei „eingabe.html“**.



- Öffnen Sie **Notepad++**.
- Richten Sie eine **FTP Verbindung** ein.
- Navigieren Sie zum **geoFOSS Webspace**.

- Erzeugen Sie in „eingabe.html“ eine **Eingabemaske** für
 - 2 Zahlen
 - 4 Grundrechnungsarten... die in einem **HTML Formular** strukturiert sind.

HTML Übung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

PHP Teil 1

Geoinformatik 2

Konstruktionsübung

WS 2011/12

Clemens Strauß

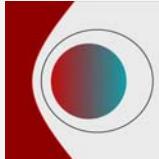


Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



- PHP - Allgemein.
- Grundgerüst.
- Typen und Variablen.
- Operatoren.
- Entscheidungskonstrukte.
- Wiederholungskonstrukte.
- Ausgabe.



Einführung

- PHP steht für **PHP: Hypertext Preprocessor**.
- Open Source Skriptsprache.
- Speziell für Webprogrammierung.
- Kann in HTML eingebettet werden.

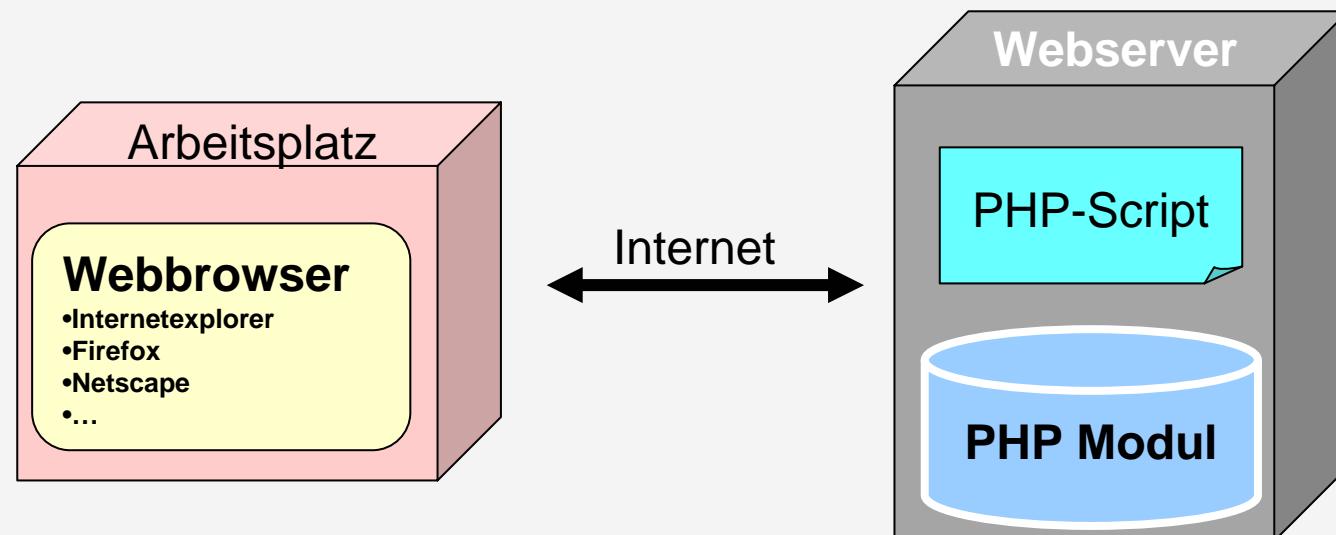
```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7     <?php
8       echo "Hallo, ich bin ein PHP-Skript!";
9     ?>
10
11   </body>
12 </html>
```





Voraussetzungen

- Webserver
- PHP Modul für Webserver
- Webbrowser





- Grundgerüst:

```
<?php  
    CODE;  
?>
```

- Zeilenabschluss mit „;“.
- Befehle sind nicht case-sensitiv.
- Kommentare „//“.
 - Dient der Verständlichkeit des Codes.
 - Befehl; // Beschreibung des Befehls.
 - Bsp.: echo "Hallo, ich bin ein PHP-Skript!"; // Textausgabe



- PHP Code in HTML eingebettet:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">  
<html>  
  <head>  
    <title>...</title>  
  </head>  
  <body>  
    <?php  
      CODE;  
    ?>  
  </body>  
</html>
```

- Dateiname trotzdem *.php und nicht *.htm!



- Skalare Typen
 - Boolean: `$var = true;` bzw. `$var = false;`
 - Integer: `$var = 12345;`
 - Float: `$var = 123.4324;`
 - String: `$var = „Das ist eine Zeichenkette.“;`
 - Zeichenkette beginnt und endet mit „“.
- Mehrdimensionale Typen
 - Array:
`$var = array(1,2,3);`
`$var[0] -> 1`
`$var[1] -> 2`
`$var[2] -> 3`

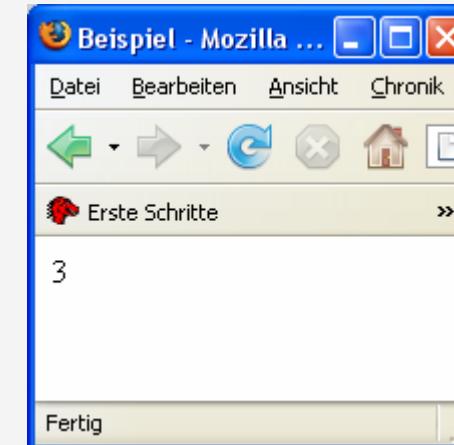


- Variablenname
 - Beginnt immer mit \$.
 - Case-sensitiv: \$var != \$Var != \$vAr != \$VAr !=
 - Darf Ziffern beinhalten aber nicht an erster Stelle
 - Bsp.: \$5setzen (NOK), \$setzten5 (OK)
 - Variablen müssen nicht einem Datentyp explizit zugewiesen werden. Wird zur Laufzeit vom Programm entschieden.
 - **Tipp:** Beabsichtigter Datentyp als Kürzel im Variablenamen verwenden.
 - Bsp.: \$strName, \$intAlter, \$bolWeiblich, \$floEntfernung,...



- Arithmetische Operatoren
 - + , - , * , / und %.
 - \$a % \$b (Modulus: Rest von \$a / \$b).

```
1  <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7     <?php
8       $intA = 1;
9       $intB = 2;
10      $intC = $intA + $intB;
11      echo $intC;
12    ?>
13
14   </body>
15 </html>
```





Operatoren

- Zuweisungsoperatoren

Zahlen: `=, +=, -=, *=, /=, %=`

Zeichen: `=, .=` (Concatenate)

```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7     <?php
8       $intA = 1;
9       $intA += 2;
10      echo $intA;
11
12    </body>
13  </html>
```



```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7     <?php
8       $strA = "Hallo";
9       $strA .= " World!";
10      echo $strA;
11
12    </body>
13  </html>
```



Operatoren

- Vergleichsoperatoren

`==, != bzw. <>, <, >, <=, >=`

```
1  <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7     <?php
8       $bolA = true;
9       $bolB = false;
10      if ($bolA == $bolB) {
11        echo "A entspricht B!";
12      }
13      else {
14        echo "A entspricht nicht B!";
15      }
16    <?>
17
18   </body>
19 </html>
```





Operatoren

- Logische Operatoren

and bzw. &&

or bzw. ||

XOR

!... NOT

!TRUE == FALSE

```
1  <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7   <?php
8     $intA = 1;
9     $intB = 2;
10    $intC = 1;
11
12    if ($intA < $intB && $intA == $intC) {
13      echo "Hurra!";
14    }
15    else {
16      echo "Juhu!";
17    }
18
19   </body>
20 </html>
```





IF - Konstruktionen

- If
 - if (Bedingung) {CODE;}
- If Else
 - if (Bedingung) {CODE;}
 - else {CODE;}
- If Elseif Else
 - if (Bedingung) {CODE;}
 - elseif (Bedingung) {CODE;}
 - else {CODE;}

```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7   <?php
8     $intA = 2;
9     if ($intA == 1) {
10       echo "Wert: 1";
11     }
12     elseif ($intA == 2) {
13       echo "Wert: 2";
14     }
15     else {
16       echo "Wert weder 1 noch 2!";
17     }
18   ?>
19
20 </body>
21 </html>
```





- For
 - for (Anweisung) {CODE;}
- Anzahl der Schleifendurchläufe in Anweisung definiert.
- Anweisung
 - „Zähle von 1 bis 10.“
 - Variable mit Startwert: \$i = 1.
 - Bedingung: \$i <= 10.
 - Zählmethode/Schrittweite: \$i++.
 - Bsp.: for (\$i = 1; \$i <= 10 ; \$i++) {CODE;}



FOR - Schleife

```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7     <?php
8       for ($intI = 1; $intI <= 10; $intI += 2) {
9         echo "{$intI} ";
10      }
11    <?>
12
13   </body>
14 </html>
15
```





WHILE - Schleife

- While
 - while (Bedingung) {CODE;}
- Anzahl der Schleifendurchläufe ist nicht vorgegeben.
- Schleife kann kein einziges mal durchlaufen werden.



```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7   <?php
8     $intI = 0;
9     $bolA = true;
10    while ($bolA) {
11      $intI++;
12      $bolA = false;
13      echo "($intI)-ter Schleifendurchlauf";
14    }
15
16  </body>
17
18 </html>
19
```



- Do While
 - do {CODE;} while (Bedingung);
- Anzahl der Schleifendurchläufe ist nicht vorgegeben.
- Schleife wird mindestens ein mal durchlaufen.



```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7   <?php
8     $intI = 0;
9     $bolA = true;
10    do {
11      $intI++;
12      $bolA = false;
13      echo "($intI)-ter Schleifendurchlauf";
14    } while ($bolA)
15
16
17   </body>
18 </html>
19
```



Textausgabe

- Einfache Textausgabe
`echo „TEXT“;`
- Erweiterung für Browserdarstellung
 - Verwendung von HTML-Tags.

```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7     <?php
8       echo "Das ist ein Text.";
9       echo "<p>Das ist ein formatierter <b>Text</b>, <br>der <i>HTML</i>-Tags beinhaltet.</p>";
10      ?>
11
12   </body>
13 </html>
```





Textausgabe mit Variablen

- Variable in Text einbetten.

```
echo „x = $intX“;
```

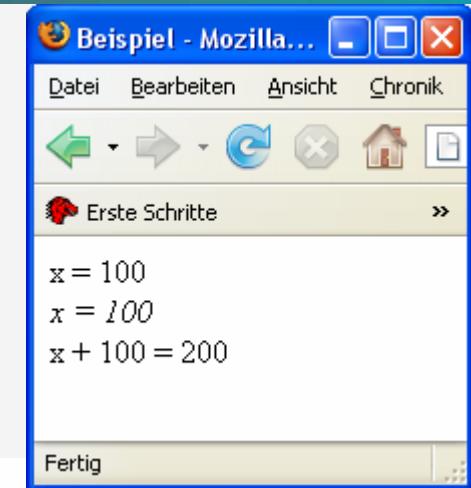
- Variable im Text mit {} umschließen.

```
echo „x = {$intX}“;
```

- Variable vom Text mit „, trennen.

```
echo „x = “, $intX;
```

```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7   <?php
8     $intX = 100;
9     echo "x = $intX <br>";
10    echo "<i>x = {$intX}</i><br>";
11    echo "x + 100 = ", $intX + 100;
12
13  </body>
14 </html>
15
16
```





- Sonderzeichen

- CR ... \r Wagenrücklauf
- LF ... \n Neue Zeile
- Tabulator ... \t
- \ ... \\
- \$... \\$
- " ... \" (Für HTML-Tags wichtig!)

```
1 <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7     <?php
8       echo "<p align=\"right\"> Text Text Text Text Text Text Text Text Text</p>";
9     ?>
10
11   </body>
12 </html>
```





- HTML – Formular
 - Tag: *<form action=„Ziel.php“>...</form>*
 - Eingabefelder
 - *<input name=„ValName“ type=„text“ size=„5“ maxlength=„5“>*
 - Daten Übermitteln
 - *<input type=“submit“ value=“Absenden“>*

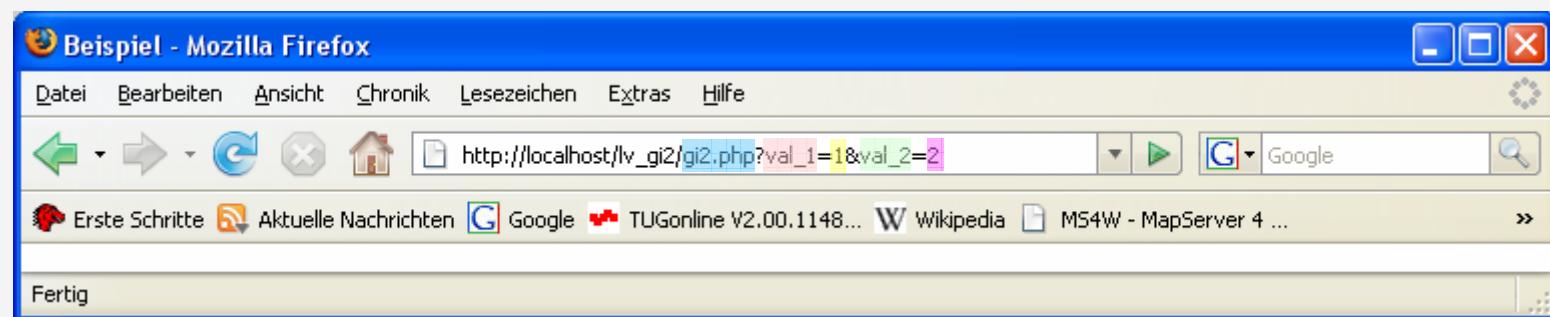
```
1 <html>
2   <head>
3     <title>Formular</title>
4   </head>
5   <body>
6
7   <form action="gi2.php">
8     Wert 1: <input name="val_1" type="text" size="5" maxlength="5"><br>
9     Wert 2: <input name="val_2" type="text" size="5" maxlength="5"><br>
10    <input type="submit" value=" Addieren "><input type="reset" value=" Abbrechen">
11   </form>
12
13   </body>
14 </html>
15
```





- Datenübermittlung

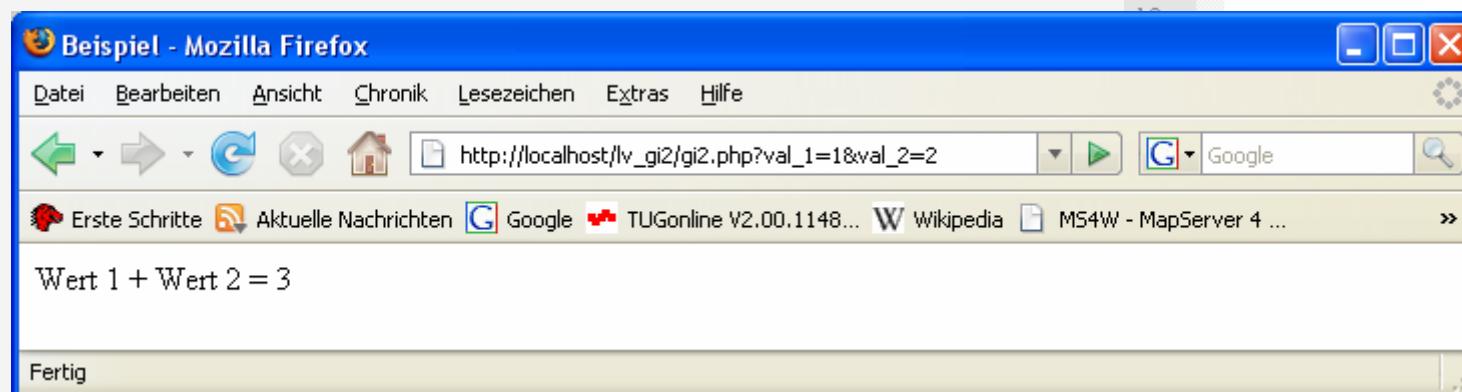
```
1 <html>
2   <head>
3     <title>Formular</title>
4   </head>
5   <body>
6
7     <form action="gi2.php">
8       Wert 1: <input name="val_1" type="text" size="5" maxlength="5"><br>
9       Wert 2: <input name="val_2" type="text" size="5" maxlength="5"><br>
10      <input type="submit" value=" Addieren "><input type="reset" value=" Abbrechen">
11    </form>
12
13   </body>
14 </html>
15
```

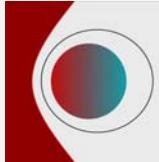




- Datenauswertung
 - Übergabeparameter im Array **\$_GET** verspeichert.

```
1  <html>
2   <head>
3     <title>Beispiel</title>
4   </head>
5   <body>
6
7   <?php
8     $intA = $_GET["val_1"];
9     $intB = $_GET["val_2"];
10
11    $intC = $intA + $intB;
12
13    echo "Wert 1 + Wert 2 = ($intC)";
14
15  ?>
16
17  </body>
18 </html>
```





Allfälliges

- Abgabe 2. Übungsprogramm
- 16.01.2012: A109
- Fragen ?

PHP Teil 1

Geoinformatik 2

Konstruktionsübung

WS 2011/12

Clemens Strauß

PHP Übung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



- Erstellen Sie eine **PHP Datei „ausgabe.php“**
- Erzeugen Sie in „ausgabe.php“ eine **Berechnungsfunktion**, die
 - Formulareinträge aus „eingabe.html“ entgegennimmt
 - die Ausgewählte Rechenart auf die eingegebenen Zahlen anwendet
 - und das Berechnungsergebnis ausgibt.

PHP Übung

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

PHP Teil 2

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3

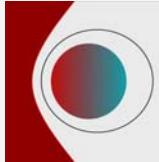


Inhalt

- Ausgangssituation.
- Funktionsablauf.
- Verwendete Funktionen.



Ausgangssituation

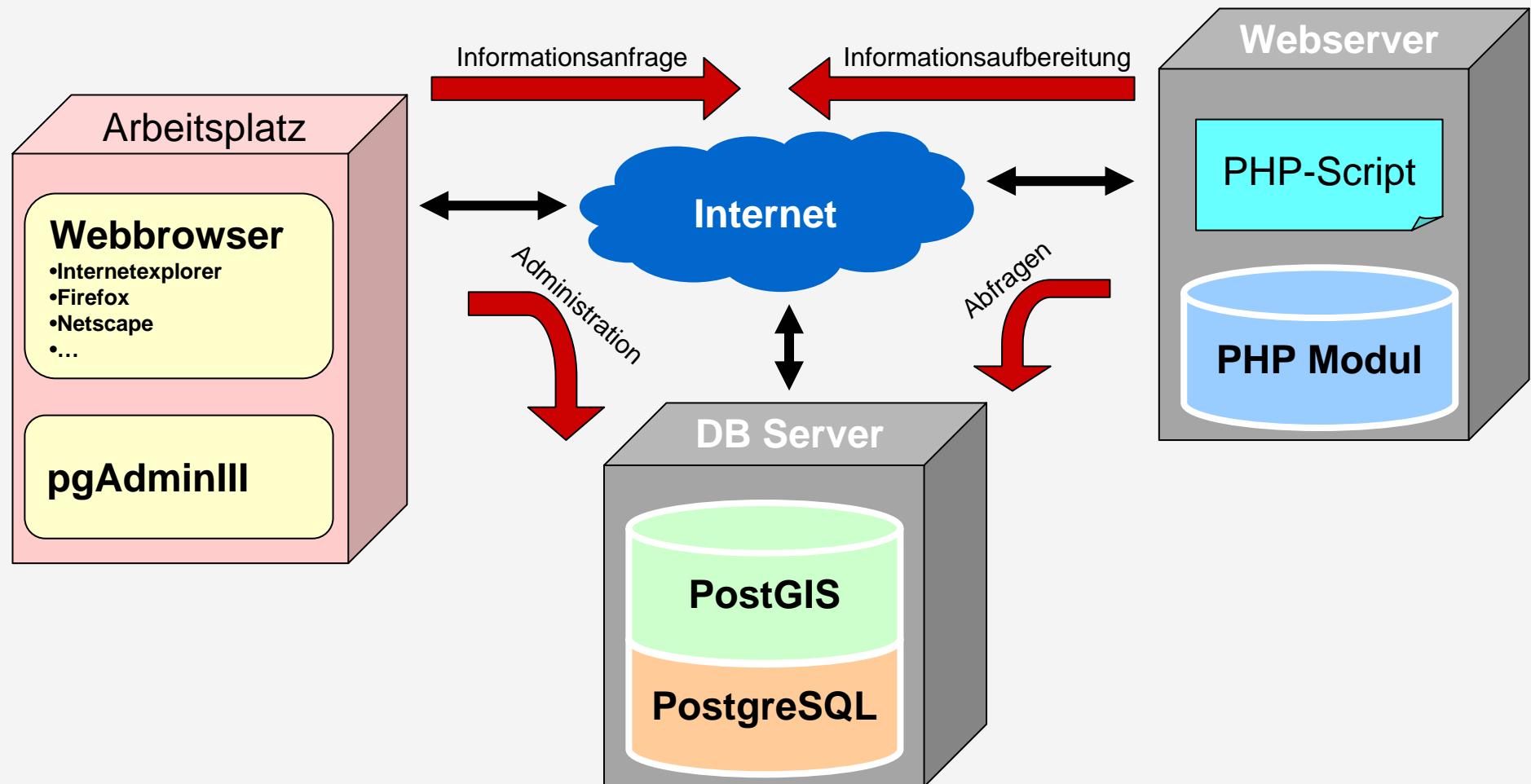


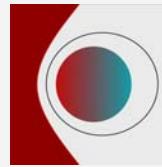
- Arbeitsplatz mit Webbrowser.
- Webserver mit PHP-Modul.
- Datenbankserver mit PostGIS Erweiterung.
- Kommunikationsmedium: Internet.

- Aktionen
 - DB Administration.
 - Informationsanfrage.
 - Datenbankabfrage.
 - Informationsaufbereitung.



Zusammenspiel der einzelnen Module

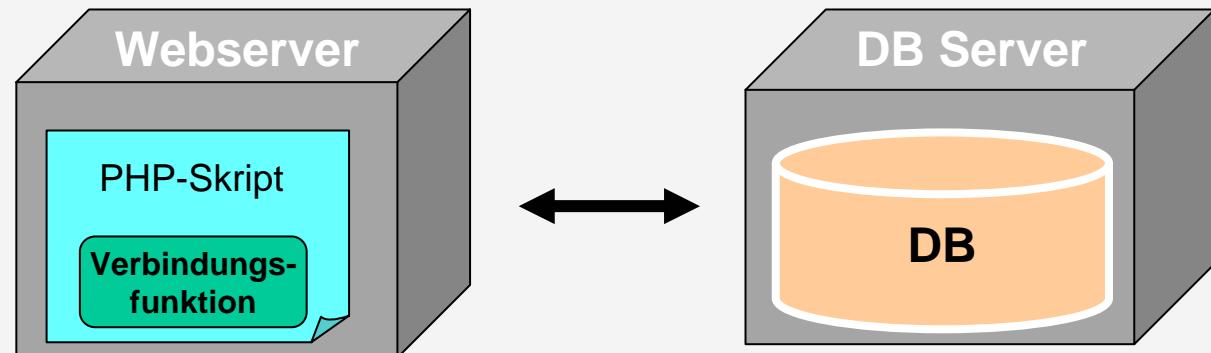




Funktionsablauf

Verbindungsauftbau

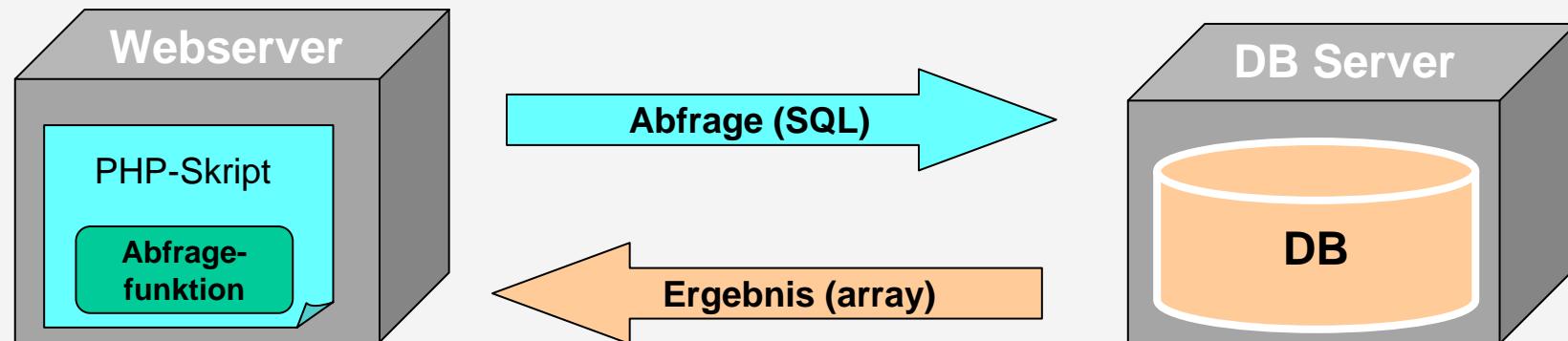
- Verbindung zwischen PHP-Skript und DB wird hergestellt.
- Benötigte Informationen
 - Hostname des DB Servers.
 - Port des DB Servers.
 - Datenbankname.
 - Benutzername.
 - Passwort.





Abfrage aus DB

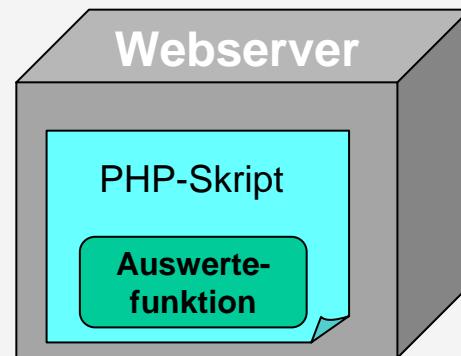
- Abfrage Formulieren (SQL).
- Abfrage an DB Server senden.
- Ergebnis der Abfrage empfangen.





Ergebnisauswertung

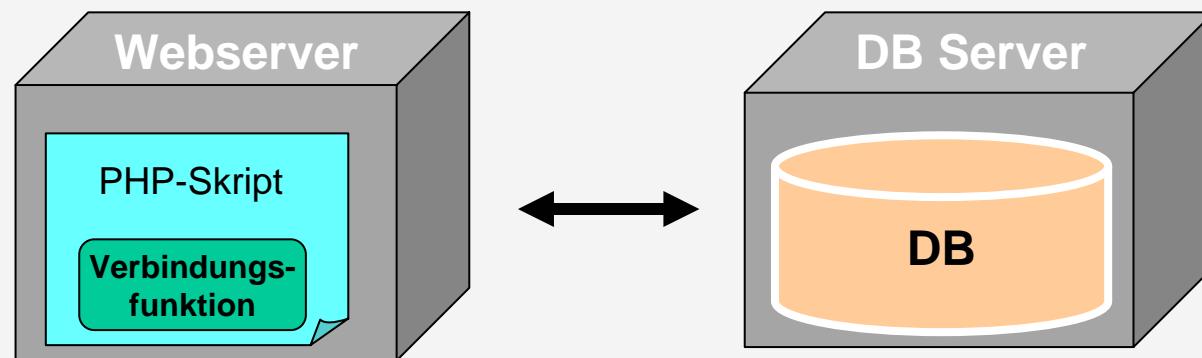
- Interpretation des Arrays.
- Elementweises auslesen.

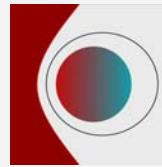




Beenden der Verbindung

- Ordnungsgemäßes beenden der Verbindung.





Verwendete Funktionen



```
$conn = pg_connect("host=$host port=$port  
dbname=$dbname user=$user password=$password");
```

- Funktionsparameter
 - Host z.B.: „129.27.89.66“.
 - Port z.B.: „5432“.
 - Datenbankname z.B.: „lv_gi2_pc31“.
 - Benutzername z.B.: „student“.
 - Passwort z.B.: „student“.
- Rückgabewert
 - Verbindungskennung.
 - FALSE bei Fehlfunktion.



Abfrage

```
$result = pg_query($conn, „SQL-String“);
```

- **Funktionsparameter**
 - Verbindungskennung (kommt aus `pg_connect`).
 - SQL-Statement als String.
- **Rückgabewert**
 - Abfrageergebnis.
 - FALSE bei Fehlfunktion.



```
$row= pg_fetch_row($result);
```

- Funktionsparameter
 - Abfrageergebnis.
- Rückgabewert
 - Array (ein Tupel).
 - \$row[i] ... Wert der i-ten Spalte (beginnend mit 0).
 - FALSE wenn kein Datensatz mehr ausgelesen werden kann.
- Verwendung mit While-Schleife
 - while(\$row = pg_fetch_row(\$result)) { ... }



```
pg_close($conn);
```

- Funktionsparameter
 - Verbindungskennung.
- Rückgabewert
 - TRUE bei erfolgreicher Verbindungstrennung.
 - FALSE Fehlfunktion.



Beispiel

- Ausgangssituation
 - DB-Server
 - 129.27.89.18:5432
 - Benutzername, Passwort
 - DB
 - Name: strauss_test
 - Tabelle
 - Name: test_tab

The screenshot shows the pgAdmin III interface. The Object Browser on the right lists database objects under 'GeoInf' (localhost:5432). It includes servers, databases (Boesenstein, postgres, strauss_test), casts, languages, and schemas (public). The 'Tables (1)' node under 'strauss_test' is expanded, showing 'test_tab'. The main window displays a data editor titled 'pgAdmin III Edit Data - GeoInf...'. It shows a table with two columns: 'id [PK] integer' and 'wochentag character var'. The data rows are:

	id [PK] integer	wochentag character var
1	1	Montag
2	2	Dienstag
3	3	Mittwoch
4	4	Donnerstag
5	5	Freitag
6	6	Samstag
7	7	Sonntag
*		

At the bottom of the data editor, it says '7 rows.'



Beispiel

- PHP-Code

```
1  <html>
2      <head>
3          </head>
4
5      <body>
6          <h4>Datenbankabfrage</h4>
7
8          <?php
9
10         // Variablen für den Verbindungsaufbau
11         $host = "129.27.89.18";
12         $port = "5432";
13         $user = "XXXXXX";
14         $password = "XXXXXX";
15         $dbname = "strauss_test";
16
17         // Aufbau der Verbindung zum DB-Server
18         $conn = pg_connect ("host=$host port=$port dbname=$dbname user=$user password=$password");
19
20         // SQL-Anweisung für DB-Abfrage
21         $sql = "select wochentag from test_tab";
22
23         // DB-Abfrage
24         $result = pg_query($conn, $sql);
25
26         // Zeilenweises Ergebnisauslesen
27         while ($row = pg_fetch_row($result)) {
28             echo "{$row[0]}<br>";
29         }
30
31         // Beenden der Verbindung zur DB
32         pg_close($conn);
33
34     ?>
```



Montag
Dienstag
Mittwoch
Donnerstag
Freitag
Samstag
Sonntag
Fertig



Allfälliges

- 23.01.2012: A109
- Fragen ?

PHP Teil 2

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

Google Chart

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß



Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3



- Überblick
- „Einfache“ Diagramme
- Dynamische Diagramme mit PHP



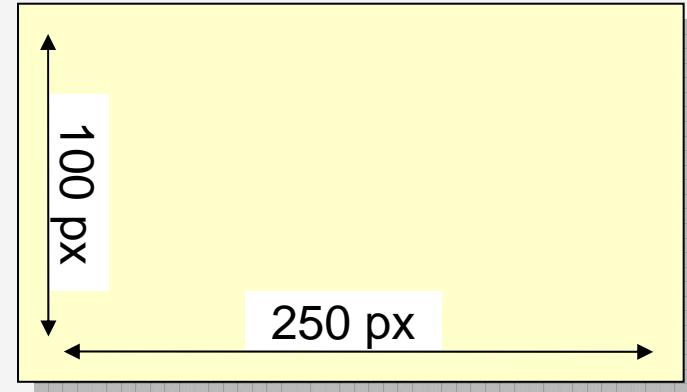
- Google Chart API (Application Programming Interface)
 - *<http://chart.apis.google.com/chart?>*
- Funktionsweise
 - Absenden von Diagrammparametern
 - Vergleichbar mit PHP Aufruf mit Übergabeparametern
 - *<URL>?<parameter name>=<parameter value>&...*
 - Erhalten einer Graphik
- Einbinden in Website
 - Verwenden eines `` Tags
 - Als **src** des Tags (entspricht der Quelle) den API Aufruf angeben.



Ausgangssituation

- Basisdaten
 - 3 Werte: 60, 10, 30
 - Textformat
 - Chart Data: ***chd=t:60,10,30***
- Diagrammgröße
 - Länge: 250 px
 - Höhe: 100 px
 - Chart Size: ***chs=250x100***

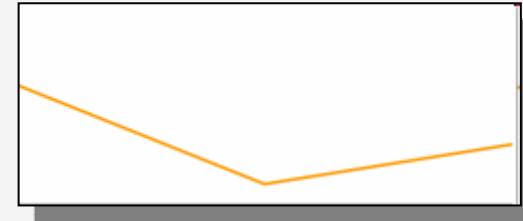
ID	Wert
1	60
2	10
3	30





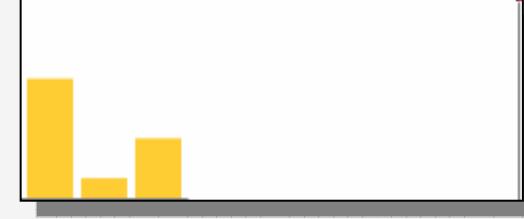
Diagrammarten 1

- Liniendiagramm
 - Chart Type: *cht=lc*



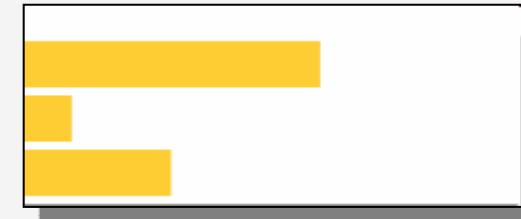
<http://chart.apis.google.com/chart?chs=250x100&chd=t:60,10,30&cht=lc>

- Vertikales Balkendiagramm
 - Chart Type: *cht=bvs*



<http://chart.apis.google.com/chart?chs=250x100&chd=t:60,10,30&cht=bvs>

- Horizontales Balkendiagramm
 - Chart Type: *cht=bhs*

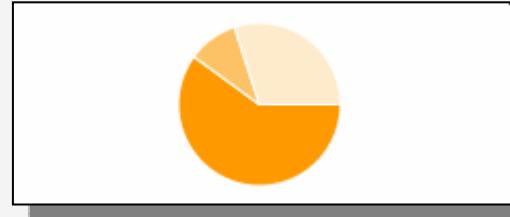




Diagrammarten 2

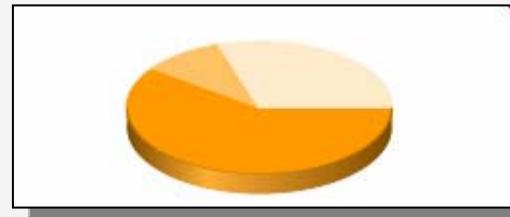
- Tortendiagramm

...cht=p



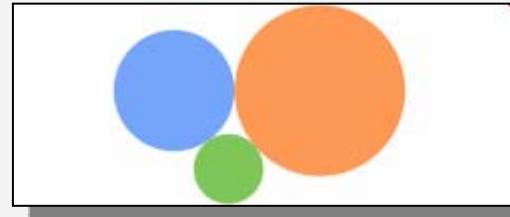
- 3D Tortendiagramm

...cht=p3



- Venn-Diagramm

...cht=v



- Google Manometer

...cht=gom

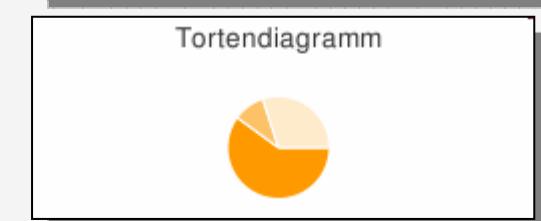
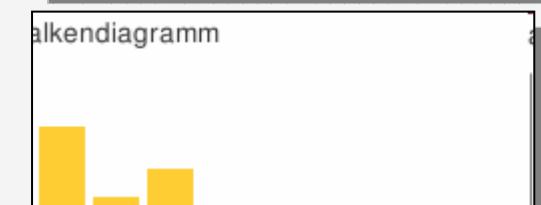
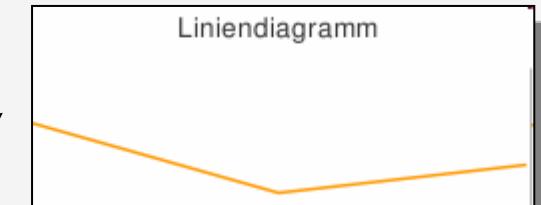




Diagrammtitel

- Parameter
 - Chart Title: ***chtt=Titel***
 - Einschränkungen
 - Keine Leerzeichen
 - Ersatzzeichen „+“ (Bsp.: Bruck+an+der+Mur)
 - Keine Umlaute

```
http://chart.apis.google.com/chart?  
chs=250x100  
&chd=t:60,10,30  
&cht=lc  
&chtt=Liniendiagramm
```

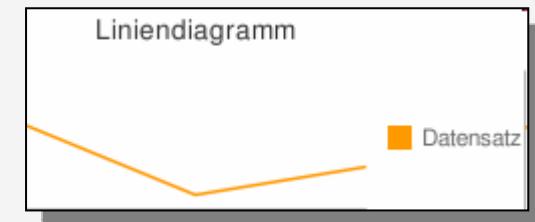




Legenden

- Parameter
 - Chart Legend: *chdl=Legendentext*
 - Liniendiagramm

```
http://chart.apis.google.com/chart?  
chs=250x100  
&chd=t:60,10,30  
&cht=lc  
&cht=Liniendiagramm  
&chdl=Datensatz
```

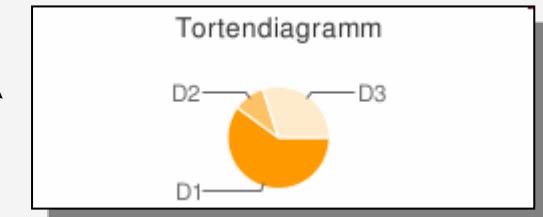
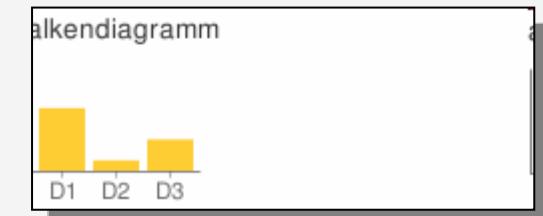
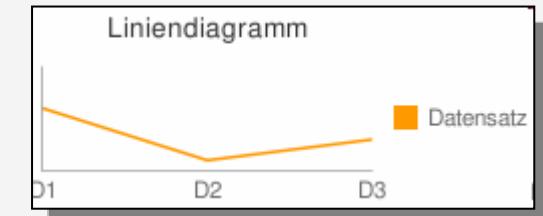




Achsenbeschriftung 1

- Parameter
 - Chart Label:
chl=Berschriftung1/Beschiftung2/Beschriftung3

```
http://chart.apis.google.com/chart?  
chs=250x100  
&chd=t:60,10,30  
&cht=p  
&chtt=Tortendiagramm  
&chl=D1|D2|D3
```

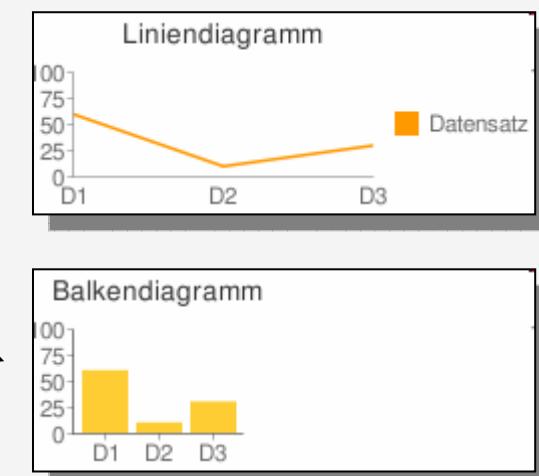




- Parameter

- Chart Axes Type: ***chxt=x,y,t,r***
 - x: X-Achse (untere horizontale Linie)
 - y: Y-Achse (linke vertikale Linie)
 - t: TOP-Achse (obere horizontale Linie)
 - r: RIGHT-Achse (rechte vertikale Linie)

```
http://chart.apis.google.com/chart?  
chs=250x100  
&chd=t:60,10,30  
&cht=bvs  
&chtt=Balkendiagramm  
&chl=D1|D2|D3  
&chxt=y
```



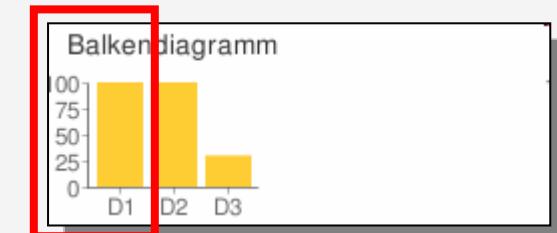


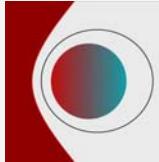
- Ausgangssituation

ID	Wert
1	150
2	100
3	30

- Ergebnis

```
http://chart.apis.google.com/chart?  
chs=250x100  
&chd=t:150,100,30  
&cht=bvs  
&chtt=Balkendiagramm  
&chl=D1|D2|D3  
&chxt=y
```





- Ausgangssituation

- Wertebereich: 0 – 100 [%]
- Maximalwert der Daten: 100 [%]
- Restliche Daten auf Maximalwert „abstimmen“

$$\bar{d}_i = \frac{d_i}{d_{max}} 100$$

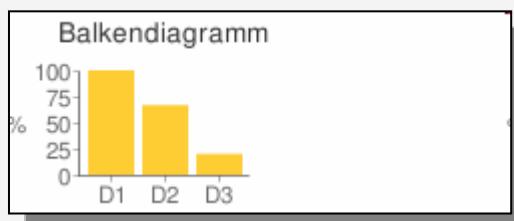
$$d_1 : \frac{150}{150} 100 = 100$$

$$d_2 : \frac{100}{150} 100 = 66.6$$

$$d_3 : \frac{30}{150} 100 = 20$$

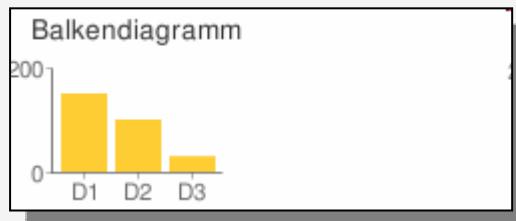
ID	Wert	%
1	150	100.0
2	100	66.6
3	30	20.0

- Ergebnis





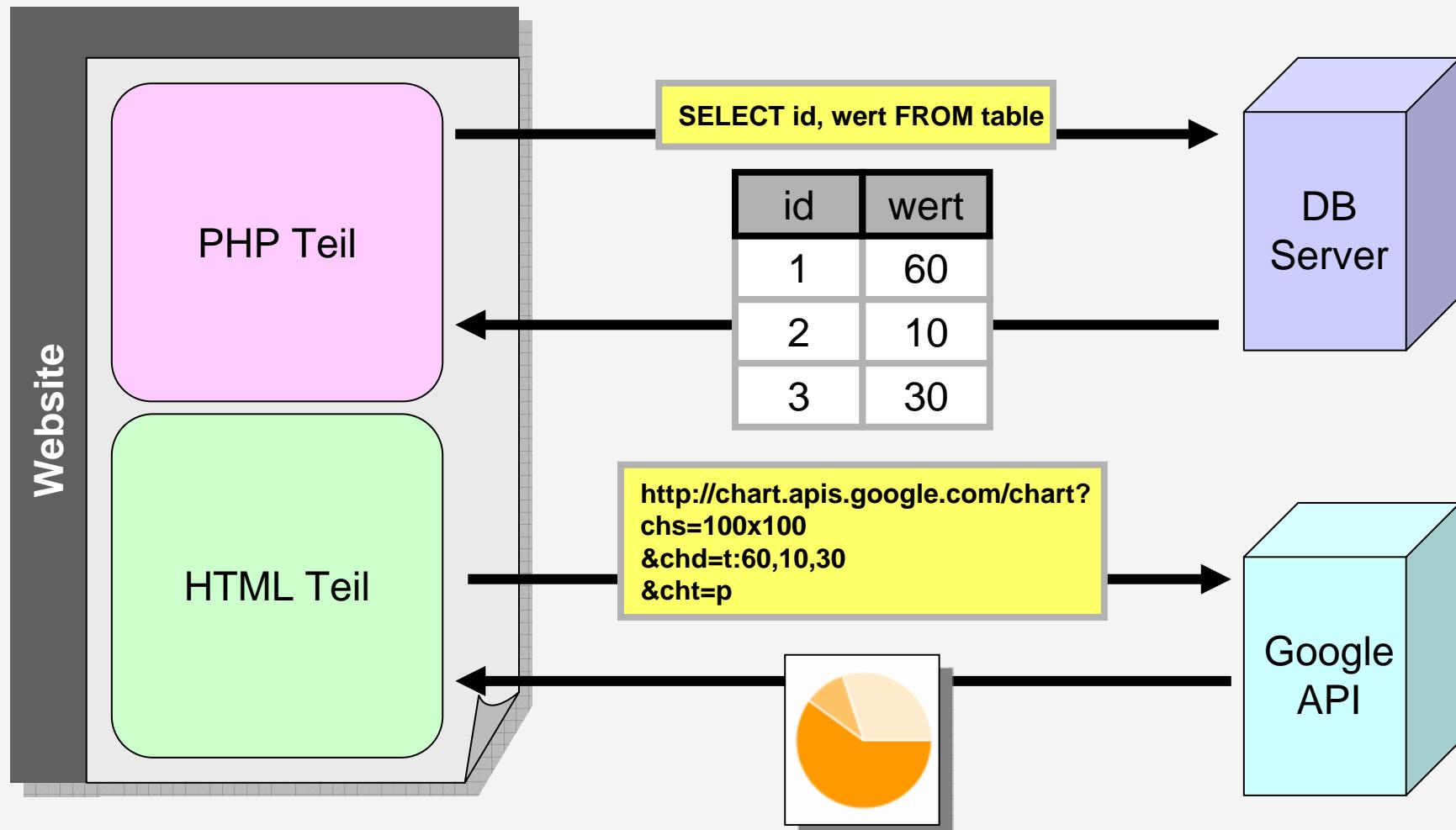
- Ausgangssituation
 - Wertebereich: absolute Werte (150, 100, 30)
 - absolutwert abhängige Skalierung
 - Chart Data Scaling: ***chds=min,max***
 - Manuelle Y-Achsenbeschriftung
 - Chart Axes Label: ***chxl=axes number:/label1/labeli/labeln***
 - axes number von chxt (Chart Axes Type) abhängig (beginnt mit 0)
- Ergebnis



```
http://chart.apis.google.com/chart?  
chs=250x100  
&chd=t:150,100,30  
&cht=bvs  
&cht=Balkendiagramm  
&chl=D1|D2|D3  
[&chxt=y]  
[&chds=0,200]  
[&chx=0:1|200]
```



Dynamische Diagramme mit PHP





Beispiel HTML Teil

```
1
2  <?php
3
4
5
6
7
8
9
10 <?php
11
12 <html>
13   <head>
14     <title>Google Chart</title>
15   </head>
16   <body>
17     <h1>Bezirksstatistik</h1>
18
19
20     
35
36     <hr>
37     <p>Clemens Strauß, TU Graz 2009</p>
38   </body>
39 </html>
```



```
1
2 <?php
3 // DB Variablen
4 $host = "129.27.89.66";
5 $port = "5432";
6 $user = "student";
7 $password = "student";
8 $dbname = "lv_gfo_master";
9
10 $conn = pg_connect("host=$host port=$port dbname=$dbname user=$user password=$password");
11 $strSQL = "select count(*), sum(flaeche), min(flaeche), max(flaeche), avg(flaeche) from bezirk";
12 $result = pg_query($conn, $strSQL);
13 $rowStat = pg_fetch_row($result);
14 $statCount = $rowStat[0];
15 $statSum = $rowStat[1];
16 $statMin = $rowStat[2];
17 $statMax = $rowStat[3];
18 $statAvg = $rowStat[4];
19 $statRelAvg = ($statAvg / $statMax) * 100;
20 $statRelMin = ($statMin / $statMax) * 100;
21 $statRelMax = ($statMax / $statMax) * 100;
```



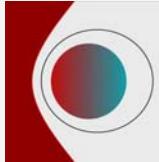
Beispiel PHP Teil b

```
23     $strSQL = "select id, name, flaeche from bezirk order by name desc";
24     $result = pg_query($conn, $strSQL);
25     $strName = "";
26     $intCounter = 1;
27     while ($row = pg_fetch_row($result)) {
28         if ($intCounter == $statCount) {
29             $strName.= $row[1];
30         }
31         else {
32             $strName.= $row[1]."," ;
33         }
34         $intCounter++;
35     }

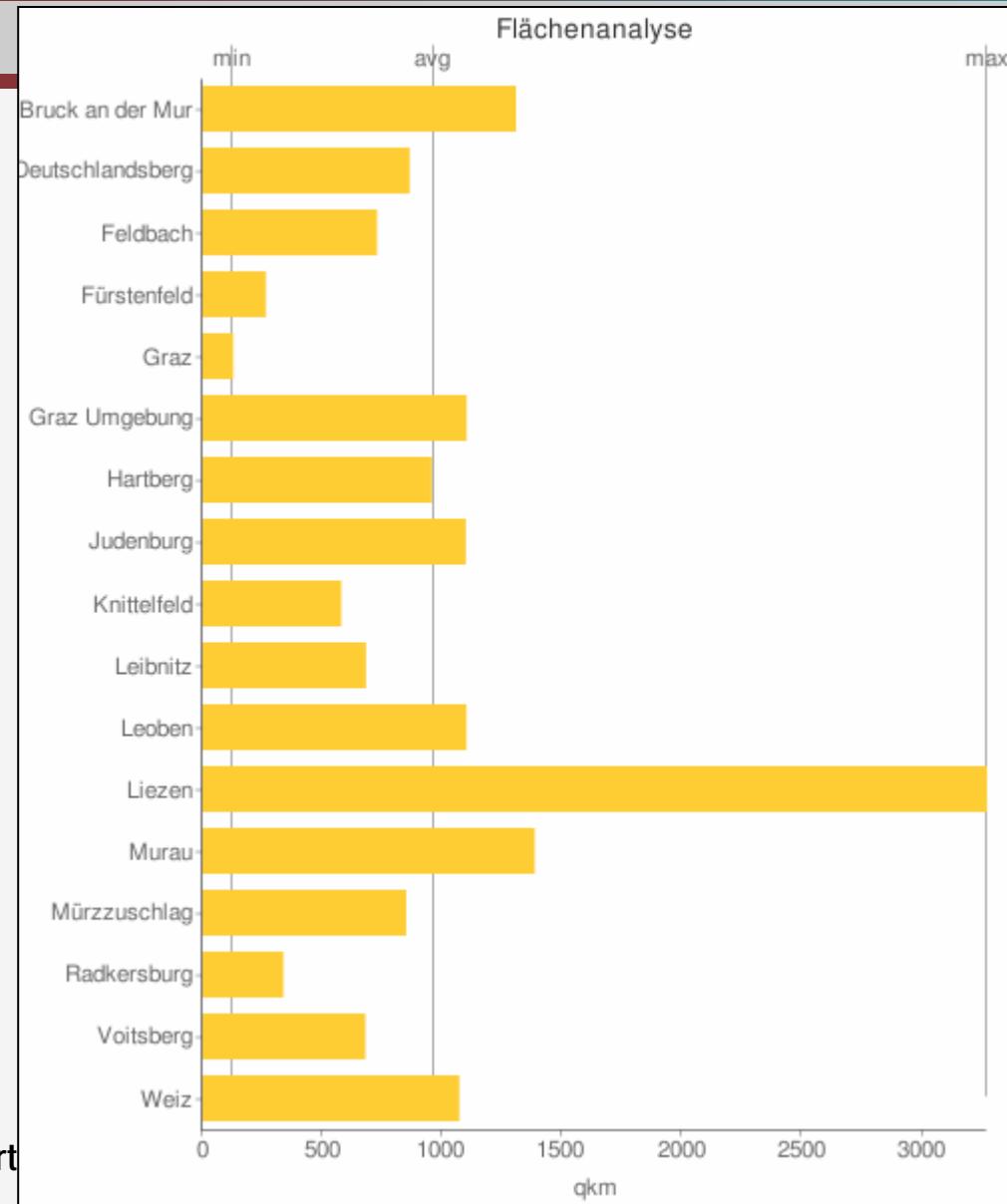
36
37     $strSQL = "select id, name, flaeche from bezirk order by name asc";
38     $result = pg_query($conn, $strSQL);
39     $strFlaeche = "";
40     $intCounter = 1;
41     while ($row = pg_fetch_row($result)) {
42         if ($intCounter == $statCount) {
43             $strFlaeche .= $row[2];
44         }
45         else {
46             $strFlaeche .= $row[2]."," ;
47         }
48         $intCounter++;
49     }

50
51     // Ersetzen der Leerzeichen durch +
52     $strName = str_replace(" ", "+", $strName);
53     // Ersetzen von ü durch ue
54     $strName = str_replace("ü", "%C3%BC", $strName);

55
56     pg_close($conn);
57 ?>
```



Beispiel Ergebnis



Google Chart

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

PHP Teil 3

Geoinformatik 2

Konstruktionsübung

WS 2011/12

Clemens Strauß



Zeitplan

Einheit	Datum	Inhalt	Übungsprogramm
1	10.10.2011	Vorbesprechung	
2	17.10.2011	Datenbankdesign	Ausgabe 1
3	24.10.2011	Normalformen	
4	31.10.2011	SQL	Ausgabe 2
5	07.11.2011	SQL Übung	
6	14.11.2011	spatialSQL	
7	21.11.2011	spatialSQL Übung	Abgabe 1
8	28.11.2011	QuantumGIS	
9	05.12.2011	Mitarbeiterüberprüfung	Ausgabe 3
10	12.12.2011	HTML (nur für Interessierte)	
11	09.01.2012	PHP	Abgabe 2
12	16.01.2012	PHP + DB	
13	23.01.2012	PHP + HTML ImageMaps	
14	30.01.2012	Klausur	Abgabe 3

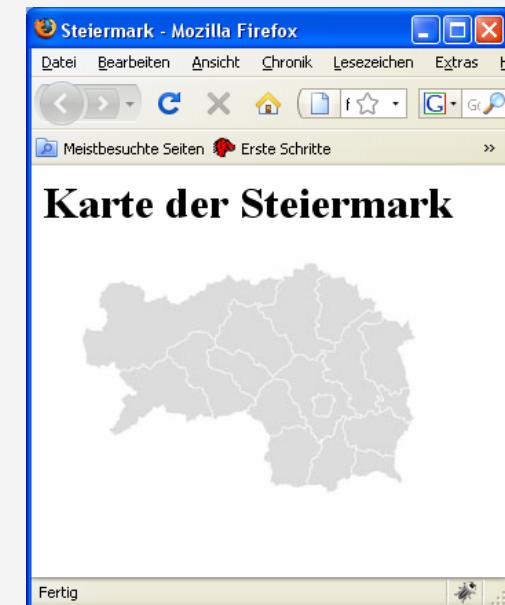


- Graphiken in HTML
- HTML Image Map
- Dynamische Map mit DB Zugriff



- Darstellen einer Graphik in einer Website
 - Minimale Parametrisierung
``
 - Beispiel:

```
1 <html>
2   <head>
3     <title>Steiermark</title>
4   </head>
5   <body>
6     <h1>Karte der Steiermark</h1>
7     
8   </body>
9 </html>
```

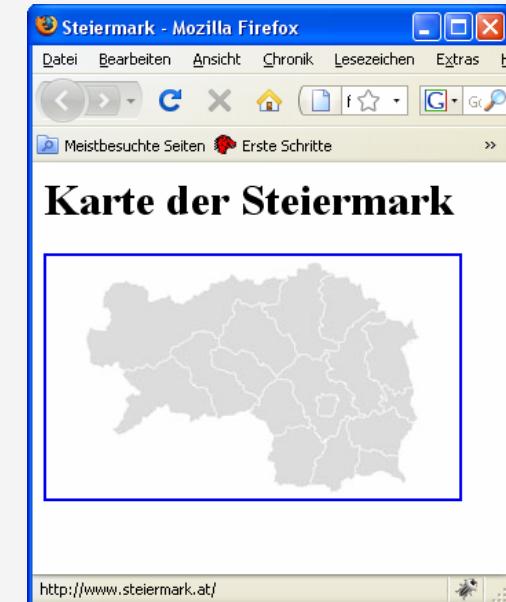




Graphik als Link

- Verweisensitive Graphik in HTML
 - Minimale Parametrisierung
 - Beispiel:

```
1 <html>
2   <head>
3     <title>Steiermark</title>
4   </head>
5   <body>
6     <h1>Karte der Steiermark</h1>
7     <a href="http://www.steiermark.at"></a>
8   </body>
9 </html>
```



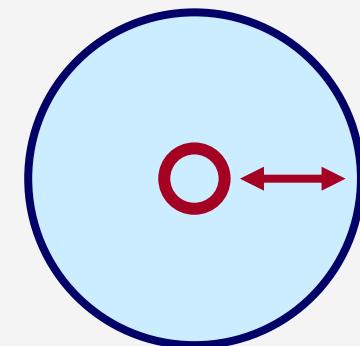
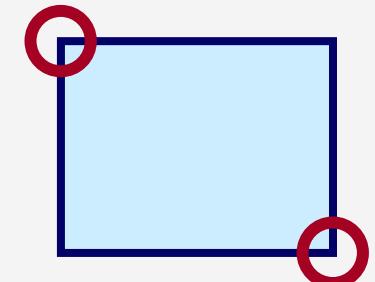
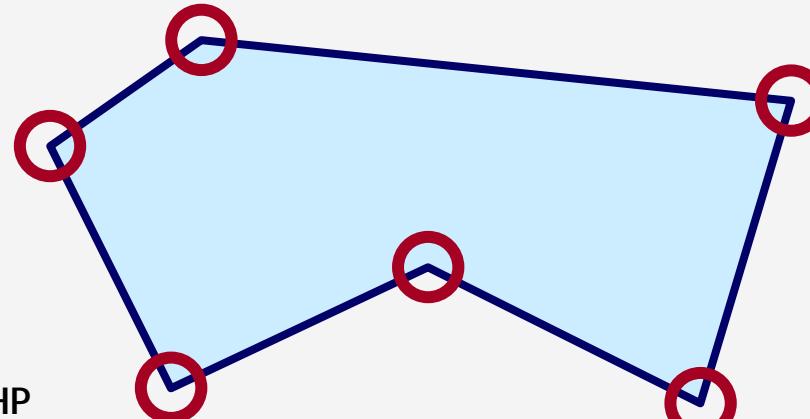


- Graphik mit verweisensensitiven Zonen
 - Definition von Flächen innerhalb einer Graphik, die auf bestimmte Adresse verweisen





- Zonendefinition
 - Form
 - Rechteck
 - Linke obere Ecke, rechte untere Ecke
 - Kreis
 - Kreismittelpunkt, Radius
 - Polygon
 - Eckpunkte des Polygons
 - Nicht geschlossen





- Zonendefinition
 - Referenzsystem der Koordinatenangaben:
Bildkoordinatensystem
 - Einheit: **Pixel**





HTML Image Map

- Erweiterung einer Graphik um verweisensitive Zonen
 - Zusätzliches Attribut innerhalb des Tags

```
<img src=„[Pfad/]Dateiname“ usemap=„#zonen“>
```
 - Definition der Zonen

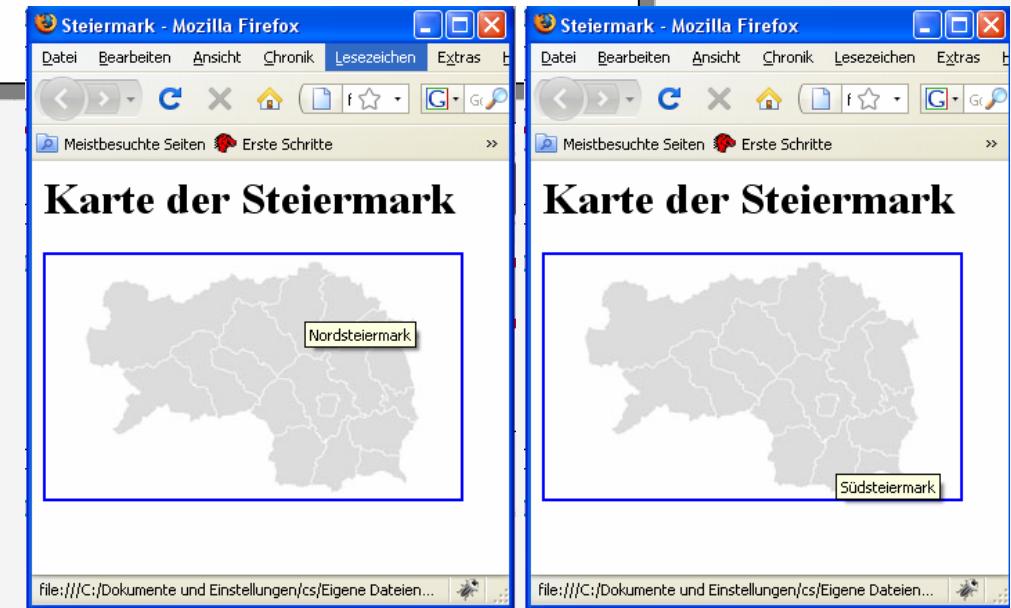
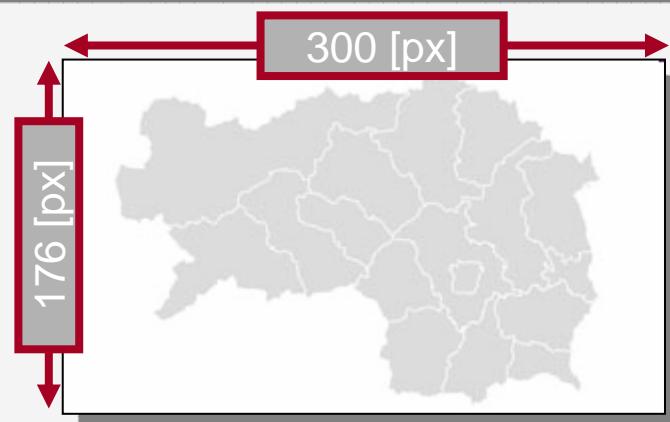
```
<map name=„zonen“>  
  <area shape=„form“  
    coords=„X,Y,...“  
    href=„Link“  
    title=„Text“>  
</map>
```



HTML Image Map

- Beispiel

```
1 <html>
2   <head>
3     <title>Steiermark</title>
4   </head>
5   <body>
6     <h1>Karte der Steiermark</h1>
7     
8     <map name="stmkzonen">
9       <area shape="rect" coords="0,0,300,88" href="nordstmk.htm" title="Nordsteiermark">
10      <area shape="rect" coords="0,89,300,176" href="sudstmk.htm" title="Südsteiermark">
11    </map>
12  </body>
13 </html>
```





Dynamische Zonendefinition

- Koordinatewerte aus PostGIS DB für Zonendefinition nutzen
- Zugriff auf PostGIS DB mittels PHP-Funktionen
- Dynamisches Zusammenstellen der <area> Tags mit PHP

```
1 <html>
2   <head>
3     <title>Steiermark</title>
4   </head>
5   <body>
6     <h1>Karte der Steiermark</h1>
7     
8     <map name="stmkzonen">
9       <area shape="rect" coords="0,0,300,88" href="nordstmk.htm" title="Nordsteiermark">
10      <area shape="rect" coords="0,89,300,176" href="sudstmk.htm" title="Südsteiermark">
11    </map>
12  </body>
13 </html>
```

dynamisch!



- Vorbereitung

- Auswahl der zu nutzenden Form (rect, circle, poly)
- Abfrage der notwendigen Daten aus PostGIS DB
 - Attribute
 - Geometrie: astext()-Funktion, x()-Funktion, y()-Funktion
- Koordinatentransformation (Ebene Drehstreckung)
 - Raumkoordinaten (aus PostGIS DB)
 - Bildkoordinaten (gesucht)
 - Benötigt: räumliche Ausdehnung der Graphik
- Nach Abschluss der Vorbereitung soll folgendes Vorhanden sein:

Titel	Link	Bildkoordinate X	Bildkoordinate Y
Graz	analyse.php?id=2	283	145
Leoben	analyse.php?id=8	170	94



Dynamische Zonendefinition

- Erzeugung der Zonen

```
1 <html>
2   <head>
3     <title>Steiermark</title>
4   </head>
5   <body>
6     <h1>Karte der Steiermark</h1>
7     
8     <map name="stmkzonen">
9       <?php
10
11       // Datenbankabfrage
12
13       // Datenvorbereitung
14
15       // Erzeugung der Zonen
16
17       ?>
18       </map>
19     </body>
20   </html>
```



```
9 <?php
10
11 // Bildvariablen
12 $dblMinX = 368844;
13 $dblMaxX = 611027;
14 $dblDX = $dblMaxX - $dblMinX;
15
16 $dblMinY = 5158927;
17 $dblMaxY = 5300603;
18 $dblDY = $dblMaxY - $dblMinY;
19
20 // Bildgrössen
21 $intDXB = 300;
22 $intDYB = 176;
23
24 // Verkürzungsfaktoren
25 $VFX = $dblDX / $intDXB;
26 $VFY = $dblDY / $intDYB;
27
28 // Variablen für DB Zugriff
29 $host = "129.27.89.66";
30 $port = "5432";
31 $user = "student";
32 $password = "student";
33 $dbname = "lv_giz2";
34
35 // DB Abfrage
36 $conn = pg_connect("host=$host port=$port dbname=$dbname user=$user password=$password");
37 $strSQL = "select id, name, x(centroid(geom)), y(centroid(geom)) from stmk_bezirke";
38 $result = pg_query($conn, $strSQL);
39 $rows = pg_num_rows($result);
40 $pg_close();
41
42 // Zeilenweise abarbeiten des Abfrageergebnisses
43 for ($i = 1; $i <= $rows; $i++) {
44
45   // Auslesen der Daten
46   $DataRow = pg_fetch_row($result);
47   $intId = $DataRow[0];
48   $strTitle = $DataRow[1];
49   $dblX10 = $DataRow[2];
50   $dblY10 = $DataRow[3];
51
52   // Koordinatentransformation
53   $intX10 = round((($dblX10 - $dblMinX) / $VFX),0);
54   $intY10 = round((($dblMaxY - $dblY10) / $VFY),0);
55
56   // Erzeugung der Zone
57   echo "<area shape=\"circle\" coords=\"$intX10,$intY10,10\" href=\"analyse.php?id=$intId\" title=\"$strTitle\">";
58 }
59 ?>
```



Dynamische Zonendefinition



Quelltext von: http://129.27.89.66/student/lv_gj2/master/steiermark.php - Mozilla Firefox

```
<html>
  <head>
    <title>Steiermark</title>
  </head>
  <body>
    <h1>Karte der Steiermark</h1>
    
    <map name="stmkzonen">
      <area shape="circle" coords="204,108,10" href="analyse.php?id=1" title="Graz Stadt">
      <area shape="circle" coords="186,40,10" href="analyse.php?id=2" title="Bruck an der Mur">
      <area shape="circle" coords="181,145,10" href="analyse.php?id=3" title="Deutschlandsberg">
      <area shape="circle" coords="242,128,10" href="analyse.php?id=4" title="Feldbach">
      <area shape="circle" coords="257,107,10" href="analyse.php?id=5" title="Fürstenfeld">
      <area shape="circle" coords="198,100,10" href="analyse.php?id=6" title="Graz Umgebung">
      <area shape="circle" coords="250,71,10" href="analyse.php?id=7" title="Hartberg">
      <area shape="circle" coords="121,87,10" href="analyse.php?id=8" title="Judenburg">
      <area shape="circle" coords="144,83,10" href="analyse.php?id=9" title="Knittelfeld">
      <area shape="circle" coords="211,149,10" href="analyse.php?id=10" title="Leibnitz">
      <area shape="circle" coords="154,59,10" href="analyse.php?id=11" title="Leoben">
      <area shape="circle" coords="68,46,10" href="analyse.php?id=12" title="Liezen">
      <area shape="circle" coords="215,37,10" href="analyse.php?id=13" title="Mürzzuschlag">
      <area shape="circle" coords="65,100,10" href="analyse.php?id=14" title="Murau">
      <area shape="circle" coords="241,151,10" href="analyse.php?id=15" title="Radkersburg">
      <area shape="circle" coords="170,111,10" href="analyse.php?id=16" title="Voitsberg">
      <area shape="circle" coords="226,80,10" href="analyse.php?id=17" title="Weiz">
    </map>
  </body>
</html>
```



- 30.01.2012: AE01 Klausur
 - 60 min.
 - 10 Fragen
 - keine Unterlagen
 - Anmeldung im TUGonline
- Fragen ?

PHP Teil 3

Geoinformatik 2
Konstruktionsübung
WS 2011/12

Clemens Strauß

Institut für Geoinformation

1. Übungsprogramm

Geoinformatik 2, Konstruktionsübung
WS 2011/12

Dipl.-Ing. Dr.techn.
Clemens Strauß
Univ. Ass.

Steyrergasse 30/III
8010 Graz

Tel.: +43 316 873-6846
Fax: +43 316 873-6845

clemens.strauss@TUGraz.at
www.geoinformation.tugraz.at

Ziel

Das Ziel der auf einander aufbauenden Übungsprogramme ist die Organisation, Analyse und Präsentation von Geodaten, die im Zuge von Vermessungstätigkeiten administriert werden.

Das erste Übungsprogramm umfasst die Konzeption einer Datenbank, in der Festpunkte (Triangulierungs-, Einschalt-, Mess-, Polygon- und Höhenpunkte nach Einteilung BEV) mit vermessungsrelevanten Eigenschaften (vgl. Punktopographien) verspeichert werden sollen. Zusätzlich sollen frei zugängige Geodaten des GIS Steiermark (<http://www.gis.steiermark.at/cms/ziel/14292094/DE/>) als Ergänzung des Datenbestands dienen (Orte, Bezirks- und Landesgrenzen).

In den weiteren Übungsprogrammen wird diese Datenbasis für Analysen und zu Präsentationszwecke weiterverwendet.

Aufgabenstellung

Die nachfolgenden Aufgabenstellungen sind, bezogen auf die erreichbare Beurteilung, gestaffelt angeführt (Note 4 bis Note 1):

Aufgabe 1 für Note 4: Entwickeln Sie ein konzeptionelles Datenbankmodell für die Organisation von Festpunkten. Die benötigten Relationen und Attribute sind so zu wählen, dass diese aus den Erfahrungen der bisher besuchten LVs (z.B.: Vermessungskunde EF) als sinnvoll zu erachten sind. Weiters sollen zu jedem Festpunkt eine beliebig hohe Anzahl von Fernzielen (ebenfalls Festpunkte) zuweisbar sein (z.B.: Von Festpunkt 1002 sieht man die Festpunkte 323, 4432 und KT234). Die GIS Steiermark Daten sind in diesem Datenbankmodell ebenfalls zu integrieren (z.B.: Festpunkt liegt in einem Bezirk, usw.).

Dieses konzeptionelle Datenbankmodell ist in einem *Entity Relationship Modell (ERM)* darzustellen und zu erläutern.

Aufgabe 2 für Note 3: Analysieren Sie das ERM aus Aufgabe 1 hinsichtlich der Erfüllung von Normalformen (NF 1 bis NF 3). Sollte das ERM nicht den Normalformen genügen, ist das konzeptionelle Modell so zu überarbeiten, dass diese Normalformen erfüllt werden.

Weiters ist ein logisches Datenbankmodell aus dem bestehenden und überprüften konzeptionellen Modell abzuleiten und in Form von Tabellengerüsten zu visualisieren.

Die Normalformüberprüfung sowie die Ableitung des logischen Modells sind in nachvollziehbarer Form zu dokumentieren.

Aufgabe 3 für Note 2: Weisen Sie jene Attribute aus, die als Schlüsselemente verwendet werden (Primärschlüssel und Fremdschlüssel). Begründen Sie, aus welchem Grund Sie diese Attribute als Schlüsselemente ausgewählt haben.

Aufgabe 4 für Note 1: Definieren Sie für alle Attributfelder den Datentyp und den dazugehörigen zulässigen Wertebereich und begründen Sie Ihre Auswahl.

Durchführung

Das Übungsprogramm ist in Kleingruppen bis zu 3 Personen zu bearbeiten und in einem Bericht zu dokumentieren.

Die Gruppeneinteilung erfolgt im TeachCenter und ist für alle drei Übungsprogramme einzuhalten.

Abgabe

Der Bericht über das Übungsprogramm ist im Zuge eines kurzen Abgabegesprächs (5 – 10 min.) abzugeben. Die Abgabe findet in der KW 47 statt, wobei die genaue Reihenfolge mit Datum und Uhrzeit im TeachCenter administriert wird.

Clemens Strauß, 17.10.2011

Institut für Geoinformation

Dipl.-Ing. Dr.techn.
Clemens Strauß
Univ. Ass.

Steyrergasse 30/III
8010 Graz

Tel.: +43 316 873-6846
Fax: +43 316 873-6845

clemens.strauss@TUGraz.at
www.geoinformation.tugraz.at

2. Übungsprogramm

Geoinformatik 2, Konstruktionsübung
WS 2011/12

Ziel

Das Ziel der auf einander aufbauenden Übungsprogramme ist die Organisation, Analyse und Präsentation von Geodaten, die im Zuge von Vermessungstätigkeiten administriert werden.

Das zweite Übungsprogramm umfasst die Realisierung des logischen Modells in einem PostgreSQL/PostGIS Datenbankmanagement Systems (DBMS).

Aufgabenstellung

Die nachfolgenden Aufgabenstellungen sind, bezogen auf die erreichbare Beurteilung, gestaffelt angeführt (Note 4 bis Note 1):

Aufgabe 1 für Note 4: Erstellen Sie jene Tabellen, die Sie im Zuge des 1. Übungsprogramms konzipiert haben, in einer PostgreSQL/PostGIS Datenbank. Die benötigten Datenbank(-zugangs)-parameter entnehmen Sie dem Abschnitt Durchführung.

Die Geometrieinformation soll dem EPSG 32633 (UTM33N) entsprechen!

Aufgabe 2 für Note 3: Befüllen Sie die in Aufgabe 1 erstellte Datenbank mit 20 Festpunkten mit beliebigen Attributwerten, wobei diese sich innerhalb der Steiermark befinden müssen.

Aufgabe 3 für Note 2: Beantworten Sie folgende Fragen mit Hilfe der Datenbank aus Aufgabe 2:

- Welche Festpunkte befinden sich im Bezirk Leoben?
- Welches ist der nördlichste Festpunkt?
- Welcher Festpunkt kommt am meisten als Fernziel vor?
- Wie weit ist der westlichste Festpunkt vom östlichsten Festpunkt entfernt (km)?
- Wie viele Festpunktepunkte befinden sich im Umkreis von 20 km um die Ortschaft Niedergams?
- Wie lauten die GK M34 Koordinaten bzw. Geographischen Koordinaten (WGS 84, Format GMS) der Festpunkte im Bezirk Graz?
- Welche zwei Festpunkte besitzen die geringste Entfernung zueinander?

- Wie groß ist die Fläche (km²), die alle Festpunkte umspannt?
- Wie viele Festpunkte liegen im flächenmäßig zweitkleinsten Bezirk?

Aufgabe 4 für Note 1: Visualisieren Sie die Antworten aus Aufgabe 3 in übersichtlicher Weise mit der Software Quantum GIS.

Durchführung

Das Übungsprogramm ist in den bestehenden Kleingruppen zu bearbeiten und in einem Bericht zu dokumentieren. Die zu benutzende Datenbank lv_gi2_pc{31-39}_up ist aus der Gruppeneinteilung im TeachCenter ersichtlich.

Datenbankparameter:

- Host: 129.27.89.66
- Port: 5432
- DB Name: lv_gi2_pc{31-39}_up
- Benutzername: student
- Passwort: student

Abgabe

Der Bericht über das Übungsprogramm ist im Zuge eines kurzen Abgabegesprächs (5 – 10 min.) abzugeben. Die Abgabe findet in der KW 02 statt, wobei die genaue Reihenfolge mit Datum und Uhrzeit im TeachCenter administriert wird.

Clemens Strauß, 31.10.2011

Institut für Geoinformation

3. Übungsprogramm
Geoinformatik 2, Konstruktionsübung
WS 2011/12

Dipl.-Ing. Dr.techn.
Clemens Strauß
Univ. Ass.

Steyrergasse 30/III
8010 Graz

Tel.: +43 316 873-6846
Fax: +43 316 873-6845

clemens.strauss@TUGraz.at
www.geoinformation.tugraz.at

Ziel

Das Ziel der auf einander aufbauenden Übungsprogramme ist die Organisation, Analyse und Präsentation von Geodaten, die im Zuge von Vermessungstätigkeiten administriert werden.

Das dritte Übungsprogramm beinhaltet die Umsetzung einer Benutzeroberfläche mit vielschichtigen Interaktionsmöglichkeiten.

Aufgabenstellung

Die nachfolgenden Aufgabenstellungen sind, bezogen auf die erreichbare Beurteilung, gestaffelt angeführt (Note 4 bis Note 1):

Aufgabe 1 für Note 4: Entwickeln Sie ein Web-Interface mit dem neue Festpunkte in der Datenbank verspeichert und bestehende in einer Liste angezeigt und gelöscht werden können. Zusätzlich soll dieses Web-Interface dem Benutzer die Möglichkeit geben, die Koordinaten in verschiedenen Projektions- bzw. Koordinatensystemen (UTM 32 N, UTM 33 N, GK M28, GK M31, GK M34, BMN, GCS WGS84) einzugeben bzw. abzufragen.

Aufgabe 2 für Note 3: Erweitern Sie das Web-Interface um eine Festpunkt-Exportfunktion mit folgenden Attribut bezogenen Auswahlmöglichkeiten:

- Zu exportierende Attribute.
- Projektions- bzw. Koordinatensystem (UTM 32 N, UTM 33 N, GK M28, GK M31, GK M34, BMN, GCS WGS84) des Geometrieattributs.
- Auswahl des Trennzeichens zwischen den einzelnen Attributwerten.

Aufgabe 3 für Note 2: Erweitern Sie das Web-Interface um eine HTML Image Map.

Hierfür soll die benötigte Graphik in QuantumGIS unter Verwendung aller Festpunkte und einer sinnvollen Auswahl an GIS Steiermark Daten entstehen. Die berührungssensitiven Zonen, dynamisch erzeugt und mittels einer Transformation von räumlichen Koordinaten in Bildkoordinaten mit der Graphik in Bezug gebracht, sollen als Kreise um die Festpunkte gestaltet sein, wobei der Radius benutzerfreundlich zu wählen ist. Die Website, auf die der Link der berührungssensitiven Zone verweist, soll alle Attribute des ausgewählten Festpunktes auflisten. Die Werte des Geometrieattributes ist in UTM 33 N und in GCS WGS84 (Format GMS) darzustellen.

Aufgabe 4 für Note 1: Ergänzen Sie die Funktionalität aus Aufgabe 3 soweit, dass zusätzlich zu den Festpunktattributen Informationen zu den Fernzielen in einem bestimmten Umkreis um den ausgewählten Festpunkt dargestellt werden. Der Umkreisradius soll hierbei vom Benutzer selbst gewählt werden können.

Durchführung

Das Übungsprogramm ist in den bestehenden Kleingruppen zu bearbeiten und in einem Bericht zu dokumentieren

Webspaceparameter

- FTP Zugang:
 - Host: 129.27.89.66
 - Port: 1234
 - Benutzername: student
 - Passwort: student
 - Verzeichnis: lv_gi2/pc{31-39}_up
- HTTP Zugang:
 - Adresse: http://129.27.89.66/student/lv_gi2/pc{31-39}_up

Abgabe

Der Bericht über das Übungsprogramm ist im Zuge eines kurzen Abgabegesprächs (5 – 10 min.) abzugeben. Die Abgabe findet in der KW 05 statt, wobei die genaue Reihenfolge mit Datum und Uhrzeit im TeachCenter administriert wird.

Clemens Strauß, 05.12.2011

HTML Gerüst

```
<html>
  <head>...</head>
  <body>...</body>
</html>
```

Dokumenttitel (head)

```
<title>...</title>
```

Metadaten des Dokuments (head)

```
<meta name = "{description | author | keywords} contend = "...">
```

Textstrukturierung / Überschrift

```
<h{1,...,6}>...</h{1,...,6}>
```

Textstrukturierung / Absatz

```
<p>...</p>
```

Textstrukturierung / Erzwungener Zeilenumbruch

```
<br>
```

Textstrukturierung / Horizontale Linie

```
<hr>
```

Referenz (Link)

```
<a href = "[mailto:]...">...</a>
```

Graphik

```
<img src = "..." alt = "...">
```

Tabelle

```
<table>
  <tr>
    <th>...</th>[<th>...</th>]
  </tr>
  <tr>
    <td>...<td>[<td>...</td>]
  </tr>
  [<tr>...</tr>]
</table>
```

Formular Gerüst

```
<form action = "..."><...></form>
```

Textfeld

```
<input type = "text" name = "... " size = "... " maxlength = "... " [value = "..."]>
```

Radio Button

```
<input type = "radio" name = "... " value = "...">
```

Checkbox

```
<input type = "checkbox" name = "... " value = "...">
```

Auswahlliste

```
<select name = "... " size = "... ">  
  <option value = "... ">...</option>  
  [<option value = "... ">...</option>]  
</select>
```

Formular übermitteln

```
<input type = "submit" value = "...">
```

Formular zurücksetzen

```
<input type = "reset" value = "...">
```

Zeichenschlüssel

<klein geschrieben>:	HTML Tag
...	Benutzereingabe
[...]:	Optionaler Teil
{... ...}:	Eines der angeführten Elemente ist zu verwenden

Aufruf eines PHP-Dokumentes ohne Übergabeparameter

<URL>/<document name>.php

Aufruf eines PHP-Dokumentes mit Übergabeparameter

<URL>/<document name>.php?<parameter name>=<parameter value>&<parameter name>=<parameter value>

Grundgerüst

```
<?php  
    <code>;  
?>
```

Wertzuweisung

```
$<variable name> = <value>;  
$<variable name> = $<variable name>;  
$<variable name> = <function name>(<value>);  
$<variable name> = <function name>($<variable name>);
```

Wertzuweisung aus Übergabeparameter

\$<variable name> = \$_GET["<parameter name>"];

Wertzuweisung auf Array-Variable

\$<variable name> = array(<value>, <value>[,...]);

Wertzuweisung aus Array-Variable

\$<variable name> = \$<variable name>[<value order>];

Mathematische Operatoren

*{<value> | \$<variable name>} {+ | - | / | * | %} {<value> | \$<variable name>}*

String Operatoren (Concatenation)

*{"<value>" | \$<variable name>} . {"<value>" | \$<variable name>}
\$<variable name> .= {"<value>" | \$<variable name>};*

Vergleichsoperatoren für Bedingungen

[!]{<value> | \$<variable name>} {== | != | <> | <= | >=} [!]{<value> | \$<variable name>}

Logische Operatoren bei Bedingungen

<condition> {AND | OR} <condition>

Entscheidungskonstruktion

```
IF (<condition>) {<execution part>;

IF (<condition>) {<execution part>;
ELSE {< execution part >;

IF (<condition>) {<execution part >;
ELSEIF (<condition>) {<execution part>;
ELSE {<execution part>;
```

Wiederholungskonstruktionen (Schleifen)

```
FOR ( $<variable name> = <value>;
      $<variable name> {<[=] | >[=] | ...} {<value> | $<variable name>};
      $<variable name>{++ | = $<variable name> + <value> | ...})
{<execution part>;

WHILE (<condition>) {<execution part>;

DO {<execution part>;} WHILE (<condition>);
```

Interaktion mit einer PostgreSQL Datenbank

```
$<connection > = pg_connect(„host=<host> port=<port> dbname=<dbname>
user=<user> password=<password>“);

$<result> = pg_query($<connection>, $<SQL string>);

$<row>= pg_fetch_row($<result>);

pg_close($<connection>);
```

Zeichenschlüssel

GROSS GESCHRIEBEN:	PHP-Standardsyntax
klein geschrieben:	Funktionsnamen
klein u. kursiv geschrieben:	benutzerdefinierte Elemente
<...>:	Anfang bzw. Ende eines benutzerdefinierten Elementes, Ausnahme: <?php und ?>
<URL>:	Hier ist ein Internetadresse einzusetzen.
<document name>:	Hier ist der Name des PHP-Dokumentes einzugeben.
<parameter name>:	Hier ist der Variablenname eines Übergabeparameter einzugeben.
<parameter value>:	Hier ist der Variablenwert eines Übergabeparameters einzugeben.
<code>:	Hier ist der PHP-Code einzugeben.
<variable name>:	Hier ist ein Variablenname einzusetzen
<value>:	Hier ist ein Wert einzusetzen
<function name>:	Hier ist der Name einer Funktion einzugeben.
<variable order>:	Hier ist die Position des Wertes innerhalb des Arrays anzugeben, beginnend mit 0

<i><execution part></i> :	Hier ist der Ausführungsteil der Funktion einzugeben
<i><condition></i> :	Hier ist ein logischer Zustand (wahr/falsch) einzugeben
<i><SQL string></i> :	Hier ist eine SQL Anweisung einzugeben
<i><connection></i> :	Hier ist der Name einer Variable für die Verbindungskennung einzugeben
<i><host></i> :	Hier ist der Host des Datenbank Servers einzugeben
<i><port></i> :	Hier ist der Port des Datenbank Servers einzugeben
<i><dbname></i> :	Hier ist der Name der Datenbank am Server einzugeben
<i><user></i> :	Hier ist der Benutzername des Datenbank Servers einzugeben
<i><password></i> :	Hier ist das Passwort des Datenbank Servers einzugeben
<i><result></i> :	Hier ist der Name der Variable einzugeben, auf der das Abfrageergebnis gespeichert wird.
<i><row></i> :	Hier ist der Name der Variable einzugeben, auf der eine Zeile des Abfrageergebnis gespeichert wird.
[...]:	Optionaler Teil, Ausnahme <code>\$_GET[...]</code>
{... ...}:	Eines der angeführten Elemente ist zu verwenden
{... / ...}:	Eines oder mehrere Elemente sind zu verwenden

Wichtige Datentypen

integer: Ganze Zahl (1, 2, 3, 4, 5, ...)
numeric: Gleitkomma- und ganze Zahlen (1.5, 2, 3.14156, ...)
varchar: Zeichenkette mit variabler Länge ('Zeichenkette')

Erstellen einer Tabelle

```
CREATE TABLE <table name>
(<attribute name> <data type> [{PRIMARY KEY | {UNIQUE / NOT NULL}}], ...);
```

Hinzufügen eines Datensatzes zu einer Tabelle

```
INSERT INTO <table name>
(<attribute name>, ...)
VALUES (<value>, ...);
```

```
INSERT INTO <table name> [(<attribute name>)]
SELECT <attribute name>
FROM <table name>
[WHERE ...];
```

Hinzufügen von Datensätzen aus einer Textdatei

```
COPY <table name> [(<attribute name>)]
FROM '<file path and file name>'
[WITH DELIMITER '<delimiter>'];
```

Werte in einer Tabelle ändern

```
UPDATE <table name>
SET <attribute name> = <value>
[WHERE ...];
```

```
UPDATE <table name>
SET <attribute name> = <table name 2>. <attribute name>
FROM <table name 2>
[WHERE ...]);
```

Löschen von Datensätzen aus einer Tabelle

```
DELETE FROM <table name>
[WHERE ...];
```

Löschen einer Tabelle

```
DROP TABLE <table name>;
```

Abfragen aus Tabellen

```
SELECT {*} | [<table name>.]<attribute name>, ...  
FROM <table name>, ...  
[WHERE ...]  
[ORDER BY <attribute name> {ASC | DESC}]  
[LIMIT <value>]  
[OFFSET <value>];
```

Bedingung

```
WHERE <attribute name> {= | < | > | <= | >= | != | <>} <value>  
WHERE <attribute name> IN (<value>, ...)  
WHERE <attribute name> BETWEEN <value min> AND <value max>  
WHERE <attribute name> LIKE '[%]<value>[%]'
```

Kombination von SELECT Anweisungen

```
WHERE <attribute name> {= | IN} (SELECT ... FROM ... [WHERE ...] [...]);
```

Kombination von Bedingungen

```
WHERE <condition 1> AND <condition 2>  
WHERE <condition 1> OR <condition 2>  
WHERE <condidion 1> OR (<condition 2> AND <condition 3>)  
WHERE (<condition 1> OR <condition 2>) AND <condition 3>
```

Aggregatsfunktionen

```
SELECT {avg(<attribute name>) /  
        count(*) /  
        max(<attribute name>) /  
        min(<attribute name>) /  
        sum(<attribute name>)}  
FROM <table name>  
[WHERE ...];
```

Erzeugen einer synthetischen Tabelle „VIEW“

```
CREATE [OR REPLACE] VIEW <view name>  
AS SELECT ... FROM ... [WHERE ...] [...];
```

Löschen einer synthetischen Tabelle „VIEW“

```
DROP VIEW <view name>;
```

Zeichenschlüssel

GROSS GESCHRIEBEN:	SQL – Standardsyntax
klein geschrieben:	Funktionsnamen
<i>klein u. kursiv geschrieben:</i>	benutzerdefinierte Elemente
<...>:	Anfang bzw. Ende eines benutzerdefinierten Elementes
<table name>:	Hier ist ein Tabellenname einzusetzen
<view name>:	Hier ist ein Viewname einzusetzen
<attribute name>:	Hier ist ein Attributname einzusetzen
<value>:	Hier ist ein Wert einzusetzen
<file path and file name>:	Hier ist der Pfad und der Name einer Datei anzugeben (Bsp.: C:/data/messung.txt)
<delimiter>:	Hier ist das Trennzeichen einzusetzen, das Attributwerte in einer Textdatei voneinander trennt. (Bsp.: , ;)
<condition>:	Hier ist eine Bedingung einzusetzen
[...]:	Optionaler Teil
{... ...}:	Eines der angeführten Elemente ist zu verwenden
{... / ...}:	Eines oder mehrere Elemente sind zu verwenden

Wichtige Geometrietypen mit WKT-Beispiel

```
'POINT(<x> <y>)'
Bsp.: 'POINT(10 3.5)'
'LINESTRING(<x> <y>, <x> <y>, ...)'
Bsp.: 'LINESTRING(0 0, 10 10)'
'POLYGON((<interior ring>) [(<exterior ring>)])'
Bsp.: 'POLYGON((0 0, 10 0, 10, 0 10, 0)(3 3, 5 3, 5, 3))'
```

Wichtige Spatial Reference System Identifiers (SRID)

4326: geogr. Koordinaten WGS 84 [$^{\circ}$]
 32633: UTM 33 N basierend auf WGS 84 [m]

Erweitern einer bestehenden Tabelle um eine Geometriespalte

```
SELECT addgeometrycolumn( '<table name>',
                           '<geometry attribute name>',
                           <SRID>,
                           '<geometry type>',
                           <dimension>);
```

Löschen einer Geometriespalte

```
SELECT dropgeometrycolumn( '<schema name>',
                            '<table name>',
                            '<geometry attribute name>');
```

Hinzufügen eines Datensatzes zu einer Tabelle mit Geometrieinformation

```
INSERT INTO <table name>
(<geometry attribute name>, ...)
VALUES (geomfromtext(<WKT>, <SRID>), ...);
```

Ausgabe von Geometrieinformation im WKT-Format

```
SELECT astext(<geometry attribute name>
FROM <table name>
[WHERE ...];
```

Anwenden von Geometriefunktionen

```
SELECT <geometry function name>
FROM <table name>
[WHERE ...];
```

```
SELECT <attribute name>
FROM <table name>
WHERE <geometry function> [<logical operator> <value>];
```

Wichtige Geometriefunktionen

```
astext(<geometry attribute name>)
geometryfromtext(<WKT>, <SRID>)
st_area(<geometry attribute name>)
st_buffer(<geometry attribute name>, <distance>)
st_centroid(<geometry attribute name>)
st_contains(<geometry attribute name>, <geometry attribute name>)
st_distance(<geometry attribute name>, <geometry attribute name>)
st_envelope(<geometry attribute name>)
st_length(<geometry attribute name>)
st_transform(<geometry attribute name>, <target SRID>)
st_x(<geometry attribute name>)
st_y(<geometry attribute name>)
```

Zeichenschlüssel

GROSS GESCHRIEBEN:	SQL – Standardsyntax
klein geschrieben:	Funktionsnamen
<i>klein u. kursiv geschrieben:</i>	benutzerdefinierte Elemente
<...>:	Anfang bzw. Ende eines benutzerdefinierten Elementes
<schema name>:	Hier ist ein Schemanname einzusetzen (meist: <i>public</i>)
<table name>:	Hier ist ein Tabellenname einzusetzen
<attribute name>:	Hier ist ein Attributname einzusetzen
<geometry attribute name>:	Hier ist ein Name des Geometriearributes einzusetzen
<value>:	Hier ist ein Wert einzusetzen
<logical operator>:	Hier ist ein logischer Operator einzusetzen (z.B.: =, <>, ...),
<WKT>:	Hier ist eine Geometrie im WKT-Format einzugeben
<SRID>:	Hier ist ein Spatial Reference System Identifier einzugeben
<target SRID>:	Hier ist der SRID des Zielkoordinatensystems einzugeben
<distance>:	Hier ist eine Länge einzugeben (Achtung: Einheit)
<x>:	Hier ist ein Rechtswert einzugeben
<y>:	Hier ist ein Hochwert einzugeben
<exterior ring>:	Hier sind Koordinaten einzugeben, die den äußeren Rand des Polygons bilden. (Achtung: erstes Koordinatenpaar muss mit letztem Koordinatenpaar übereinstimmen)
<interior ring>:	Hier sind Koordinaten einzugeben, die eine Aussparung innerhalb des Polygons bilden. (Achtung: erstes Koordinatenpaar muss mit letztem Koordinatenpaar übereinstimmen)
[...]:	Optionaler Teil
{... ...}:	Eines der angeführten Elemente ist zu verwenden
{... / ...}:	Eines oder mehrere Elemente sind zu verwenden

Stand: 22.09.2009

FTP - Server

FileZilla 0.9.27

Host: 129.27.89.66
Port: 21
Benutzername: student
Passwort: student
Verzeichnis: F:/ms4w/Apache/htdocs/student/lv_{Kürzel}/
sichtbar: /lv_{LV Kürzel}/

Web - Server

Apache 2.2.9

Host: 129.27.89.66
Webspace (lokal): F:/ms4w/Apache/htdocs/student/lv_{LV Kürzel}/
Aufruf: http://129.27.89.66/student/lv_{LV Kürzel}/

DB - Server

PostGIS 1.3.3

Host: 129.27.89.66
Port: 5432
Benutzername: student
Passwort: student
Lehrdatenbanken: lv_{LV Kürzel}

WMS - Server

UMN Mapserver 5.2.0

Aufruf: http://129.27.89.66/cgi-bin/mapserv
GetCapabilites: http://129.27.89.66/cgi-bin/mapserv?
MAP=F:/ms4w/Apache/htdocs/student/lv_{LV Kürzel}/.../{Mapfile}.map&
SERVICE=WMS&
VERSION=1.0.0&
REQUEST=GetCapabilities

WFS (-T) - Server

Geoserver 1.7.0

Administration: http://129.27.89.66:8081/geoserver/
Benutzername: admin
Passwort: geoserver
Get Capabilities: http://129.27.89.66:8081/geoserver/wfs?
VERSION=1.0.0&
SERVICES=WFS&
REQUEST=GetCapabilities