

Geodatenbank

Sun Sen 孙森

Web Mapping (Wise2012-13)

Dozent: Oliver Roick

Geographisches Institut, Universität Heidelberg

❖ Inhalt

1. Datenbank – Allgemeines

- Aufbau und Konzepte von Datenbanksystemen
- Datenbankmodelle
 - Relationale DB & SQL
 - Objektorientierte DB
 - Objektrelationale DB
 - NoSQL & Dokumentorientierte DB

2. Geodatenbank

- Modellierung von Geodaten
- Standards
 - Simple Feature Modell
 - SQL/MM Spatial
- PostGIS

1. Datenbank-Allgemeines

- **Aufbau und Konzepte von Datenbanksystemen**
- **Datenbankmodelle**
 - Relationale DB & SQL
 - Objektorientierte DB
 - Objektrelationale DB
 - NoSQL & Dokumentorientierte DB

❖ Aufgaben von GIS

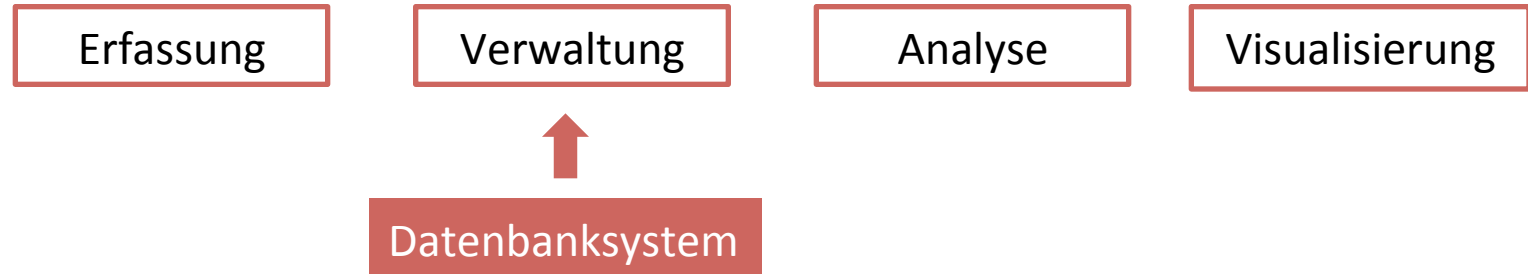
Erfassung

Verwaltung

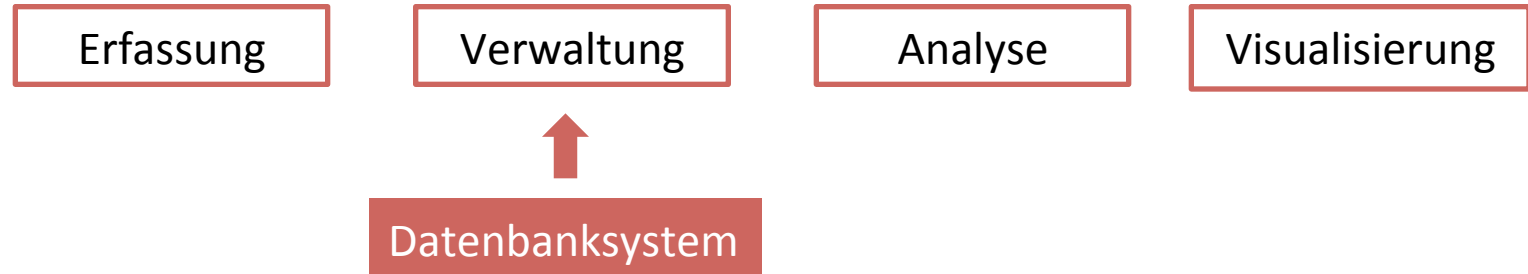
Analyse

Visualisierung

❖ Aufgaben von GIS

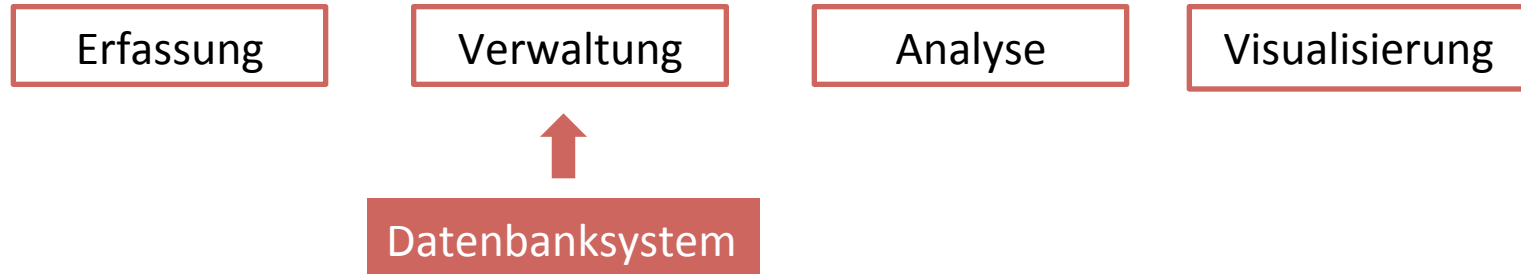


❖ Aufgaben von GIS

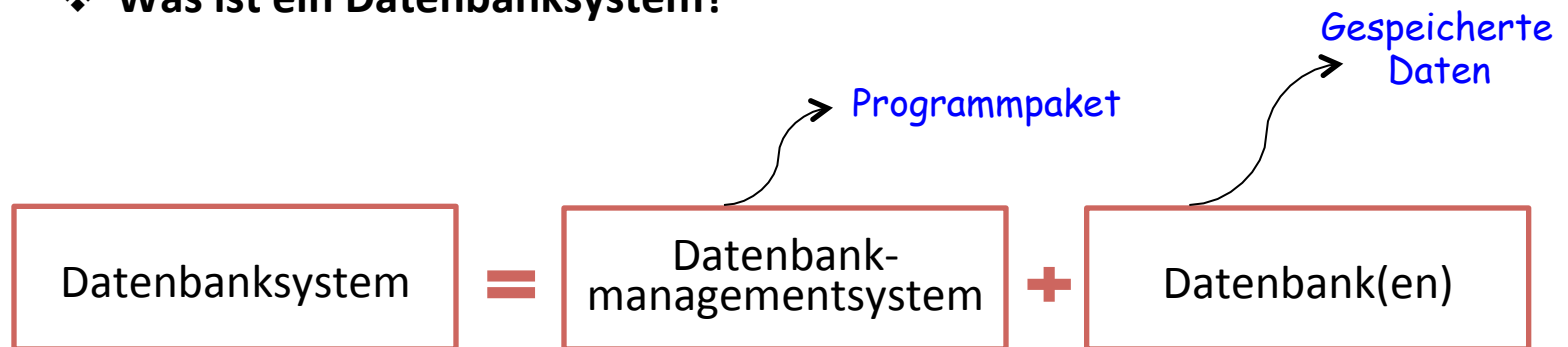


❖ Was ist ein Datenbanksystem?

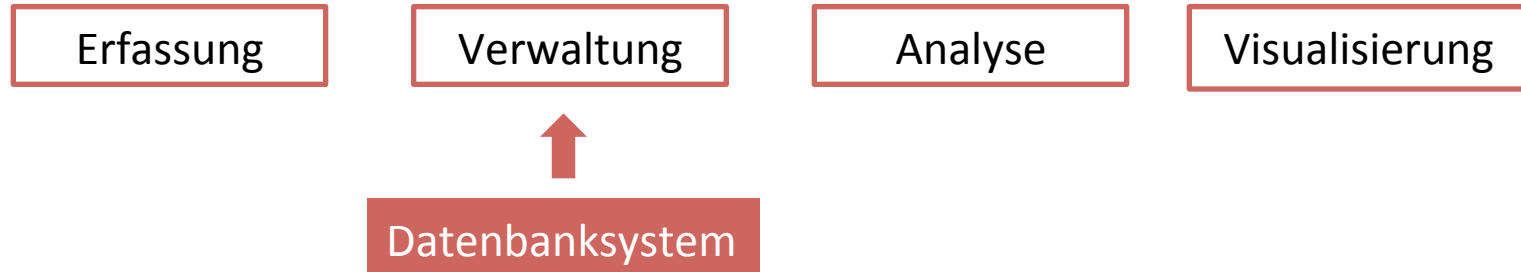
❖ Aufgaben von GIS



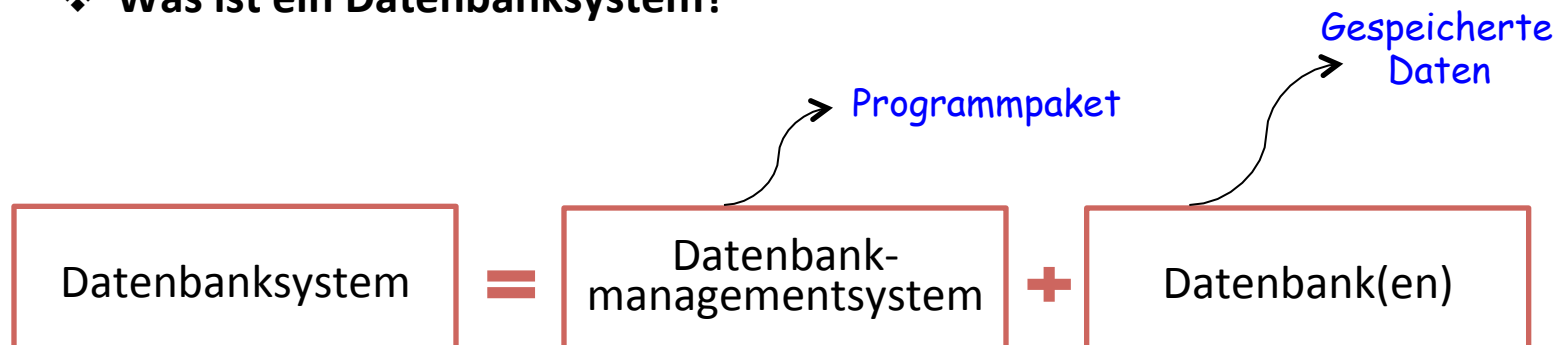
❖ Was ist ein Datenbanksystem?



❖ Aufgaben von GIS

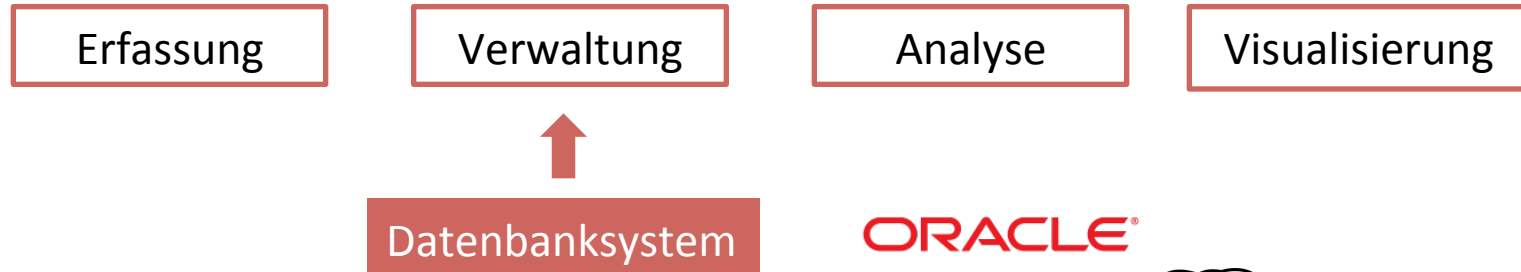


❖ Was ist ein Datenbanksystem?

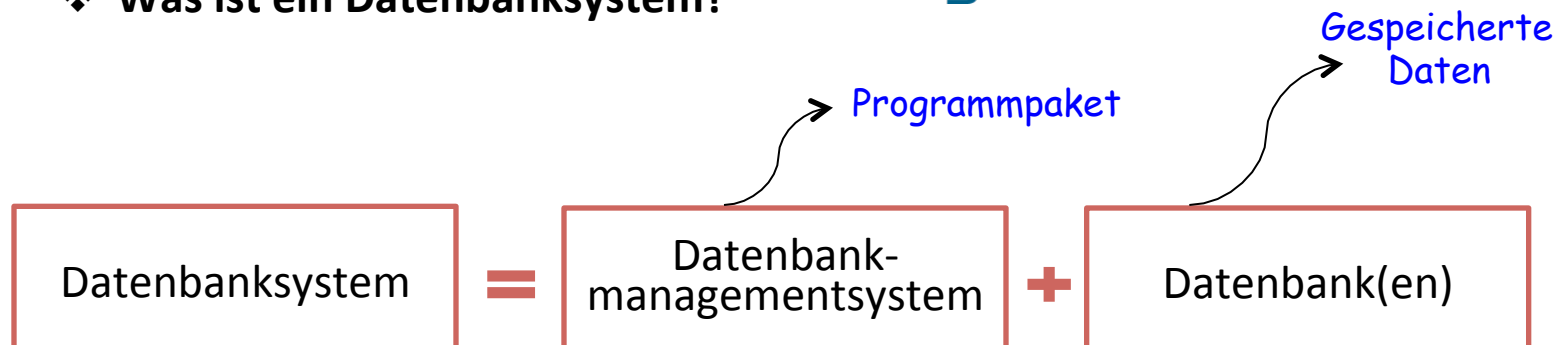


In einer Datenbank werden Daten **strukturiert** und **zentral** gespeichert. Jedes Datenbanksystem beruht auf einem **Datenbankmodell**. Zur Abfrage und Verwaltung von Daten bietet ein Datenbanksystem eine **Datenbanksprache** an.

❖ Aufgaben von GIS

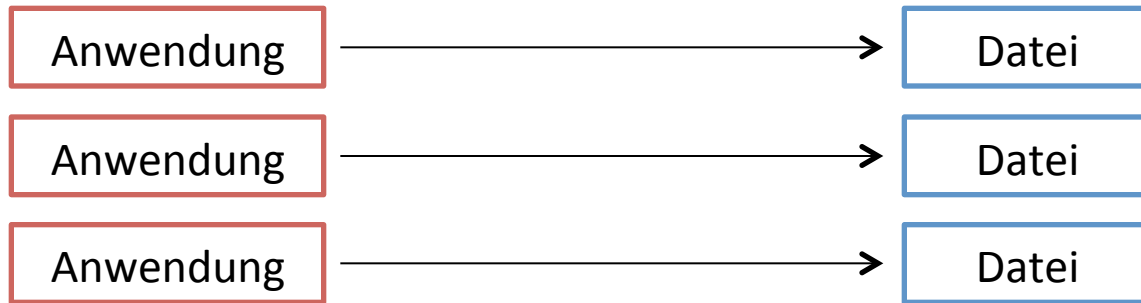


❖ Was ist ein Datenbanksystem?

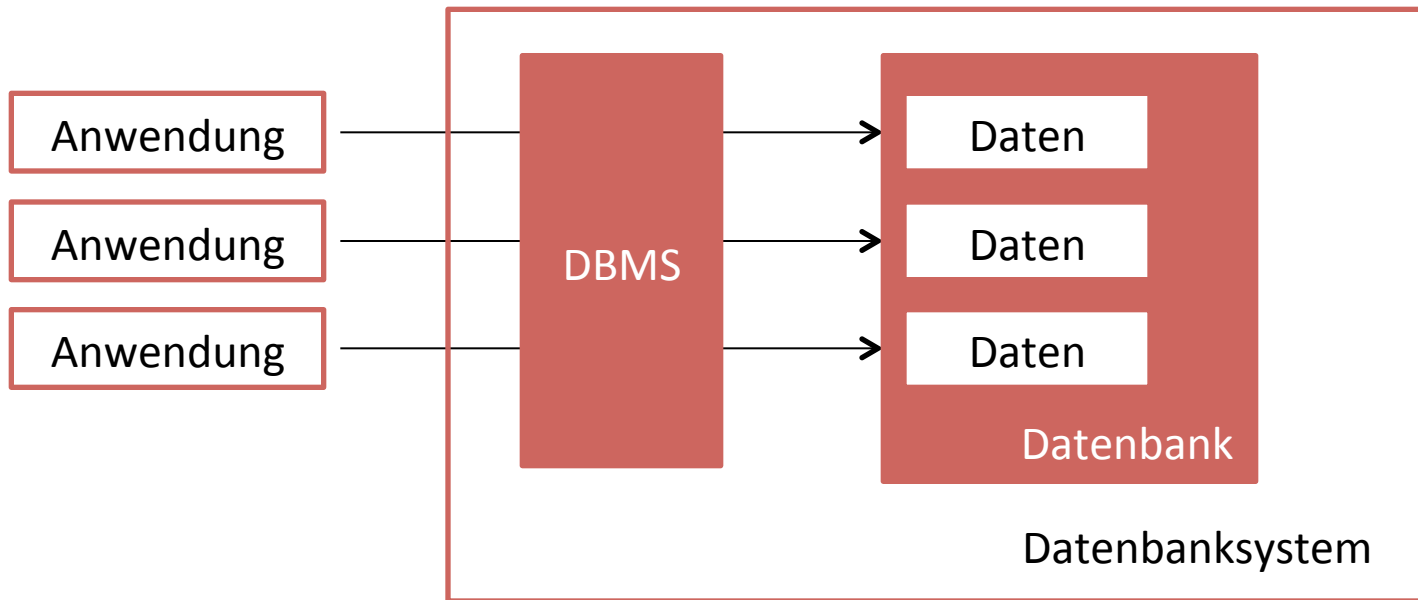


In einer Datenbank werden Daten **strukturiert** und **zentral** gespeichert. Jedes Datenbanksystem beruht auf einem **Datenbankmodell**. Zur Abfrage und Verwaltung von Daten bietet ein Datenbanksystem eine **Datenbanksprache** an.

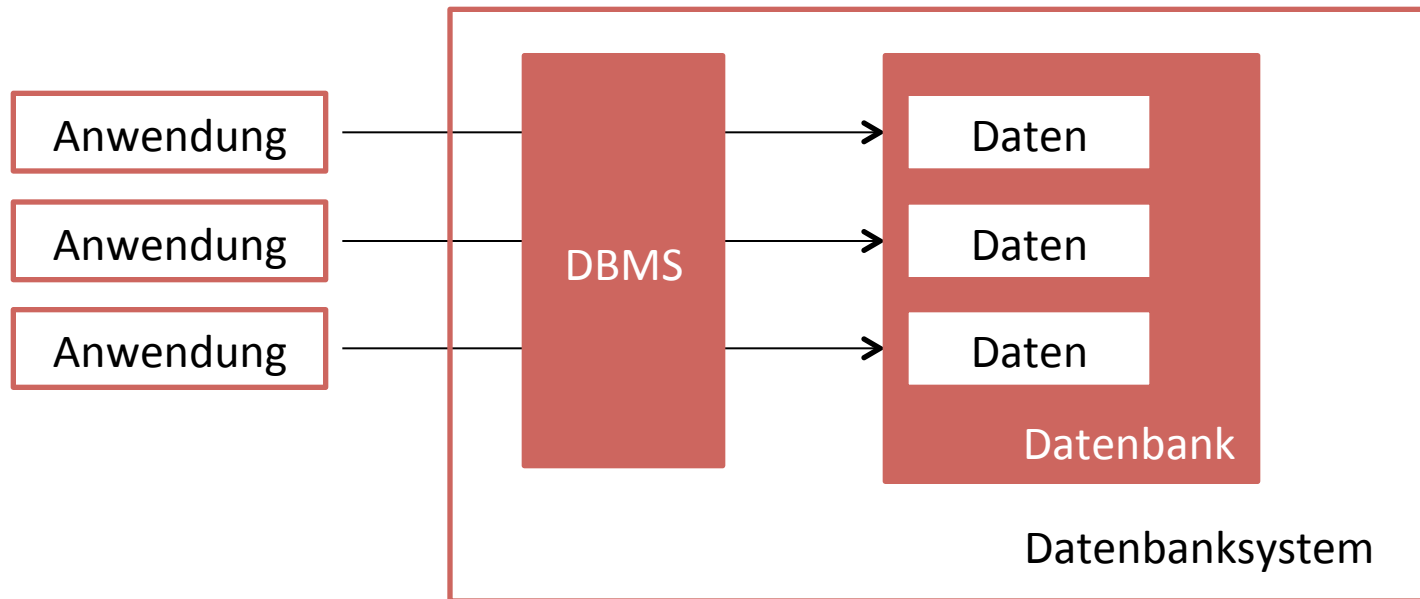
❖ Konzepte von einem DBS



❖ Konzepte von einem DBS



❖ Konzepte von einem DBS



In einem Datenbanksystem greifen die Anwendungen nicht mehr direkt auf die Daten zu, sondern durch eine neue „Schicht“ – das DBMS.

Diese neue Schicht hat folgende Ziele zu erreichen:

- **Datenunabhängigkeit**
- **Datenintegrität**
- **Mehrbenutzerbetrieb**
- **Effiziente Anfragebearbeitung**

❖ Konzepte von einem DBS

Datenunabhängigkeit

Die Änderungen an den Daten sollen möglichst wenig Einfluss auf die Anwendungen haben.

Datenintegrität

Korrektheit von den Daten, z.B.:

Geburtsdatum: 23.07.1983

Gebäude dürfen sich nicht überlappen

Mehrbenutzerbetrieb

1 Transaktionen - stabilen Datenzustand sichern
Transaktion: eine Folge von Anweisungen, die entweder alle oder überhaupt nicht durchgeführt werden.

z.B.: Geldüberweisung: Abbuchen + Hinzubuchen
parallele Zugriffe

2 Benutzerverwaltung

Verschiedene Lese- und Schreibrechte

z.B.: Dozenten und Studierende in Moodle

Effiziente Anfrage

Index: Signatur von Büchern in einer Bibliothek

❖ Datenbankmodell

Jedes Datenbanksystem beruht auf einem **Datenbankmodell**.

Ein Datenbankmodell ist ein System von Konzepten zur Definition und Evolution von Datenbanken. (Can Türker, Gunter Saake, 2006)

Die Evolution von Datenbankmodellen hängt von den mathematischen Theorien und Programmiersprachen ab.

Im Laufe der Zeit wurden **viele** Datenbankmodelle entwickelt.

4 wichtige Datenbankmodelle:

- **Relationales** Datenbankmodell – relationale Datenbanken
- **Objektorientiertes** Datenbankmodell – objektorientierte Datenbanken
- **Objektrelationales** Datenbankmodell – objektrelationale Datenbanken
- **Dokumentorientiertes** Datenbankmodell – dokumentorientierte Datenbanken

❖ Datenbankmodell – Relationale Datenbanken

In einer relationalen Datenbank werden die Daten in **Tabellen** gespeichert.

Matrikelnummer	Name
101	Max
102	Monika
103	Hans

❖ Datenbankmodell – Relationale Datenbanken

In einer relationalen Datenbank werden die Daten in **Tabellen** gespeichert.

Spalten
Attribute

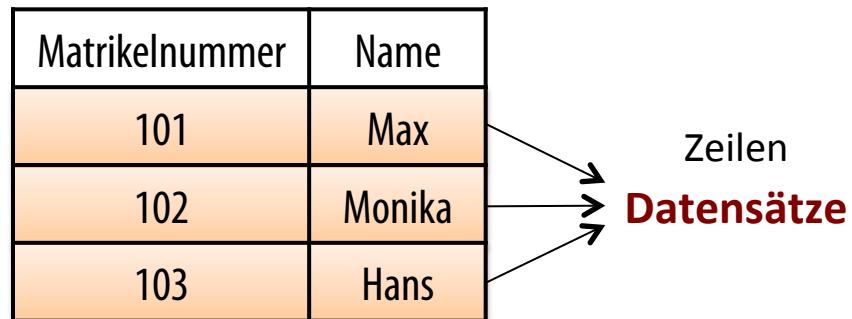
Matrikelnummer	Name
101	Max
102	Monika
103	Hans

❖ Datenbankmodell – Relationale Datenbanken

In einer relationalen Datenbank werden die Daten in **Tabellen** gespeichert.

Matrikelnummer	Name
101	Max
102	Monika
103	Hans

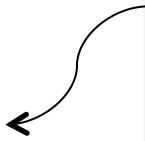
Zeilen
Datensätze



❖ Datenbankmodell – Relationale Datenbanken

In einer relationalen Datenbank werden die Daten in **Tabellen** gespeichert.

Primärschlüssel



Matrikelnummer	Name
101	Max
102	Monika
103	Hans

- muss beim Anlegen der Tabelle werden *
 - Identifikation der Datensätze
 - einziges Attribut oder eine Kombination von mehrere Attributen
 - muss eindeutig sein
 - automatisch ein Index erzeugt *
- * von DBMS abhängig

❖ Datenbankmodell – Relationale Datenbanken

Mehrere Tabellen werden durch **Fremdschlüssel** miteinander verbunden.

Tabelle 1 Kurse

Titel	Dozent
Mathe	Alex
Deutsch	Kathrin

Tabelle 2 Studenten

Matrikelnummer	Name
101	Max
102	Monika
103	Hans

Tabelle 3 Teilnehmer von Kursen

Kurse	Teilnehmer
Mathe	101
Mathe	102
Mathe	103
Deutsch	102
Deutsch	103

❖ Datenbankmodell – Relationale Datenbanken

Mehrere Tabellen werden durch **Fremdschlüssel** miteinander verbunden.

Tabelle 1 Kurse

Titel	Dozent
Mathe	Alex
Deutsch	Kathrin

Tabelle 2 Studenten

Matrikelnummer	Name
101	Max
102	Monika
103	Hans

Fremdschlüssel

Tabelle 3 Teilnehmer von Kursen

Kurse	Teilnehmer
Mathe	101
Mathe	102
Mathe	103
Deutsch	102
Deutsch	103

- in Tabelle 3 durch SQL definiert
- muss auf einen Primärschlüssel verweisen

❖ Datenbankmodell – Relationale Datenbanken

Der Attributwert in einer relationalen Datenbank muss **atomar** sein.

atomar: als eine „Einheit“ behandelt.

atomare Datentypen: Zeichenketten, Zahlen, Zeitangabe...

✓

Kurse	Teilnehmer
Mathe	101
Mathe	102
Mathe	103
Deutsch	102
Deutsch	103

✗

Titel	Teilnehmer
Mathe	(101, 102, 103)
Deutsch	(102, 103)

❖ Datenbankmodell – Relationale Datenbanken

Die Datenbanksprache für relationale Datenbanken ist **SQL**.
SQL ist auch eine ISO-Norm.

1. Tabellen mit Attributen anlegen und Primärschlüssel definieren

```
CREATE TABLE Studenten (  
    Matrikelnummer DECIMAL(3),    -- 3-stellige Zahl  
    Name VARCHAR(20),  
    CONSTRAINT pk_stu PRIMARY KEY (Matrikelnummer)  
);  
COMMIT;
```

```
CREATE TABLE Kurse (  
    Titel VARCHAR(10),  
    Dozent    VARCHAR(10),  
    CONSTRAINT pk_kurs PRIMARY KEY (Titel)  
);
```

...

❖ Datenbankmodell – Relationale Datenbanken

Die Datenbanksprache für relationale Datenbanken ist **SQL**.
SQL ist auch eine ISO-Norm.

1. Tabellen mit Attributen anlegen und Primärschlüssel definieren

```
CREATE TABLE Studenten (  
    Matrikelnummer DECIMAL(3)  
    Name VARCHAR(20),  
    CONSTRAINT pk_stu PRIMARY  
);  
COMMIT;
```

Matrikelnummer	Name

```
CREATE TABLE Kurse (  
    Titel VARCHAR(10),  
    Dozent VARCHAR(10),  
    CONSTRAINT pk_kurs PRIMARY  
);
```

Titel	Dozent

...

❖ Datenbankmodell – Relationale Datenbanken

2. Datensätze einfügen

```
INSERT INTO Studenten (Matrikelnummer, Name) VALUES  
(101, 'Max');  
...
```

3. Tabelle mit Fremdschlüssel definieren

```
CREATE TABLE Kursteilnahme (  
    Kurs VARCHAR(10),  
    Teilnehmer DECIMAL(3),  
    CONSTRAINT pk_kurs_stu PRIMARY KEY (Kurs,  
    Teilnehmer),  
    CONSTRAINT fk_stu FOREIGN KEY (Teilnehmer)  
REFERENCES Studenten(Matrikelnummer) ON DELETE  
CASCADE,  
    CONSTRAINT fk_kurs FOREIGN KEY (Kurse)  
REFERENCES Kurse(Titel) ON DELETE CASCADE  
);
```

❖ Datenbankmodell – Relationale Datenbanken

2. Datensätze einfügen

```
INSERT INTO Studenten (Matrikelnummer, Name)  
(101, 'Max');  
...
```

Matrikelnummer	Name
101	Max

ES

3. Tabelle mit Fremdschlüssel definieren

```
CREATE TABLE Kursteilnahme (  
    Kurs VARCHAR(10),  
    Teilnehmer DECIMAL(3),  
    CONSTRAINT pk_kurs_stu PRIMARY KEY (Kurs,  
    Teilnehmer),  
    CONSTRAINT fk_stu FOREIGN KEY (Teilnehmer)  
REFERENCES Studenten(Matrikelnummer) ON DELETE  
CASCADE,  
    CONSTRAINT fk_kurs FOREIGN KEY (Kurse)  
REFERENCES Kurse(Titel) ON DELETE CASCADE  
);
```

❖ Datenbankmodell – Relationale Datenbanken

4. Anfrage

```
SELECT [Attribute] FROM [Tabelle] WHERE [Bedingung];
```

❖ Datenbankmodell – Relationale Datenbanken

4. Anfrage

SELECT [Attribute] **FROM** [Tabelle] **WHERE** [Bedingung];

```
SELECT Name FROM Studenten;
```

```
SELECT * FROM Studenten;
```

```
SELECT Name FROM Studenten WHERE  
Matrikelnummer=101;
```

```
SELECT char_length(Name) FROM Studenten WHERE  
Matrikelnummer=101;
```

```
SELECT avg(char_length(Name)) FROM Studenten;
```

Matrikelnummer	Name
101	Max
102	Monika
103	Hans

❖ Datenbankmodell – Objektorientierte Datenbanken

In einer objektorientierten Datenbank werden die Daten als **Objekte** gespeichert.

Ein Objekt hat nicht nur **Attribute**, sondern auch **Methoden**.

Objekte, die die gleichen Attribute und Methoden haben, bilden eine **Klasse**.

Klasse: Menschen

Attribute:

Name;
Geburtsdatum;

Methoden:

essen();
laufen();

❖ Datenbankmodell – Objektorientierte Datenbanken

Klassen sind in einer **Klassenhierarchie** angeordnet.

Unterklassen können die Attribute und Methoden von ihren **Oberklassen** vererben.

abstrakte Klassen: gemeinsame Methoden von Unterklassen bündeln

Oberklasse: Menschen

Attribute:

Name;
Geburtsdatum;

Methoden:

essen();
laufen();

Unterklasse: Kinder

Attribute:

Schule;

Methoden:

zurSchuleGehen();

Unterklasse: Erwachsene

Attribute:

Beruf;

Methoden:

arbeiten();

❖ Datenbankmodell – Objektorientierte Datenbanken

Klassen sind in einer **Klassenhierarchie** angeordnet.

Unterklassen können die Attribute und Methoden von ihren **Oberklassen** vererben.

abstrakte Klassen: gemeinsame Methoden von Unterklassen bündeln

Oberklasse: Menschen

Attribute:

Name;
Geburtsdatum;

Methoden:

essen();
laufen();

nicht verbreitet,
Risiko & Kosten

Unterklasse: Kinder

Attribute:

Schule;

Methoden:

zurSchuleGehen();

Unterklasse: Erwachsene

Attribute:

Beruf;

Methoden:

arbeiten();

❖ Datenbankmodell –objektrelationale Datenbanken

relationale Datenbanken + Objektorientierung = objektrelationale Datenbanken

Vorteile von Relationalen Datenbanken und Objektorientierung kombinieren:

- SQL, Integritätsbedingung, effiziente Anfrage
- strukturierte Datentypen, objektspezifische Methoden

❖ Datenbankmodell –objektrelationale Datenbanken

relationale Datenbanken + Objektorientierung = objektrelationale Datenbanken

Vorteile von Relationalen Datenbanken und Objektorientierung kombinieren:

- SQL, Integritätsbedingung, effiziente Anfrage
- strukturierte Datentypen, objektspezifische Methoden

für Geodaten
wichtig

❖ Datenbankmodell –objektrelationale Datenbanken

relationale Datenbanken + Objektorientierung = objektrelationale Datenbanken

Vorteile von Relationalen Datenbanken und Objektorientierung kombinieren:

- SQL, Integritätsbedingung, effiziente Anfrage
- strukturierte Datentypen, objektspezifische Methoden

für Geodaten
wichtig

mächtig
aber nicht allmächtig

```
{  
Name: Max  
Geschlecht: männlich  
}
```

```
{  
Name: Monika  
PLZ: 69126  
}
```

```
{  
Name: Kathrin  
Email: kathrin@yahoo.de  
}
```

```
{  
Name: Max  
Geschlecht: männlich  
}
```

```
{  
Name: Monika  
PLZ: 69126  
}
```

```
{  
Name: Kathrin  
Email: kathrin@yahoo.de  
}
```

relationale Datenbank:

Name	Geschlecht	PLZ	Email
Max	männlich		
Monika		69126	
Kathrin			kathrin@yahoo.de

```
{  
Name: Max  
Geschlecht: männlich  
}
```

```
{  
Name: Monika  
PLZ: 69126  
}
```

```
{  
Name: Kathrin  
Email: kathrin@yahoo.de  
}
```

relationale Datenbank:

Name	Geschlecht	PLZ	Email
Max	männlich		
Monika		69126	
Kathrin			kathrin@yahoo.de

Speicherplatz
verschwenden

❖ Datenbankmodell –NoSQL Datenbanken

Dachbegriff: **NoSQL**

Speicherung und Verwaltung riesiger Datenmenge mit „lockerer“ Struktur

```
{  
Name: Max  
Geschlecht: männlich  
}
```

```
{  
Name: Monika  
PLZ: 69126  
}
```

```
{  
Name: Kathrin  
Email: kathrin@yahoo.de  
}
```

relationale Datenbank:

Name	Geschlecht	PLZ	Email
Max	männlich		
Monika		69126	
Kathrin			kathrin@yahoo.de

Speicherplatz
verschwendet

❖ Datenbankmodell –NoSQL Datenbanken

Dachbegriff: **NoSQL**

Speicherung und Verwaltung riesiger Datenmenge mit „lockerer“ Struktur

```
{  
Name: Max  
Geschlecht: männlich  
}
```

```
{  
Name: Monika  
PLZ: 69126  
}
```

```
{  
Name: Kathrin  
Email: kathrin@yahoo.de  
}
```

relationale Datenbank:

Name	Geschlecht	PLZ	Email
Max	männlich		
Monika		69126	
Kathrin			kathrin@yahoo.de

Speicherplatz
verschwendet

Beispiele: Google Big Table Amazon Dynamo



❖ Datenbankmodell –dokumentorientierte Datenbanken

eine Art NoSQL-Datenbanke

Dokumente : strukturierte Dateien mit einem Standard-Dateiformat, nicht weiter strukturiert im Sinne eines Datenbankzugriffs (z.B. Binary Large Objects, JSON-Objekte, XML-Dokumente)

Quelle: (Wikipedia, dokumentorientierte Datenbank)

dokumentorientierte DBMS: CouchDB, MongoDB



Beispiel: foursquare (MongoDB)



❖ Datenbankmodell – ein Vergleich

a. riesige Datenmenge ohne
Attributgleichheit

relationale DB

b. komplexe Datenstruktur,
datentypspezifische
Methoden

objektrelation
ale DB

c. einfache und klare
Datenstruktur,
effiziente Anfrage
erforderlich

NoSQL DB

❖ Datenbankmodell – ein Vergleich

a. riesige Datenmenge ohne
Attributgleichheit

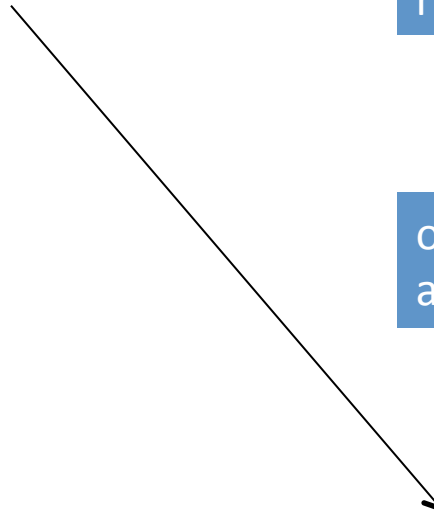
relationale DB

b. komplexe Datenstruktur,
datentypspezifische
Methoden

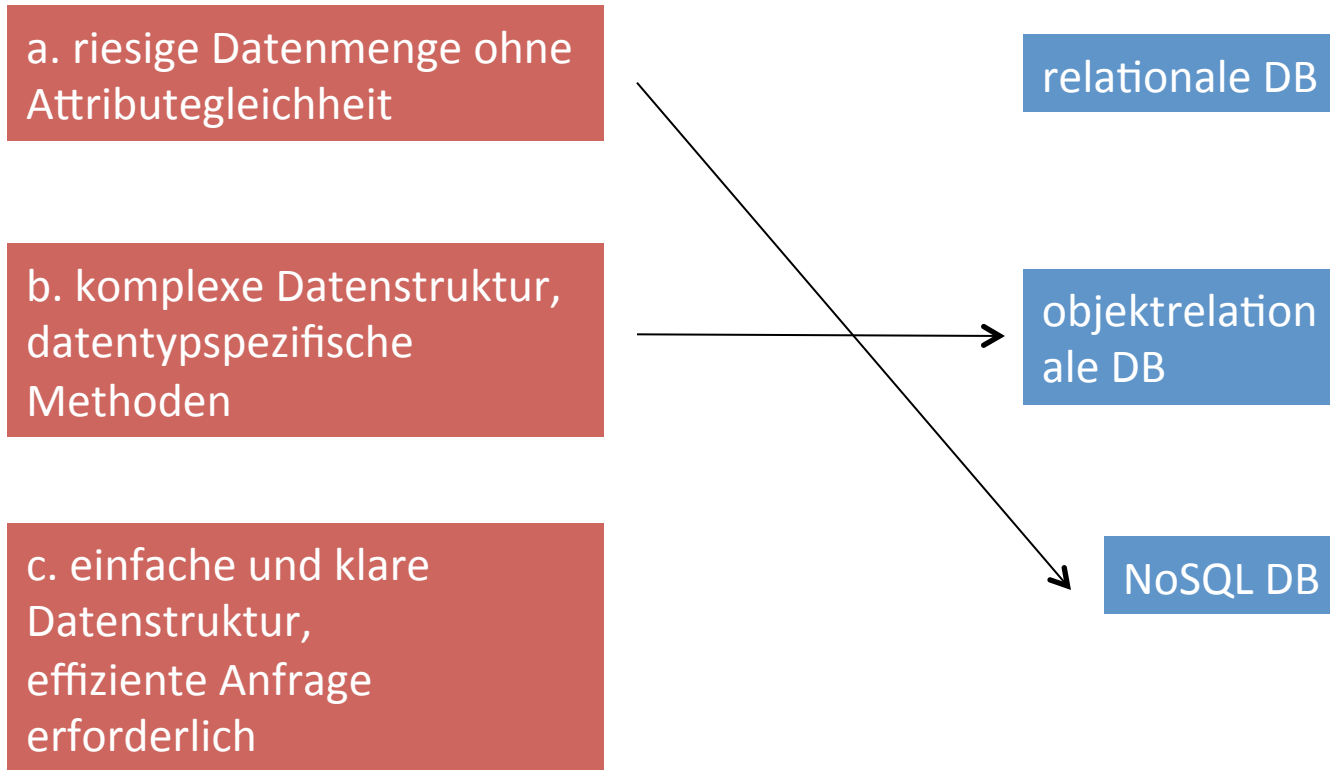
objektrelationale DB

c. einfache und klare
Datenstruktur,
effiziente Anfrage
erforderlich

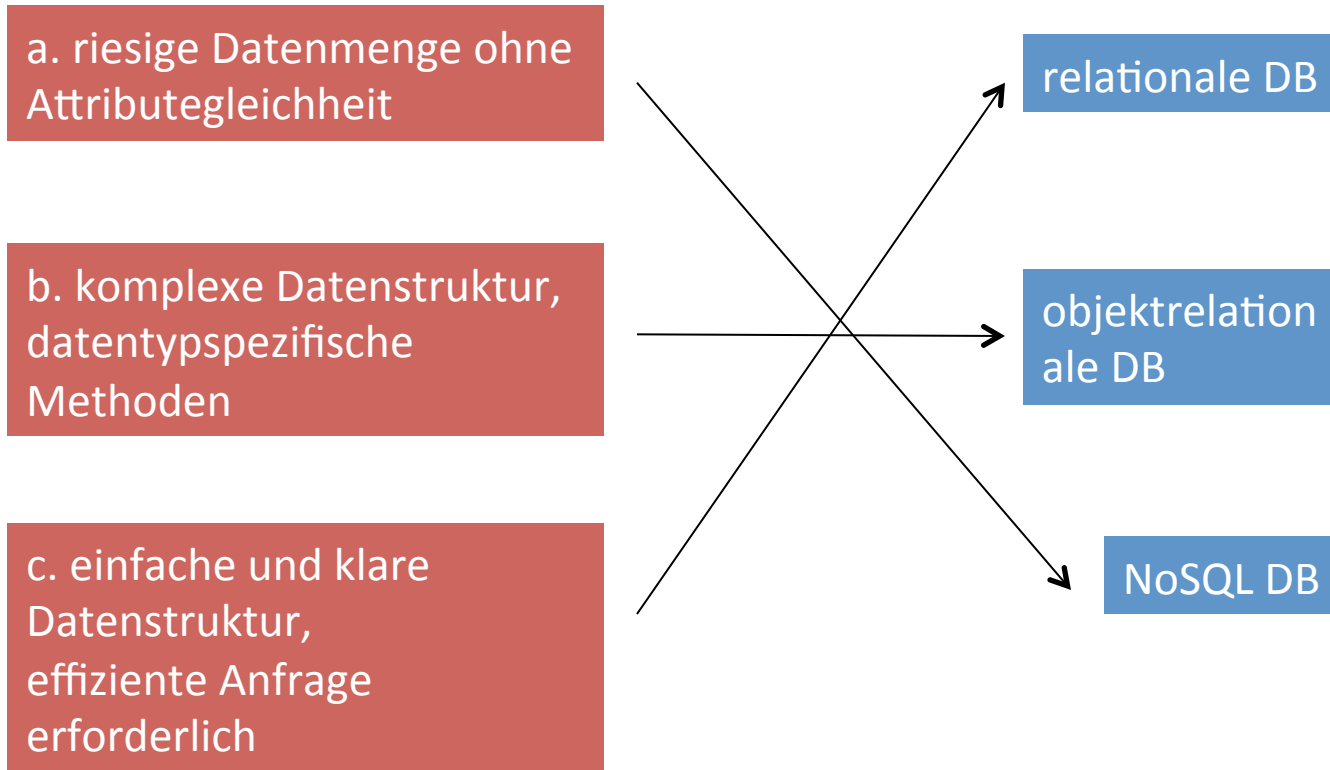
NoSQL DB



❖ Datenbankmodell – ein Vergleich



❖ Datenbankmodell – ein Vergleich



2. Geodatenbank

- **Modellierung von Geodaten**
- **Standards**
 - Simple Feature Modell
 - SQL/MM Spatial
- **PostGIS**

❖ Modellierung von Geodaten

Für die Speicherung und Verwaltung von Geodaten ist ein **Datenmodell** erforderlich.

Ein Datenmodell ist eine vereinfachte Abbildung von der realen Welt.

z.B: **Telefonzellen, Bushaltestellen -> Punkte;**
Straßen -> Zentrallinien

Dimension:

- **2D-Modell**
- 3D-Modell
- spatio-temporales Modell

Eigenschaften von Geodaten:

- Sachattribute
- Geometrie (Lage und Ausdehnung)
- Topologie
- Metainformationen

❖ Modellierung von Geodaten

Für die Speicherung und Verwaltung von Geodaten ist ein **Datenmodell** erforderlich.

Ein Datenmodell ist eine vereinfachte Abbildung von der realen Welt.

z.B: **Telefonzellen, Bushaltestellen -> Punkte;**
Straßen -> Zentrallinien

Dimension:

- **2D-Modell**
- 3D-Modell
- spatio-temporales Modell

Eigenschaften von Geodaten:

- Sachattribute
- Geometrie (Lage und Ausdehnung)
- Topologie
- Metainformationen

alphanumerisch

❖ Modellierung von Geodaten

Für die Speicherung und Verwaltung von Geodaten ist ein **Datenmodell** erforderlich.

Ein Datenmodell ist eine vereinfachte Abbildung von der realen Welt.

z.B: **Telefonzellen, Bushaltestellen -> Punkte;**
Straßen -> Zentrallinien

Dimension:

- **2D-Modell**
- 3D-Modell
- spatio-temporales Modell

Eigenschaften von Geodaten:

- Sachattribute → **alphanumerisch**
- Geometrie (Lage und Ausdehnung) → **direkt speichern & ableiten**
- Topologie → **direkt speichern & ableiten**
- Metainformationen → **direkt speichern & ableiten**

❖ Modellierung von Geodaten

Für die Speicherung und Verwaltung von Geodaten ist ein **Datenmodell** erforderlich.

Ein Datenmodell ist eine vereinfachte Abbildung von der realen Welt.

z.B: **Telefonzellen, Bushaltestellen -> Punkte;**
Straßen -> Zentrallinien

Dimension:

- **2D-Modell**
- 3D-Modell
- spatio-temporales Modell

Eigenschaften von Geodaten:

- Sachattribute
- Geometrie (Lage und Ausdehnung)
- Topologie
- Metainformationen

alphanumerisch

direkt speichern & ableiten

Raster & **Vektor**

❖ Speicherung von Geometrie - Standards

• Simple-Feature-Modell

- ISO-Norm 19125-1: Simple Feature Access – Common Architecture
- ISO-Norm 19125-2: Simple Feature Access – SQL Option

• SQL/MM Spatial

Hauptunterschied:

SQL/MM Spatial unterstützt Kreisbögen, während Simple-Feature-Modell nur gerade Linien erlaubt.

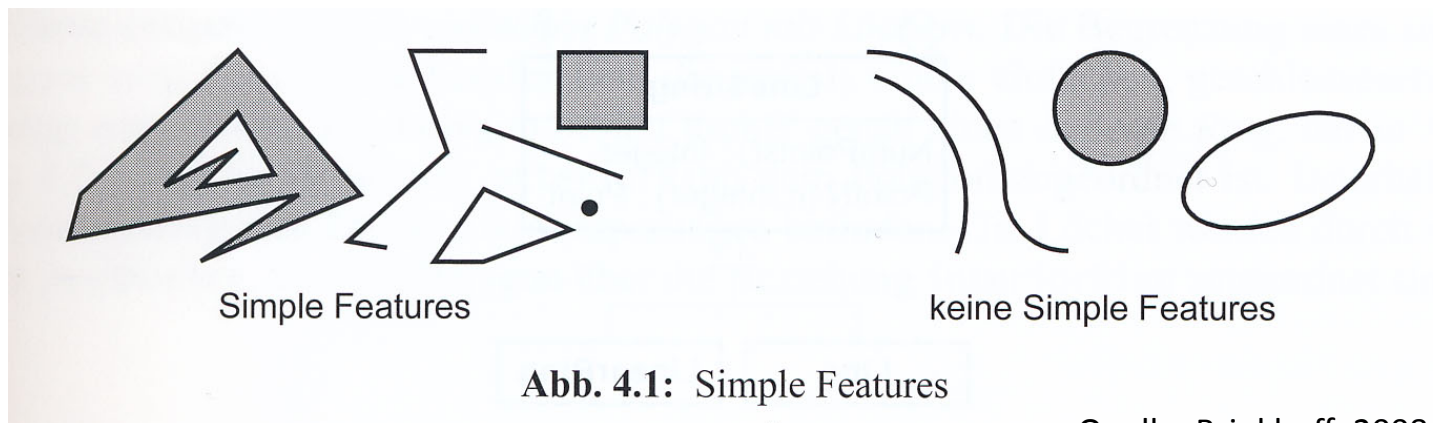


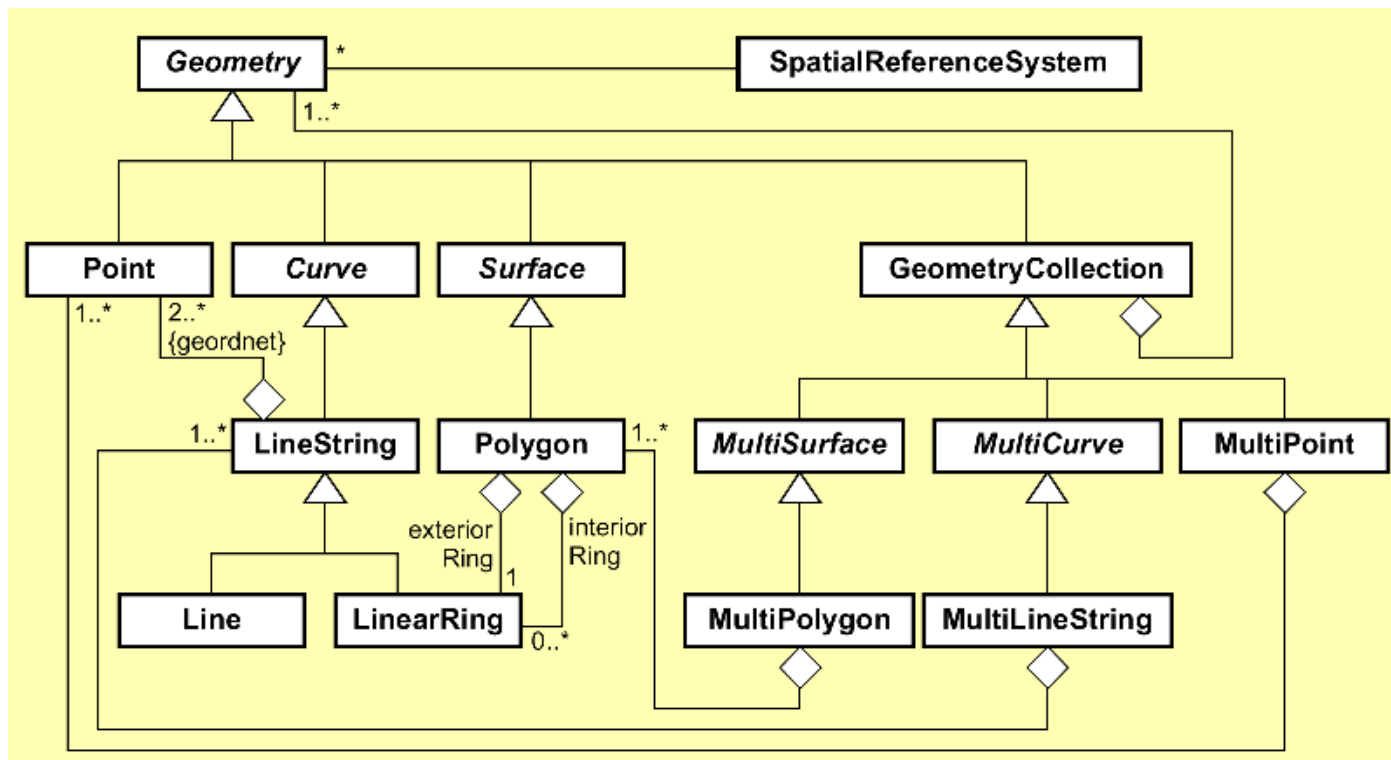
Abb. 4.1: Simple Features

Quelle: Brinkhoff, 2008

❖ Simple-Feature-Modell

1. geometrische Klassenhierarchie:

- allgemeine abstrakte Oberklasse: Geometry (verweist auf ein räumliches Bezugssystem)
- geometrische Primitive: Punkte, Linien und Flächen
- abstrakte Klassen: Curve, Surface (für Erweiterungen vorgesehen?)



❖ Simple-Feature-Modell

2. Raprensation von Geometrie:

- **WKT** (Well Known Text)
- **WBT** (Well Known Binary)

Point(10 10)

LineString(10 10, 20 20, 30 40)

Polygon((10 10, 10 20, 20 20, 20 15, 10 10))

Polygon((0 0, 0 20, 20 20, 20 0, 0 0),(5 5, 5 15, 15 15, 15 5, 5 5))

MultiPolygon(((10 10, 10 20, 20 20, 20 15, 10 10)),((60 60, 70 70, 80 60, 60 60)))

❖ Simple-Feature-Modell

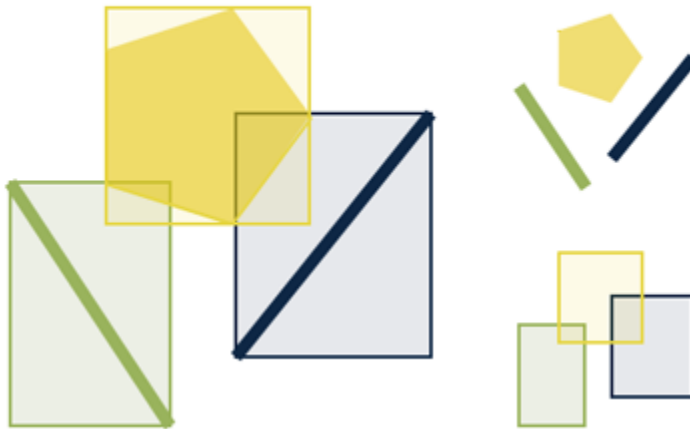
3. Methoden von Geometrie:

Das Simple Feature Model beinhaltet u.a. **Methoden** zur:

- Prüfung **topologischer Beziehungen**
- **Approximation** von Geometrien
- Berechnung **geometrischer Eigenschaften** (z.B. Länge, Abstand)
- **Verschneidung** von Geometrien

Quelle: FerGI, Geodatenbanksysteme

Bounding Boxes



Quelle: http://workshops.opengeo.org/postgis-intro/_images/boundingbox.png

❖ PostGIS – ein standardkonformes räumliches DBMS



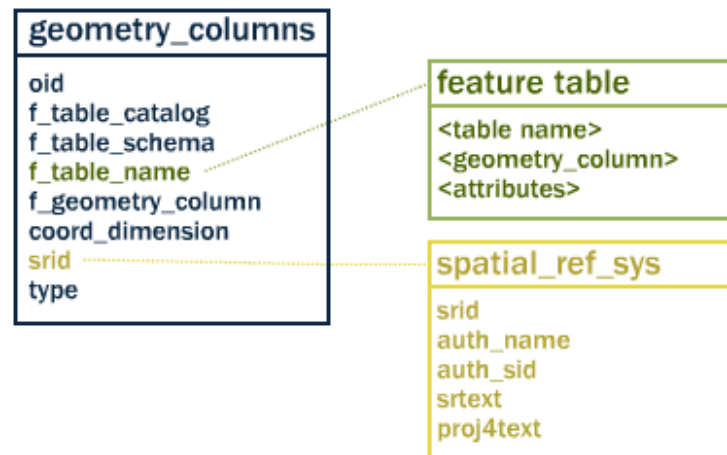
PostgreSQL um räumliche Erweiterungen

- Räumliche Bezugssysteme
- Geometrische Datentypen
- Geometrische Methoden

ein oft verwendetes open source räumliches DBMS: z.B. **OpenStreetMap**

zwei wichtige Tabellen: **geometry_columns**, **spatial_ref_sys**

Ein Register für die
Tabellen, die
geometrische Datentypen
enthalten



Quelle: http://workshops.opengeo.org/postgis-intro/_images/table01.png

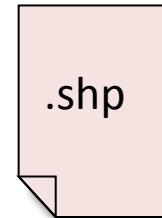
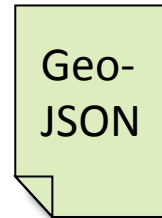
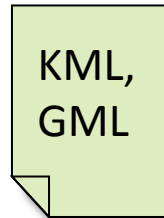
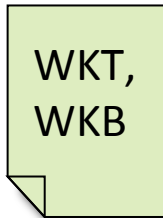
über 3000 räumliche
Bezugssysteme

❖ PostGIS – Input und Output



Input:

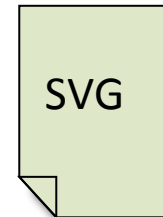
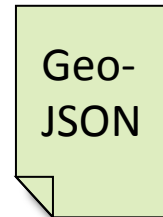
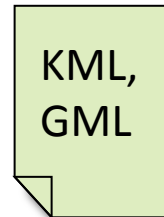
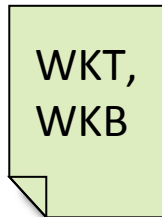
```
ST_GeomFromText (WKT, SRID) ;  
ST_GeomFromKML (KML) ;
```



durch Plug-in




Output:

```
ST_AsText (geometry) ;
```



❖ PostGIS – geometrische Datentypen und Methoden



 <p>Point</p>	<ul style="list-style-type: none">• <code>ST_X(geometry)</code>• <code>ST_Y(geometry)</code>
 <p>Simple non-closed linestring</p>	<ul style="list-style-type: none">• <code>ST_Length(geometry)</code>• <code>ST_StartPoint(geometry)</code>• <code>ST_EndPoint(geometry)</code>• <code>ST_NPoints(geometry)</code>
 <p>Polygon defined by exterior ring</p>	<ul style="list-style-type: none">• <code>ST_Area(geometry)</code>• <code>ST_NRings(geometry)</code>• <code>ST_ExteriorRing(geometry)</code>• <code>ST_InteriorRingN(geometry, n)</code>• <code>ST_Perimeter(geometry)</code>

❖ PostGIS – geometrische Datentypen und Methoden



Multipoint with 4 parts



Simple multilinestring defined by 4 endpoints of 2 elements



Multipolygon consisting of 2 elements defined by exterior rings and 3 interior rings

- `ST_NumGeometries(geometry)`
- `ST_GeometryN(geometry, n)`
- `ST_Area(geometry)`
- `ST_Length(geometry)`

❖ PostGIS – räumliche Beziehungen von Geometrien

z.B:

```
ST_Intersects(geometry A, geometry B);  
ST_Distance(geometry A, geometry B)
```



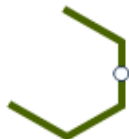
Intersects



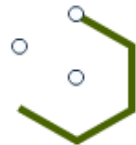
Point & Multipoint



Multipoint & Multipoint



Point & Linestring



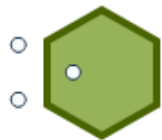
Multipoint & Linestring



Linestring & Linestring



Linestring & Polygon



Multipoint & Polygon

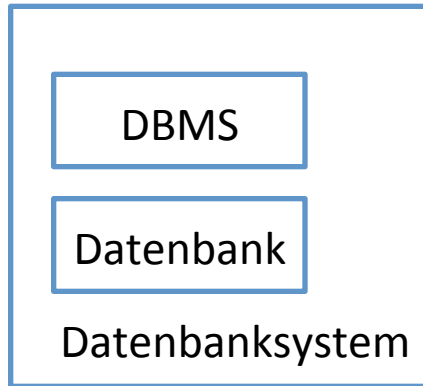


Linestring & Multipolygon

mehr in
[Dokumentation](#)
von PostGIS

❖ Zusammenfassung

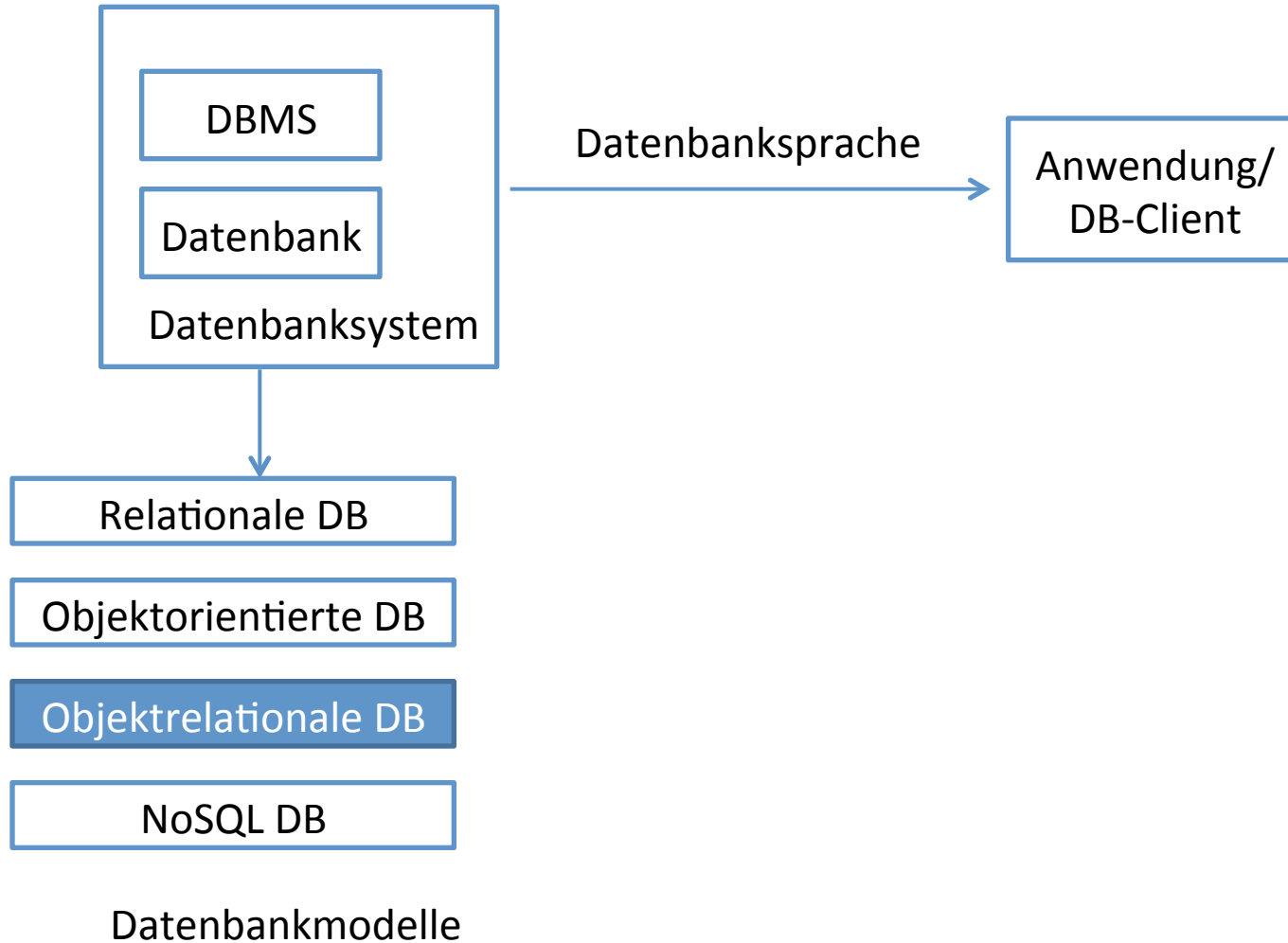
❖ Zusammenfassung



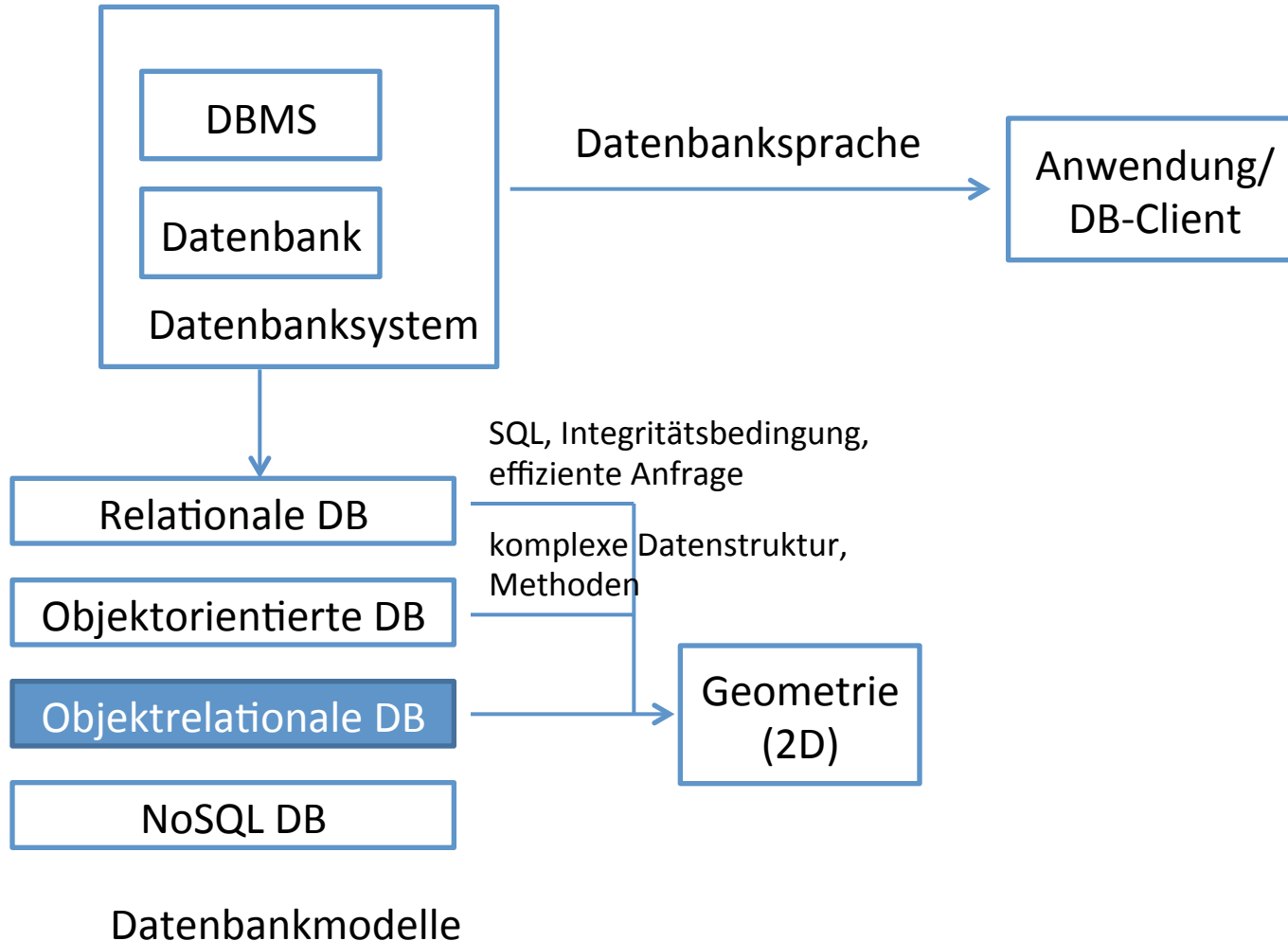
❖ Zusammenfassung



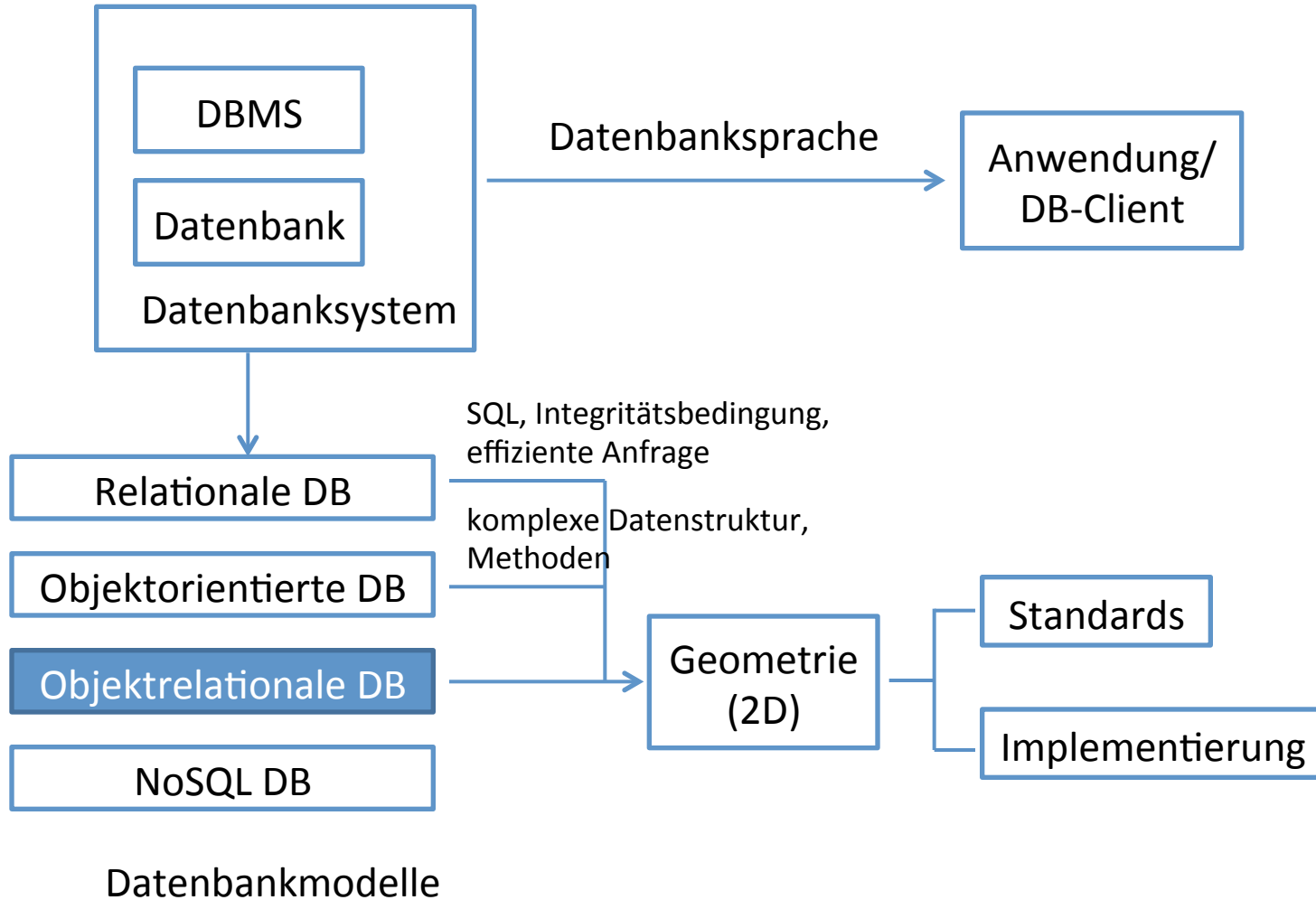
❖ Zusammenfassung



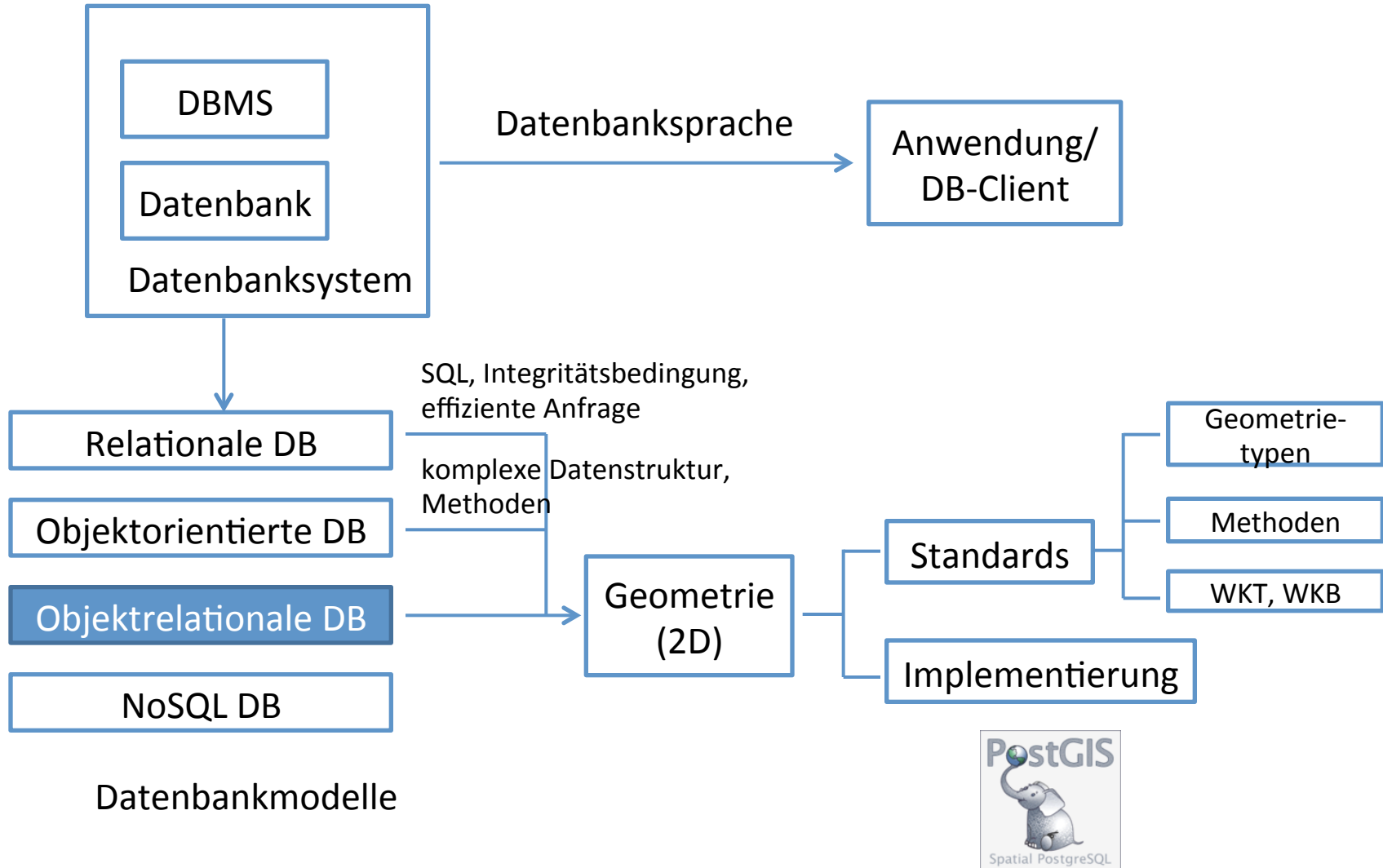
❖ Zusammenfassung



❖ Zusammenfassung



❖ Zusammenfassung



❖ Literatur

- Ressourcen im Internet

<http://www.fergi.uni-osnabrueck.de/moodle/>

<http://workshops.opengeo.org/postgis-intro/index.html>

<http://postgis.net/docs/manual-2.0/>

- Lehrbücher

**Brinkhoff T. (2008): Geodatenbanksysteme in Theorie und Praxis.
Wichmann Verlag.**

Türker C.; Saake G. (2006): Objektrelationale Datenbanken. dpunkt.Verlag.
Abraham Silberschatz ; Henry F. Korth ; S. Sudarshan. (2002): Database
system concepts. McGraw-Hill, Boston.

Danke für eure
Aufmerksamkeit.