Datenmodelle und Anfragesprachen

- Entity-Relationship-Modell
- Relationale Datenbanken
  - Relationenmodell
  - SQL
- Semistrukturierte Daten
  - Graphenbasierte Modelle und XML
  - XML-Anfragesprachen und XML-DOM
- Multimedia-Datenbanken
  - Anfragesprachen

Datenbankmodell

- Datenbankmodell :=
  - System von Konzepten zur Beschreibung von Datenbanken
  - legt Syntax und Semantik von Datenbankbeschreibungen fest
- Datenbankbeschreibung = Datenbankschema

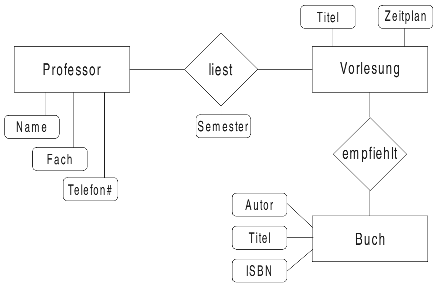
Datenbankmodell

- Datenbankmodell legt fest
  - statische Eigenschaften
    - Objekte
    - Beziehungen
    - inkl. Standard-Datentypen, die Daten über Beziehungen und Objekte darstellen können
  - dynamische Eigenschaften
    - Operationen
    - Beziehungen zwischen Operationen
  - Integritätsbedingungen an
    - Objekte und Operationen

Entity-Relationship-Modelle [Chen 1976]

- **Entity:**
  - Objekt der realen oder der Vorstellungswelt über das Informationen zu speichern sind
  - Bsp.: Vorlesungsveranstaltung, Buch, Lehrperson, Prüfung
- **Relationship:**
  - Beziehungen zwischen Entities
  - Bsp.: Lehrperson hält Vorlesung
- **Attribut:**
  - Eigenschaft von Entities oder Beziehungen
  - Bsp.: ISBN eines Buches, Titel einer Vorlesung

ER-Modell: Beispiel



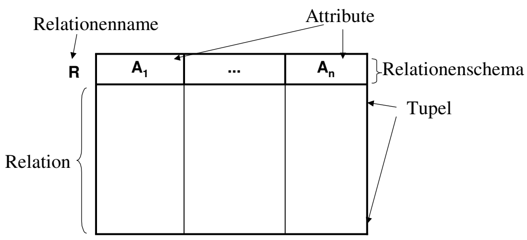
Erweiterungen des ER-Modells

- Spezialisierung
  - IST-Beziehung, Vererbung von Attributen
  - Bsp.: Professor als Spezialisierung von Mitarbeiter
- Generalisierung
  - Entities in einen allgemeineren Kontext
  - Bsp: Person oder Institut als Ausleiher
- Partitionierung
  - Spezialfall der Spezialisierung, mehrere disjunkte Entity-Typen
  - Bsp: Partitionierung von Büchern in Monographien und Sammelbänden

Erweiterungen des ER-Modells

- Komplexe Objekte
  - Aggregierung
    - Entity aus einzelnenen Instanzen anderer Entity-Typen zusammengesetzt
    - Bsp.: Fahrzeug aus Motor, Karosserie
  - Sammlung oder Assoziation
    - Mengenbildung
    - Team als Gruppe von Personen
- Beziehungen höheren Typs
  - Spezialisierung, Generalisierung für Beziehungstypen

Relationenmodell [Codd 1970]



Begriffe des Relationenmodells

- **Attribut:** Spalte einer Tabelle
- **Wertebereich (Domäne):** mögliche Werte eines Attributs
- **Attributwert:** Element eines Wertebereichs
- **Relationenschema:** Menge von Attributen
- **Relation:** Menge von Zeilen einer Tabelle
- **Tupel:** Zeile einer Tabelle
- **Datenbankschemata:** Menge von Relationenschemata
- **Datenbank:** Menge von Relationen

Begriffe des Relationenmodells

- **Schlüssel:** minimale Menge von Attributen, deren Werte ein Tupel einer Tabell eindeutig identifizieren
- **Primärschlüssel:** ein beim Datenbankentwurf ausgezeichneter Schlüssel
- **Fremdschlüssel:** Attributmenge, die in einer anderen Relation Schlüssel ist
- **Fremdschlüsselbedingung:** alle Attributwerte des Fremdschlüssels tauchen in der anderen Relation als Werte des Schlüssels auf

Relationenalgebra

- Grundlage des Relationenmodells
- Operationen
  - Selektion σ: Auswahl von Tupeln aus einer Relation
  - Projektion π: Auswahl bestimmer Spalten
  - Verbund: Verknüpfung von zwei Relationen über alle gemeinsamen Attribute
  - Mengenoperationen: $\cap$, $\cup$, $-$
  - Umbenennung β: Veränderung von Attributen

Anfragen und Änderungen

- Bisher:
  - Relationenschemata mit Basisrelationen, die in der Datenbank gespeichert sind
- Jetzt:
  - „abgeleitete“ Relationenschemata mit virtuellen Relationen die aus Basisrelationen berechnet werden
  - Basisrelationen bleiben unverändert

Anfragen und Änderungen

- **Anfrage:** Folge von Operationen, die aus den Basisrelationen eine Ergebnisrelation berechnet
- **Sicht:** Folge von Operationen, die unter einem Sichtnamen langfristig abgespeichert wird und unter diesem Namen wieder aufgerufen werden kann; ergibt Sichtrelation
- **Snapshot:** Ergebnisrelation einer Anfrage, die unter einem Snapshot-Namen abgelegt wird, aber nie ein zweites Mal (mit geänderten Basisrelationen) berechnet wird

Kriterien für Anfragesprachen

- **Ad-hoc-Formulierung:** Formulierung einer Anfrage ohne vollständiges Programm zu schreiben
- **Deskriptivität:** „Was“ soll formuliert werden, nicht „Wie“
- **Mengenorientiertheit:** jede Operation soll auf Mengen von Daten gleichzeitig arbeiten, nicht navigierend auf einzelnen Elementen
- **Abgeschlossenheit:** Ergebnis ist wieder Relation und kann als Eingabe für nächste Anfrage verwendet werden
- **Adäquatheit:** Unterstützung aller Konstrukte des zugrundeliegenden Datenmodells
- **Orthogonalität:** Sprachkonstrukte in ähnlichen Situationen auch ähnlich anwendbar

Kriterien für Anfragesprachen

- **Optimierbarkeit:** Sprache besteht aus wenigen Konstrukten, für die es Optimierungsregeln gibt
- **Effizienz:** jede Operation ist effizient ausführbar
- **Sicherheit:** keine syntaktisch korrekte Anfrage darf in Endlosschleige geraten oder ein unendliches Ergebnis liefern
- **Eingeschränktheit:** keine komplette Programmiersprache
- **Vollständigkeit:** muß mind. Anfragen einen Standardsprache (z.B. Relationenalgebra) ausdrücken können

SQL-Kern

SELECT	▪ Projektionsliste ▪ arithmetische Operationen und Aggregatfunktionen
FROM	▪ zu verwendende Relationen ▪ eventuelle Umbenennungen (Tupelvariable, Alias)
WHERE	▪ Selektionsbedingungen ▪ Verbundbedingungen ▪ Geschachtelte Anfragen (SFW-Block)
GROUP BY	▪ Gruppierung für Aggregatfunktionen
HAVING	▪ Selektionsbedingung an Gruppen

FROM-Klausel

- Syntax:
  select *
  from relationenliste
- Beispiel:
  select *
  from Bücher

  liefert die gesamte Relation Bücher

Kartesisches Produkt

- FROM-Klausel mit mehr als einer Relation
  select * **from** Bücher, Ausleih
- Berechnung des kartesischen Produktes
- Einführung von Tupelvariablen (mehrfacher Zugriff auf eine Relation)
  select * **from** Bücher eins, Bücher zwei
- Ergebnis mit acht Spalten
  eins.Invnr, eins.Titel, eins.ISBN, eins.Autor
  zwei.Invnr, zwei.Titel, zwei.ISBN, zwei.Autor
- Selbst-Verbund für tupelübergreifende Selektionen

SQL-92-Spezialitäten

- Verbunde als explizite Operatoren
  - Kartesisches Produkt
    select * **from** Bücher, Ausleih
    select * **from** Bücher **cross join** Ausleih
  - Verbund über Verbundbedingungen
    select * **from** Bücher, Ausleih
    where Bücher.Invnr = Ausleih.Invnr
  - Join-Operator: θ-Verbund
    select * **from** Bücher **join** Ausleih
    on Bücher.Invnr = Ausleih.Invnr

Weitere Verbunde in SQL-92

- Gleichverbund
  select * **from** Bücher **join** Ausleih
  using (Invnr)
- Natürlicher Verbund
  select * **from** Bücher **natural join** Ausleih
- Nach FROM: jeder SFW-Block (Orthogonalität in SQL-92)

Äußere Verbunde

- Statt **inner join** nun **outer join**
  - *Dangling tuples* übernehmen
  - Mit Nullwerten auffüllen
- **full outer join**: in beiden Operanden
- **left outer join**: im linken Operanden
- **right outer join**: im rechten Operanden

Äußere Verbunde

LINKS

A	B
1	2
2	3

RECHTS

B	C
3	4
4	5

NATURAL JOIN

A	B	C
2	3	4

OUTER

A	B	C
1	2	⊥
2	3	4
⊥	4	5

LEFT

A	B	C
1	2	⊥
2	3	4

RIGHT

A	B	C
2	3	4
⊥	4	5

SELECT-Klausel

- Relationenalgebra: abschließende Projektion
  select [**distinct**] { attribut |
  arithmetischer-ausdruck |
  aggregat-funktion }
  from ...
- Attribute aus FROM-Relationen
- Arithmetische Ausdrücke über Attribute und Konstanten
- Aggregatfunktionen über Attributen

Projektionsergebnis

- Ergebnismenge statt Multimenge: **distinct**

select Name
from Ausleih

Name
Meyer
Schulz
Müller
Meyer

select distinct Name
from Ausleih

Name
Meyer
Schulz
Müller

Tupelvariablen und Relationennamen

- Angabe der Attributnamen durch Präfix ergänzen
  select ISBN **from** Bücher
  select Bücher.ISBN **from** Bücher
- Benutzung von Tupelvariablen
  select eins.ISBN, zwei.Titel
  from Bücher eins, Bücher zwei

WHERE-Klausel

- Selektionsbedingung der Relationenalgebra oder Verbundbedingung
  select ... **from** ... **where** bedingung
- Bedingung:
  - Konstanten-Selektion
    attribut θ konstante
  - Attribut-Selektion zwischen zwei Attributen mit kompatiblen Wertebereichen
    attribut1 θ attribut2
  - wie folgt

Verbundbedingung

- Angabe der Bedingung für Verknüpfung der Tupel
  relation1.attribut = relation2.attribut
- Bsp.: natürlicher Verbund
  select Bücher.Titel, Buch_Stichwort.Stichwort
  from Bücher, Buch_Stichwort
  where Bücher.ISBN = Buch_Stichwort.ISBN
- Auch Gleichverbund und θ–Verbund erlaubt

Bereichs- und Ungewißheitsselektion

- Bereichsselektion
  attribut **between** konstante1 **and** konstante2
  - Abkürzung für
    attribut >= konstante1 **and** attribut <= konstante2
- Ungewißheitsselektion mit **like**
  - Abkürzung fur disjunktiv verknüpfte Bedingungen
  - Spezialkonstante mit
    - % - kein oder beliebig viele Zeichen
    - _ - genau ein Zeichen

Weitere Bedingungen

- Null-Selektion
  attribut **is null**
- Quantifizierte Bedingungen
  - wenn ein Argument in Vergleich Menge liefert (**all**, **any**, **some** und **exists**)
- Boolesche Ausdrücke mit Konnektoren
  - **or**, **and** und **not**

Schachtelung von Anfragen

- WHERE-Klausel kann geschachtelt werden
- SFW-Blöcke liefern im allgemeinen mehrere Werte
- Vergleich mit Wertemengen
  - Standardvergleiche in Verbindung mit Quantoren **all** ($\forall$) oder **any** ($\exists$)
  - Spezielle Prädikate für den Zugriff auf Mengen: **in**, **exists**

IN-Prädikat

- Anwendung
  attribut **in** (SFW-Block)
- Beispiel
  select Titel **from** Bücher
  where ISBN **in** (**select** ISBN **from** Empfiehlt)
- Abarbeitung:
  - Ergebnis der inneren **select**-Anweisung hinter **in** als Liste von Konstanten einsetzen
  - Modifizierte Anfrage abarbeiten

Verzahnt geschachtelte Anfragen

- In der inneren Anfrage Relationen- oder Tupelvariablen-Namen aus dem FROM-Teil der äußeren Anfrage verwenden
  select Nachname
  from Personen
  where 1.0 **in**
  (**select** Note
  from Prüft
  where PANr = Personen.PANr)

Abarbeitung

- In der äußeren Anfrage erstes Personen-Tupel untersuchen; Ergebnis in innere Anfrage einsetzen
- Innere Anfrage auswerten
  select Note **from** Prüft **where** PANr = 4711
  liefert Werteliste (2.0, 2.3)
- Ergebnis der inneren Anfrage in die äußere einsetzen
  1.0 **in** (2.0, 2.3)
- Nächstes Tupel untersuchen usw.

EXISTS-Prädikat

- Test, ob Ergebnis der inneren Anfrage leer ist

```
select ISBN
from Buch_Exemplare
where exists
    (select * from Ausleihe
     where Invnr = Buch_Exemplare.Invnr)
```


Vereinigung, Durchschnitt, Differenz

- **union**, **intersect** und **except** orthogonal in andere Anfragen einsetzbar

```
select count(*)
from ( (select PANr from Professoren)
    union
    (select PANr from Studenten))
```

- **corresponding**-Klausel: zwei Relationen nur über ihre gemeinsamen Attribute vereinigen

```
select count(*)
from (Professoren union corresponding
    Studenten)
```


Kompatible Attribute

- Attribute sind kompatibel bei kompatiblen Wertebereichen
- Wertebereiche sind kompatibel, wenn sie
  - gleich sind oder
  - beide auf character basieren (unabhängig von der Länge)
  - beide numerisch sind (unabhängig vom genauen Typ)
- benutzbar in Vergleichen und Mengenoperationen

Relationenalgebra vs. SQL

Relationenalgebra	SQL-92
Projektion	**select distinct**
Selektion	**where** ohne Schachtelung
Verbund	**from**, **where** **from** mit **join**
Umbenennung	**from** mit Tupelvariable **as**
Differenz	**where** mit Schachtelung **except corresponding**
Durchschnitt	**where** mit Schachtelung **intersect corresponding**
Vereinigung	**union corresponding**

Operationen auf Wertebereichen

- Innerhalb von SELECT und WHERE: statt Attribute auch skalare Ausdrücke
  - Numerische Wertebereiche: +, -, *, /
  - Strings: **char_length**, Konkatenation ||, **substring**
  - Datumstypen, Zeitintervalle: **current_date**, **current_time**, +, -, *
- Ausdrücke werden tupelweise ausgewertet

Aggregatfunktionen

- Tupelübergreifend
  - **count**: Anzahl der Werte einer Spalte (Spezialfall: **count**(*))
  - **sum**: Summe der Werte einer Spalte
  - **avg**: arithmetisches Mittel der Werte einer Spalte
  - **max** bzw. **min**: größter bzw. kleinster Wert einer Spalte
- Argumente einer Aggregatfunktion
  - Attribut der durch FROM spezifizierten Relation
  - gültiger skalarer Ausdruck
  - Bei **count** auch *
  - Vor Argument: **distinct** oder **all** (Voreinstellung)

Aggregatfunktionen - Beispiele

select sum(Preis) **from** Buch_Versionen

select count(**distinct** PANr) **from** Prüft

select avg(**all** Note)
from Prüft
where V_Bezeichnung = 'Datenbanken I'

Gruppierung

- Erzeugung virtuell geschachtelter Relationen
  select ... **from** ... [**where** ...]
  [**group by** attributliste]
  [**having** bedingung]
- Semantik:
  - Relationenschema R und Attributmenge hinter Gruppierung G schachteln nach Attributen R – G, d.h. für gleiche G-Werte werden Resttupel in Relation gesammelt
  - **having** ist Selektionsbedingung auf gruppierter Relation

Gruppierung: Schema

1. Schritt:
   aus **from** und **where** resultierende Relation

A	B	C	D
1	2	3	4
1	2	4	5
2	3	3	4
3	3	4	5
3	3	6	7

2. Schritt:
   group by A, B

A	B	C	D
1	2	3	4
		4	5
2	3	3	4
3	3	4	5
		6	7

Gruppierung: Schema

3. Schritt:
   select A, **sum** (D)

A	sum(D)	C	D
1	9	3	4
		4	5
2	4	3	4
3	12	4	5
		6	7

4. Schritt:
   having A < 4 **and**
   sum(D) < 10 **and**
   max(C) = 4

A	sum(D)
1	9

Quantoren und Mengenvergleiche

- Allquantor: **all**
- Existenzquantoren: **any**, **some**
  attribut θ { **all** | **any** | **some** } (
  select attribut **from** ... **where** ..)
- Anwendbarkeit eingeschränkt
- Beispiel:
  select PANr, Immatrikulationsdatum
  from Studenten
  where Matrikelnummer = **any** (
  select Matrikelnummer **from** Prüft)

Selbst-Verbund

- Verbund über eine Relation
  Gib alle Bücher aus, an denen 'Vossen' und 'Witt' gemeinsam als Autoren beteiligt waren

  select B1.ISBN
  from Buch_Autor B1, Buch_Autor B2
  where B1.ISBN = B2.ISBN **and**
  B1.Autor='Vossen' **and** B2.Autor='Witt'

ORDER BY-Klausel

- Menge von Tupeln → Liste
  order by attributliste

  select Matrikelnummer, Note **from** Prüft
  where V_Bezeichnung = 'Datenbanken I'
  order by Note **asc**
- Sortierung
  - Aufsteigend: **asc**
  - Absteigend: **desc**
  - Anwendung auf Ergebnis der vorangehenden SFW-Anfrage !!

Änderungsoperationen

- **insert**: Einfügen von Tupeln in Basisrelationen oder Sichten
- **delete**: Löschen von Tupeln
- **update**: Ändern von Tupeln

- Operationen als
  - Eintupel-Operation
  - Mehrtupel-Operationen

UPDATE

update basisrelation
set attribut_1 = ausdruck_1,
...
attribut_n = ausdruck_n
[**where** bedingung]

update Angestellte
set Gehalt = Gehalt + 1000
where Gehalt < 5000

INSERT

insert
into basisrelation [(attribut_1, ..., attribut_n)]
values (konstante_1, ..., konstante_n)

insert into Buch (Invrnr, Titel)
values (4867, ´Wissensbanken´)

insert
into basisrelation [(attribut_1, ..., attribut_n)]
SQL-Anfrage

insert into Kunde
(**select** LName, LAdr, 0 **from** Lieferant)

DELETE

delete from basisrelation [**where** bedingung]

delete from Ausleihe **where** Invnr = 4711

- Standardfall: Löschen mehrer Tupel
  delete from Ausleihe **where** Name = ´Meyer´
- Löschen der gesamten Relation:
  delete from Ausleihe

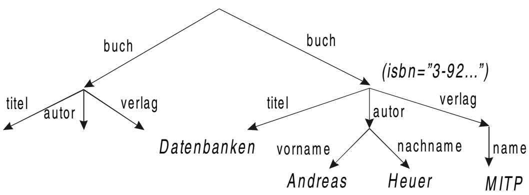
Semistrukturierte Daten

- **Semistrukturierte Daten/Dokumente**
  - Daten mit einer internen, oft wechselnden und nicht streng typisierten Struktur
- **Merkmale**
  - Kein zentrales Schema, sondern implizit in jedem Dokument
  - Wechselnde Struktur
  - Daten ohne weitere Struktur
  - Keine Datentypen bzw. Datentypen nicht als Integritätsbedingungen
  - große Anzahl (möglicher) Attribute
  - Unscharfe Trennung von Daten und Schema

Datenmodell für semistrukturierte Daten

- Graphenbasierte Modelle
  - Beispiele: OEM, XML
  - Dokument modelliert als **Graph** mit
    - **Kanten** bezeichnet mit Element-Tag-Bezeichnern
    - **Knoten** bezeichnet mit Attribut-Wert-Paaren
    - **Blätter** bezeichnet mit Werten (Strings)
    - **Wurzelknoten**
  - Knoten = Objekt

Datenmodell für semistrukturierte Daten



XML – Überblick

- XML = Extensible Markup Language
  - Metasprache als Weiterentwicklung von SGML
  - Nachfolger von HTML ?
- Metasprache: Definition von Dokumenttypen
  - XML-Dokument = Definition (DTD) + Instanz
- DTD (Document Type Definition):
  - formale Grammatik zur Definition einer bestimmten XML-Sprache
  - Namen der in Instanzen erlaubten Tags sowie deren mögliche Schachtelung

XML – Strukturen

- Elemente (Tags), Attribute (Typ)
- Schachtelung von Elementen
  - Sequenz: (E_1, E_2)
  - Alternative: (E_1 | E_2)
  - Iteration:
    - 0...n Wiederholungen: E*
    - 1...n Wiederholungen: E+
    - Optionales Element: E?

XML – Beispiel einer DTD

```
<!element   buch   (titel, verlag, autor*, abstrakt?)>
<!attlist   buch   isbn cdata #required>
<!element   titel     (#PCDATA)>
<!element   verlag (name, adresse)>
<!element   name  (#PCDATA)>
<!element   adresse (#PCDATA)>
<!element   autor   (vorname?, nachname)>
<!element   abstrakt (#PCDATA)>
```


XML-Instanzen

- **Instanz** = Dokument, dessen Inhalt mit Tags einer bestimmten DTD ausgezeichnet ist
  - Wohlgeformt: enthalten Tags, die XML-Regeln entsprechen
  - Gültig: enthalten DTD und dürfen Inhaltsmodell nicht verletzen
  - DTD **intern** (im Dokument definiert) oder **extern** (Verweis auf DTD)

XML – Beispieldokument

```
<?xml version="1.0" ?>
<!DOCTYPE buch SYSTEM "buch.dtd">
<buch isbn= "3-929821-31-1">
    <titel>Datenbanken</titel>
    <verlag>
        <name>MITP</name><adresse>Bonn</adresse>
    </verlag>
    <autor>
        <vorname>Andreas</vorname>
        <nachname>Heuer</nachname>
    </autor>
</buch>
```


XML – Anwendungen

- (Semi-)strukturierte Dokumente
  - Dokumentation
  - Web-Inhalte (erfordert Stylesheets: Zuordnung der Präsentationsform)
- Datenaustausch
  - Import/Export von Daten aus Datenbanken
  - Austausch von Geschäftsdaten (Rechnungen, Bestellungen etc.)
  - Erfordert standardisierte DTDs
- Informationsintegration

XML – Anfragesprachen

- Anfragen in XML-Datenbeständen (Dokumente, Repositories)
  - Extraktion
  - Transformation
  - Integration
- Vorschläge
  - XML-QL (AT&T, INRIA)
  - XQL (Microsoft)

XML – Anfragesprachen

- XML-QL
  - Anlehnung an klassische Anfragesprachen (SELECT-WHERE)

```
WHERE <buch>
  <verlag><name>MITP</name></verlag>
  <titel>$t</titel>
  <autor>$a</autor>
</buch> IN www.a.b.c/bib.xml
CONSTRUCT <result>
        <autor>$a</autor> <titel>$t</titel>
  </result>
```


XML – Anfragesprachen

- Ergebnis

```
<result>
   <autor>
        <vorname>Andreas</vorname>
        <nachname>Heuer</nachname>
   </autor>
   <titel>Datenbanken</titel>
</result>
```


XML – Anfragesprachen

- XQL
  - nutzt Elemente von XSL (Stylesheet-Sprache) zur Mustererkennung
  - Konstrukte
    - Elemente in einem Kontext: buch
    - Pfadausdrücke: buch/titel, buch/*
    - Filter (WHERE): buch[titel = 'XYZ']
    - Boolesche Ausdrücke: and, or
    - Vereinigung und Durchschnitt: $union$, $intersect$

XML – Anfragesprachen

- XQL-Beispiele
  - Verlag mit Namen 'MITP'
    verlag[name = 'MITP']
  - Alle Autoren
    autor
  - Alle Bücher vom Verlag 'MITP'
    buch[verlag/name = 'MITP']

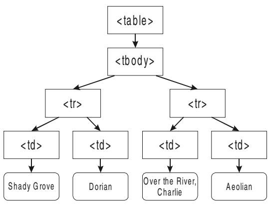
XML – Document Object Model

- API (Application Programming Interface) für XML und HTML-Dokumente
- Definiert logische Struktur und Mechanismen für Zugriff und Manipulation eines Dokumentes
- Unterstützt
  - Erzeugen von Dokumenten
  - Navigation über Struktur
  - Hinzufügen, Löschen und Modifieren von Elementen

DOM - Strukturmodell

- Hierarchie von Knoten (*Node*)
- Knotentypen: *Document*, *Element*, *Attr*, *Comment*, *Text*, ...

```
<table>
<tbody>
<tr>
<td>Shandy Grove</td>
<td>Aoelian</td>
</tr>
<tr>
<td>Over the River, Charlie</td>
<td>Dorian</td>
</tr>
</tbody>
</table>
```



DOM - Spezifikation

- Schnittstellenspezifikation in IDL
- Sprachanbindungen für Java, JavaScript, ...
- Beispiel:

```
interface Node {
        attribute unsigned short nodeType;
        attribute DOMString nodeName
        attribute DOMString nodeValue;
        attribute Node parentNode;
        attribute NodeList childNodes;
        ...
};
```


Anfragen in MMDBS

- **Inhaltsbasiert:** Suche anhand des Inhaltes der Medienobjekte
- **Räumlich:** Anfragen bzgl. der räumlichen Anordnung von Objekten
- **Zeitbezogen:** Suche über zeitliche Zusammenhänge innerhalb und zwischen den Medienobjekten
- **Unscharf:** unscharfe Formulierung einer Anfrage unter Ausnutzung von Methoden des Information Retrieval

Inhaltsbasierte Anfragen

- Abbildung des Inhaltes
  - **Schlüsselwort-basiert:** Benutzer-Kommentare
  - **Feature-basiert:** Menge von automatisch (d.h. aus dem Inhalt) abgeleiteten Features, z.B. Farbe, Textur, Gestalt, Position
  - **Konzept-basiert:** Interpretation des durch Features abgeleiteten Inhaltes mit Hilfe von Anwendungswissen (erfordert Benutzerunterstützung)

Inhaltsbasierte Anfragen

- Abschwächung bzw. Verschärfung von Anfragen bei leeren oder zu umfangreichen Ergebnissen
- Methoden
  - **Manuell:** iterativer Anfrageprozeß, Relevanz-Feedback
  - **Automatisch:** Ausführung durch DBMS, Hierarchie mit Beziehungen zwischen Objekten (Musik → Klassik → Beethoven)

Räumliche Anfragen

- Repräsentation
  - Minimale umschreibende Rechtecke
  - Objekt als Punktmenge
- Beziehungen
  - Richtungsbeziehungen: räumliche Lage von Objekten zueinander (Referenzobjekt) (N, S, O, W usw.)
  - Topologische Beziehungen: Abbildung von Nachbarschaftsverhältnissen (equal, inside, cover,overlap, touch, disjoint)
  - Distanzbeziehungen: Beschreibung des räumlichen Abstands durch Prädikate

Räumliche Anfragen

- Operationen
  - Testoperationen
    - equal, touch, inside, cover, ...
  - Operationen zur Berechnung numerischer Werte
    - Abstand, Länge, Fläche
  - Mengenoperationen zwischen Regionen
    - Durchschnitt, Vereinigung, Differenz

Zeitbezogene Anfragen

- Für zeitabhängige Daten (Video, Audio): Berücksichtigung zeitlicher Abhängigkeiten
- Zeittypen
  - Zeitintervall
  - Zeitpunkt
  - Zeitspanne (Abstand zwischen Ober- und Untergrenze eines Intervalls)
- Operationen
  - Intervall-, Punkt- und Zeitspannebezogen

Unscharfe Anfragen

- Ergebnis: bezüglich der *Relevanz* geordnete Liste
- Beschreibung der Relevanz von Objekten in der Anfrage:
  - Prädikate zur Beschreibung von Ähnlichkeit oder Nähe von Objekten
    - Vorgabe eines Objektes, zu dem ähnliche Objekte zu suchen sind
    - Räumliche Umgebungssuche
    - Suche nach Texten, die Worte oder Terme in ähnlicher Weise enthalten
  - Verwendung unpräziser Ausdrücke und Prädikate
  - Zuordnung von Gewichten zu Anfragebedingungen

MM-Anfragesprachen: MOQL

- Erweiterung von OQL (Object Query Language)
  select ...
  from ...
  where condition
- WHERE-Bedingung
  - Beschreibung räumlicher Beziehungen
  - Beschreibung zeitlicher Beziehungen
  - Beschreibung von Beziehungen zwischen abgeleiteten Objekten und Mediendaten

MM-Anfragesprachen: MOQL

- Beispiele:
  - „Suche alle Waldgebiete in Brandenburg und berechne deren Größe“

```
select wald, area(wald.region)
from Waldgebiet wald, Bundesland land
where wald.region inside land.region
        and land.name.equal ("Brandenburg")
```

  - „Suche alle Bilder mit Personen“

```
select b
from Bilder b, Personen p
where b contains p
```


MM-Anfragesprachen: SQL/MM

- Zukünftiger ISO-Standard für Multimedia-Erweiterung zu SQL
- Definition abstrakter Datentypen und Funktionen für Repräsentation von Multimedia-Objekten
- Unterstützung (Repräsentation, Suche) von
  - Textdokumenten
  - räumlicher Daten
  - Bildern