Klausurrelevante Themen:

Grundlagen:

- **DBMS**: Datenbank-Management-System
- **DBS**: Datenbanksystem (DBMS + Datenbank)
- Prinzipien
  - **3-Ebenen-Architektur** (intern: Datenverwaltung, konzeptuell: Gesamtstruktur, extern: verschiedene Anwendungen)
  - Trennung zwischen Schema (Struktur) und Instanz (Inhalt)
- Codd'sche Anforderungen
  1. **Integration**: einheitliche nichtredundante Datenverwaltung
  2. **Operationen**: Speichern, Suchen, Ändern (ad hoc und "fest verdrahtet")
  3. **Katalog**: Zugriffe auf Datenbankbeschreibung
  4. **Benutzersichten**: unterschiedl. ~ auf Datenbankausschnitte
  5. **Konsistenzüberwachung**: Gewährleistung der Korrektheit des Datenbankinhaltes gegenüber Schema
  6. **Datenschutz**: Ausschluss unautorisierter Zugriffe
  7. **Transaktionen**: mehrere Operationen als Funktionseinheit
  8. **Synchronisation**: parallele Transaktionen koordinieren
  9. **Datensicherung**: Wiederherstellung von Daten nach Systemfehlern
- **Komponenten** eines DBSs
  - **Datendefinition**: konzeptuelle Definition aller Daten (konzeptuelles Schema)
  - **Sichtdefinition**: Definition von Benutzersichten (externes Schema)
  - **Dateiorganisation**: Definition des (internen) Dateiaufbaus und Zugriffsstrukturen (internes Schema)
  - **Masken**: Entwurf der Benutzeroberfläche
  - **Operationen**: Anfragen und Änderungsoperationen (Updates) in Programmen
  - **Einbettung**: Einbettung von Datenbankoperationen in Programme
  - verschiedene **Anwendungsprogramme**
  - **Optimierung**: Umformung von Abfragen in effiziente Gestalt
  - **Auswertung**: Durchführung der umgeformten Abfragen
  - **Plattenzugriff**: effiziente Ablage der Daten
- Entwicklungslinien
  60er: Netzwerkmodell; 70er: relational; 80er: Wissenssysteme, objektorientierte ~, geographische ~

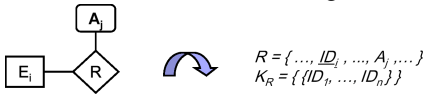
Konzeptioneller Entwurf

- Beschreibung der zu verwaltenden Daten unabhängig von ihrer Realisierung
  Quellen: Analyseklassendiagramm oder (bei kleineren Projekten) auch ad hoc
  1. **Sichtenentwurf**
     Modellierung von Sichten z.B. für verschiedene Anwendungen durch einzelne ER- oder Klassendiagramme
  2. **Sichtenanalyse**
     Analyse der vorliegenden Sichten in Bezug auf Konflikte: Namen, Typen, Bedingungen, Struktur (gleicher Sachverhalt durch unterschiedliche Konstrukte wie z.B. Klassen/Attribute)
  3. **Sichtenintegration**
     Auflösen der Konflikte und Integration in ein Gesamtschema
- **Datenbankmodell**
  Def.: Ein Datenbankmodell ist ein System von Konzepten zur Beschreibung von Datenbanken. Es legt Syntax und Semantik von Datenbankbeschreibungen für ein Datenbanksystem fest.
- **Datenbankbeschreibung**
  - statische Eigenschaften
    - Objekte
    - Beziehungen zwischen Objekten
    - Attribute von Objekten und Beziehungen
  - dynamische Eigenschaften wie
    - Operationen
    - ihre Abfolge und Koordination
  - Integritätsbedingungen an
    - Objekte
    - Operationen
- **Schema**: Beschr. der Menge der erlaubten Datenbankzustände
- **Instanz**: Datenbankzustand zu einem Zeitpunkt t
- Entity-Relationship
  $\mu(E)$: Menge der möglichen Entities vom Typ E
  $\sigma(E)$: Menge der aktuellen Entities vom Typ E im Zustand σ (Teilmenge von $\mu(E)$ und immer endlich)
  Seien $E_1, \ldots, E_n$ Entities, R Beziehungstyp. Es gilt:
  $\sigma(R) \subseteq \sigma(E_1) \times \ldots \times \sigma(E_n)$
  Sei A vom Typ D ein Attribut des Entitiy-Typs E. Es gilt:
  $\sigma(A) : \sigma(E) \rightarrow \mu(D)$ ist Abbildung
  Optionale Attribute ergeben partielle Funktion
  Notation für Kardinalitätsangaben an einem Beziehungstyp
  $R(E_1, \ldots, E_i[\min_i, \max_i], \ldots, E_n)$
- **Funktionale Beziehungen**
  sind wie Entity-wertige Attribute
  Notation: $\sigma(bat)(Lehrer1)=Schüler1$
- IST-Beziehung: Beschreibung von Mengeninklusionen

Relationaler Entwurf

- Beschreibung in einem Relationenschema
  - **Ausgangsschemata**
    aus integriertem Schemata des konzeptionellen Entwurfs (ER-Diagramm) abgeleitet oder ad hoc entworfen
  - **Normalisierte Schemata**
    durch wiederholte Analyse und Zerlegung (Normalisierung) aus Ausgangsschemata gewonnen
- **Relationenmodell**
  - **Attribut**: Bezeichnung der Spalte einer Tabelle, eine Attributmenge ist identifizierend für eine Relation r, wenn gilt:
    $\forall t_1, t_2 \in r: t_1 \neq t_2 \Rightarrow \exists B \in K: t_1(B) \neq t_2(B)$
  - **Wertebereich**: Mögliche Werte eines Attributs (auch Domäne)
  - **Attributwert**: Element eines Wertebereichs
  - **Relationenschema**: Menge von Attributen
  - **Relation**: Menge von Zeilen einer Tabelle, formal: Menge von Abbildungen $R \rightarrow \bigcup_{i=1}^{m} D_i$,
    2 Arten: abgeleitete Relation vs. Basisrelation
  - **Tupel**: Zeile einer Tabelle
  - **Datenbankschema**: Menge v. Relationenschemata, Notation: S
  - **Datenbank**: Menge von Relationen (Basisrelationen), $d(S)$
  - **Universum**: Menge ($\neq \varnothing$) aller Attribute d. Datenbankschemas

  Beispiel:
  $S = \{Personen, Pers\_Telefon\}$, $d(S)=\{r(Personen), r(Pers\_Telefon)\}$
- Abbildung von **ER- auf Relationenschemata**
  eine Transformation ist **kapazitätserhaltend**, falls es eine bijektive Abbildung zwischen der Menge der Instanzen des ER-Schemas und der Menge der Instanzen des relationalen DB-Schemas gibt
  Transformation von Beziehungstypen:
  - **Ohne Kardinalitätsbeschränkung:**

  - **Mit Kardinalitätsbeschränkung:**
    Wenn E_i Kardinalität [0, 1] hat: Primärschlüssel des Relationenschema E_i wird ein Schlüssel von R

  - Δ Wenn E_i Kardinalität [1, 1] hat: Relationenschemata R und E_i werden verschmolzen
- **Funktionale Abhängigkeiten**
  Notation: F – Menge von funktionalen Abhängigkeiten, f – funktionale Abhängigkeit, $\mathrm{SAT}_R(F)$ – Menge aller Relationen über dem Relationenschema R, die die funktionalen Abhängigkeiten aus F erfüllen. Gilt für f über R: $\mathrm{SAT}_R(F) \subseteq \mathrm{SAT}_R(f)$, dann impliziert F die funktionale Abhängigkeit f; kurz: $F \models f$.
  - **Transitive Hülle** einer FD f: Menge der implizierten Abhängigkeiten von f.
  - **RAP-Algorithmus**: Berechnung der Menge X_F^* aller mit F von X abhängigen Attribute
  - F heißt **äquivalent** zu G, falls $F^+ = G^+$, F heißt **minimal**, wenn F in allen äquivalenten Mengen enthalten ist. Eine FD $f \in F$ heißt **redundant**, falls $F \setminus \{f\} \equiv F$ gilt.
- **1. Normalform**
  - Attribute der Relation müssen atomar sein (ist eine schon vorausgesetzte Eigenschaft von Relationen): Strukturierte Attribute (wie Adresse) müssen aufgeteilt werden in ihre Teilattribute (z.B. in PLZ, Ort, Straße und Hausnummer).
  - Relationen in erster Normalform: „flache Relationen“
  - Aufgrund von funktionalen Abhängigkeiten (PLZ bestimmt Ort) ergeben sich in 1NF-Relationen Redundanzen.
- **2. Normalform**
  - **Vermeidung partieller funktionaler Abhängigkeiten** (diese bewirken Redundanzen)
  - besteht, wenn Attribute (die in keinem Schlüssel vorkommen) funktional von einem Teil eines Schlüssels abhängen. Die zweite Normalform kann durch Elimination der abhängigen Attribute und Auslagerung in eine eigene Relation erreicht werden.
- **3. Normalform**
  - Kein Nicht-Schlüsselattribut ist funktional abhängig von einer Attributgruppe ohne Schlüsseleigenschaft ist.
  - Zur Beseitigung kann man das transitiv abhängige Attribut in eine neue Relation kopieren (gemeinsam mit der bestimmenden Attributmenge) und aus der ursprünglichen Relation entfernen.
- **Boyce-Codd-Nornamlform**
  - Kein Attribut ist funktional abhängig von einer Attributgruppe ohne Schlüsseleigenschaft ist. (Verschärfung der 3. Normalform, bei der Teile eines Schlüssel von anderen Schlüsseln funktional abhängig sein dürfen)
- **Abhängigkeitstreue**: Menge der FD‘s äquivalent zur Menge der Schlüsselabhängigkeiten, d.h.: Alle gegebenen Abhängigkeiten sind durch Schlüssel repräsentiert
- **Verbundtreue**: Die Originalrelationen können durch den Verbund der Basisrelationen wiedergewonnen werden

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 eine Sichtrelation
- **Snapshot**: Ergebnisrelation einer Anfrage, die unter einem Snapshot-Namen abgelegt wird, aber nie ein zweites Mal (mit geänderten Basisrelationen) berechnet wird (etwa Jahresbilanzen)
- **Eigenschaften relationaler Anfragesprachen**
  - **Ad-Hoc-Formulierung**: Benutzer kann Anfragen formulieren,ohne ein vollständiges Programm schreiben zu müssen
  - **Deskriptivität**: Der Benutzer kann formulieren „Was will ich haben?“ und nicht „Wie komme ich an das, was ich haben will?“.
  - **Mengenorientiertheit**: Operationen arbeiten auf ganzen Mengen von Daten, statt navigierend einzelnen Elementen.
  - **Abgeschlossenheit**: Das Ergebnis einer Anfrage ist eine Relation und kann wieder als Eingabe für weitere Anfrage verwendet werden.
  - **Adäquatheit**: Alle Konstrukte des zugrunde liegenden Datenmodells werden unterstützt.
  - **Orthogonalität**: Die Operationen sind frei kombinierbar.
  - **Optimierbarkeit**: Die Sprache besteht aus wenigen Operationen, für die es Optimierungsregeln gibt.
  - **Effizienz**: Anfragen sind effizient ausführbar. (Im Relationenmodell hat jede Operation eine Komplexität $\leq O(n^2)$, n = Anzahl der Tupel einer Relation.
  - **Sicherheit**: Keine syntaktisch korrekte Anfrage gerät in eine Endlosschleife oder liefert ein unendliches Ergebnis.
  - **Eingeschränktheit**: Die Anfragesprache ist keine komplette Programmiersprache (folgt aus Sicherheit, Optimierbarkeit, Effizienz).
  - **Vollständigkeit**: Die Sprache kann mindestens die Anfragen Standardsprache (wie die in diesem Kapitel einzuführende Relationenalgebra oder der sichere Relationenkalkül) ausdrücken.
- **Anfragealgebren**
  Operationen (minimale Operationenmenge, ergibt relationale Vollständigkeit):
  - **Projektion** π [*Attributmenge*] (*Relation*)
    Spalten ausblenden
  - **Selektion** σ [*Bedingung*] (*Relation*)
    Zeilen auswählen
  - **Verbund** (Join) $Relation_1 \bowtie Relation_2$
    bei gleich benannten Spalten Tupel mit übereinstimmenden Werten verknüpfen
  - **Vereinigung** $Relation_1 \cup Relation_2$
    Tupel aus zwei Relationen sammeln, Duplikate weglassen
  - **Differenz** $Relation_1 - Relation_2$
    Tupel aus der ersten Relation herausnehmen, falls sie auch in der zweiten Relation vorkommen
  - **Spalten umbenennen** β [$Attributname_{neu} \leftarrow Attributname_{alt}$](*Relation*)
    in Relation ein Attribut in ein anderes umbenennen (wichtig für $\bowtie$ und $\cup, -$)
- **Relationale Abfragekalküle**
  - **Bereichskalkül**: Variablen nehmen Werte elementarer Datentypen (Bereiche) an
    Bsp.: $\{ x \mid Kunde(x, \text{'MD'}, \_) \}$.
    „Welcher Kunde hat Waren unter 1,- EUR bestellt?“
    $\{ x, w \mid Kunde(x, y, z) \wedge bestellt(x, w, a) \wedge Ware(w, p, v) \wedge a > 0 \wedge p < 1.00 \}$
  - **Tupelkalkül**: Variablen nehmen Tupelwerte an
    Bsp.: $\{t.Kname, t.Adresse \mid Kunde(t) \wedge \exists b: (bestellt(b) \wedge b.WName = \text{‘Papier‘} \wedge t.KName = b.KName)\}$
- **Semantisch sichere Abfragen**
  liefern für jeden Datenbankzustand ein endliches Ergebnis
  Semantische Sicherheit ist nicht entscheidbar im Sinne der Berechenbarkeit
- **Änderungsoperationen**
  - insert t into $r_i(R_i)$
  - delete t from $r_i(R_i)$
  - replace $t \rightarrow t'$ into $r_i(R_i)$

Structured Query Language

- **Sprachanteile**
  - Definition von Datenbankschemata (**DDL** - Data Definition Language)
  - Formulierung von Anfragen (**DQL** - Data Query Language)
  - Formulierung von Datenbankänderungsoperationen (**DML** - Data Manipulation Language)