Themengebiete „Datenbanken“

- **Grundlagen**
  - Datenbankbegriff
  - Objekte – Relationen – Beziehungen
  - Anfrage
  - Transaktion ➔ ACID
  - DBMS
  - Relationales Datenmodell
    - 3 Abstraktionsebenen
  - ANSI/SPARC Architektur
  - DDL – SDL – DML
  - Integritätsbedingungen
- **Relationenalgebra**
  - Projektion
  - Selektion
  - Vereinigung
  - Differenz
  - Natürlicher Verbund (Join)
  - Umbenennung
  - Ausdrücke in Relationenalgebra
    - Äquivalenzbegriff
- **Relationenkalkül**
  - Syntax
  - Anfragen
  - Projektion, Selektion, Verbund, Vereinigung, Differenz, Division
  - Sichere R-Formeln
- **SQL**
  - Grundstruktur
  - Relation, Tupel, Attribut
  - Aggregierungsfunktionen
  - Orthogonalität (Schachtelung von Ausdrücken)
  - Umwandlung Algebra ➔ SQL
  - Umwandlung Kalkül ➔ SQL
  - SQL Grundoperationen
  - NULL-Werte ➔ 3wertige Logik
  - Sichten (Views)
    - WITH CHECK OPTION
  - Fehlende Turing-Vollständigkeit von SQL
  - Impedance Mismatch
  - Dynamisches SQL
  - Referentielle Integrität
  - Dynamische Integrität
    - Kompensation
  - Trigger
- **Datenbankentwurf**
  - Anforderungsanalyse
  - Konzeptueller Entwurf
    - E/R Modell
    - Implementationsentwurf
    - Schwache Entitätstypen
    - Generalisierung
    - Aggregattypen
    - Mengenwertige und strukturierte Attribute
  - Vermeidung von Redundanz
    - Funktionale Abhängigkeit
    - Sat(R)
    - Hülle
    - Schlüssel/Superschlüssel
    - Schlüsselattribut vs. Nichtschlüsselattribut
    - Armstrong Axiome

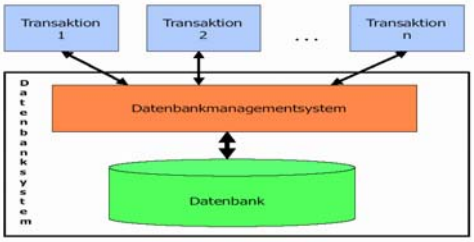
                - Reflexivität
                - Augmentation
                - Transitivität
                - Vereinigung
                - Pseudotransitivität
                - Dekomposition
                - Reflexivität
                - Akkumulation
            - X^+ Algorithmus
            - Äquivalenz von FAs
        - Normalformen
            - 1NF
            - 3NF
            - BCNF
            - 4NF
        - Eindeutigkeitsannahme
- **Physische Datenbank**
    - Datei und Puffermanager
    - Primär-, Sekundär-, Tertiärspeicher
    - Adressierung
        - Seitenorganisierung (Statisch, Dynamisch)
        - 3 Dateiorganisationsformen
    - Indexstrukturen
        - Ballung
        - Dichte
        - (voll) Invertiert
        - Primär-/ Sekundärschlüssel und zusammengesetzte
        - B-Baum
        - Hashing
        - Anfragen über räumliche Daten
        - Mehrdimensionale Indexstrukturen
            - Quadbaum/R-Baum
            - Space-Filling Curves
        - Dynamische Hashverfahren
        - Lineares Hashing
    - Große Datenbanken
        - Merge-Sort
        - Geballter B-Baum
- **Transaktionsverwaltung**
    - ACID
    - Mehrbenutzerbetrieb
    - Verzahnte Transaktionen
        - Fehlersituationen
            - Lost Update
            - Dirty Read
            - Non Repeatable Read
            - Phantom
        - Serialisierbarkeit
            - Korrektheitskriterium
            - Semantik einer Transaktion formalisieren
        - Augmentierter Schedule
        - D-Graphen (Dependencygraph)
        - Äquivalenz von Schedules
        - C-Graph (Conflictgraph)
    - Scheduler
        - 2-PL
            - Optimalität von 2-PL
        - Überwachen der C-Graphen
        - Zeitmarken
        - Optimistische Verfahren
        - Livelocks

        - Deadlocks
        - Intentionlocks
        - Hot-Spots
            - Increment-/Decrementlocks
        - Sperrprotokoll für B-Bäume
- **Fehlerbehandlungs**
    - Arten von Fehlern
        - Transaktionsfehler
        - Systemfehler
        - Mediafehler
    - Logfiles
        - Undo vs. Redo
        - Stealing Frames und Forcing Pages
        - Log-Granulat
        - WAL Regel
        - Restartalgorithmus
        - Checkpoints
        - Schutz vor Mediafehlern
            - Mirroring
            - Dumps
- **Verteilte DBS**
    - 3-tiered Architektur
    - Homogene vs. Heterogene Föderation
    - Globale Serialisierung
        - Globaler Schedule
        - Deadlock
        - Verteiltes Zeitmarkenverfahren
        - Serialisierbarkeit bei heterogenen Föderationen
            - GTM/LTM
        - 2-Phasen Commit-Protokoll
            - Recovery
        - Rollback Work und Rollback Work i
        - Geschachtelte Transaktionen
        - Sagas
- **Sicherheit**
    - Sicherheitsmechanismen
        - DAC
        - Privilegien
        - Rechte ver- und weitergabe
        - REVOKE…CASCADE vs. REVOKE…RESTRICT
        - Autorisierungsgraph
        - Angriffsszenarien
            - Trojanische Pferde
            - Lösung: Bell-La Padula
                - Zugriffsklassen
                - Polyinstantiierung
            - Angriffe auf statistische Datenbanken
        - Verschlüsselung
            - Symmetrisch
            - Public Key Verfahren
            - Digitale Signatur
- **Anfrageoptimierung**
    - Kataloge
        - Baum, Hashindex
        - Qualitätsmaß
    - Verschieden Ausgangspunkte für Selektionen
    - Verschieden Ausgangspunkte für Projektionen
    - Joins
        - Nested-Loop-Join
        - Sort-Merge Join
        - Hash-Join

        - ➔ Abgeleitete Verfahren
    - Optimierungsarten
        - Semantische Optimierung
        - Logische Optimierung
        - Physische Optimierung
    - Heuristiken zur Optimierung
        - Selektivität
    - Pipelining

Fragenkatalog Datenbanken

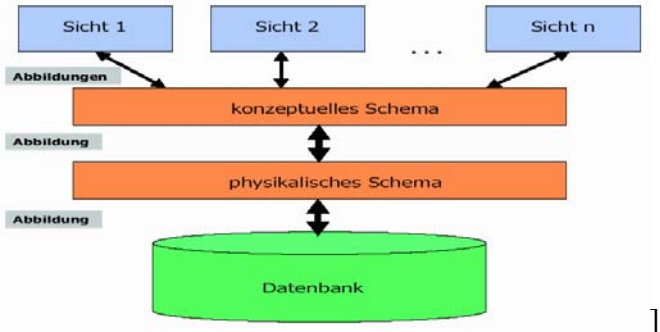
1. **Was ist eine Datenbank?** [Sie besteht aus: Relationen, Beziehungen, Objekte, Tupel]
2. **Was ist ein Objekt?** [Spalten einer Tabelle, also z:b. Vorname der Relation Student]
3. **Was ist eine Relation?** [Eine Tabelle mit Objekten ➔ Entität]
4. **Was ist eine Beziehung?** [Verknüpfung zwischen 2 Relationen z.B. 1:1 , 1:n etc.]
5. **Was ist eine Anfrage?** [Ein mengenwertiger, deklarativer Ausdruck zur Extraktion von Daten]
6. **Was ist eine Transaktion? Beispiele?** [Eine Aktionenfolge die etwas bestimmtes auf der Datenbank ausführt, z.B. Erstelle für jeden Studierenden eine Lister der belegten Kurse]
7. **Welche Eigenschaften müssen Transaktionen haben?** [ACID ➔ Atomizität (➔ Eine nicht zerlegbare Einheit, entweder muss die Transaktion vollständig durchgeführt werden, oder gar nicht) , Konsistenz (➔ Die Datenbank muss zu jedem Zeitpunkt, d.h. vor und nach der Transaktion konsistent sein. Zwischenzustände dürfen inkonsistent sein), Isoliertheit (➔ Verzahnte Ausführungen von Transaktionen sollen dem Benutzer verborgen bleiben und so wirken als würden sie seriell ausgeführt (Simulierter Einbenutzerbetrieb), Dauerhaftigkeit(➔Persistenz: Wenn eine Transaktion zum Ende gekommen ist, dann sind die Änderungen auch von Dauer)]
   a. **Wie können diese Eigenschaften erreicht werden?** [Mit einem Datenbankmanagementsystem (DBMS)]
8. **Was ist denn ein Datenbankmanagementsystem (DBMS)?** [Eine Menge von (Teil-) Programmen, die auf einer Datenbank die entsprechende Transaktionen ermöglichen. Der Zugriff er folgt nur über bestimmte Schnittstellen. Sie repräsentieren ein Modell der entsprechenden Miniwelt]
9. **Wie ist die Basisarchitektur eines kompletten Datenbanksystems? Skizze?** [Ein DBS besteht aus einer Datenbank, dem aufgesetzten DBMS und den jeweiligen Transaktionen:



10. **Was sind die charakteristischen Eigenschaften eines DBS?** [Integrierte Verealtung persistenter Daten (Integritätserhaltung) | Verarbeitung der Daten und Ihre Abspeicherung sind weitgehend getrennt | Deklarative, mengenorientierte Anfragesprache (SQL) | kontrollierter Mehrbenutzerbetrieb | Datensicherheit im Fehlerfall | Sicherheitsmechanismen]
11. **Wo werden Datenbanken eingesetzt?** [Wirtschaft, Verwaltung, Technik, ➔ Überall dort, wo persitente Datenmengen mit einem hzohen Qualitätsanspruch verarbeitet werden müssen]
12. **Was ist ein Datenmodell? Für was brauchen wir das?** [Eine Abstraktion zur Repräsentation von Daten (z.B. Dateien) ➔ Direktes Arbeiten auf den Daten hat Probleme wie das Y2K-Problem zur Folge, da die Verarbeitung der Daten hier direkt etwas mit dem Format der gespeicherten Daten zu tun hatte]
13. **Was versteht man dann unter dem „Relationalen Datenmodell“?** [Man versteht darunter die Übertragung von reinen Daten in passende Relationen]
14. **Welche 3 Abstraktionsebenen werden dabei unterschieden (Schema)?** [physikalische/konzeptuelle/externe]
    a. **Erkläre diese einzelnen Ebenen!**
       physikalische Ebene: [Es wird festgelegt wo und wie die Daten auf dem Speichermedium physisch abgelegt werden und welche Hilfstrukturen zur Effizienzsteigerung zur Verfügung gestellt werden sollen]
       konzeptuelle Ebene: Setzt auf der physikalischen Ebene auf. Die Datentypen werden unabhängig von den Details einer physischen Implementierung definiert. Ziel ist eine möglichst adäquate Repräsentation der relevanten Objekte der Miniwelt.
       Externe Ebene: Für einzelne Benutzergruppen werden spezielle für deren Belange geeignete Datentypen definiert. Es werden Sichten auf das konzeptuelle Schema

definiert. Die Sichten basieren auf den Datentypen des konzeptuellen Schemas]

15. **Wie sieht die Architektur nach ANSI/SPARC aus?**



[ ]

16. **Welchen Vorteil bringt die Trennung zwischen physischer und logischer Datenunabhängigkeit?** [Die Anwendungen arbeiten nicht direkt auf den Daten sondern auf den Datentypen des konzeptuellen Schemas. Die Abbildung auf die Daten geschieht über das DBMS. Physische und konzeptionelle Änderungen sind kein Problem, da das DBMS ja als Schnittstelle fungiert und die Daten in der richtigen Form an die Anwendung weitergibt]
17. **Was versteht man unter DDL, SDL, DML und Integritätsbedingungen?** [
DDL➔ Data Definition Language: Definitionssprache für konzeptuelle und externe Schemata
SDL ➔ Speicherdefinitionssprache: Teil von DDL beeinflusst die physische Speicherung zur Performancesteigerung
DML ➔ Datamanipulationssprache: z.B. SQL]
    a. **Was sind dabei statische bzw. dynamische Integritätsbedingungen?** [Statische Bedingungen überprüfen einen festen Zustand (Gehalt kleiner 5000€) | Dynamische Bedingungen betreffen einen Zustandsübergang (Gehalt darf nur um 100€ ansteigen]
    b. **Was sind strukturelle bzw. semantische Integritätsbedingungen?** [
    Strukturelle Bedingungen wie Typ Bedingungen, Schlüsselbedingungen und Fremdschlüsselbedingungen |
    Semantische Bedingungen: Bedingt durch die konkreten Anwendungen. Sie werden mittels logischer Ausdrücke definiert]
18. **Was sind Relationsbezeichner, Attribute und Wertebereiche im Relationsschema?** [
Relationsbezeichner: Der Name der Relation |
Attribute: Die Menge von Eigenschaften mit nichtleerem Wertebereich]
19. **Was ist eine Instanz eines Schemas?** [Eine Belegung des Schemas]
20. **Was ist Relationenalgebra?** [Eine Anfragesprache auf Datenbanken mit bestimmten, abgeschlossenen Operationen auf die Mengen, d.h. Relationen der Datenbank. Datenbanken arbeiten intern mit Relationenalgebra und sind Gegenstand zahlreicher Optimierungen. Die Relationenalgebra ist stark prozedural, d.h. man sagt der Datenbank wie man etwas haben möchte]
21. **Geben Sie ein Beispiel für eine Projektion in Relationenalgebra an!** [Selektion von Attributen

Sei $r \subseteq \mathsf{Tup}(X)$ und $Z \subseteq X$.

$$\pi[Z]r = \{\mu \in \mathsf{Tup}(Z) \mid \exists \mu' \in r, \text{ so dass } \mu = \mu'[Z]\}.$$

Beispiel:

$$r = \begin{array}{ccc} A & B & C \\ \hline a & b & c \\ a & a & c \\ c & b & d \end{array} \qquad \pi[A,C](r) = \begin{array}{cc} A & C \\ \hline a & c \\ c & d \end{array}$$

]

22. **Geben Sie ein Beispiel für eine Selektion in Relationenalgebra an!** [Filter auf Zeilen

Sei $r \subseteq \mathrm{Tup}(X)$ und α eine Selektionsbedingung zu X.

$$\sigma[\alpha]r = \{\mu \in \mathrm{Tup}(X) \mid \mu \in r \wedge \mu \text{ erfüllt } \alpha\}.$$

Beispiel:

$$r = \begin{array}{|ccc|}\hline A & B & C \\ \hline a & b & c \\ d & a & f \\ c & b & d \\ \hline \end{array} \qquad \sigma[B = b](r) = \begin{array}{|ccc|}\hline A & B & C \\ \hline a & b & c \\ c & b & d \\ \hline \end{array}$$

23. **Geben Sie ein Beispiel für eine Vereinigung in Relationenalgebra an!** [

$$r \cup s = \{\mu \in \mathrm{Tup}(X) \mid \mu \in r \vee \mu \in s\}.$$

$$r = \begin{array}{|ccc|}\hline A & B & C \\ \hline a & b & c \\ d & a & f \\ c & b & d \\ \hline \end{array} \qquad s = \begin{array}{|ccc|}\hline A & B & C \\ \hline b & g & a \\ d & a & f \\ \hline \end{array} \qquad r \cup s = \begin{array}{|ccc|}\hline A & B & C \\ \hline a & b & c \\ d & a & f \\ c & b & d \\ b & g & a \\ \hline \end{array}$$

]

24. **Geben Sie ein Beispiel für eine Differenz in Relationenalgebra an!** [

$$r - s = \{\mu \in \mathrm{Tup}(X) \mid \mu \in r, \text{ wobei } \mu \notin s\}.$$

$$r = \begin{array}{|ccc|}\hline A & B & C \\ \hline a & b & c \\ d & a & f \\ c & b & d \\ \hline \end{array} \qquad s = \begin{array}{|ccc|}\hline A & B & C \\ \hline b & g & a \\ d & a & f \\ \hline \end{array} \qquad r - s = \begin{array}{|ccc|}\hline A & B & C \\ \hline a & b & c \\ c & b & d \\ \hline \end{array}$$

]

25. **Geben Sie ein Beispiel für einen natürlichen Verbund (Join) in Relationenalgebra an!**

XY sei im folgenden eine Kurzschreibweise für $X \cup Y$.

Seien X, Y beliebig und seien weiter $r \subseteq \mathrm{Tup}(X), s \subseteq \mathrm{Tup}(Y)$.

$$r \bowtie s = \{\mu \in \mathrm{Tup}(XY) \mid \mu[X] \in r \wedge \mu[Y] \in s\}.$$

Beispiel:

$$r = \begin{array}{|ccc|}\hline A & B & C \\ \hline 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 6 \\ \hline \end{array} \qquad s = \begin{array}{|cc|}\hline C & D \\ \hline 3 & 1 \\ 6 & 2 \\ 4 & 5 \\ \hline \end{array} \qquad r \bowtie s = \begin{array}{|cccc|}\hline A & B & C & D \\ \hline 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 2 \\ 7 & 8 & 6 & 2 \\ \hline \end{array}$$

[ ]

26. **Geben Sie ein Beispiel für eine Umbenennung in Relationenalgebra an!** [

Sei $X = \{A_1, \ldots, A_k\}$, $Y = \{B_1, \ldots, B_k\}$ und $\mathrm{dom}(A_i) = \mathrm{dom}(B_i)$, $1 \leq i \leq k$.

Sei $r \subseteq \mathrm{Tup}(X)$.

$$[X \to Y]r = \{\mu \in \mathrm{Tup}(Y) \mid \exists \mu' \in r, \text{so dass } \mu(B_i) = \mu'(A_i), 1 \leq i \leq k\}$$

Beispiel: $X = \{A, B, C\}, Y = \{D, E, C\}$.

$$r = \begin{array}{|ccc|}\hline A & B & C \\ \hline a & b & c \\ d & a & f \\ c & b & d \\ \hline \end{array} \qquad [X \to Y]r = \begin{array}{|ccc|}\hline D & E & C \\ \hline a & b & c \\ d & a & f \\ c & b & d \\ \hline \end{array}$$

]

27. **Wie sehen die Basisoperatoren für diese Operationen aus?** [

Vereinigung $\cup$

Differenz $-$

Projektion π

Selektion σ

Verbund $\bowtie$

Umbenennung $[\ldots]$

]

28. **Welche weiteren Operationen kann man durch Kombination der Basisoperatoren erreichen?** [

Durchschnitt: Sei $X_1 = X_2$.

$$r_1 \cap r_2 = r_1 - (r_1 - r_2).$$

Produkt: Sei $X_1 \cap X_2 = \emptyset$.

$$r_1 \times r_2 = r_1 \bowtie r_2.$$

Allgemeiner natürlicher Verbund: $\bowtie_{i=1}^{n} r_i = \{\mu \in \mathrm{Tup}(\cup_{i=1}^{n} X_i) \mid \mu[X_i] \in r_i\}.$

θ-Verbund: Sei $X_1 \cap X_2 = \emptyset$ und sei α eine beliebige Selektionsbedingung über $X_1 \cup X_2$.

$$r \bowtie_\alpha s = \sigma[\alpha](r \times s).$$

Equi-Verbund: Wenn α nur Gleicheitsvergleiche enthält

Division: Sei $Y \subset X$, $Z = X - Y$ und weiter $s \neq \emptyset$.

$$r \div s = \{\mu \in \mathrm{Tup}(Z) \mid \{\mu\} \times s \subseteq r\} = \pi[Z]r - \pi[Z]((\pi[Z]r \times s) - r).$$

Beispiel:

$$r = \begin{array}{cccc} A & B & C & D \\ \hline a & b & c & d \\ a & b & e & f \\ b & c & e & f \\ e & d & c & d \\ e & d & e & f \\ a & b & d & d \end{array} \qquad s = \begin{array}{cc} C & D \\ \hline c & d \\ e & f \end{array} \qquad r \div s = \begin{array}{cc} A & B \\ \hline a & b \\ e & d \end{array}$$

]

29. **Wie kann man die Relationenalgebra als Anfragesprache betrachten?** [Man ersetzt in den Asudrücke die Relationen durch entsprechende Relationsbezeichner bzw. Konstanten]
30. **Was ist ein Relationsbezeichner, bzw. die Relationskonstante in der Relationenalgebra?** [Ein Relationsbezeichner ist ein Ausdruck der Relationenalgebra, Mit ihm kann man bei einer Anfrage einen Wert übergeben der angefragt wird. | Eine Relationskonstante ist auch ein Ausdruck der Relationenalgebra. Es ist eine Belegung eines Attributes A]
31. **Welche Ausdrücke sind in Relationenalgebra gültig?** [Atomare Ausdrücke und diejenigen die durch Vereinigung, Differenz, Projektion, Selektion, Verbund und Umbenennung entstehen können]
32. **Wann ist eine Anfragesprache relational vollständig? Was bedeutet das?** [Wenn sie mindestenes, die Mächtigkeit der Algebra besitzt]
33. **Wann ist eine Anfragesprache Turing vollständig?** [Wenn man die Anfrage auf einer Turing-Maschine berechnen kann]
    a. **Treffen diese beiden Bedingungen auf die Relationenalgebra zu?** [Nein (?)]
34. **Wann sind zwei Ausdrücke der Relationenalgebra äquivalent?** [Wenn die beiden Ausdrücke für jede beliebige im Bezug auf die Datenbank mögliche Belegung gleich sind]
    a. **Für was wird dies benötigt?** [Grundlage für Optimierung]

b. **Nennen Sie einige wichtige äquivalenzerhaltende Umformungen!** [

- $Z_1 \subseteq Z_2 \subseteq X \Longrightarrow \pi[Z_1](\pi[Z_2]R) \equiv \pi[Z_1 \cap Z_2]R.$
- $\text{attr}(\alpha) \subseteq Z_1 \subseteq X \Longrightarrow \pi[Z_1](\sigma[\alpha]R) \equiv \sigma[\alpha](\pi[Z_1]R).$
- $R \bowtie S \equiv S \bowtie R.$
- $R \bowtie R \equiv R.$
- $(R \bowtie S) \bowtie T \equiv R \bowtie (S \bowtie T).$
- $\text{attr}(\alpha) \subseteq X, \text{attr}(\alpha) \cap Y = \emptyset \Longrightarrow \sigma[\alpha](R \bowtie S) \equiv (\sigma[\alpha]R) \bowtie S.$

]

35. **Was ist denn das Relationenkalkül?** [Das Relationenkaklül ist im Gegensatz zur Relationenalgebra eher deklarativ, d.h. man sagt nur was man haben möchte und nicht wie man da hin kommt. Das RK ist eine Ableitung des Prädikatenkalküls 1. Ordnung ➔ Ein Relationenkalkül legt fest, wie aus einer Menge von vorhandenen Daten (Inhalten einer Datenbank) in Form von Relationenschemata und darauf basierenden Relationenausprägungen weitere Daten in Form von Relationen syntaktisch abgeleitet werden können. Die Menge von Regeln, die beschreibt, wie diese Umformung der vorhandenen Daten(Relationen) aussehen darf, ist ein Relationenkalkül.]
    a. **Wie werden die Formeln gebildet (Syntax)?** [Die R-Formeln werden aus Konstanten, Variablen, Relationsbezeichnern, Junktoren, Quantoren und Hilfszeichen wie bei den Formeln des Prädikatenkalküls gebildet]
    b. **Was ist neu im Gegensatz zum Prädikatenkalkül?** [Gleichheits und Ungleichheitsoperator]
36. **Wie sieht eine einfache Anfrage im Relationenkalkül aus?** [3 Beispielanfragen:

$$r = \begin{array}{|cc|}\hline A & B \\ \hline a & a \\ d & a \\ c & b \\ \hline \end{array} \qquad s = \begin{array}{|c|}\hline B \\ \hline b \\ c \\ \hline \end{array}$$

$$\exists Y R(X,Y) \quad \Longrightarrow \quad \begin{array}{|c|}\hline A \\ \hline a \\ d \\ c \\ \hline \end{array}$$

$$R(X,Y) \wedge X = Y \quad \Longrightarrow \quad \begin{array}{|cc|}\hline A & B \\ \hline a & a \\ \hline \end{array} \qquad R(X,Y) \wedge S(Y) \quad \Longrightarrow \quad \begin{array}{|cc|}\hline A & B \\ \hline c & b \\ \hline \end{array}$$

]

37. Wie werden die folgenden Abfragen im Relationenkalkül formuliert:
    a. **Projektion? Selektion? Verbund? Vereinigung? Differenz? Division?** [

Seien Relationsschemata $R(A,B)$ und $S(B)$ gegeben.

Projektion $\pi[A]R$:	$\{X \mid \exists Y R(X,Y)\}$
Selektion $\sigma[A=B]R$:	$\{(X,Y) \mid R(X,Y) \wedge X = Y\}$
Verbund $R \bowtie S$:	$\{(X,Y) \mid R(X,Y) \wedge S(Y)\}$
Vereinigung $R \cup (\{1\} \times S)$:	$\{(X,Y) \mid R(X,Y) \vee (X = 1 \wedge S(Y))\}$
Differenz $R - (\{1\} \times S)$:	$\{(X,Y) \mid R(X,Y) \wedge \neg(X = 1 \wedge S(Y))\}$
Division $R \div S$:	$\{X \mid \forall Y(S(Y) \Rightarrow R(X,Y))\}$

]

38. **Nenne eine Anfrage, bei der die Antwortmenge nicht endlich ist?** [z.B. Wenn man eine Anfrag auf alle Elemente stellt, die nicht in einer Relation sind: {X | ¬R(X)} ➔ Man darf also um dies zu verhindern nur Anfragen stellen, deren Resultat von den Relationen der betrachteten Instanz sind]
39. **Wann sind zwei Relationen Q und F wertebereichsunabhängig?** [???]
40. **Wann ist eine Teilformel G von F maximal konjunktiv?** [Eine Teilformel G einer Formel F heisst maximal konjunktiv, wenn F keine konjunktive Teilformel der Form H∧G oder G∧H enthält]
41. **Wann nennt man freie Variablen begrenzt?** [Eine freie Variable ist begrenzt , wenn folgendes gilt: Sei 1≤j≤m

1. Eine Variable ist begrenzt, wenn sie in einer Formel F_j frei ist, wobei F_j weder ein Vergleichausdruck noch negiert ist
2. Falls F_j die Form X=a oder a=X hat, a eine Konstante, dann ist X begrenzt
3. Falls F_j die Form X=Y oder Y=X hat und Y ist begrenzt, dann ist auch X begrenzt]]

42. **Wann ist eine R-Formel (syntaktisch) sicher? [3 Bedingungen] Beispiel?** [
1. Wenn Sie keine Allquantoren besitzt |
2. Wenn $F_1 \vee F_2$ Teilformel von F ist, dann müssen F_1 und F_2 dieselben freien Variablen besitzen
3. Wenn $F_1 \wedge \ldots \wedge F_m \geq 1$, eine maximal konjunktive Teilformel von F ist dann müssen alle freien Variablen begrenzt sein]
43. **Was haben Algebra und sicheres Kalkül gemeinsam?** [Sie sind äquivalent!]
44. **Was ist SQL?** [Die Structured (Standard) Query Language ist eine Anfragesprache auf Datenbanken]
45. **Wie sieht die Grundstruktur eines typischen SQL Ausdruckes aus?** []
46. Wie sieht der Ausdruck in der Relationenalgebra aus?

SELECT $A_1, \ldots, A_n$ (... entspricht π in der Algebra)
FROM $R_1, \ldots, R_m$ (... gibt die betreffenden Relationen an)
WHERE F (... entspricht σ in der Algebra)

... ist äquivalent zu dem Algebraausdruck: $\pi[A_1, \ldots, A_n](\sigma[F](r_1 \times \ldots \times r_m))$

47. **Welche Teileausdrücke korrespondieren dabei miteinander?** [Select entspricht der Projektion in Algebra, das From gibt die betreffenden Relationen an und das Where entspricht der Selektion in der Algebra]
48. **Ordne die Begriffe Relation, Tupel, Attribut zu den Begriffen Spalte, Zeile, Tabelle zu!** [Relation ➔ Tabelle | Tupel ➔ Zeile | Attribut ➔ Spalte]
49. **Was sind Aggregierungsfunktion?** [Sie beziehen sich auf eine Spalte einer Relation, es gibt die Operatoren SUM, AVG,MIN,MAX und COUNT]
50. Wie kann man gruppieren? Was macht die Having Klausel? [Gruppiert wird mit z.B. Group by Country. Diese Gruppierung kann man einschränken in dem man z.B. sagt, man will nur diejenigen Länder, in denen die Einwohnerzahl in den Städten kleiner ist als 0.1. Ausgedrückt wird dies durch die HAVING Klausel, also:

```
SELECT country, AVG(City.population)
   FROM City
   GROUP BY country
   HAVING AVG(population) < 0.1;
```

51. **Was besagt die Orthogonalität von SQL Ausdrücken?** [Dass man überall da, wo eiegntlich eine Relation steht auch ein neuer Ausdruck stehen kann (Schachtelung)

```
SELECT COUNT(*)
   FROM ((SELECT name
             FROM City) UNION
         (SELECT name
             FROM Country));
```

]
52. **Wie kann man einen Algebra bzw. Kalkülausdruck in SQL transformieren?** [Algebra ➔ SQL:

Relationsschemata $R(A,B), S(B)$ und $T(A)$ seien gegeben.

$$T = R \div S$$
$$=$$
$$\{X \mid \{X\} \times S \subseteq R\}$$
$$=$$
$$\{X \mid \{X\} \times S - R = \emptyset\}$$
$$=$$
$$\{X \mid (S - \pi[B](\sigma[A = X]R)) = \emptyset\}$$

kann in einen SQL-Ausdruck transformiert werden:

```
SELECT DISTINCT A FROM R as X
   WHERE NOT EXISTS
      ((SELECT DISTINCT B FROM S)
      EXCEPT
      (SELECT DISTINCT B FROM R
         WHERE R.A = X.A));
```

Kalkül in SQL:

Relationsschemata $R(A,B), S(B)$ und $T(A)$ seien gegeben.

$$T = R \div S$$
$$=$$
$$\{X \mid \forall Y(S(Y) \Rightarrow R(X,Y))\}$$
$$=$$
$$\{X \mid \neg\exists Y(S(Y) \wedge \neg R(X,Y))\}$$

kann nach SQL transformiert werden wie folgt:

```
SELECT DISTINCT A FROM R as X
   WHERE NOT EXISTS
      (SELECT * FROM S as Y
         WHERE NOT EXISTS
         (SELECT * FROM R
            WHERE R.A = X.A
             AND R.B = Y.B));
```

```
SELECT DISTINCT A FROM R as X
   WHERE NOT EXISTS
      (SELECT * FROM S as Y
         WHERE X.A NOT IN
          (SELECT A FROM R
             WHERE R.B = Y.B));
```

]

53. **Welches sind die Grundoperationen in SQL zum Erstellen, Löschen, Einfügen, Ändern von Tabellen und zum Erstellen von Sichten?** [Erstellen: CREATE TABLE [NAME] (Attribute mit Wertebereich) | Löschen: DELETE R WHERE P | Einfügen: INSERT INTO R [Attribute] VALUES [Werte] ➔ Man kann auch eine Menge von Tupeln einfügen in dem man z:b.via Select aus einer anderen Relation einfügt. | Ändern: UPDATE R SET K WHERE P | Views: CREATE VIEW V AS <E> [WITH CHECK OPTION]]
54. **Welche Probleme treten bei NULLwerten in Tabellen auf?** [Die Frage ist, ob der Wert einfach nur unbekannt war, oder ob das Attribut nicht anwendbar war ➔ Man kann mit den Operatoren IS NULL und IS NOT NULL auf Existenz von Nullwerten prüfen]
    a. **wie werden diese behandelt?** [Bei skalare Ausdrücken ist das Ergebnis NULL wenn einer der Operanden NULL ist | Aggregierunbsfunktionen ignorieren NULL mit Ausnahme von COUNT(*)]
    b. **Bei Vergleichsoperatoren bzw. bei logischen Verknüpfungen**? [Man verwendet dann ein dreiwertige Logik. Die Wahrheitstabellen sehen dann folgendermaßen aus:

AND	t	u	f
t	t	u	f
u	u	u	f
f	f	f	f

OR	t	u	f
t	t	t	t
u	t	u	u
f	t	u	f

NOT	
t	f
u	u
f	t

]

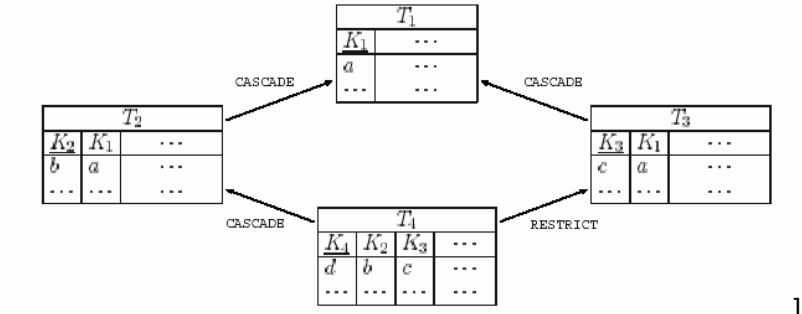
55. **Für was kann man Sichten in SQL gebrauchen?** [Man kann die Daten damit in eine bestimmte Form bringe und Teile abgrenzen von der Sicht die entweder nicht relevant sind oder dem jeweiligen Benutzer verborgen bleiben soll]
    a. **Nenne Vor- und Nachteile von virtuellen und materialisierten Sichten!** [virtuelle Sichten: Vorteile: werden nicht beeinträchtigt durch Änderungen in der Datenbank | Nachteil: Hoher Rechenaufwand | Materalisierte Sicht: Vorteil: Schneller Zugriff auf die Sichten, Nachteil: Hoher Aufwand bei Änderungen von Basisrelationen, viel mehr Speicher (Redundanz) und eine Indexstruktur]
    b. **Welche Änderungen auf Sichten sind erlaubt?** [Bei existierender Transformation (Zwischen Sicht und zugehörigen Relationen)

   c. **Auf welches Tabellenattribut sollte man bei einer Projektionssicht nicht verzichten?** [Primärschlüssel]

56. **Was bewirkt die Angabe der Klausel „With Check Option"?** [Bei einem Update auf einen View kann es sein, dass die Bedingungen, nach der der View generierte worden ist, auf dem aktualisierten Tupel nicht mehr gelten. WITH CHECK OPTION gibt dann an, ob das überprüft werden soll ➔ z.B. View mit „alle Länder größer als 100 Einwohner", Update auf den View mit „Population eines bestimmten Landes auf 90]
57. **Datenbankanfragesprachen, wie SQL, sind deklarativ, mengenorientiert, aber nicht turing-vollständig. Warum hat man bewusst auf diese Eigenschaft verzichtet?** [Um einer automatischen Optimierung zugänglich zu sein.]
58. **Wie kann man die Turing Vollständigkeit jedoch trotzdem erreichen?** [Einbettung in Hostsprache (C,C++,Java,etc) oder über Aufrufschnittstellen]
59. Was versteht man unter Impedance Mismatch? [SQL ist Mengenorientiert, Programmiersprachen sind Satzorientiert]
    a. **Welches Konzept gibt es diesen zu lösen?** [Cursor Konzept: Man gibt die einzeln Datensätze der angefragten Menge der Reihe nach aus. Ein „Cursor" läuft dann von Satz zu Satz]
    b. **Was ist der Nachteil dieses Konzepts?** [Wenn ein Tupel z.B. mehrmals benötigt wird, muss man mehrere Cursor laufen lassen oder wenn die Tupel nicht sequentiell benötigt werden]
60. **Wann redet man von dynamischem SQL?** [Wenn die Anfrage erst während der Laufzeit des Programmes in Abhängigkeit von z.B. einer bestimmten Eingabe generiert werden können]
61. **Wir haben bereits von Integritätsbedingungen gesprochen, auf welche Bereiche der Datenbank kann man diese definieren?** [Wertebereichsbedingung: domain constraints | Allgemeine Bedingungen: general constraints | Basistabellenbedingung: base table constraints | Spaltenbedingungen: column constraints ➔z.B. NOT NULL, PRIMARY KEY, UNIQUE]
    a. **Wo werden die einzelnen Bedingungen deklariert, wann werden sie überprüft und wie sieht es mit der Gültigkeit aus?** [Basistabellen- und Spaltenbedingungen sind Teil der CREATE TABLE Klausel, Die allgemeinen Constraints kann man unabhängig mit einem CREATE ASSERTION definieren]
    b. **Wie lautet das Konstrukt, eine solche Bedingung in SQL auszudrücken**? [CREATE ASSERTION [Name] CHECK … | CREATE TABLE [NAME] CHECK …]
62. **Was versteht man unter referentieller Integrität?** [Wenn man eine Referenz (mit Foreign Key) auf eine andere Relation hat und in dieser Relation etwas gelöscht wird, auf was die eine Relation eigentlich referenziert, dann ist die Ref. Integrität verletzt]
63. **Was passiert bei Zyklischen Fremdschlüsselbedingungen?** [Das geht, aber nicht direkt, denn die Basistabelle zu einem Fremdschlüssel muss immer existieren. Man muss also zunächst eine Tabelle erzeugen und diese im nachhinein mit ALTER TABLE verändern]
64. **Welche möglichen Zeitpunkte gibt es die Bedingungen zu überprüfen? Wie heißen die Konstrukte?** [IMMEDIATE ➔ Direkt nach dem jeweiligen SQL Ausdruck DEFERED ➔ Am Ende der Transaktion. Eine Bedingung kann als „DEFERABLE" oder „NOT DEFERABLE" definiert werden.Bei einer Transaktion steht immer ein „INITIALLY" dabei, welches angibt ob DEFERED ODER IMMIDIATE verwendet wird. Default ist IMMIDIATE. Während der Ausführung kann die Bedingung noch über SET CONSTRAINT geändert werden]
65. **Was sind dynamische Integritätsbedingungen?** [Definieren zulässige Zustandsübergänge wie z.B. „Gehälter dürfen nicht fallen"]
66. **Was passiert wenn in der C-Tabelle (Child) etwas geändert oder gelöscht wird?** [Wenn die referentille Integrität nicht verletzt wird, ist das okay, ansonsten wird zurückgesetzt bz.w nicht ausgeführt]
    a. **Welche Möglichkeiten der Kompensation hat die Integritätsbedingung nun?** [Man kann der referentiellen Aktion sagen wie sie verfahren soll: ON DELETE { NO ACTION | RESRICT | CASCADE | SET DEFAULT | SET NULL} , Gleiches gilt für ON UPDATE ➔ Default ist „NO ACTION"]
67. **Weshalb ist ein Insert auf einer P-Tabelle unkritisch, wogegen es auf einer C-Tabelle Probleme bereitet?** [Die P-Tabelle enthält z.B. Stamminformationen, die C-Tabelle erweiterte Informationen eines Datensatzes. Man darf nun nicht erweiterte Informationen

einfügen bzw. einen vorhandenen Primärschlüssel ändern zu denen es keine Stamminformationen gibt]

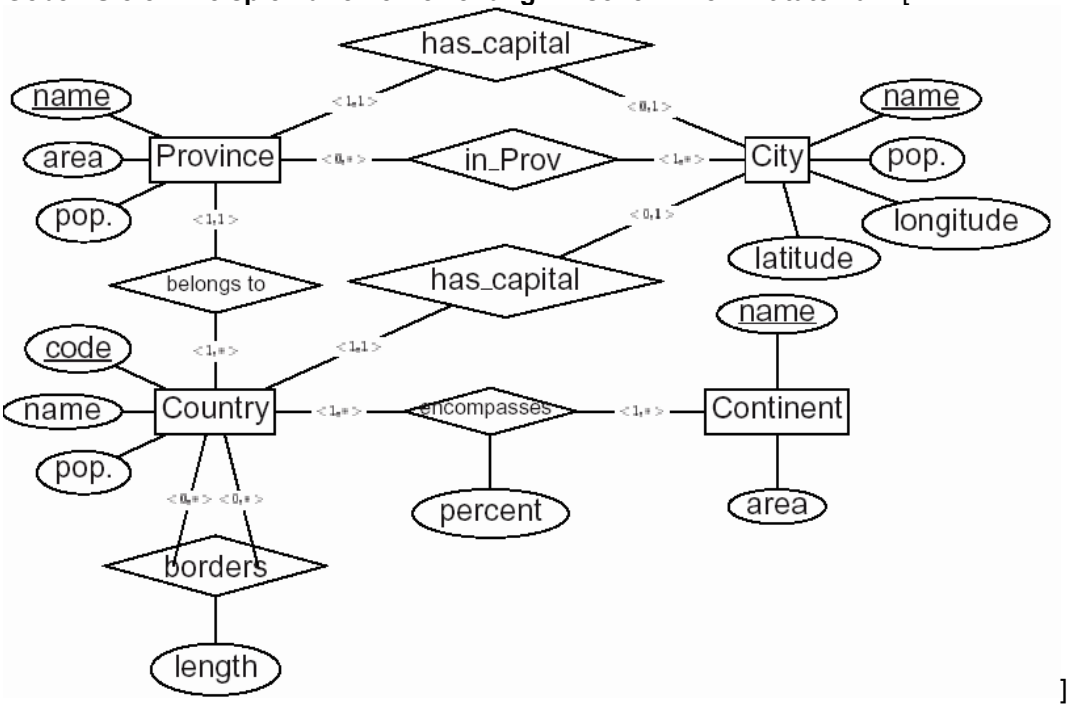
68. **Wie sieht es mit einem Delete/Update auf den beiden Tabellen aus?** [Delete im C-Table ist kein Problem und ein. Bei der P-Tabelle darf man nicht einfach löschen ohne nicht auch die referenzierten Informationen zu löschen. Ein Update darf auf P- und C-Tabellen nur geschehen, wenn der Primärschlüssel nicht verändert wird]
69. **Wie kann es passieren, dass durch die Aktionen der Integritätsbedingungen mehrdeutige Ergebnisse entstehen?** [Die Abarbeitungsreihenfolge liefert unterschiedliche Ergebnisse. T1 wird gelöscht 1. dann folgt T3 vor T4 vor T2 ➔ T1 und T3 werden zwar gelöscht, aber durch das Restrict wiederhergestellt | 2. dann folgt T2 vor T4 vor T3 bewirkt, dass alle gelöscht werden (3x Cascade)]
    a. **Wie kann man das verhindern?** [Man ersetze das RESTRICT durch NO ACTION]



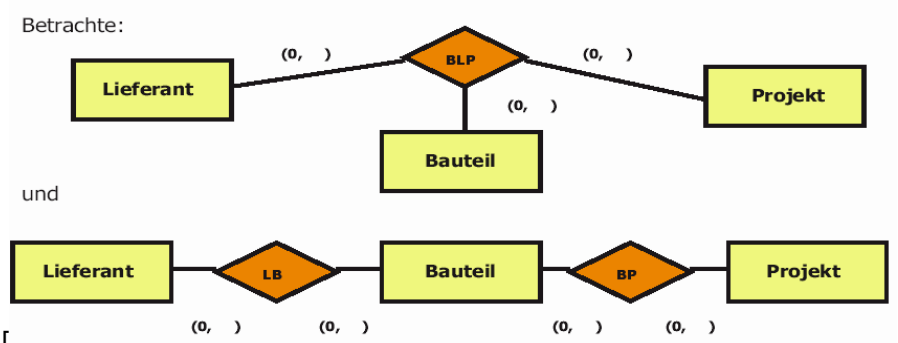
]

70. **Was sind Trigger?** [Trigger sind ein Spezialfall aktiver Regeln ➔ EventConditionAction-Paradigma, d.h. unter bestimmten Events werden bestimmte Aktionen „getriggert“. Man könnte z.B. einen Trigger setzen, der bei jeder Gehaltserhöhung noch einen einmaligen Bonus von 100€ dazurechnet]
71. **Welche verschiedene Sorten von Triggern gibt es?** [Before: Vor Ausführung der Aktion wird der Trigger ausgfeührt |After: Nach Ausführen er Aktion | InsteadOf: Die Aktion wird gar nicht ausgeführt, sondern nur der Trigger]
72. **Wie können die Trigger auf die jeweils alten bzw. neuen Werte in einer Tabelle zugreifen, wenn ein Insert/Delete/Update ausgeführt wurde?** [Bei Insert ist mit einem Before und Instead of Trigger ja noch nichts gespeichert, man kann aber mit dem Befehl NEW bzw. NEW Table auf die zukünftige Information zugreifen. Beim Aftertrigger ist die Änderung schon materialisiert und kann direkt aus der Tabelle gelesen werden | Beim Delet ist das genau umgekehrt wie beim Insert, man kann nun beim Aftertrigger mit OLD bzw. OLD TABLE auf die bereits gelöschten Daten zugreifen | Beim Update betrifft das alle Trigger, diese können dann jeweils mit NEW/OLD TABLE auf die benötigten Daten zugreifen]
73. **Auf welchen 3 Schritten baut der Datenbankentwurf auf?**
[Anforderungsanalyse/konzept. Entw./Implementationsentwurf]
74. **Welche Bestandteile hat die Anforderungsanalyse?** [Identifikation von Organisationseinheiten | Identifikation der zu unterstützenden Aufgaben | Anforderungs-Sammelplan: Ermittlung der zu befragenden Personen | Anforderungssammlung | Filterung: gesammelte Information auf Verständlichkeit und Eindeutigkeit überprüfen | Satzklassifikation: Information wird Objekten, Beziehungen zwischen Objekten, Operationen und Ereignissen zugeordnet | Formalisierung bzw. Systematisierung: Übertragung auf Verzeichnisse, die in ihrer Gesamtheit das Pflichtenheft repräsentieren ➔ Bei sehr großen Anwendungen macht man die Anforderungsanalyse typischerweise für die Organisationseinheiten getrennt]
75. **Was ist der konzeptuelle Entwurf?** [Strukturierung der Informationsanforderungen einer Miniwelt, unabhängig von den konkreten Anwendersichten. Der Entwurf wird typischerweise unter Verwendung des ER-Modells vorgenommen. Das Resultat ist das Konzeptuelle Schema]
76. **Was ist der Implementationsentwurf?** [Definition der Zusammenhänge eines konzeptuellen Schemas mit den Konzepten des zum Einsatz kommenden Datenbanksystems. Typischerweise wird ein relationales DBS verwendet. Das Resultat ist das logische Schema. Im Unterschied zum konzeptuellen Schema, das als Spezifikationesdokument betrachtet werden kann, ist das logische Schema Teil des DBS]

77. **Im konzeptuellen Entwurf wird typischerweise ein so genanntes ER Modell entwickelt, was wird dabei gemacht?** [Es werdem Emtitäten, Entitätstypen, Beziehungen und Beziehungstypen generiert]
    a. **Entitäten?** [Sind wohl unterscheidbare physisch oder gedanklich existierende Objekte der zu modellierenden Miniwelt]
    b. **Entitätstypen?** [Gleichartige Entitäten werden zu Entitätstypen abstrahiert]
    c. **Berziehungen?** [Entitäten können zueinander in wohlunterscheidbaren Beziehungen stehen]
    d. **Beziehungstypen?** [Gleichartige Beziehungen werden zu Beziehungstypen abstrahiert]
78. **Welche beiden wesentlichen Strukturierungskonzepte gibt es im ER Schema?** [Entitätstypen➔ Attribute mit Schlüsseln/Beziehungstypen ➔Beziehung v eines Beziehungstyps B ist definiert durch die beteiligten Entitäten gemäß den B zugeordneten Rollen; Sie besitzt zu jedem Attribut genau einen Wert]
79. **Geben Sie ein Beispiel für eine Beziehung zwischen zwei Entitäten an!** [

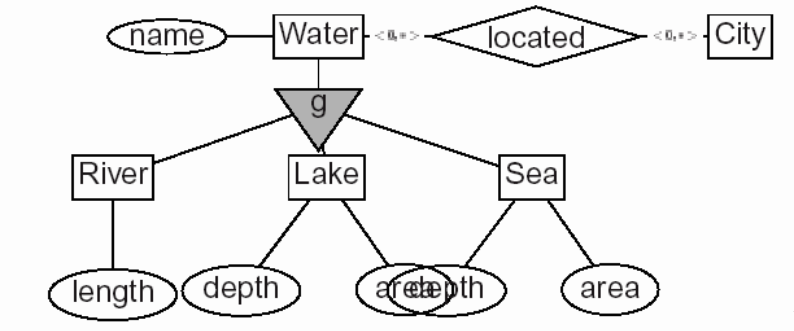


]
80. **Was sind Beziehungskomplexitäten?** [Eine Beziehungskomplexität ist eine einem Beziehungstyp zugeordnete Integritätsbedingung. Mit ihrer Festlegung wird die Mindest- und Maximalzahl von Beziehungen ausgedrückt, in denen eine Entität eines Typs unter einer bestimmten Rolle in einer Beziehungsmenge beteiligt sein darf ➔ Allgemein (min,max), wenn min>0 dann redet man von einem Participation-Constraint. Falls max=1 redet man von einem Key Constraint. Typische Beziehungen sind (1:1), (1:n), (n:m)]
81. **Nehmen wir an, es hängen 3 Entitäten an einer Relation. Kann man diese „3er Konstellation“ auflösen, welche Probleme ergeben sich dabei?**



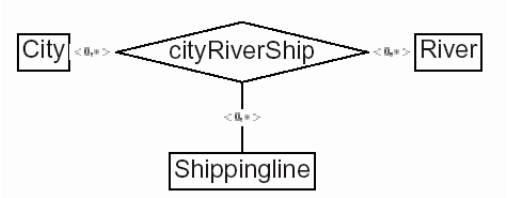
[U.U. wichtige Informationen werden eliminiert. Mit der binären Beziehung kann man z.B. nicht mehr ausdrücken, dass ein Lieferant nur Bauteile für ein bestimmtes Projekt liefert, die Dreierbeziehung drückt dies korrekt aus]

82. **Was ist ein schwacher Entitätstyp?** [Eine Entität ohne kompletten eigenen Schlüssel]. ➔ Schwache Entitätstypen müssen mit einem starken Entitätstyp in einer n:1-Beziehung stehen. Sie müssen einen lokalen Schlüssel besitzen, d.h. Attribute, die erweitert um den Schlüssel des betreffende (starken) Entitätstyps einen Schlüssel des schwachen Entitätstypes ergeben. (Schlüsselvererbung).
83. **Was versteht man unter Generalisierung im ER Modell?** [Eine hierarchische Anordnung der Entitäten ➔ z.B. Fluss, See, Meer sind alles Gewässer und werden zu Gewässer generalisiert.

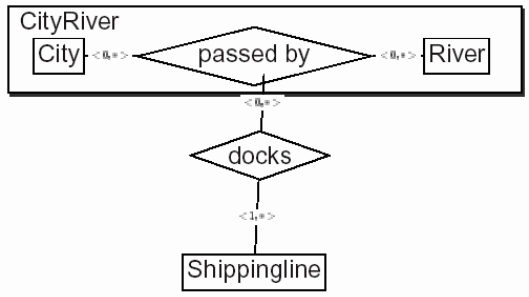


]

84. **Wie sieht diese Generalisierung aus?** [Jede Entität eine Untertyps ist auch Entität des Obertyps. Als Konsequenz sind die Attribute und Beziehungen des Obertyps auch anwendbar auf die Untertypen (Vererbung)
    a. **Was für Integritätsbedingungen gibt es zwischen den Mengen?** [ISA (Is-a) ist erfüllt, wenn die betreffenden Entitätsmengen der Untertypen Teilmenge der Entitätsmenge des Obertyps sind | Disjunktheit der Untertypen | Überdeckung des Obertyps durch die Untertypen]
85. **Was ist ein Aggregattyp? Wie wird diese im ER Diagramm dargestellt?** [Wir haben möglicherweise folgende 3er Beziehung

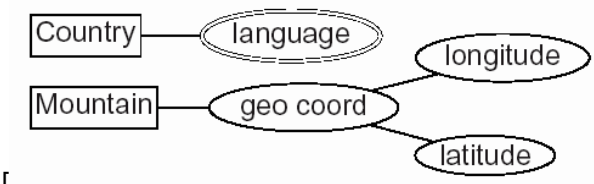


Was passiert aber, wenn es eine Stadt mit Fluss gibt, die gar keine Schifffahrtslinie besitzt? Das geht mit dieser 3er Beziehung nicht. Wir müssen also einen Aggregattyp verwenden, bei dem diese Information im Prinzip optional ist:



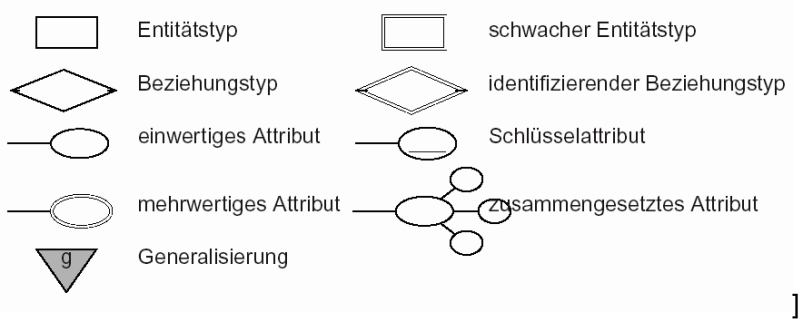
Der Schlüssel des Aggregatstyps ergibt sich aus den Primärschlüsseln der beteiligten Entitätstypen]

86. **Was sind mengenwertige und strukturierte Attribute?**



[
Mengenwertige Attribute sind z.B. Länder mit „Sprachen“ als Attribut. Das Attribut ist eine Menge | Strukturierte Attribute sind z.B. Attribute, die sich aus 2 oder mehreren Werten zusammensetzen wie bei „GEO COORD“ setzt sich zusammen aus „longitude und latitude“]

87. **Geben Sie eine Übersicht der graphischen Notationen eines ER-Diagramms!** [



]

88. **Oft haben wir das Problem, das Informationen redundant gespeichert sind, wie kann man durch Verfeinerungen Redundanz vermeiden?**

[

Lieferung	Name	Adr	Artikel	Menge	Preis
	Meier	MA	a	10	1
	Meier	MA	b	15	1
	Meier	MA	c	20	2
	Müller	KA	b	12	1
	Müller	KA	c	23	2

enthält redundante Informationen (Meier-Mannheim und Müller-Karlsruhe, b-1, c-2) Durch Verfeinerung kann man nun diese Tabelle aufteilen und die redundante Information somit eliminieren:

Kunde	Name	Adr
	Meier	MA
	Müller	KA

Lieferung	Name	Artikel	Menge
	Meier	a	10
	Meier	b	15
	Meier	c	20
	Müller	b	12
	Müller	c	23

Artikel	Artikel	Preis
	a	1
	b	1
	c	2

Wichtig ist es jetzt noch zu testen ob unser verfeinerte Entwurf auch das ist, was wir wollen, d.h. ob die Anomalien beseitigt sind und ob die Schemas äquivalent sind im Bezug auf die Anfrage]

89. **Was ist die Funktionale Abhängigkeit?** [Attribute sind dann funktional abhängig, wenn sich der Inhalt eines oder mehrere Attribute durch die Belegung anderer Attribute ergibt]
    a. **Beispiel?** [Im Beispiel oben gibt es z.B. die FAs Name➔ Adresse und Artikel ➔ Preis]
    b. **Wann nennt man eine FA trivial?** [Wenn A eine Teilmenge von B ist dann gilt immer A➔B | Trivialstes Beispiel wäre Name➔Name]
90. **Was ist Sat(R)?** [Sat)R) ist die Menge aller Relationsschema, die man zu einer vorgegebenen Attributmenge und gegebenen FAs erhalten kann.:

$X = \{A, B, C\}, \Sigma_X = \{A \to B, B \to C, C \to A\}.$

A	B	C
1	1	1
2	2	2
3	3	3
1	2	3
~~3~~	~~2~~	~~1~~
3	1	2

Wir haben auf jeden Fall Instanzen, wenn alle Spalten gleich sind ➔ 111 222 333, es ist aber auch eine Einstanz, wenn in jeder Spalte, jeder Wert nur einmal vorkommt (Zyklische Verschiebung) ➔ 123 312 231]

91. **Was ist die Hülle von F?** [F ist die Menge der FAs | F+ ist nun die Hülle dazu, denn es könnte sich evtl noch weitere FAs daraus ableiten, z.B. bedingt durch Transisivität]

92. **Was ist ein Schlüssel?** [Ein Attribut von welchem alle anderen Attribute funktional abhängig sind. Außerdem muss der Schlüssel minimal sein, d.h. keine Teilmenge des Schlüssels soll eine funktionale Abhängigkeit zu allen andere Attributen ergeben.]
93. **Was ist ein Superschlüssel?** [Wenn X ein Schlüssel ist und Y eine Obermenge des Schlüssel, dann ist Y ein Superschlüssel]
94. **Wann ist ein Attribut (Nicht-)Schlüsselattribut?** [Wenn ein Attribut teil eines Schlüssel ist (SA) bzw. nicht (NSA)]]
95. **Was besagt das Membership Problem?** [Ist eine FA in F+?]
96. **Welches sind die 3 (Basis)-Armstrong-Axiome?** [Die Armstrong Axiome sind ein Ansatz um das Membership Problem zu lösen, d.h.
A1 Reflexivität➔ Die trivialen FAs |
A2 Augmentation ➔ Wenn man zu einer bestehende FA auf beiden Seiten das gleiche Attribut hinzufügt, hat man wieder eine FA |
A3 Transitivität ➔ Wenn es eine Fa von x→y und eine von y→z, dann gibt es auch eine von x→z]
97. **Warum ist die Anwendung der 3 Armstrong Axiome als Lösung für das Membership-Problem nicht sehr effizient?** [Weil man u.U. alle enthaltenen FAs aufzählen muss, deshalb hat er mindestens die Zeitkomplexität $O(2^n)$, wobei n die Anzahl der gegebenen FAs in F ist]
98. **Welches sind die 5 erweiterten Armstrong Axiome?** [
A4 Vereinigung➔X→y und x→z ==> X→YZ |
A5 Pseudotransitivität ➔ X→Y und WY→Z ==> XW→Z |
A6 Dekomposition ➔ X→Y und Z $\subseteq$Y ==> X→Z |
A7 Reflexivität ➔ X→X |
A8 Akkumulation ➔ X→YZ, Z→AW ==> X→YZA (Klar wäre X→YZAW aber es wird noch die Dekomposition angewandt was uns erlaubt das W wegzustreichen]
99. **Wir haben eine einfache Variante des Membership Algorithmus kennen gelernt, welcher auf den Armstrong Axiomen 1-3 aufbaut. Mit Hilfe der Axiome A6-A8 kann man nun einen wesentlich effizienteren Algorithmus (X^+) konstruieren, wie geht dieser?** [Man leitet sich die FAs mit Hilfe vonA6-A8 her und bricht ab, sobald mal die gesuchte gefunden hat, deshalb ist X^+ auch nur eine Teilmenge von F^+]
    a. **Wie schnell ist dieser?** [O(||F||)]
100. **Was kann man noch mit dem X^+ Algorothmus berechnen?** [Man kann in Zeit O(|V| ||F|| ein minimaler Schlüssek für eine Attributmenge berechnen, indem man mit einem Superschlüssel beginnt und sukzessive Attribute streicht. Man streicht solange, bis aus dem Superschlüssel ein Schlüssel geworden ist. Das finden sämtlicher Schlüssel ist im übrigen NP-vollständig]
101. **Um funktionale Abhängigkeiten zu beseitigen zerlegt man die Attributmenge. Wann ist eine solche Zerlegung verlustfrei?** [Die Zusammengehörigkeit der Komponenten der einzelnen Tupel bleibt mittels Verbund rekonstruierbar, ohne das zusätzliche Tupel (quasi als Nebeneffekt) gebildet werden]
102. **Wann ist eine Zerlegung abhängigkeitsbewahrend?** [Die Abhängigkeiten des Relationsschemas bleiben auch bzgl. der Zerlegung (ohne den Verbund zu der Zerlegung zu berechnen) überprüfbar. Die bedeutet insbesondere, dass es bei Einfügungen und Änderungen von Tupeln genügt, die Zerlegung zu betrachten]
103. **Wann sind 2 Mengen von funktionalen Abhängigkeiten äquivalent?** [Wenn die zugehörigen Hüllen gleich sind, also F≡G gdw. $F^+ = G^+$]
104. **Wann ist eine Menge von funktionalen Abhängigkeiten (F) minimal?** [(3 Bedingungen:
Rechtsreduktion➔ X→Y $\in$ F, wenn Y aus einem Attribut besteht |
Linksreduktion➔ X→A $\in$ F wenn F\{X→A} nicht äquivalent zu F ist (d.h. Die Menge ist noch nicht minimal, wenn man durch Entfernen einer FA nicht äquivalenz zu F verletzt) |
3. Regel: ➔ X→A $\in$ F, Z $\subset$X wenn (F\{X→A})$\cup${Z→A} nicht äquivalent zu F ➔ d.h. Verkleinere die linke Seite, wenn möglich]
105. **Mit welchem Algorithmus lässt sich F^{min} in polynomieller Zeit berechnen?** [Mit Hilfe von X^+ und zwar ohen die Berechnung von F^+ in polynomieller Zeit]
    a. **Ist F^{min} eindeutig?** [Nicht notwendigerweise]

106. **Gibt es verlustfreie Belegungen die nicht abhängigkeitsbewahrend sind?**

Es gibt verlustfreie Zerlegungen, die nicht abhängigkeitsbewahrend sind !
Betrachte $R = (V, \mathcal{F})$, *wobei*
- V = { *Stadt, Adresse, PLZ* }, *und*
- $\mathcal{F}$ ={ *Stadt Adresse* $\rightarrow$ *PLZ, PLZ* $\rightarrow$ *Stadt* }.

R_1(*Adresse, PLZ*), R_2(*Stadt,PLZ*) *ist verlustfrei wg.* $(R_1 \cap R_2) \rightarrow (R_2 \setminus R_1) \in \mathcal{F}$, *jedoch offensichtlich nicht abhängigkeitsbewahrend.*
Beachte R *hat die Schlüssel Adresse PLZ und Stadt Adresse.*

[Man kann die Anfrage nach Stadt, Adresse mit Ausgabe PLZ nicht eindeutig beantworten (Mehrmals Freiburg mit Berliner Allee). Man müsste nun erst wieder den Verbund berechnen, um die Abhängigkeiten wieder zu erhalten]

107. **Es gibt beim Datenbankentwurf auch verschiedene Normalformen, welche Eigenschaften werden dabei betrachtet?** [genestete Realtionen auflosen, d.h. eine mininmal Zerlegung der Attribute zu finden die noch verlustfrei und Abhängigkeitsbewahrend ist und diese in eine Normalform zu bringen]
108. **Wann ist ein Relationsschema in 1NF?** [Wertebereiche der Attribute Atomar]
    a. **Was ist dabei das Problem?** [Man muss alles ausmultiplizieren und erhält viele Redundanzen]
109. **Wann ist ein Relationsschema in 3NF?** [Zu jedem NSA A muss es eine FA zu einer Menge X (X→A) geben, wobei A nicht Teil von X sein darf und X einen Schlüssel enthält d.h. falls rechts ein NSA steht, muss links ein Schlüssel enthalten sein und das NSA darf natürlich nicht Teil von X sein]
    a. **Wie schnell kann man überprüfen ob ein Schema in 3NF ist?** [NP Vollständig ➔ Exponentielle Laufzeit]
110. **Wann ist ein Relationsschema in BCNF?**[X→A∈F A∉X→X enthält einen Schlüssel, d.h. auf der linken muss immer eine Menge von Attributen stehen mit Schlüssel und A darf nicht in der Menge links enthalten sein]
    a. **Was gilt gegenüber 3NF zusätzlich?** [[Erweiterung der Bedingung auch auf SA und nicht nur auf NSAs]
    b. **Wie schnell kann man überprüfen ob ein Schema in BCNF ist?** [in polynomieller Zeit mit dem X^+ Algorithmus]
111. **Wann ist ein Relationsschema gleichzeitig in BCNF und in 3NF?** [Wenn das Schema nur genau einen Schlüssel enthält, dieser muss dann zwingend links stehen]
112. **Wenn ein Relationsschema R in BCNF ist, in welcher Normalform ist dann die zugehörige Hülle?** [auch in BCNF]
113. **Was versteht man unter Verbundabhängigkeit, mehrwertiger Abhängigkeit und Inklusionsabhängigkeit?** [Verbundabhängigkeit: Wir haben eine eine Verlustfreie Zerlegung, die wenn man sie durch Verbund wieder zusammenführt, erhalten wir die Ursprungsrelation zurück | Mehrwertige Abhängigkeiten (MVD): Spezialfall der Verbundabhängigkeiten ➔ für n=2 gilt $X_1 \cap X_2 \twoheadrightarrow X_1 \setminus X_2$, *bzw.*, $X_1 \cap X_2 \twoheadrightarrow X_2 \setminus X_1$. ➔ d.h. man hat einen Schlüssel der eine FA auf zwei Attribute hat, die ihrerseits aber unabhängig voneinander sind. | Inlusionsabhängigkeiten: Man betrachte im Gegensatz zu FAs, und VA immer 2 Schemata. Man nimmt aus diesen beiden Schemata die Schnittmenge (z.B. wie bei Hauptrelation „Name“ und Erweiterung „Zusatzinformation) der Attribute. Die Projektion der erweiterten Menge ist dann immer Teilmenge(Gleich) der Projektion der Hauptmenge ➔ Test auf referentielle Integrität möglich, d.h. wenn die IA nicht erfüllt ist, haben wir „dangling references und damit ist die referentielle Integrität verletzt

Seien X_1, X_2 *Mengen von Attributen und* $r_1 \in Rel(X_1), r_2 \in Rel(X_2)$ *Relationen. Sei* $Y \subseteq X_1 \cap X_2$.
r_1, r_2 *erfüllen die* **Inklusionsabhängigkeit (IA)** $R_1[Y] \subseteq R_2[Y]$ *gdw. gilt:*

$$\pi[Y](r_1) \subseteq \pi[Y](r_2).$$

]
114. **Was ist das Ziel der 4NF?** [Voneinander unabhängige Zusammenhänge sollen nicht in der selben Relation repräsentiert werden]
115. **Wie ist die 4NF definiert?** [Eine Relation ist in Vierter Normalform, wenn sie in Boyce-Codd Normalform ist und für jede mehrwertige Abhängigkeit einer Attributmenge Y von

einer Attributmenge X gilt:
- Die mehrwertige Abhängigkeit ist trivial ist oder
- X ist ein Schlüsselkandidat der Relation]

116. **Wie hängen die 5 möglichen Normalformen zusammen, d.h. welche ist in welcher enthalten?** [1NF←(2NF)←3NF←BCNF←4NF]
117. **Ist es möglich in BCNF bzw. 4NF abhängigkeitsbewahrende UND Verlustfreie Zerlegungen zu erzeugen?** [i.A. nicht nur bis zur 3NF]
118. **Gibt es stets eine verlustfreie Belegung in 4NF?** [Ja]
119. **Wie funktioniert der Algorithmus zur Berechnung der BCNF?**

[*Sei $R = (V, \mathcal{F})$ ein Relationsschema gegeben. R sei nicht in BCNF.*

1. *Sei $X \subset V$, $A \in V$ und $X \to A \in \mathcal{F}$ eine FA, die die BCNF-Bedingung verletzt.*
   Sei $\rho = (V - A, XA)$ eine Zerlegung von V und entsprechend
   $$R_1 = (V - A, \pi[V - A]\mathcal{F}), R_2 = (XA, \pi[XA]\mathcal{F})$$
   die entsprechenden Schemata.
2. *Teste die BCNF-Bedingung bzgl. R_1 und R_2 und wende den Algorithmus gegebenenfalls rekursiv an.*

]

120. **Die BCNF ist ja deutlich schneller zu berechnen als die 3NF, dennoch benötigt man die 3NF, warum?** [Weil die BCNF im Allgemeinen zwar verlustfrei ist, aber nicht abhängigkeitsbewahrend. Bei der 3NF kann man beide Bedingungen erfüllen]
121. Zur Berechnung der 3NF gibt es zwei unterschiedliche Ansätze, wie funktionieren sie? Wie schnell gehen diese? [3NF-Analyse: d.h. man hat ein Schema in BCNF gegeben und macht dieses abhängigkeistbewahrend:

Sei $R = (V, \mathcal{F})$ ein Relationsschema gegeben und sei $\rho = (X_1, \ldots, X_k)$ eine Zerlegung von V, so daß die entsprechenden Schemata $R_1 = (X_1, \pi[X_1]\mathcal{F})$, …, $R_k = (X_k, \pi[X_k]\mathcal{F})$ in BCNF. Sei $\mathcal{F}$ minimal.

1. *Identifiziere die Menge N derjenigen FA's, für die die Abhängigkeitsbewahrung verletzt ist.*
2. *Für jede solche FA der Form $X \to A$ erweitere ρ um XA; die entsprechenden Schemata haben die Form $(XA, \pi[XA]\mathcal{F})$.*

3NF-Synthese: Die Eingabe ist hier das Relationsschema und das zugehörige F^{min}, als Ergebnis erhält man das Schema in 3NF abhängigkeitsbewahrend und verlustfrei:

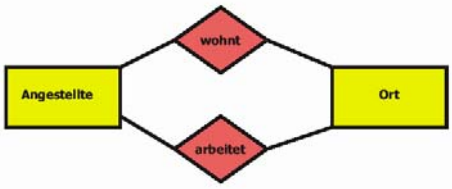
3NF-Synthese-Algorithmus
Eingabe: *Relationsschema $R = (V, \mathcal{F})$ und dazugehöriges $\mathcal{F}^{min}$.*
Ausgabe: *Verlustfreie und abhängigkeitsbewahrende Zerlegung $\rho = (X_1, \ldots, X_k)$, wobei alle $R_i = (X_i, \pi[X_i](\mathcal{F}))$ in 3NF.*

(1) Betrachte jeweils maximale Klassen von FA's aus F^{min} mit derselben linken Seite. Sei $\{X \to A_1, X \to A_2, \ldots\}$ eine solche Klasse. Bilde zu jeder solchen Klasse ein Schema mit Format $XA_1A_2\ldots$.

(2) Sofern unter den in (1) gebildeten Schemata kein Format einen Schlüssel für R enthält, berechne einen Schlüssel für R; sei K ein solcher Schlüssel. Dann bilde zu K ein Schema mit Format K.

➔ Polynomielle Zeit ➔ Korrekt ➔ Test auf 3NF ist NP vollständig ➔ Ergebnis ist nicht immer minimal]

122. **Was ist das Problem bei der Eindeutigkeitsannahme?** [Man nimmt an, dass mit gleichem Namen ausgedrückte Zusammenhänge auch identisch sind!

[

Man kann aufgrund dieser Beziehung direkt in der Relation nicht sagen, ob ein Ort eine Arbeitsstätte oder Wohnstätte ist, wenn beides gleich codiert wird d.h. die beiden Attribute PLZ,ORT können sowohl für Arbeitsstätten als auch für Wohnstätten stehen]

123. **Wie wird das konzeptuelle Datenbankschema auf das physische abgebildet?** [

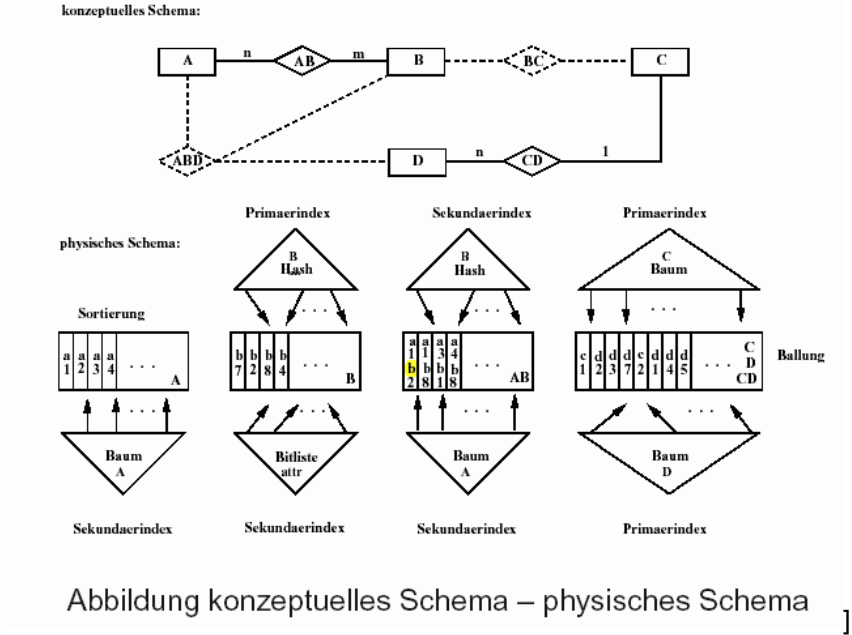


Abbildung konzeptuelles Schema – physisches Schema]

124. **Welches Ziel muss die Physische Datenbank stets verfolgen?** [Anfragen sollten möglichst effizient sein]
125. **Aus was besteht eine physische Datenbank letztendlich?** [aus einer Menge von Dateien]
126. **Aus was besteht eine Datei einer Datenbank? Erkläre die Begriffe Seiten, Blöcke, Satz und Felder?** [Eine Seite ist Datenmenge von typischer 4 bis 8 kb und enthält mehrere Sätze. Ein Block besteht aus aufeinanderfolgenden Seiten mit einer Größe von bis zu 32kb (➔4-8Seiten) | Jeder besteht aus einem oder mehreren Feldern.| In einer Datei sind typischerweise Sätze des gleichen Typs gespeichert. Hierarchische Abfolge: Datei➔Block➔Seite➔Satz➔Feld]
127. **Was versteht man unter Primär-,Sekundär- und Tertiärspeicher, nenne Einsatzbereich, Zugriffsart und Zugriffszeiten?** [

Typ	Einsatzbereich	Zugriffsart	Zugriffszeit
Primärspeicher (Cache, Hauptspeicher)	Verarbeitung der Daten	direkt	10 - 100ns
Sekundärspeicher (Platten)	permanente Speicherung	direkt	10 msec
Tertiärspeicher (Bänder, CD)	Archivierung	sequentiell	n.a.

]

128. **Die Zugriffszeit bei Festplatten setzt sich zusammen aus Seek time, rotational delay und transfer time. Auf was sollte man unter diesem Gesichtspunkt bei der Anordung der Daten auf der Platte achten?** [z.B. Blocked Access]
129. **Was sind die Aufgaben von Datei- bzw. Puffermanager?** [Der Dateimanager lokalisiert die Seite des gesuchten Satzes und beauftragt den Puffer-Manager die Seite in den Hauptspeicher zu übertragen. Der Puffermanager führt den Auftrag aus und meldet dies dem DM]
    a. Wie hängen diese zusammen? [pin_count: Die Anzahl der Transaktionen die gerade auf eine bestimmte Seite zugreifen.| Das dirty-Bit zeigt an, ob eine Seite verändert wurde ➔ Wenn nun eine Seite angefordert wird, die nicht im Puffer ist, so wird eine Seite entfernt die Pin_count = 0 hat, also auf die keiner mehr zugreift. Bei gesetztem Dirty Bit muss diese noch materialisiert werden. Gibt es keine Seite mit pin_count=0 muss entweder gewartet oder abgebrochen werden. Eine Seite mit pin_count>0 darf nicht entfernt oder abgebrochen werden | prefetching: Aufgrund bekannter Zugriffsmuster kann man Daten im Vorraus laden]
130. **Wie werden Sätze adressiert, welche Aufgabe hat dabei das Adressbuch?** [Das Adressbuch verwaltet die Adressen der einzelnen Seiten und speichert dazu Tupel mit der Seitennummer und laufender der Sätze innerhalb der Seite. Wird ein Satz gelöscht, wird der Speicher im Adressbuch als frei gekennzeichnet]
131. **Wie sieht die Seitenorganisation für Sätze fester Länge aus?** [Bei fester Länge von Sätzen hat jeder Block eine Adresse die sich ergibt aus der Multiplikation des Satznummer

und der Länge eines Satzes. Bei PACKED stehen alle belegten Sätze am Anfang im Speicher, man braucht deshalb nur die Menge der gespeicherten Sätze abspeichern. Beim Entfernen müssen Sätze nachgerückt werden. Bei UNPACKED gibt es Zwischenräume, welche über eine Bitmaske markiert werden.]

132. **Wie geht das auch für Sätze variabler Länge?** [Hier hat man ein SLOT-Directory in dem abgespeichert ist, wie lange ein SLOT ist, wie viele SLOTs es gibt und einen Zeiger auf den Anfang des leeren Bereichs. Sätze können verschoben werden, in dem man Verweisketten verwendet, d.h. Man vermerkt dort wo der Satz vorher abgespeichert war, wo er jetzt zu finden ist.]

133. Nenne 3 Organisationsformen für Dateien und die jeweiligen Laufzeiten für die Operationen Scan, Equality Search, Range Search, Insert und Delete! [

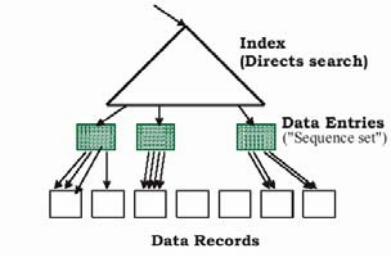
Typ	Scan	Equality Search	Range Search	Insert	Delete
Haufen	n	$0.5n$	n	2	$0.5n+1$
Sortierung	n	$\log n$	$\log n + \#matches$	$\log n + 1$	$\log n + 1$
Hash	$1.25n$	1	$1.25n$	2	2

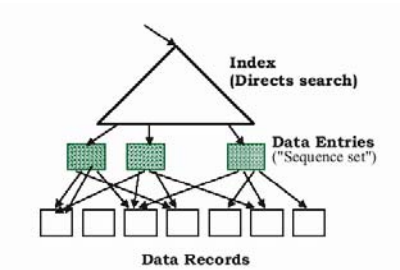
]

a. **Was versteht man unter Haufenorganisation?** [Die Datei wird ohne spezielle Organisation in einer Menge von Blöcken abgespeichert. Es wird angenommen, dass die Nummern dieser Blöcke bekannt sind (intern gehaltenes Blockverzeichnis)]
b. **Was versteht man unter Sortierung?** [Die Sätze in den Seiten sind sortiert. Die Anordnung der Seiten und entsprechend der Blöcke berücksichtigt die Sortierung]
c. **Was versteht man unter Hash-Organisation?** [die Datei wird Abhängigkeit einer Hash-Funktion in Kacheln (Blöcke) aufgestellt]

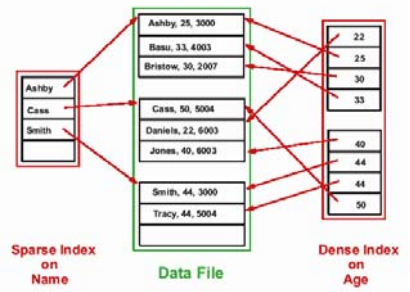
134. **Was ist eine Indexstruktur? Für was wird sie benötigt?** [Ein Index zu einer Datei ist eine Hilfstruktur um Opertaionen zu beschleunigen, die aufrgund der Organisationsform der datei nicht effizient ausgeführt werden können. Ein Suchschlüssel ist eine Feldkombination einer Datei, bzgl. der der Zugriff unterstützt werden soll. Eine Index ist eine Menge von Dateneinträgen zusammen mit einer effizienten Zugriffstechnik um alle Dateneinträge zu einem gegebenen Suchschlüsselwerz zu lokalisieren.]

135. **Was sind die Eigenschaften von Indexstrukturen, gehen sie dabei auf die Begriffe Ballung, Dichte, invertiert, voll invertiert, Primär- und Sekundärschlüssel und zusammengesetzte Suchschlüssel ein!** [
Ballung (geballt oder ungeballt) bedeutet, dass logisch zusammenhängende Sätze/Seiten auch physisch zusammenhängend gespeichert werden |



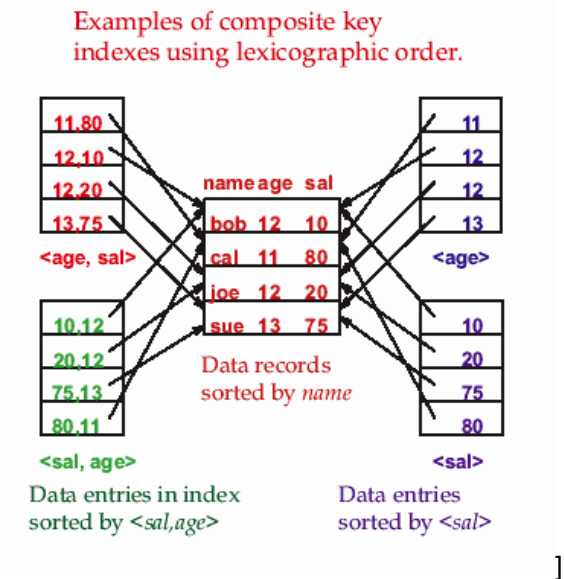


Dichte Indexstrukturen enthalten pro Satz der betreffenden Datei einen Eintrag. Spärliche Strukuren pro Seite der datei einen Eintrag |



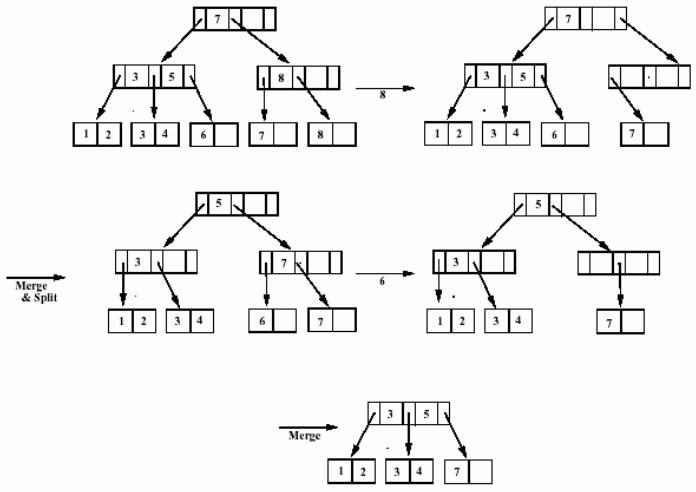
Eine Datei heisst invertiert bzgl. einem Feld, wenn ein dichter Sekundärindex zu diesem Feld existiert. Sie heisst vollinvertiert, wenn zu jedem Feld, dass nicht Teil des *Primärschlüsels* ist, ein dichter Sekundärindex existiert. |
Bei einem Primärindex enthält der Suchschlüssel den Primörschlüssel, andernfall redet man

von einem Sekundärindex. |
Ein zusammengesetzter Suchschlüssel besteht aus mehreren Feldern, sind nicht alle Felder durch Konstanten gebunden, so liegt eine Bereichsanfrage vor



]

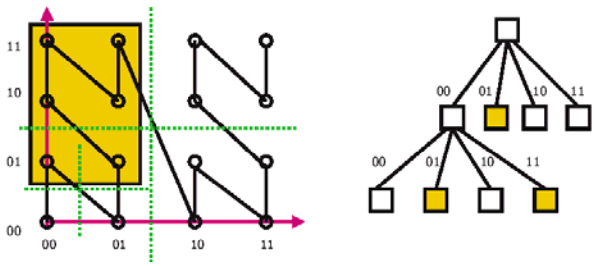
136. **Was ist ein B-Baum?** [Ein B-Baum der Odnung (m,l) mit m ≥ 3 und l ≥ 1 ist ein Vielweg-Suchbaum mit den folgenden Eigenschaften, dass:
1. Die Wurzel ist entweder ein Blatt oder hat mindestens zwei Söhne
2. Jeder innere Knoten ausser der Wurzel hat mindestens m/2 und höchsten m Söhne
➔ Mindestspeicherplatzausnutzung ist 50%
3. Alle Blätter sind auf dem gleichen Level. Bei N Blättern liegt die Höhe des Baumes also zwischen $\log_m N$ und $\log_{m/2}(N/2)$
4. Einfügen und Löschen geht in Zeit proportional zur Höhe des Baumes
5. Die inneren Knoten sind Tupel, die Suchschlüsselwerte sind geordnet
6. Blätter können auch verkettet sein, wenn Zugriff gemäß der Sortierfolge unterstützt werden soll
➔ B-Bäume eignen sich für alle Arten von Indexstrukturen (geballt/ungeballt;primär/sekundär;dicht,spärlich)]
    a. **Löschen und Einfügen gefährden die Ausgeglichenheit, wie kann man das wieder herstellen?** [Beim Einfügen werden Söhne gesplittet und das neue Element bei einem der beiden eingetragen | Beim entfernen kann sein, dass ein Element nur noch alleine in einem Zweig steht (verwaist). Dann wird das Element eins hochgezogen und die Blätter auf der Ebene neu verteilt



]

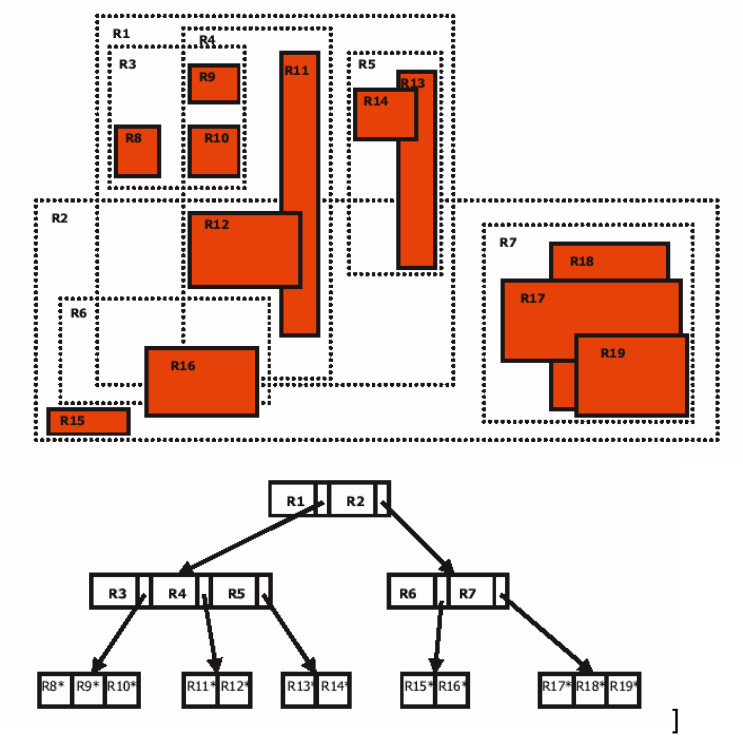
137. **Wie funktioniert Hashing?** [Verteilung der Schlüssel auf „Kacheln“ mittels einer Hashfunktion z.b. h(v=v mod k, d.h. der einzufügende Schlüssel wird in eine ganze Zahl konvertiert und und auf eine der k Kacheln gelegt. Überläufer werden in einer verketteten Liste gespeichert.]

138. **Wie werden Anfragen über räumliche Daten realisiert?** [Man meint damit z.B. Geographische Daten. Die Datenbank kann Punkte speichern oder Regionen, die z.B. durch Punkte, Linien, Polygone oder Kuben approximiert werden.]
    a. **Wie funktionieren dann die Bereichsanfragen auf solchen Daten?** [Man möchte z.B. wissen ob ein Punkt z.B. eine Sehenswürdigkeit in einer bestimmten Region liegt. Dabei gibt es verschiedene Anfragetypen. Partial Match: Heisst man übergibt den Schlüssel nur teilweise, möchte also z.b. alle Rechtecke die ihr Ecke an einer bestimmten Koordinate haben | Nearest Neighbour Anfragen | Verbundanfragen für räumliche Prädikate wie z.B. Intersects, contains, is_enclosed_by, distance(…), adjacent, northwest…]
    b. **Warum reichen hier eindimensionale Indexstrukturen nicht aus?** [1. Man könnte bei einem Rechtecke z:b. vier B-Baum Indexstrukturen verwenden, aber diese gemeinsam zu unterhalten ist schwierig, Anfragen der Form $X_1 \geq Y_2$ gehen nicht. 2. Man könnte einen B-Baum für die Konkatenation der 4 Indexe $X_1Y_1X_2Y_2$ aufbauen. Anfragen der Form $Y_1 \geq a$ sind nicht möglich. Im allgemeinen muss für jede Permutation der Koordinaten ein Index aufgebaut werden.]
    c. **Welche mehrdimensionalen Indexstrukturen kennen Sie?** [Quadbaum/R-Baum; Es gibt auch Hash basierte=> nicht besprochen]
139. **Erkläre die Funktionsweise der Space Filling Curves!**



[
Es geht darum Punkte im Raum auf eine lineare Liste abzubilden um sie einbfach abspeichern zu können ohne Informationen zu verlieren und Bereichsanfragen zu ermöglichen. Jeder Attributwert wird durch k Bits repräsentiert. Mittel seiner space-filling Kurve wird eine lineare Ordnung auf allen Punkten im Raum erzeugt, die die räumliche Nähe berücksichtigt. Gemäß dieser Ordnung wird ein B-Baum Index aufgebaut. Der Z-Wert eines Punktes ergibt sich alternierend aus den Bits seines X- und Y-Wertes.]

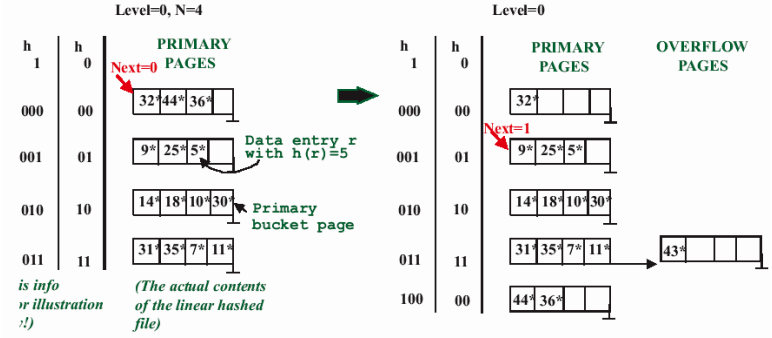
140. **Was ist ein Quad-Baum?** [Jedem Teilbaum des Quadbaumes werden Quadranten der Space-Filling Kurve zugeordnet (siehe Baum oben). Die Wurzel steht für die 4 Quadranten der ersten Unterteilung, jeder der 4 Quadranten wir in unserem Beispiel wieder in 4 zerteilt, dies könnte auch noch weiter gehen bei mehr Punkten. Die obere Anfrage zeigt, dass der Quadrant unten links (repräsentiert durch den angehängten Baum die gewünschten Punkte in seinen Quadranten „01“ und „11“ enthält, während der obere linke Quadrant komplett im Bereich liegt, markiert durch das „01“ an der Wurzel]
141. **Was ist ein R-Baum, was sind die Bounding Boxes?** [Variation von B-Bäumen, in dem wir als Suchschlüssel Intervalle nehmen, welche in x-und y-Richtung jeweils y und x-Achsen paralelle Rechtecke beschreiben, sogenannte Bounding Boxes. In jeder Boundingbox können nun wieder weitere Bounding Boxes als begrenzer liegen oder das oder die gesuchten Objekte, falls keine weitere Unterteilung nötig ist. Jeder Dateneinträg speichert eine Bounding auf unterster Ebene und das Objekt, das sie enthält]. Die Bounding Boxes sollen möglichst überschneidungsfrei sein, indem jede Boundingbox die kleinste Boundingbox ist die ihre Nachfolge enthält. Der R-Baum orientiert sich im Gegensatz zum Quad-Baum direkt an den Daten➔ Die Bounding Boxes sind immer anders, je nach Daten, die z-Curves nicht.



]

142. **Wir haben mehrere dynamische Hash-Verfahren kennen gelernt. Wie funktioniert „erweiterbares Hashing“?** [Jede „Hash-Kachel“ enthält einen Zeiger auf einen „Bucket“ der die Datensätze enthält. Der Bucket hat eine maximale Größe. Wird ein Datensatz auf einen vollen Bucket gelegt, wird dieser gesplittet. Die größe des Kachelverzeichnisses wird verdoppelt.]
    a. **Wie wird der Hashtable vergrößert bzw. verkleinert?** [Die Adressierung der Kacheln wird um ein Bit erhöht, d.h. der Adressraum verdoppelt. Sinkt die Belegung unter einen bestimmten Wert, so kann das Kachelverzeichnis wieder um die Hälfte verkleinert werden und die Globale Tiefe um 1 verringert werden (#der Adressbits)]
    b. **Was sind die Split Partner?** [Wenn ein Bucket gesplittet wird, sind die beiden getilten Buckets die Splitpartner]
    c. **Wie werden die Schlüssel auf die Buckets verteilt?** [Die letzten d Bits des Schlüssels sind der Adress-Offset für den Bucket]
    d. **Was ist die lokale und die globale Tiefe?** [Die Globale Tiefe bestimmt die Anzahl der Bits des Adressraumes. Sie steht im Header der Datei. Die lokale Tiefe ist die Anzahl der Adressbit über die ein Bucket definiert, hat z.B. ein Bucket nur Tiefe 2 wobei die globale Tiefe 3 ist, zeigen 2 Zeiger aus dem Adressraum auf ihn. Beim Split wird die lokale Tiefe um 1 erhöht, das heisst die Beiden Zeiger werden nun auf die Spltpartner aufgeteilt und die Schlüssel entsprechend verteilt. Muss gesplittet werden, obwohl die Tiefe des Buckets, der globalen Tiefe entspricht muss die Tiefe global um 1 erhöht werden (verdopplung des Adressraumes]
143. **Wie funktioniert lineares Hashing?** [Wir haben kein Kachelverzeichnis, sondern es wird eine Familie von Hash-Funktionen $h_1,\ldots,h_n$ verwendet. Der Wertebereich der Hashfunktion i ist doppelt so groß wie der der Funktion i-1.Typischerweise sieht das so aus: $h_i(v)=h(v) \bmod (2^i N)$. In jeder i-ten Runde wird die i-te Hash-Funktion zum verteilen

verwendet und der Adressraum um eines erhöht, da der Wertebereich in jeder Runde

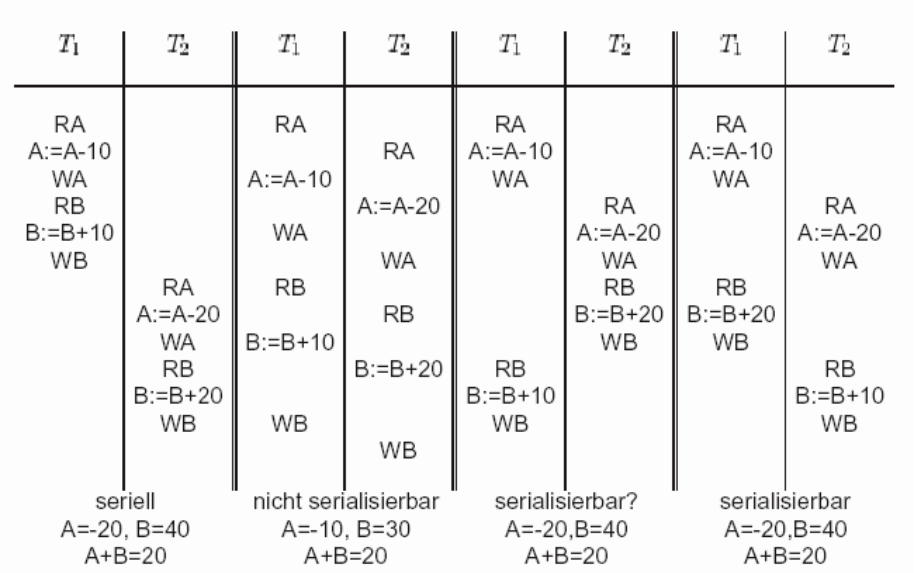


verdoppelt wird.]

a. **Wann wird ein Bucket gesplittet?** [Wenn der „next“ Zeiger auf ihn zeigt und irgendwo (das kann auch in einem anderen Bucket sein) ein überlauf auftritt. Der Next Zeiger wandert nach dem Split um einen Bucket weiter]
b. **Wie viele Zugriffe sind im schlechtesten Fall erforderlich?** [Wenn die Sätze nicht gleichverteilt sind, könnte es einen wort-case geben, der linear in der Anzahl der Sätze ist]
c. **Was passiert, wenn ein Bucket aufgrund von Löschungen leer wird?** [Falls next>0 wird dieser um eines verringert, falls next=0, wird der Level um eines veringert und der next-Zeiger auf den letzten Datensatz gesetzt]

144. **Welches Problem kann bei sehr großen Datenbanken auftreten, wenn man diese sortieren will?** [Dass sie nicht komplett in den Hauptspeicher passen]
    a. **Auf welchem Sortier-Verfahren beruht die Lösung dieses Problems?** [Merge-Sort➔ Am Anfang macht man ganz normales Mergesort bis die Teilstücke eine gewisse Größe haben. Diese nennt man Runs. Aus diesen Runs werden nun in den folgenden Schritten B-1 Seiten (B ist die Größe des Hauptspeichers) genommen und gemerged]
    b. **Kann einem die B-Baum Indexstruktur beim Sortieren hilfreich sein?** [Ja, wenn es geballter B-Baum ist, d.h. wenn zusammengehörende Datensätze im B-Baum nah beeinander gespeichert sind.]
145. **Bei der Transaktionsverwaltung spricht man von den sogenannten ACID-Eigenschaften, was bedeutet das?** [Atomicity/Consistency/Isolation/Durability]
146. **Was sind die elementaren Aktionen einer Transaktion?** [Read/Write/Begin Work/Commit Work/Rollback Work]
147. **Welche Probleme treten beim Mehrbenutzerbetrieb auf?** [Verzahnungen ➔ Die Transaktionen auf gemeinsamen Daten ab. Diese gefährden die Korrektheit der Ergebnisse]
148. **Was versteht man unter verzahnten Transaktionen?** [Transaktionen die nebenläufig oder parallel ablaufen u.U. auf die gleichen Daten zugreifen]
149. **Es gibt vier typische Fehlersituationen, erklären Sie die Begriffe „Lost Update“, „Dirty Read“, „Non-Repeatable Read“ und „Phantom“!** [Lost-Update: Wenn zwei Transaktionen verzahnt, das gleiche lesen, und am Ende auf die gleichen Daten schreiben, dann ist das Update der Transaktion die zuerst geschrieben hat verloren | Dirty-Read: z.B. Eine Transaktion mach ein Update auf Daten. Die Transaktion ist noch nicht vollständig beendet, die Daten aber schon materialisiert. Jetzt kommt eine 2. Transaktion, die ein Update auf diese Daten macht. Die Erste Transaktion beendet nun ganz am Schluss mit einem Abort. D.h. die zweite Transaktionen hat unsauber geschriebene Daten gelesen | Non-Repeatable Read (Inkonsistente Analyse): Wenn z.B. währende einer mehrere Datensätze betreffenden Transaktion verzahnt änderungen auf diesen Stattfinden. (Aufsummieren von Gehältern, wobei sich die Gehälter währende des aufsummierens ändern) | Phantom-Problem: Wenn eine neue berechnete zB. Von Gehältern mit einer bereits berechnete Summe (die redundant gespeichert ist) verglichen werden werden soll. Zwischen Berechnung und Vergleich wird aber ein zusätzlicher, die Berechnung verändernder Datensatz eingefügt (z.B. neuer Mitarbeiter mit bestimmten Gehalt]
150. **Die Probleme kann man durch Serialisierung beheben, aber was genau bedeutet Serialisierbarkeit?** [Parallele Transaktionen so verändern, dass sie sich verhält, wie wenn

sie seriell ausgeführt



würde]

a. **Wie heißt das Korrektheitskriterium für parallele Transaktionen?** [Ein Schedule heisst serialisierbar, gdw. zu ihm ein äquivalenter serieller Schedule derselben Transaktion existiert]
b. **Wann sind zwei Schedule äquivalent?** [Sie heissen dann äquivalent, wenn für jeden Startzustand und jede Interpretation der Schreibaktionen (1,) dieselben Werte gelesen werden und (2.) dieselben Endzustände erzeugt werden]

151. **Wie lässt sich die semantik einer Transaktion formalisieren?** [Bei einer Schreibtransaktion muss man all diejenigen Werte berücksichtigen die zuvor gelesen worden sind. Der Wertebereich des Werte von A bei einer Schreib-Transaktion WA ergibt sich also, aus den Wertebereichen der zuvor gelesenen Werte]
    a. **Für was kann man das nutzen?** [Um festzustellen wo Verzahnungen problematisch sind

Schedule T_1T_2	Schedule T_2T_1
T_1 :	T_2 :
$RA \quad A_0$	$RA \quad A_0$
$WA \quad f_{T_1,A}(A_0)$	$WA \quad f_{T_2,A}(A_0)$
$RB \quad B_0$	$RB \quad B_0$
$WB \quad f_{T_1,B}(A_0, B_0)$	$WB \quad f_{T_2,B}(A_0, B_0)$
T_2 :	T_1 :
$RA \quad f_{T_1,A}(A_0)$	$RA \quad f_{T_2,A}(A_0)$
$WA \quad f_{T_2,A}(f_{T_1,A}(A_0))$	$WA \quad f_{T_1,A}(f_{T_2,A}(A_0))$
$RB \quad f_{T_1,B}(A_0, B_0)$	$RB \quad f_{T_2,B}(A_0, B_0)$
$WB \quad f_{T_2,B}(f_{T_1,A}(A_0), f_{T_1,B}(A_0, B_0))$	$WB \quad f_{T_1,B}(f_{T_2,A}(A_0), f_{T_2,B}(A_0, B_0))$

]

152. **Was versteht man unter dem augmentierten Schedule?** [Der augementierte Schedule S' zu einem Schedule S ist der um zwei Transaktionen erweiterte Schedule, d.h. S'=T_0S T∞. Mit der Transaktion T_0 erzeugt man sozusagen die Vorbedingungen für S und T∞ liest den Endzustand von S. T_0 schreibt zuvor jeden Wert, der später ins S gelesen oder geschrieben wird]
153. **Was drückt ein D-Graph aus und wie konstruiert man diesen?** [Die Knoten des Graphen sind alle möglichen Aktionen und die Kanten sind Abhängigkeiten zwischen den Transaktionen. Eine Abhängigkeit zwischen zwei Aktionen exitistiert dann,
(1.) wenn die gleiche Transaktion eine Aktion hat die die Daten liest und diese möglicherweise in einer nachfolgende Transaktion schreibt, dann sind diese zwei Aktionen abhängig
(2.) wenn eine Transaktion einen Wert liest, den eine andere Transaktion unmittelbar zuvor geschrieben hat, dann sind auch diese beiden Transaktionen abhängig]
154. **Wann sind dann zwei Schedule äquivalent?** [Wenn die beiden Abhängigkeitsgraphen (D-raphen) gleich sind]
155. **Was drückt ein C-Graph aus und wie konstruiert man diesen?** [Die Knoten sind nun nicht mehr nur die Aktionen, wie beim D-Graph sondern es sind die ganzen Transaktionen.

Die Kanten beschreiben Konflikte zwischen den Transaktionen. Es gibt 3 Arten von Konflikten:
(1.) WR-Konflikt, d.h. Transaktion j liest einen Wert den, eine andere Transaktion i unmittelbar zuvor geschrieben hat.
(2.) WW-Konflitk d.h. Transaktion j schreibt einen Wert den, eine andere Transaktion i unmittelbar zuvor bereits geschrieben hat.
(3.) RW-Konflikt, d.h. Transaktion j schreibt einen Wert den, eine andere Transaktion i unmittel zuvor bereits gelesen hat ohne dass noch ein weiterer Schreibzugriff dazwischen stattgefunden hat.]

a. **Was sagt der C-Graph über Serialisierbarkeit aus?** [Ein Schedule ist serialisierbar, wenn der zugehörige C-Graph keine Zyklen enthält]
b. **Gilt auch die Umkehrung?** [Nein, denn es gibt serialisierbare Schedules, deren C-Graph nicht zyklenfrei ist]
c. **Wann nennt man einen Schedule C-Serialisierbar?** [Wenn sein C-Graph Zyklenfrei ist]

156. **Was ist die Aufgabe eines Schedulers?** [Der Scheduler stellt sicher, dass nur serialisierbare Schedule zur Ausführung kommen. Der Scheduler ist ein Teil des Datenbanksystems. Formal gesehen ist der Scheduler eine Abbildung, die eien Eingabefolge von Aktionen einer Menge von Transaktionen in eine serialisierbare auszuführende Ausgabefolge von Aktionen der Transaktionen transformiert.]
157. **Welche 4 Verfahren kann der Scheduler anwenden?** [2-PL (2-Phasen-Sperren) | Überwachen der C-Graphen | Zeitmarken | Optimistische Verfahren]
158. **Eine Möglichkeit der Konfliktvermeidung sind Sperren, wie läuft das im Allgemeinen ab?** [Man sperrt die betreffenden Objekt, sobald eine Transaktion diese bearbeiten für andere Transaktionen und gibt sie erst wieder frei, wenn die Transaktion beendet ist. Es wird unterschieden zwischen WLOCK,d.h. Lese-/ und Schreibzugriffe sind für andere Transaktionen gesperrt, und RLOCK, d.h. nur Lesezugriffe sind für andere Transaktionen gesperrt. Lock und Unlock können wie Aktionen einer Transaktion im Schedule auftauchen]

a. **Welche Sperren sind miteinander kombinierbar?**

[

angefordertes Privileg: \ gehaltenes Privileg:	LOCK
LOCK	N

	RLOCK	WLOCK
RLOCK	J	N
WLOCK	N	N

]

159. **Was sind Livelocks?** [Wenn eine Transaktion ein Privileg (zb. einen Schreibzugriff) anfordert, diesen jedoch nie bekommt, weil andere Transaktionen immer vorgezogen werden ➔ Lösung First-Come First Serve]
160. **Was sind Deadlocks?** [Wenn z.B. zwei Transaktionen auf ein gesperrtes Objekt der jeweils anderen Transaktion warten:

T_1: LOCK A; LOCK B; UNLOCK A,B; T_2: LOCK B; LOCK A; UNLOCK A,B;

T_1: LOCK A;
T_2: LOCK B;
↑
Deadlock

T1 warte auf LOCK B und T2 warte auf LOCK A ➔ Nix geht mehr]

a. **Wie kann man sie auflösen bzw. vermeiden?** [
(1.) Bevor eine Transaktion gestartet wird, fordert diese beim Scheduler in einer atomaren Aktion alle notwendigen Sperren auf einmal an.
(2.) Auf den Objekten wird eine lineare Ordnung definiert nach welcher die Privilegien angefordert werden können.
(3.) Man nimmt einen Wartegraphen für die Transaktionen, (d.h. Knoten sind die Transaktionen und Kanten sind vorhanden, wenn sich eine Transaktion um ein Privileg bewirbt, welches eine andere Transaktion hat) sobald sich dieser in einem Zyklus befindet sind wir in einem Deadlock ➔ Auflösung: Abbrechen einer beteiligten Transaktion]

161. **Leider garantieren reine Sperren noch keine Serialisierbarkeit, das 2 Phasen Sperrprotokoll löst dieses Problem, wie funktioniert es?** [Sobald ein Unlock ausgeführt wurde, darf diese Transaktion kein Lock mehr anfordern]

a. **Welches sind die beiden Phasen?** [Sperrphase und Unlockphase]
b. **Wann nennt man das Protokoll strikt?** [Wenn alle Unlocks ganz am Schluss der Transaktion sind]

c. **Was ist der Sperrpunkt?** [Der Punkt an dem die letzte Sperroperation ausgeführt wurde, d.h. alle Sperren gesetzt sind]

162. **Warum garantiert das 2-PL Protokoll serialisierbarkeit?** [Beweis durch Widerspruch. Man nimmt an S sei nicht serialisierbar ➔ der CG(S) hat einen Zyklus. …]
163. **Warum ist das 2 Phasen Sperren optimal?** [Da es zu jeder 2-Phasigen Transaktion eine einphasige Transaktion gibt, die zusammen mit der zweiphasigen nicht serialisierbar ist]
    a. **Gibt es serialisierbare Schedule, die nicht mittel 2-PL serialisierbar sind?** [Ja, z.B. $S = R_1A\ R_2W_2A\ R_3W_3B\ W_1B$ ist nicht mittel 2-PL serialisierbar, obwohl er serialisierbar ist]
164. **Möchte ein Transaktion nur auf wenige Tupel einer Relation zugreifen muss man ja nicht gleich die ganze Relation sperren, welchen Vorteil hat das und welche Probleme ergeben sich dabei?** [Vorteil: Höhere Parallelität möglich, da weniger Sperren vorhanden sind | Problem: Wenn ein Tupel gesperrt wurde und danach eine Transaktion die ganze Relation sperren will ➔ Das darf nicht sein]
    a. **Wie werden diese Probleme gelöst?** [Warnsperre an der Relation ➔ Intention Lock]

	IL	WL
IL	ja	nein
WL	nein	nein

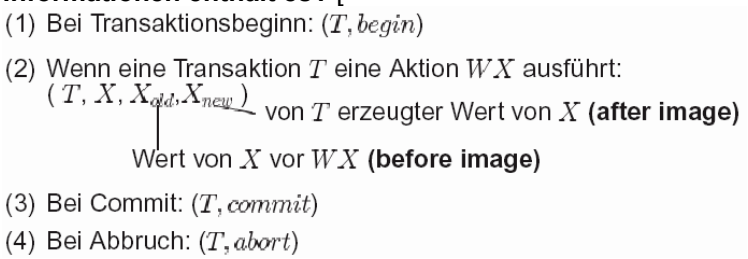
    b. **Welche Locks sind dann noch miteinander vereinbar?** [IL=Intention Lock | WL =WriteLock]
165. **Reine Tupel-Sperren reichen z.B. beim Phantom Problem nicht aus um die Transaktion zu serialisieren, wie muss man hierbei vorgehen?** [Eine Tupelsperre unterbindet kein „Insert“, was bewirkt, das Phantomdatensätze entstehen können➔ Lösung: Sperren ganzer Tabellen, Schlüsselbereichen, Prädikaten, Indexen]
166. **Was ist ein Hot Spot?** [Ein Hot Spot ist ein Punkt in der Datenbank, welcher durch viele Zugriffe häufig gesperrt wird, was eine verlangsamung der Transaktion zur Folge hat.]
    a. **Werden auf einen Hot Spot nur Additionen und Subtraktionen ausgeführt, so sind die normalen Sperren eigentlich nicht notwendig, da die Aktion kommutativ ist. Welche speziellen Sperren gibt es in diesem Fall und was sperren sie?** [Wir haben dazu sogenannte Increment und Decrement Locks, die beliebig oft vergeben werden können. Ist aber ein Increment- oder Decrement Lock gesetzt, bekommt keine andere Transaktion ein RLOCK bzw. WLOCK. Komplette Kompatibilitätstabelle:

	RLOCK	WLOCK	INCRLOCK	DECRLOCK
RLOCK	J	N	N	N
WLOCK	N	N	N	N
INCRLOCK	N	N	J	J
DECRLOCK	N	N	J	J

]
167. **Was ist das Problem beim 2-PL?** [u.U. lange unnötige Hot-Spots]
168. **Wie könnte ein einfaches Sperr-Protokoll für B-Bäume aussehen, welches das Hot-Spot Problem löst?** [Ziel ist es ein Protokoll auf B-Bäume zu erzeugen, welches mehr parallelität ermöglicht als 2-PL.

	insert/delete	**read**
(1)	Sperre die Wurzel	Sperre die Wurzel
(2)	Lese die Wurzel und mache sie aktuell	Lese die Wurzel und mache sie aktuell
(3)	Solange der aktuelle Knoten kein Blatt ist,	Solange der aktuelle Knoten kein Blatt ist,
(3.1)	Sperre betreff. Nachfolger	Sperre betreff. Nachfolger
(3.2)	Lese ihn und mache ihn aktuell	Gib Sperre des Vorgängers frei
(3.3)	Falls er sicher ist, gib alle Sperren der Vorgänger frei	Lese den betreff. Nachfolger und mache ihn aktuell
(4)	Führe insert/delete aus	Führe read aus
(5)	Gib alle Sperren frei	Gib Sperre frei

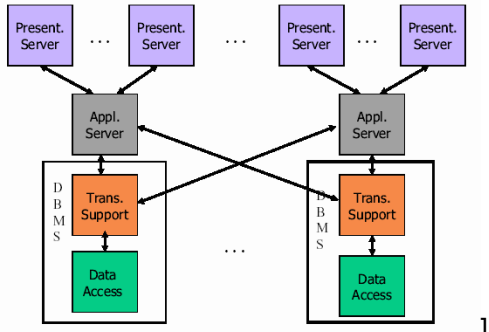
]
    a. **Wann heisst ein Knoten eines B-Baumes sicher?** [Wenn er bzgl. Insert noch nicht ganz voll ist und bzgl. Delete mindestens halbvoll ist ➔ Keine Gefahr von Split oder Löschung des Knotens durch diese Operation]

169. **Bei Transaktionen kann es auch zu Fehlern, welche Arten Unterscheidet man da?** [Transaktionsfehler: z.B. Fehler im Awendungsprogramm, Abbruch durch User oder DBS, Verletzung von Systemresriktionen, Auflösen eines Deadlocks…
Systemfehler: Primärspeicherverlust, Hardwarefehler, falsche Werte in den Systemtabellen
Mediafehler: Fehler auf dem Speichermedium ➔ Sekundärspeicherverlust]
170. **Sämtliche Transaktionen werden in einem Logfile mitgeschrieben, welche Informationen enthält es?** [


Also im Wesentlichen, den Beginn einer Transaktion, was ausgeführt wurde, welche Daten zuvor und hinterher in der DB standen und ob es ein commit gab oder einen Abbruch.]
171. **Wann wird bei einem Fehler die Transaktion rückgängig gemacht (Undo) und wann wird sie wiederholt (Redo)?** [
Undo: Fehler vor erfolgtem Commit, aber schon materialisierte Daten
Redo: Fehler nach Commit, d.h. Daten evtl. noch nicht materialisiert]
172. **Was bedeutet Stealing Frames und Forcing Pages?** [Steal: Der Speicherbereich der Transaktion darf von einer anderen Transaktion „gestohlen" werden, dazu müssen aber die Daten in die Datenbank zurückgeschrieben werden, noch bevor ein Commit kam.
Force: Erzwingen, die Daten bis zum Zeitpunkt des Commits zu materialisieren ➔ Viele Datenbankzugriffe, ansonsten könnte man einige Transaktionen ansammeln und gleichzeitig schreiben]
173. **Welche Eigenschaft muss das Log-Granulat (d.h. die einzelnen Einträge im Logfile) erfüllen?** [Es muss klein genug, also präzise genug, sein, um auch z.B. Sperren auf Satzebene zu erkennen und nicht z.B. nur auf Seitenebene]
174. **Was fordert die die WAL-Regel?** [Write-ahead-log: Bevor etwas in die Datenbank materialisiert wird, muss auf jeden Fall ins LogFile geschrieben werden]
175. **Was passiert während und nach der Commitphase einer Transaktion?** [Eine Transaktion heisst noch aktiv, wenn noch kein Commit ausgeführt wurde, ansonsten abgeschlossen. Ein abgeschlossenes Commit bedeutet, dass alle Schreibaktionen zumindest im LogFile materialisiert sind. Nach dem Commit dürfen die Sperren freigegeben werden.]
176. **Was macht das System bei einem Transaktionsfehler?** [Wenn es vor dem Commit einen Transaktionsfehler gab, macht das System ein Undo, d.h. die alten Werte werden in umgekehhrter Reihenfolge bis zum Beginn der Transaktion im Logfile zurückgeschrieben]
177. **Was macht der Restart Algorithmus, wenn ein Systemfehler aufgetreten war und kein Sicherungspunkt gesetzt war?** [Das DBS erstellt zwei Listen redone und undone und initialisiert diese „leer". Dann wird das Logfile ausgewertet. Falls zu einer Transaktion bereits ein Commit im Logifile steht, dann wird ein Redo gemacht und dies in der Redone Liste vermerkt. Steht noch kein Commit im Logfile, müsen wir die Transaktion rückgängig machen und sie im Undone vermerken. Sobald alle Transaktionen des Logfiles in einer der beiden Listen steht, sind wir fertig.]
178. **Was geschieht beim Setzen eines Sicherungspunktes (Checkpoint)?** [Man unterbindet den Start neuer Ttansaktionen, und warte bis alle anderen Transaktionen abgeschlossen sind. Erzwingen des Schreibens aller Änderungen in die Datenbank. Checkpoint am Ende im Logfile vermerken➔ Bei einem Restart, muss das Logfile nur bis zum letzte Sicherungspunkt verarbeitet werden]
179. **Welche beiden Strategien kann man zum Schutz vor Mediafehlern anwenden?** [Mirror➔Schreiben auf die Mirrorplatte, erst wenn auf der Originalplatte bestätigt wurde | Periodische Datensicherung ➔ Regelmässiges erstellen eines Dumps. Der Restartalgorithmus muss dann nur bis zum letzten „archive" Eintrag im Logfile angewandt werden.]
180. **Was ist ein verteiltes Datenbanksystem?** [Die Einträge der Datenbank sind lokal bei den einzelnen Suborganisationen gespeichert (nicht redundant). Verbunden durch LAN, WAN

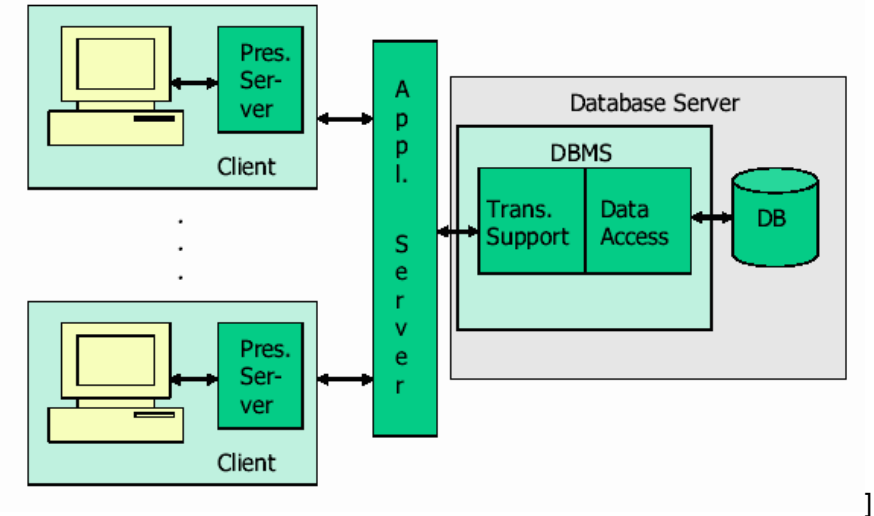
oder Telefon. Man unterscheidet zwischen homogener und heterogener Föderation. Bei homogener Föderation gibt es distribution transparency, d.h. die Verteilung ist von aussen nicht zu unterscheiden]

181. **Wie sieht so ein verteiltes DBS aus?** [



]

182. **Wie sieht die Systemarchitektur einer Site aus (3-tiered Architektur)?** [

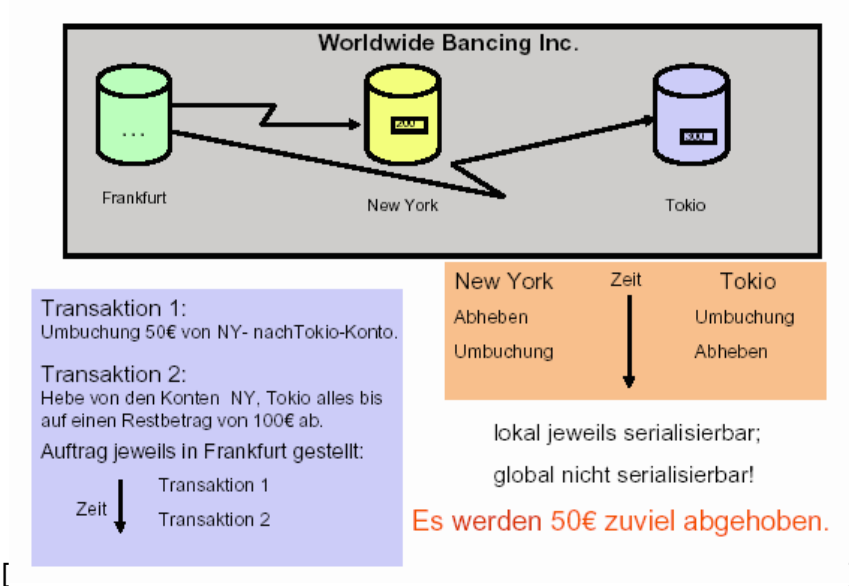


]

a. **Was muss man dabei bei globalen Verarbeitungen beachten?** [Serialisierbarkeit Recovery und Transaktionsverwaltung müssen jeweil global verwaltet werden]

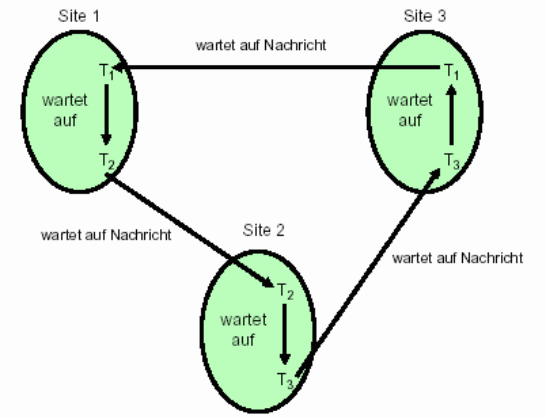
183. **Bei globaler Serialisierbarkeit unterscheidet man den Rechnerverbund zwischen homogener Föderation und einer heterogenen Föderation. Welche Techniken kommen jeweils zur Anwendung?** [Bei der homogenen Föderation können die gleichen Techniken angewandt werden, wie auch bei einer nicht verteilten Datenbank, diese müssen nur ein wenig adaptiert werden. Bei der heterogenen Föderation sind die Mehrbenutzerkontrollen gegenseitig unbekannt. Die globale serialisierbarkeit sollte sich aus der jeweils lokalen serialisierbarkeit der einzelnen Sites ergeben. Man muss zwischen lokalen und globalen Transaktionen unterscheiden]

184. **Nennen Sie ein Szenario, bei dem die Serialisierung (Global gesehen) schief läuft, obwohl die Transaktionen Lokal gesehen Serialisierbar wären?**



[ ]

185. **Was ist ein globaler Schedule und welche Bedingungen muss er erfüllen?** [Der Globale Schedule projiziert auf die Site i, ergibt gerade den lokalen Schedule der Site i (in der richtigen Reihefolge der Transaktionen. Gilt dies nicht, ist der Schedule nicht global]
186. **Welche Probleme ergeben sich, wenn man das 2-Phasen Sperren auf verteilte Systeme erweitern will?** [Sperrverwaltung wird einer ausgewählten Site zugewiesen (➔ Single Point of Failure und Flaschenhals) | Vor der Freigabephase müssen sich die einzelnen Subtransaktion an einer globalen Transaktion synchrionisieren ➔ Hoher Nachrichtenverkehr. Sperren werden gehalten bis zur Einleitung eines Globalen Commits. Deadlocks sind oft über mehrere Sites verteilt, deren Erkennung ist aufwendig]]
187. **Wie sieht ein verteilter Deadlock aus?** [



]

a. **Wie kann man diesen erkennen?** [Zentraler Monitor: Bei einem erkannten Zyklus wird eine Transaktion abgebrochen. Die Auswahl dazu ist problematisch, desweitren kann es passieren, dass durch die Zeitverzögerung der Deadlock bereits aufgelöst wurde | Time Out: Falls der Time-Out zu niedrig ist, wird ein Deadlock vermutet wo keines ist und unnötigerweise eine Transaktion abgebrochen | Edge Chasing: (Probe Nachrichten werden an die blockierenden Transaktionen geschickt, diese sendet die Probenachrichten ihrerseits an die sie blockierenden Transaktionen weiter. Erhält eine Transaktion ihre eigene Probenachricht, bricht sie sich selber ab | Path Pushing(Wartegraph): Eine wartende Seit schickt ihre „Wartekante“ an die blockierende Site, diese ergänzt diesen „Graphen“ um ihre eigenen Wartekanten in Richtung des nächsten Blockierers. Entsteht in diesem Graphen irgendwann ein Zyklus, wird dieser von einer der Sites erkannt und es können gezielt Transaktionen abgebrochen werden, da man die beteiligten Transaktinen kennt]

188. **Wie funktioniert das verteilte Zeitmarken-Verfahren?** [Globale Vergabe der jeweiligen Zeitmarken für die Transaktionen]

189. **Wie stellt man beim Zeitmarkenverfahren sicher, dass alle Uhren der verteilten Sites synchron laufen**? [Lamport Uhr: Eine Uhr ist hier ein Zähler! In jeder Kommuniktionsnachricht der einzelnen Sites werden die lokalen Uhrzeiten übermittel. Ist die Uhr eines Gegenübers größer als die eigene, wird die eigene Uhrzeit hochgesetzt auf diesen Wert]
190. **In einer heterogenen Föderation ist die Verwaltung von Transaktionen aufwendiger. Wie wird hier die Serialisierbarkeit gewährleistet?** [Ein globaler Transaktionsmanager (GTM) verwaltet die globalen Transaktionen der einzelnen Sites und kontrolliert die Ausführung und die Verteilung an die einzelnen Lokalen Transaktionsverwalter (LTM). Der GTM läuft auf einer ausgewählten Site. Lokale Transaktionen sind nur ihrem LTM bekannt. Die globalen Transaktionen werden vom GTM untereinander serialisiert und die berechneten Schedules an die LTMs weitergegeben]
191. **Trotz lokaler Serialisierbarkeit kann es dennoch sein, dass obwohl die serielle Reihenfolge global nicht verändert wurde hat der LTM dennoch die ursprüngliche Reihenfolge der einzelnen Transaktionen verändert und damit zwar nicht lokal, aber global einen Konflikt verursacht. Diese Problem kann man mit Hilfe so genannter Expliziter-Tickets lösen, wie funktioniert dieses Verfahren?** [Man bezieht den LTM mit in die Transaktion ein, indem man die Subtransaktionen einer Globalen Transaktionen ein Ticket auf der Site ziehen lässt. Die Subtransaktion liest den Wert, erhöht ihn um eins und schreibt ihn zurück (Take-A-Ticket-Operation). Dadurch verhindert man, dass der LTM die Reihenfolge der Subtransaktionen verändert.]
192. **Unter bestimmten Voraussetzungen wäre es möglich auf den GTM gänzlich zu verzichten, welche Vorraussetzungen müssen dabei erfüllt sein?** [LTM=>Globale Subtransaktionen. Die globalen Subtransaktionen enthalten Take-A-Ticket Operationen. ACA➔ Avoids Cascading Abort gilt, d.h. eine Transaktion kann ein Ticket erst lesen, wenn die zuvor lesende Transaktion das Ticket zurückgeschrieben hat. Das heisst der LTM gewährleistet die Serialisierbarkeit der globalen Subtransaktionen und der lokalen Transaktionen, damit benötigen wir den GTM eigentlich nicht mehr]
193. **Welche Fehlersituationen gibt es in einem verteilten Datenbanksystem?** [Sites Failure➔ Eine Site arbeitet entweder korrekt oder gar nicht, wobei ein Teil der Sites oder auch alle ausfallen können.(Partiell vs. Totaler Ausfall | Communication Failure➔ Keine Verbindungen, deshalb verlorene Nachrichten. Eine Partitionieren des Netzte ist möglich | Undeliverable Message➔ Empfänger ist z.B. down oder ist in einer anderen Partition ➔ Unzustellbare werden zerstört | Timeout ➔ Annahme eines Site bzw. Kommunikationsfehlers (Keine Unterscheidung möglich)]
194. **Was ist das 2 Phasen Commit Protokoll?** [Votierungsphase: Will eine Site eine Transaktion ausführen, ist Sie die sogenannte Homesite und wird damit zum Koordinator der Transaktion. Alle beteiligten Sites nennt man Teilnehmer. Nach Ausführung der Transaktion sendet der Koordinator ein Vote-Req an alle Teilnehmer, diese antworten mit „Yes“ oder „No“ je nach dem ob bei der Transaktion alles geklappt hat. Im Falle von No, bricht der betreffende Teilnehmer seine beteiligte Transaktion ab.
Die Entscheidungsphase: Der Koordinator sammelt die Antworten. Wenn alle Antworten „Yes“ sind, sendet er jedem Teilnehmer ein Commit, ansonsten ein Abort. Die Teilnehmer mit „Yes“ warten auf die Antwort und führen entsprechen Commit oder Abort aus (die mit „no“ haben ohnehin schon abgebrochen]
195. **Welche möglichen unsicheren Wartesituationen gibt es?** [
    1. Teilnehmer wartet auf Vote-Req ➔ Nach Time-Out bricht er sich ab
    2. Koordinator wartet auf Votes ➔ Nach Time-Out Abort an alle schicken
    3. Teilnehmer hat mit „Yes“ geantwortet und warte auf das Commit bzw. Abort]
196. **Was passiert, wenn ein Transaktionsteilnehmer die Entscheidung aus irgendeinem Grund nicht mitbekommt?** [Der Teilnehmer ist ein unsicheren Situation in der Zeit nach der Voteabgabe und dem Erhalt des Commit. Decision-Req an andere Teilnehmer➔wenn Misserfolg➔blockiert]
197. **Wie kann man ein Recovery während der Commit-Phase ausführen?** [Mit Hilfe des Logfiles, aber etwas problematisch ist das recovery einen unsicheren Prozesses, da dieser nicht weiss in welcher Situation er ist. Das Logfiles muss also lange genug gehalten werden, biss alle Teilnehmer erfolgreich ihr Commit oder Abort durchgeführt haben]
198. **Unter Umständen kann es sinnvoll sein eine Transaktion nur Teilweise rückgängig zu machen, zum Bsp. Bei der Routensuche für Flugreisen. Was ist der Unterschied zwischen „Rollback Work i“ und „Rollback Work“?** [Es ist z.B. gerade bei einer

Routenplanung sinnvoll nur einen Teil der berechneten Route rückgängig zu machen. Dies geschieht durch setzen von Sicherungspunkten mit „Save Work“. Mittels „Rollback Work i“ gelangt man gezielt zum Sicherungspunkt i zurück. Alle ab diesem Punkt gemachten Teiltransaktionen werden rückgängig gemacht. „Rollback Work“ setzt die gesamte Transaktion bis „Begin Work“ zurück. Mit „Commit Work“ wird die gesamt Transaktion bestätigt. Gibt es bei diesem Commit einen Fehler gibt es einen Rollback auf die gesamte Transaktion]

199. **Geschachtelte Transaktionen bestehen ihrerseits aus Sub-Transaktionen, welche der ACID Eigenschaften müssen diese noch erfüllen?** [Atomizität und Isolation: um isoliertes Rücksetzen zu ermöglichen. Konsistenz ist nicht nötig, da die Vatertransaktion die Konsistenz wiederherstellen kann. Dauerhaftigkeit: nicht möglich ➔ Beim Rücksetzen der übergeordneten Transaktion, werden die Subtransaktionen auch zurückgesetzt]
200. **Erkläre die Commit-, Rücksetz- und die Sichtbarkeitsregel für Subtransaktionen!** [Commitregel: Der endgültige Commit einer Vater-Transaktion setzt sich zusammen aus den Ergebnissen der Subtransaktionen
Rücksetzregel: Untergeordneter Subtransaktionen werden beim Abbruch einer übergeordneten Transaktion mit abgebrochen unabhängig von deren Commit-Status
Sichtbarkeitsregel: Die Änderungen einer Subtransaktion werden der Vatertransaktion bei erfolgtem Commit der Subtransaktion sichtbar (Isolation bis zum Commit). Die Geschwister Transaktionen wissen nichts voneinander]
201. **Wann heisst eine Transaktion „geschlossen geschachtelt“ und wann „offen geschachtelt“?** [„geschlossen geschachtelt“: Wenn die Vatertransaktion die Änderungen einer Subtransaktion erst mit erfolgtem Commit einsehen kann. Kann sie dies immer sehen, nennt man eine Transaktion „offen geschachtelt“]
202. **Nennen Sie ein Beispiel für eine langlaufende Transaktion (Saga)! Warum muss jede Teil- Transaktion auf jeden Fall Kompensiert werden können, nennen Sie auch hier ein Beispiel!** [Eine Saga von Transaktionen besteht aus in sich abgeschlossenen Transaktionen, die alle für sich die ACID Eigenschaften erfüllen. Bricht man also eine Transaktion der Saga ab, sind die vorherigen Transaktionen aber bereits ausgeführt. Man kann diese nicht mehr abbrechen sondern nur noch kompensieren, dazu muss zu jeder Transaktion der Saga eine Kompensationstransaktion vorhanden sein]
203. **Langlaufende Transaktionen werden normalerweise sequentiell aber kontinuierlich abgearbeitet. Der Prozess ist somit blockiert Mit welchem Prinzip könnte man das ändern?** [Die Transaktionen werden in einer Queue an das DBS übergeben welches diese abarbeiten kann. Diese Queue muss aber Recoverable bleiben. Desweiteren muss gelten: Bei einem enque, welches abgebrchen wird, muss das Element wieder aus der Schlange entfernt werden. Umgekehrt muss bei einem abgebrochenene Dequeue das Element wieder in die Schlange eingefügt werden. Erst wenn das enqueuen des Elementes Commitet wurde, daf es von einer Transaktion abgearbeitet werden]
204. **Was Bedeutet Sicherheit in einer DB?** [Geheimhaltung: Informationen dürfen nicht unauthoisierten Benutzern offengelegt werden | Integrität: Daten dürfen nur durch autorisierte Benutzer geändert werden | Verfügbarkeit:: Berechtigter Zugriff darf nicht verweigert werden | Sicherheitspolitik: Vorgehensweise um das Ziel der Sicherheit zu erreichen]
205. **Welche Faktoren können die Sicherheit in einem Gesamtsystem gefährden?** [Sicherheitslöcher im Betriebssystem oder Netzwerk etc.]
206. **Welche Sicherheitsmechanismen bringt dabei ein DBMS mit?** [Views und Zugriffskontrolle]
207. **Was versteht man unter Discretionary Access Controll (Diskrete Zugangskontroller DAC)?** [SQL2 biete für die Zugriffskontrolle GRANT und REVOKE Kommandos an: Zugriffsrechte zu Basisrelationen und Sichten können zugewiesen werden oder wieder entzogen werden

```
GRANT privileges ON object TO users [ WITH GRANT OPTION ]

REVOKE [ GRANT OPTION FOR ] privileges ON object FROM users {
                  RESTRICT | CASCADE }
```

]
208. **Auf welche üblichen Einzeloperationen eines DBS kann man die Rechte (Priviligien) vergeben?** [Select: Recht zum lesen von Spalten | Insert: Recht zum Einfügen von Zeilen wobei man das auf einzelne Spalten beschränken kann | Update: wie bei Insert | Delete: Recht Zeilen zu löschen | References: Das Recht einen Fremdschlüssel (Referenz) zu

definieren, die auf die betreffende Relation verweisen. Beschränkung auf bestimmte Spalten ist wieder möglich]

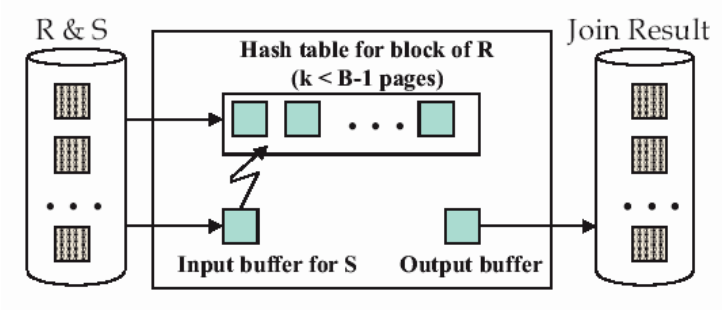
209. **Welche Probleme treten bei der Rechtevergabe auf, wenn ein Benutzer z.B. ein Recht oder auch Teilrecht von verschiedenen Personen erhält?** [Rechte dürfen an andere Benutzer weitergegeben werden. Problem: Wenn ein Benutzer mir das Recht wieder entzieht, habe ich das Recht vom anderen Benutzer noch, ja oder nein?]
210. **Was passiert, wenn man ein Recht besitzt, das man aber aufgrund des Verlustes anderer Rechte nicht mehr nutzen kann?** [Ein solches Recht nennt man Abandoned (verlassen) ➔ z.B. Wenn man einen Tresorschlüssel hat, den schlüssel für die Haupteingangstür aber nicht, bringt einem der Tresorschlüssel wenig]
211. **Wie kann man das Entstehen solcher verlassener Rechte verhindern?** [REVOKE …CASCADE: Das verlieren des Rechts bewirkt auch den Verlust von Rechten, die sonst verlassen wären | REVOKE … RESTRICT: Falls beim Entfernen des Hauptrechts, verlassene Rechte entstehen würden, wird das REVOK abgebrochen und das Recht bleibt erhalten]
212. **Wie sieht ein Autorisierungsgraph aus?** [Wir haben eine Knotenmenge, welche die Benutzer repräsentiert und einen Systemknoten, welcher alle Rechte hat. Die Kanten sind nun Rechte, die gerichtet und auch gewichtet sind, d.h. die Gewichtung besteht aus einem 3-Tupel, bestehend aus dem erhaltenen Privileg, das betreffende Objekt und ob er das recht weitergeben darf ➔ [Privileg | Objekt | WITH GRANT OPTION (YES/NO)]]
213. **Nennen Sie ein Szenario bei dem die Zugriffskontrolle unterwandert werden kann!** [Trojanische Pferde: Ein Benutzer TRicky Dick möchte z.B. an die Bewertungstabelle von Trustin Justin herankommen. Er kreiert eine neue Tabelle MineAllMine und gibt Trustin Justin ein Insert Recht auf diese Tabelle ohne dass Trustin Justin das merkt. Er manipuliert den Code einer Datenbankanwendung von Justin Trustin so, dass diese Anwendung ab sofort immer die Bewertungstabelle von Trustin Justin liest und sie in MineAllMine kopiert. Die Anwendung fällt nicht unter die Sicherheitsmechanismen des DBMS. Tricky Dick muss nun nur noch warten, bis Trustin Justin die Anwendung das nächste mal startet]
214. **Wie kann man auch dieses Sicherheitsloch stopfen?** [Zugriffsklassen mit Bell-La Padula]
215. **Wie ist das Bell-LaPadula Modell aufgebaut? Was ist das Prinzip?** [Es verfügt über vier Sicherheitsklassen: TopSecret>Secret>Confidential>unclassified. Das Lesen und Schreiben zu Objekten muss nun folgenden Bedingungen genügen:
Subjekt U darf Objekt O lesen, wenn class(U)≥class(O) und schreiben wenn classu(U)≤class(O) ist. Jeder Benutzer ist in genau einer Klasse. Das oben genannte Szenario ist nun nicht mehr möglich, da Trustin Justin in einer höheren Sicherheitsklasse ist als Tricky Dick und nicht auf Tabellen dieser Sicherheitsklasse (von Tricky Dick) schreiben darf.]
216. **Das Bell-LaPadula Modell ist auf den ersten Blick etwas unintuitiv, besonders das Recht zu Schreiben. Erklären Sie warum es funktioniert?** [Eine niedrigere Klasse darf nur in eine höhere Klasse schreiben, darf diese aber nicht lesen. Man verhindert damit, dass ein Benutzer der Klasse Top-Secret vertrauliche Informationen in niedrigere Klassen weitergibt. Lesen darf er alles, nur schreiben nicht.]
217. **Was versteht man unter Polyinstantiierung?** [Bei verschiedenen Zugriffsklassen, ergeben sich zwangsläufig Relationen, die sich über mehrere Klassen erstrecken (Mehr-Ebenen-Relationen) mit der Konsequenz, dass nicht jeder Benutzer das selbe einer Tabelle sieht. Es gibt also verschiedene Ansichten genannt Polyinstantiierungen]
218. **Was passiert, wenn man einen Datensatz einfügen will der die Schlüsselbedingung einer höheren Klasse verletzen würde?** [Damit ist gemeint, dass ein Benutzer z.B. einen Datensatz einfügen will, dessen primärschlüssel aber in einer höheren Sic herheitsklasse bereits definiert ist. Die sieht der Benutzer aber nicht. Würde nun eine Fehlermeldung vom System kommen, könnte der Benutzer auf die Existenz des Schlüssels in der höheren Ebene schließen, erhält also Informationen, die ihm nicht zustehen. Lösung: Die Einfügung wir erlaubt indem der schlüssel mit der Sicherheitsklasse kombiniert wird]
219. **Was versteht man unter einer statistischen Datenbank?** [Nur Aggregationsfunktionen erlaubt wie MIN,MAX,AVG, SUM etc.]
220. **Wie kann man individuelle Informationen aus einer Statistischen Datenbank erspähen, auf die man eigentlich keinen Zugriff hat?** [Man muss eine Teilinformation haben. Sneaky Pete möchte die Bewertung seines Admirals wissen und weiß, dass dieser der älteste ist. Er stellt also solange Anfragen vom Typ „Wieviele Segler existieren mit dem

Alter größer als X“ und inkrementiert X sukzessive, bis als Ergebnis „1“ herauskommt. Dann weiss er das Mindestalter des Admirals, nämlich z.b. „65“. Nun kann er weiter anfragen und zwar „Was ist die maximale Bewertung aller Segler mit Alter größer 65?“. Das Ergebnis ist die gewünschte Bewertung]

221. **Wie kann man genau dieses Problem verhindern?** [Bestimmte Anzahl von Tupeln erzwingen]
222. **Das Problem ist damit allerdings noch nicht ganz gehoben, da man über den Schnitt zweier Ergebnismengen auch auf ein einzelnes Ergebnis kommen kann. Wie kann man auch das verhindern?** [Schnitt muss auch aus mehreren Tupeln bestehen]
223. **Auch jetzt könnte man noch weiter Informationen versuchen zu erspähen, allerdings wird es jetzt unrentabler, da sehr viele Anfragen notwendig sind (c*n/m viele). Wie kann man Schlussendlich dieses Problem noch versuchen in den Griff zu bekommen?** [Anfragen Protokollieren undgefährliche Muster erkennen]
224. **Wie funktioniert Verschlüsselung im Allgemeinen?** [Man verschlüsselt eine Information mit einem Schlüssel und kann diese mit dem gleichen Schlüssel wieder entschlüsseln]
225. **Was ist der unterschied zwischen symmetrischer Verschlüsselung und asymetrischer Verschlüsselung?** [Symmetrisches Verschlüsseln: Ein Schlüssel zu ver- und entschlüsseln | Asymmetrisches Verschlüsseln: Zwei Schlüssel: Public und Private Key, einer zum verschlüsseln und einer zum entschlüsseln.]
226. **Welche 6 Anforderungen muss ein Asymetrisches Verfahren erfüllen?** [
    1. Geringer Rechenaufwand die beiden Schlüssel zu erzeugen
    2. Die Verschlüsselung soll in Kenntnis des öffentlichen Schlüssels sehr schnell möglich sein
    3. Die Entschlüsselung soll in Kenntnis des privaten Schlüssels schnell möglich sein
    4. Es soll einem Gegner praktisch unmöglich sein, aus dem öffentlichen Schlüssel den privaten Schlüssel herzuleiten
    5. Es soll einem Gegner in Kenntnis des öffentlichen Schlüssel und der damit verschlüsselten Nachricht ohne Kenntnis des privaten schlüssels praktisch unmöglich sein, die Nachricht zu entschlüsseln
    6. Verschlüsselung und Entschlüsselung sollen kommutativ sein.]
227. **Was ist eine digitale Signatur?** [Man verschlüsselt nicht mit dem offentlichen Schlüssel sondern mit dem privaten Schlüssel. Jeder der den öffentlichen Schlüssel hat, kann die Nachricht entschlüsseln und weiss dadurch sicher, dass die Nachricht vom richtigen Sender kommt
    a. **Was ist der Message Digest?** [Eine Funktion, die die Nachricht eindeutig auf einen kleinen Wertebereich abbildet ➔ z.B. Hashfunktion mit Modulo. Die Funktion ist nicht umkehrbar]
    b. **Wie geht das nun mit der Signatur und Message Digest?** [Die zu versendende Nachricht wird mit der Message Digest Funktion umgerechnet. Das Ergebnis wird mit dem privaten Schlüssel verschlüsselt. Dann wird die Nachricht mit der Signatur versand. Der Empfänger wendet die öffentlich zugängliche Message Digest Funktion auf die Nachricht an und berechnet den Message Digest dieser. Nun entschlüsselt er die Signatur der Nachricht mit dem öffentlichen Schlüssel und vergleicht sie mit dem eben berechneten Messagedigest. Sind diese identisch, ist die Nachricht authentisch. Zusätzlich kann man die Nachricht selber natürlich auch noch verschlüsseln vor dem versenden mit dem öffentlichen Schlüssel des Empfängers]
228. **Welche Grundlegenden Techniken gibt es zur Auswertung relationaler Operatoren?** [Grundsätzlich geht es darum einen Anfrageoptimierer zu konstruieren, dessen Aufgabe es ist, einen möglichst optimalen Auswertungsplan der einzelnen Anfragen zu bilden. Folgende Techniken sollte der Anfrageoptimierer verwenden:
    Iteration: Alle Tupel in den Eingaberelationen werden iterativ berücksichtigt. Sofern möglich, kann man statt der Tupel ein entsprechender Index verarbeitet werden
    Indexverwendung: Im Falle einer Selektions- oder Verbundbedingung wird ein Index verwendet, um die die Bedingung erfüllenden Tupel zu bestimmen.
    Partitionierung: Eine Menge von Tupeln wird partitioniert, damit Operationen durch eine Reihe von effizienteren Operationen realisiert werden können. Partitionierungen könne mittels Sortierung oder Streuung erreicht werden.]
229. **Welche Informationen werden in einem Katalog gespeichert?** [Anzahl von Tupeln und Anzahl von Seiten jeder Relation | Anzahl unterschiedlicher Schlüsselwerte und Anzahl der Seiten für jeden Schlüsselindex | Höhe und minimalen/maximalen Schlüsselwert für jedem

Baum-Index ➔ Kataloge werden periodisch aktualisiert | Enthalten unter Umständen auch Statistische Informationen über Werteverteilungen z.B. in Form von Histogrammen]

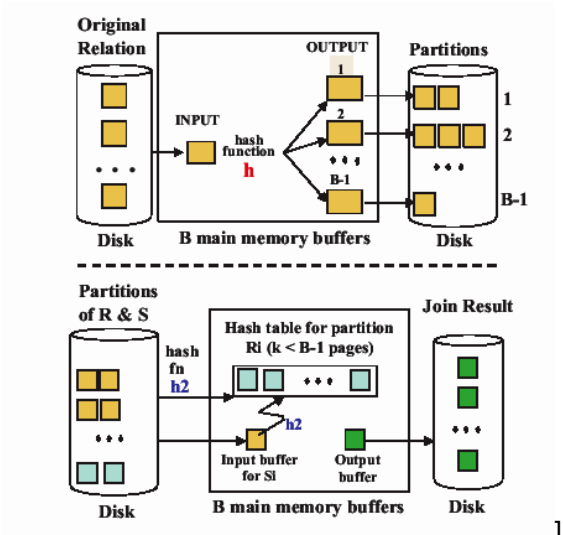
230. **Unter welchen Vorraussetzungen kann man einen Baum- bzw. Hashindex verwenden um einen geeigneten Zugriffspfad auf eine Menge von Tupeln zu finden?** [Je nach dem welche Art von Selektion verwendet wird. Wenn die verwendeten Attribute der Selektion einen Präfix des betreffenden Suchschlüssel ergeben, dann verwendet man den Baumindex (a,b,c), da er Anfragen der Form $a=5 \wedge b>3$ erlaubt, Anfragen der Form b=3 sind nicht möglich, da b kein Präfix ist | Beim Hashindex gehen nur gleichheitsanfragen die auf alle Attributen beruhen, also z.b. $a=5 \wedge b=3 \wedge c=6$]
231. **Welches Maß gibt die Qualität eines Indexes an?** [Die Selektivität: Die die Anzahl der für die Durchführung einer Indexoperation benötigten zu lesenden Seiten. Die Selektivität ist umso größer, je kleiner ihr Wert ist]
232. **Welche verschiedenen Ausgangssituationen gibt es für die Selektion?** [
Kein Index, unsortierte Daten➔ Scan der kompletten Datei in O(n)
Kein Index, sortierte Daten ➔ Binäre Suche möglich in O (log n)+O(n)
B-Baum Index: Suche den Punkt im Präfixbaum, in dem der Suchschlüssel steht und geben auch alle anderen Werte aus, die den Schlüssel als Präfix haben O(log n)+O(n)
gestreute Speicherung (Hashing) ➔ geeignet für Gleichheitsoperationen. Großer Effizienzegewinn bei Ballung, wenn mehrer qualifizierende Sätze existieren]
233. **Wie kann man bei einer Selektion ohne Disjunktionen den Zugriff Beschleunigen, wenn ein oder mehrere passende Indexe vorhanden sind?** [Man nimmt die Konunktion mit dem selektivsten Index als erstes und wendet die restlichen Konjunktionen auf das Ergebnis an. Existiert kein passender Index müssen wir die Datei sequentiell durchscannen | Wenn es mehrer passende Index gibt, dann verwende alle möglichen und schneide die Ergebnisse. Für die Konjunktionen, für die kein Index exitiert, muss man nun Schritt für Schritt auf die Ergebnisschnittmenge prüfen]
234. **Wie sieht es aus bei Selektionen mit Disjunktionen?** [Sobald für eine der Disjunktionen kein Index vorhanden ist, muss man einen Dateiscan anwenden. | Wenn eine Disjunktion konjunktiv mit einem Term bzw, einer Konjunktion von Termen verknüpft, kann man soweit es für diese Konjunktion einen Index, diesen zuerst anwenden | Wenn zu allen Termen der Disjunktion Indexe existieren, dann kann man alle Disjunktionen berechnen und die Ergebnisse vereinigen]
235. **Wie kann man eine Projektion mittels Sortierung optimieren?** [Man sortiert um einfach die Duplikate entferne zu können. Ein kompletter Scan der Datei ist auf jeden Fall erforfderlich. Man erzeugt hierzu eine Datei mit den gewünschten zu projezierenden Tuppeln, sortiert diese und entfernt benachbarte Duplikate]
236. **Wie kann man eine Projektion mittels Streuung (Hashing) optimieren?** [Man erzeugt auch wieder eine neue Datei in der die nicht benötigten Felder entfernt sind. Mittels Streufunktion werden die Elemente verteilt. Duplikate liegen nun auf der jeweils gleichen Kacheln nur wissen wir nicht wo. Nun wenden wir eine zweite Hashfunktion an, die die Elemente auf einen Hauptspeicherhshtable verteilt. Sobald ein Element zweimal auf die gleiche Kachel gelegt würde, wird getestet, ob es ein Duplikat ist oder ein uneindeutigkeit durch die Hashfunktion. Nun braucht man nur noch aus dem Hauptspeicherhashtable die duplikatfreie Projektion in eine Datei schreiben]
237. **Welche 3 Basis-Verfahren gibt es einen Verbund (Join) auszuführen?** [nested-loop-join/sort-merge-join/hash-join]
    a. **Erkläre Nested-Loop-Join!** [Zwei ineinander geschachtelte Schleifen ➔ Sehr ineffizient, da quadratische Laufzeit]
        i. **Welche beiden Varianten gibt es, die den nested-loop-join deutlich verbessern?** [Verbesserung durch Parallelisierung, d.h. eine der Relationen wird partitioniert und jeder Partition ein Prozessor zugewiesen | Seitenorientierte Implementierung: Betrachte alle Tupel von r und s, die sich gerade im Hauptspeicher befinden]
        ii. **Welche weiteren Verfahren des Nested-Loop Join gibt es?** [block-nested-loop-join: Wähle als äußere Relation die kleinere Relation und lege sie im Hauptspeicher ab, falls das nicht geht, muss partitoniert werden. Untertsütze den Zugriff zu den Sätzen der Blöcke durch dynamischen Aufbau eines Hashtables



|
index-nested-loop-join: Falls es einen Index für eine der beiden Relationen über dem Verbundattribut gibt, wähle diese als innere Relation. Alternativ kann man den Index auch beim ersten Durchlauf der inneren Relation on-the-fly aufbauen]

b. **Erkläre Sort-Merge-Join!** [Erst werden beiden Relationen bzgl. des Verbundattributes sortiert, soweit noch nicht geschehen. Lasse zwei Zeiger von Anfang loslaufen. Halte das jeweils größere Element fest und suche in der zweiten Liste ob es dieses gibt. Sobald ein größeres Element gefunden wurde und nicht das gesucht, kann man den Zeiger anhalten und sucht in der anderen Liste bezgl. Dieses Elementes usw. ➔ Bei Multimengen muss darauf geachtet werden, dass jedes mit jedem gepaart wird, dafür benötigt man einen dritten Zeiger]

c. **Erklären Sie die 2 Phasen von hash-join?** [
partitioning(building-phase): Partinieren der beiden Relationen mittels einer Hashfunktion h_1
phase/matching(probing)-phase: Potenzielle Verbundpartner liegen nun in gegenüberliegenden Hashkacheln. Bilden den Join mit Hilfe einer zweiten Hasfunkiton h_2 indem wir die nacheinander die Partitionen von R in die Hauptspeicherstreutabelle verteilen und die korrespondierenden Sätze mittels h_2 bestimmt werden



]

238. **Wie kann man die Mengen Operationen Vereinigung, Differenz, Durchschnitt und Produkt aus den obigen Verfahren ableiten?** [Durchschnitt und kartesisches Produkt sind spezialfälle der Verbundoperation | Vereinigung und Differenz kann man auf zwei Arten Ableitung, einmal mittels Sortierung (Sort-Merge-Join) und einmal mittel Streuung (hash-join]
239. **Wie kann man bei Aggregierungen die GROUP BY Bedingung beschleunigen?** [auch wieder durch hashen bzw. sortieren. In Baum-Index kann zusätzlich unterstützen, wenn der Suchschlüssel alle für die Aggregierung benötigten Attribute enthält]
240. **Welche 3 Bereiche kann man bei einer deklarativen Datenbankanfrag optimieren?** [Semantische Optimierung: Ausnutzen der Integritätsbedingungen, wie z:b. Gilt, dass kein Angestellter mehr verdienen Verdienen darf als sein direkter Vorgesetzter, so braucht man danach erst gar nicht zu fragen |
Logische(algebraische) Optimierung: Umwandlung des SQL Ausdrucks in Relationenalgebra um und forme diesen äquivalenzerhaltend um zu optimieren.

Physische Optimierung: Umwandlung in Algebra. Weise den Operatoren einen sogenannten Iterator zur Bildung eines Auswertungsplanes zu. Der Iterator realisiert in Form eines abstrakten Datentyps einen Operator oder auch eine Indexstruktur]

241. **Wie sehen typische algebraische Umformungen aus die äquivalenzerhaltend sind?**

- $Z \subseteq Y \subseteq X \Longrightarrow \pi[Z](\pi[Y](Q)) \equiv \pi[Z](Q)$.
- $Z, Y \subseteq X \Longrightarrow \pi[Z](\pi[Y](Q)) \equiv \pi[Z \cap Y](Q)$.
- $\sigma[\alpha_1](\sigma[\alpha_2](Q)) \equiv \sigma[\alpha_2](\sigma[\alpha_1](Q))$.
- $\text{attr}(\alpha) \subseteq Y \subseteq X \Longrightarrow \pi[Y](\sigma[\alpha](Q)) \equiv \sigma[\alpha](\pi[Y](Q))$.
- $Q1 \bowtie Q2 \equiv Q2 \bowtie Q1$.
- $(Q1 \bowtie Q2) \bowtie Q3 \equiv Q1 \bowtie (Q2 \bowtie Q3)$.
- $\text{attr}(\alpha) \subseteq X_1, \text{attr}(\alpha) \cap X_2 = \emptyset \Longrightarrow \sigma[\alpha](Q1 \bowtie Q2) \equiv \sigma[\alpha](Q1) \bowtie Q2$.
- $\text{attr}(\alpha) \subseteq X_1 \cap X_2 \Longrightarrow \sigma[\alpha](Q1 \bowtie Q2) \equiv \sigma[\alpha](Q1) \bowtie \sigma[\alpha](Q2)$.

[ ]

242. **Was könnte eine Heuristik zur Optimierung sein?** [Die Selektivität einer Teiloperation gibt an, was man als erstes macht. Die Selektivität berechnet man durch Ermittlung der Kardinalität der Ausgegebene Tupel bzgl. der Kardinalität der Relation(en) selber. Z.B. für die Selektion: $sel_p := \frac{|\sigma[p](R)|}{|R|}$ oder für den Join: $sel_{RS} := \frac{|R \bowtie S|}{|R \times S|} = \frac{|R \bowtie S|}{|R| \cdot |S|}$]

243. **Was ist das Grundprinzip von Pipelining bei Datenbanken?** [Zwischenergebnisse von Teiloperationen werden direkt an Folgeoperationen weitergereicht ohne dass diese materialisiert werden. Die Folgeoperation kann dann bereits früher starten.]