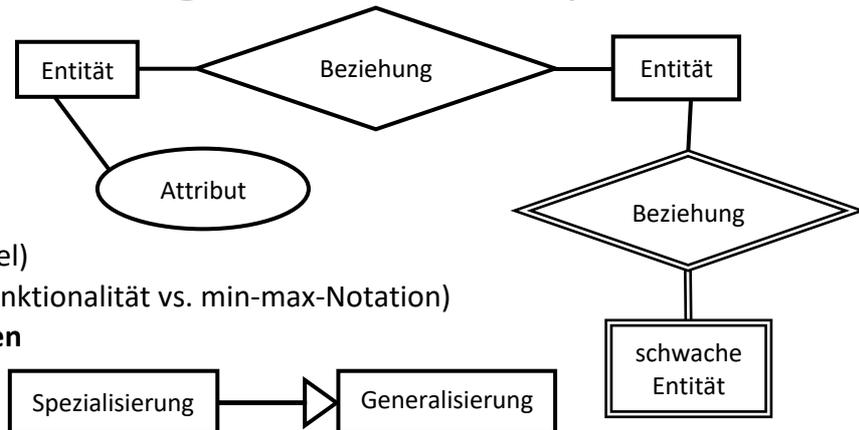


Zusammenfassung Datenbanksysteme

ER-Modell

Bestandteile:

- **Entitäten**
- **Beziehungen**
- **Attribute** (Schlüssel)
- **Kardinalitäten** (Funktionalität vs. min-max-Notation)
- **schwache Entitäten**
- **Generalisierung**



Relationenschema

Relation

Eine **Relation** R ist eine Teilmenge des kartesischen Produkts der Domänen:

$$R \subseteq D_1 \times \dots \times D_n$$

⇒ Eine Datenbank ist eine Menge von Relationen.

Tupel

Ein **Tupel** t ist ein Element einer Relation R : $t \in R$

Schema

Das **Schema** einer Relation besteht aus dem **Namen** der Relation und der Liste der **Attribute**:

$$Name: (A_1: D_1, \dots, A_n: D_n)$$

Schlüssel

Ein **Schlüssel** ist eine **minimale Menge** von Attributen, deren Werte ein Tupel **eindeutig** identifizieren. Ein **Fremdschlüssel** ist eine Menge von Attributen, welche auf den Schlüssel einer anderen Relation verweist. Schlüssel werden im Relationenschema unterstrichen, Fremdschlüssel zusätzlich kursiv geschrieben.

Anomalien

- **Problem-Schlüssel**
- **Update-Anomalie**
- **Einfüge-Anomalie**
- **Lösch-Anomalie**

Relationale Algebra

Basisoperatoren

- σ : **Selektion**
- π : **Projektion**
- \cup : **Vereinigung**
- \cap : **Durchschnitt**
- $-$: **Mengendifferenz**
- \times : **kartesisches Produkt**
- \div : **Division**
- ρ : **Umbenennung**

Selektion: $\sigma_F(R)$

- Schema: $att(\sigma_F(R)) = att(R)$

- Ausprägung: $\sigma_F(R) = \{t \in R \mid t \text{ erfüllt } F\}$

Projektion: $\pi_A(R)$

- Vorbedingung: $A \subseteq \text{att}(R)$
- Schema: $\text{att}(\pi_A(R)) = A$
- Ausprägung: $\pi_A = \{t' \mid \exists t \in R: t.A = t'\}$
- \Rightarrow Duplikate werden eliminiert.

Vereinigung: $R \cup S$

- Vorbedingung: $\text{att}(R) = \text{att}(S)$
- Schema: $\text{att}(R \cup S) = \text{att}(R) = \text{att}(S)$
- Ausprägung: $R \cup S = \{t \mid t \in R \vee t \in S\}$

Mengendifferenz: $R - S$

- Vorbedingung: $\text{att}(R) = \text{att}(S)$
- Schema: $\text{att}(R - S) = \text{att}(R) = \text{att}(S)$
- Ausprägung: $R - S = \{t \mid t \in R \wedge t \notin S\}$

kartesisches Produkt: $R \times S$

- Schema: $\text{att}(R \times S) = \text{att}(R) \cup \text{att}(S)$
- Ausprägung: $R \times S = \{t \mid \exists t_R \in R: t.\text{att}(R) = t_R \wedge \exists t_S \in S: t.\text{att}(S) = t_S\}$
- Ergebnisgröße: $|R \times S| = |R| \cdot |S|$

Division: $R \div S$

- Vorbedingung: $S \subseteq R$
- $R \div S = T \Leftrightarrow R = T \times S$
- Schema: $\text{att}(R \div S) = \text{att}(S)$
- Ausprägung: $R \div S = \{t \mid \forall s \in S: \exists r \in R: r.\text{att}(S) = s \wedge r.(\text{att}(R \setminus \text{att}(S))) = t\}$

Tupelkalkül

Anfragen **relationalen Tupelkalküls** sind von der Form:

$$\{t \mid P(t)\}$$

Die **Tupelvariable** t ist eine freie Variable der Formel $P(t)$ und durch keinen Quantor gebunden. Ein Tupel t ist im Ergebnis, wenn es die Formel $P(t)$ erfüllt.

Domänenkalkül

Anfragen **relationalen Domänenkalküls** sind von der Form:

$$\{[v_1, v_2, \dots, v_n] \mid P(v_1, v_2, \dots, v_n)\}$$

Die **Domänenvariablen** v_1, v_2, \dots, v_n sind freie Variablen der Formel $P(v_1, v_2, \dots, v_n)$ und durch keinen Quantor gebunden. Ein Tupel $[v_1, v_2, \dots, v_n]$ ist im Ergebnis, wenn es die Formel $P(v_1, v_2, \dots, v_n)$ erfüllt.

funktionale Abhängigkeit

Sei \mathcal{R} ein Relationenschema mit $\alpha \subseteq R, \beta \subseteq R$: Eine **funktionale Abhängigkeit** ist eine Beziehung $\alpha \rightarrow \beta$. Eine Relation R erfüllt eine funktionale Abhängigkeit $\alpha \rightarrow \beta$ genau dann, wenn gilt:

$$\forall r, t \in R: r.\alpha = t.\alpha \Rightarrow r.\beta = t.\beta$$

Attributhülle

Die Menge γ^+ der Attribute, welche von γ funktional abhängen nennt man die **Hülle** der Attributmengens γ :

$$\gamma^+ = \{A \in R \mid \gamma \rightarrow A\}$$

Hülle von funktionalen Abhängigkeiten

Die Menge F_2 von FDs ist aus der Menge F_1 ableitbar, wenn jede Relation, die alle FDs in F_1 erfüllt, auch alle FDs in F_2 erfüllt. Die Menge aller aus F ableitbaren FDs wird Hülle F^+ von F genannt:

$$F \vdash \{\gamma \rightarrow \text{AttrHülle}(F, \gamma)\} = F^+ \\ F \vdash \{\alpha \rightarrow \beta\} \Leftrightarrow \beta \subseteq \text{AttrHülle}(F, \alpha)$$

Armstrong Axiome

- **Reflexivität:** $\beta \subseteq \alpha \Rightarrow (\alpha \rightarrow \beta \wedge \alpha \rightarrow \alpha)$
- **Verstärkung:** $\alpha \rightarrow \beta \Rightarrow (\alpha\gamma \rightarrow \beta\gamma)$
- **Transitivität:** $(\alpha \rightarrow \beta \wedge \beta \rightarrow \gamma) \Rightarrow (\alpha \rightarrow \gamma)$
- **Vereinigung:** $(\alpha \rightarrow \beta \wedge \alpha \rightarrow \gamma) \Rightarrow (\alpha \rightarrow \beta\gamma)$
- **Dekomposition:** $(\alpha \rightarrow \beta\gamma) \Rightarrow (\alpha \rightarrow \beta \wedge \alpha \rightarrow \gamma)$
- **Pseudotransitivität:** $(\alpha \rightarrow \beta \wedge \gamma\beta \rightarrow \delta) \Rightarrow (\alpha\gamma \rightarrow \delta)$

Äquivalenz

Zwei Mengen F, G sind **äquivalent**, wenn sie **dieselbe Hülle** besitzen:

$$F \equiv G \Leftrightarrow F^+ = G^+ \\ F \equiv G \Leftrightarrow (F \subseteq G^+ \wedge G \subseteq F^+)$$

kanonische Überdeckung

F_C heißt **kanonische Überdeckung** einer Menge F von FDs, wenn folgende Kriterien erfüllt sind:

- $F_C^+ = F^+$
- F_C enthält keine FDs mit überflüssigen Attributen
- Jede linke Seite einer FD in F_C ist einzigartig.

Berechnung:

1. **Zerlegung** durch **Dekomposition** auf der rechten Seite
2. **Kürzung** überflüssiger Attribute
 - a. **Linksreduktion:**
 $B \in \text{AttrHülle}(F, \alpha - A) \Rightarrow$ ersetze $\alpha \rightarrow B$ durch $(\alpha - A) \rightarrow B$
 - b. **Rechtsreduktion:**
 $B \in \text{AttrHülle}(F - (\alpha \rightarrow B), \alpha) \Rightarrow$ streiche $\alpha \rightarrow B$
3. **Zusammenfassung** mittels Vereinigungsregel

Schlüssel

$\gamma \subseteq R$ ist ein Schlüssel, wenn:

- $\gamma \rightarrow R: \text{AttrHülle}(F, \gamma) = R$
- γ ist minimal: $\forall A \in \gamma: (\gamma - \{A\}) \not\rightarrow R$

Zerlegung

Eine Zerlegung eines Relationenschemas \mathcal{R} ist eine Menge von Schemata $\mathcal{R}_1, \dots, \mathcal{R}_n$, sodass gilt:

$$\text{att}(\mathcal{R}_1) \cup \dots \cup \text{att}(\mathcal{R}_n) = \text{att}(\mathcal{R})$$

Verlustlosigkeit

Eine Zerlegung $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_n$ von \mathcal{R} ist **verlustlos**, falls für jede Ausprägung R von \mathcal{R} , welche $F_{\mathcal{R}}$ erfüllt gilt:

$$R = \pi_{\mathcal{R}_1}(R) \bowtie \pi_{\mathcal{R}_2}(R) \bowtie \dots \bowtie \pi_{\mathcal{R}_n}(R)$$

Eine Zerlegung von \mathcal{R} in \mathcal{R}_1 und \mathcal{R}_2 ist **verlustlos**, wenn die Joinattribute in einer der Teilrelationen (Super-)Schlüssel sind.

Abhängigkeitstreue

Sei F eine Menge an FDs über \mathcal{R} und $\mathcal{R}' \subseteq \mathcal{R}$:

$$F[\mathcal{R}'] = \{\alpha \rightarrow \beta \in F \mid \alpha \cup \beta \subseteq \mathcal{R}'\}$$

Eine Zerlegung $\mathcal{R}_1, \dots, \mathcal{R}_n$ von \mathcal{R} ist **abhängigkeitstreu**, wenn:

$$F \equiv (F^+[\mathcal{R}_1] \cup \dots \cup F^+[\mathcal{R}_n])^+$$

Normalformen

1. Normalform

Ein Relationenschema \mathcal{R} ist in 1. Normalform, wenn die Domänen von \mathcal{R} atomar sind.

2. Normalform

Ein Relationenschema \mathcal{R} ist in 2. Normalform, wenn es in 1. Normalform ist und jedes Nichtschlüsselattribut von jedem Schlüssel funktional abhängig ist.

3. Normalform & Boyce-Codd-Normalform

1. $B \in \alpha$
2. α ist Superschlüssel von \mathcal{R}
3. $\exists \sigma: \sigma$ ist Schlüssel $\wedge B \in \sigma$

Ein Relationenschema \mathcal{R} mit F ist in dritter Normalform, wenn gilt:

$$\forall (\alpha \rightarrow B) \in F, \alpha \subseteq \mathcal{R}, B \in \mathcal{R}: 1 \vee 2 \vee 3$$

Ein Relationenschema \mathcal{R} mit F ist in Boyce-Codd-Normalform, wenn gilt:

$$\forall (\alpha \rightarrow B) \in F, \alpha \subseteq \mathcal{R}, B \in \mathcal{R}: 1 \vee 2$$

Die Normalformen hängen wie folgt zusammen:

$$BCNF \subseteq 3NF \subseteq 2NF \subseteq 1NF$$

Synthesealgorithmus

Der Synthesealgorithmus erzeugt eine verlustlose, abhängigkeitstreu Zerlegung von \mathcal{R} Teilschemata, die jeweils in dritter Normalform liegen:

1. Bestimmung der **kanonischen Überdeckung** F_C
2. für jede FD $\alpha \rightarrow \beta \in F_C$:
 - a. Erstellung des Relationenschemas $\mathcal{R}_i := \alpha \cup \beta$
 - b. Zuordnung von $F_i := F_C[\mathcal{R}_i]$ zu \mathcal{R}_i
3. falls keines der in 2. erzeugten Teilschemata einen Schlüssel von \mathcal{R} enthält: Definition von zusätzlichem Schema $\mathcal{R}_\kappa := \kappa$ mit $F_\kappa := \emptyset$ (beliebiger Schlüssel κ)
4. Elimination von Schemata, die bereits einem anderen Schema enthalten sind

Dekompositionsalgorithmus

Der Dekompositionsalgorithmus erzeugt eine verlustlose, aber nicht zwingend abhängigkeitstreu, Zerlegung von \mathcal{R} in Teilschemata, die jeweils in Boyce-Codd-Normalform liegen:

1. $Z := \{(\mathcal{R}, F)\}$
2. Existenz eines Teilschemas $(\mathcal{R}_i, F_i) \in Z$, dass die BCNF verletzt:
 - a. beliebige Auswahl $\alpha \rightarrow \beta \in F_i$
 - b. $\mathcal{R}_{i_1} := (\alpha \cup \beta); F_{i_1} := F_i^+[\mathcal{R}_{i_1}]$
 - c. $\mathcal{R}_{i_2} := (\alpha \cup \beta); F_{i_2} := F_i^+[\mathcal{R}_{i_2}]$
 - d. Entfernung von (\mathcal{R}_i, F_i) und Ersetzung durch $(\mathcal{R}_{i_1}, F_{i_1})$ und $(\mathcal{R}_{i_2}, F_{i_2})$

Transaktionsverwaltung

Transaktion

Eine **Transaktion** ist eine Folge von Datenbankoperationen, welche die Datenbasis von einem konsistenten Zustand in einen anderen konsistenten Zustand überführen.

ACID

- **Atomicity:** Eine Transaktion ist die kleinste, nicht weiter zerlegbare Einheit.
- **Consistency:** Eine Transaktion hinterlässt nach Beendigung einen konsistenten Zustand der Datenbasis.
- **Isolation:** Nebenläufig ausgeführte Transaktionen dürfen sich nicht beeinflussen.
- **Durability:** Die Auswirkungen einer erfolgreich abgeschlossenen Transaktion gehen nicht mehr verloren.

Komponenten

- **Mehrbenutzersynchronisation:** Stellt Isolation sicher, kontrolliert Nebenläufigkeit.
- **Recovery:** Stellt Atomicity und Durability sicher.

Recovery

Einbringstrategie

- **Update-in-Place:** jede Seite hat genau einen Platz im Hintergrundspeicher
- **Twin-Block-Verfahren:** für jede Seite zwei Kopien im Hintergrundspeicher
- **Schattenspeicherkonzept:** nur für veränderte Seiten zwei Kopien

Auslagerungsstrategie

- **steal vs. nosteal:** Seiten aktiver Transaktionen dürfen nicht ausgelagert werden
- **force vs. noforce:** sofort auslagern

Fehlerkategorien

- lokaler Fehler
- Fehler mit Verlust des DBMS-Puffers
- Fehler mit Hintergrundspeicherverlust

angenommene Systemkonfiguration

steal, noforce, update-in-place, kleine Sperrgranulate auf Datensatzebene

Protokollierung

- **physisch:** before- & after-image
- **logisch:** wie eines aus dem anderen erzeugt wird

Struktur der Logeinträge

[LSN, TransaktionsID, PageID, Redo, Undo, PrevLSN]

<LSN, TransaktionsID, PageID, Redo, PrevLSN, UndoNxtLSN>

Wideranlauf

1. **Analyse:** Identifiziere Loser-Menge, finde höchste LSN für jede Transaktion
2. **Redo:** Datenbasis wird auf Stand bei Absturz gebracht
3. **Undo:** Wiederherstellung des before-image für Loser in umgekehrter chronologischer Reihenfolge – spezieller Log-Eintrag

Sicherungspunkte

- **transaktionskonsistent:** Hintegrundspeicher enthält alle Änderungen von zum Zeitpunkt S_i abgeschlossenen Transaktionen. Zum Zeitpunkt S_i darf es keine aktiven Transaktionen geben.
- **aktionskonsistent:** Alle aktiven Änderungsoperationen sollen beendet werden.

- **unscharfe** (fuzzy): Zum Sicherungspunkt werden nur die Kennungen aller modifizierten Seiten ausgeschrieben.

Mehrbenutzersynchronisation

Fehler bei unkontrollierter Nebenläufigkeit

- Lost Update (W-W)
- Dirty Read (W-R)
- Unrepeatable Read (R-W)
- Phantomproblem (R-W)

Serialisierbarkeit

- **Serialisierbarkeit:** Eine Historie ist serialisierbar, falls ihre Auswirkung identisch zu einer seriellen Historie derselben Transaktionen ist.
- **Konfliktserialisierbarkeit:** Eine Historie ist genau dann konfliktserialisierbar, wenn der Serialisierbarkeitsgraph azyklisch ist.
- **Rücksetzbarkeit:** Eine Historie ist rücksetzbar, wenn keine Transaktion ihr commit vor einer Transaktion durchführt, von welcher sie gelesen hat.
- **kaskadierendes Rücksetzen:** Änderungen einer Transaktion werden erst nach einem commit zum Lesen freigegeben.
- **strikte Historie:** Auf eine von einer Transaktion geschriebene Seite dürfen andere Transaktionen erst nach deren Beendigung schreibend oder lesend zugreifen.

Sperrmodi

| Sperrmodus | | aktuell | | |
|-------------|---|---------|---|---|
| | | NL | S | X |
| angefordert | S | ✓ | ✓ | X |
| | X | ✓ | X | X |

Zwei-Phasen-Sperrprotokoll (2PL)

- Jedes von Transaktion benützte Objekt muss vor Verwendung gesperrt werden.
- Eine Transaktion fordert keine Sperre an, die sie schon besitzt.
- Kann eine Sperre nicht gewährt werden, wartet die Transaktion, bis die Sperre gewährt werden kann.
- Nachdem eine Transaktion eine Sperre freigegeben hat, darf sie keine weiteren Sperren mehr anfordern (Wachstums- vs. Schrumpfphase).
- Am Transaktionsende müssen alle Sperren freigegeben worden sein.

⇒ garantiert Konfliktserialisierbarkeit, aber nicht Rücksetzbarkeit

strenges Zwei-Phasen-Sperrprotokoll (strict 2PL)

- Alle Sperren werden bis Transaktionsende gehalten.

⇒ lässt nur strikte Historien zu

Conservative 2PL (Preclaiming)

- Transaktion fordert alle Sperren, die sie brauchen wird, bereits zu Beginn, und hält alle Sperren bis zum Schluss.

⇒ Deadlock-Vermeidung

Multiple Granularity Locking (MGL)

| Sperre | | aktuell | | | | |
|-------------|----|---------|---|---|----|----|
| | | NL | S | X | IS | IX |
| angefordert | S | ✓ | ✓ | X | ✓ | X |
| | X | ✓ | X | X | X | X |
| | IS | ✓ | ✓ | X | ✓ | ✓ |
| | IX | ✓ | X | X | ✓ | ✓ |

- **Anforderung:** Bevor eine Transaktion einen Knoten sperren kann, benötigen alle Vorgängerknoten eine IS-/IX-Sperre.
- **Freigabe:** Eine IS-/IX-Sperre darf erst freigegeben werden, wenn alle Sperren auf Nachfolgeknoten bereits freigegeben worden sind.

Einfügen/Löschen

- **Löschen:** erfordert X-Sperre, andere Transaktionen können Sperre nach erfolgreichem Abschluss nicht mehr erhalten
- **Einfügen:** Transaktion erhält X-Sperre – zusätzliche Maßnahmen zur Vermeidung des Phantomproblems

Zeitstempel-basierende Synchronisation

- Jede Transaktion T erhält eindeutigen Zeitstempel $TS(T)$
- Jedem Datum A werden zwei Werte zugewiesen:
 - $readTS(A)$: TS der letzten Transaktion die A gelesen hat
 - $writeTS(A)$: TS der letzten Transaktion die A geschrieben hat
- **Lesezugriff:** $TS(T_i) \geq writeTS(A)$
- **Schreibzugriff:** $TS(T_i) \geq readTS(A) \wedge TS(T_i) \geq writeTS(A)$

⇒ keine Deadlocks, konfliktserialisierbare Historie, Historien ev. nicht rücksetzbar

optimistische Synchronisation

1. **Lesephase:** alle Operationen der Transaktion ausgeführt, Schreiboperationen nur auf lokalen Kopien
2. **Validierungsphase:** Überprüfung, ob Transaktion committed werden darf, Konflikte mittels Zeitstempel erkannt
3. **Schreibphase:** nach erfolgreicher Validierung Änderungen in Datenbasis eingebracht

Isolation Levels

- **read uncommitted:** schwächste Stufe, kann nicht festgeschriebene Änderungen lesen und daher auch inkonsistente Datenbankzustände sehen, nur für READ ONLY Transaktionen erlaubt
- **read committed:** Jede Operation sieht nur Datensätze, die vor dem Beginn der Operation committed waren.
- **repeatable read:** Alle Operationen der Transaktion sehen nur Datensätze, die vor der ersten Aktion der Transaktion committed waren.
- **serialisable:** höchste Stufe

| Isolation Level | Dirty Read | Unrepeatable Read | Phantom Problem |
|------------------|------------|-------------------|-----------------|
| read uncommitted | möglich | möglich | möglich |
| read committed | - | möglich | möglich |
| repeatable read | - | - | möglich |
| serializable | - | - | - |

Vergleich – Protokolle

| Protokoll | äquivalente serielle Historie | weitere Eigenschaften | Deadlock möglich |
|---|---|------------------------------|-------------------------|
| 2PL | Reihenfolge der Sperranforderungen bei Konflikten | i.a. nicht rücksetzbar | ja |
| Stric 2PL | wie 2PL | strikt | ja |
| Strict 2PL + Deadlock-Vermeidung | wie 2PL | strikt | nein |
| Zeitstempel-basierend | Zeitpunkt von BOT | strikte Variante existiert | nein |
| optimistisch | Zeitpunkt der Validierung | strikt | nein |