Gliederung

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

1. Die Bedeutung des Datenmanagements
2. Datenbank-Architektur
3. Modellierung und Entwurf von DB-Systemen
4. Relationale Algebra und Normalisierung
5. Definition und Abfrage von Datenbank-Systemen
6. Dateiorganisation und Zugriffsstrukturen
7. Optimierung von Anfragen
8. **Transaktionen**
9. weitere Aspekte der Datenbanken

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

1. Transaktionsmodelle

1. *Transaktionen im Mehrbenutzerbetrieb*
2. *Transaktionseigenschaften*
3. *Probleme im Mehrbenutzerbetrieb*
4. *Problemvermeidung*
5. *Verklemmung (Deadlock)*

2. Sicherung der Integrität

1. *Sicherung der Zugriffsintegrität*
2. *Sicherung der physischen Integrität*
3. *Aufrechterhaltung der semantischen Integrität*

8.1 Transaktionsmodelle

Semantische Integritätsbedingung

- Forderung: **fehlerhafte Datenbankzustände** sind **auszuschließen** (Semantische Ebene)
- Problem: **fehlerhafte Datenbankzustände** beim **gleichzeitigen Zugriff mehrerer Benutzer**

→ **Ablaufintegrität (Operationale Integrität)**

Ablaufintegrität (1)

- i.d.R mehrere **Programme simultan** (nebenläufige, konkurrierende Prozesse)
- Probleme (exemplarisch):
  - **Platzreservierung** für Flug- und Bahnreisen
  - **Überschneidende Kontooperationen** einer Bank
  - bei **statistischen Datenbankoperationen**, wenn während der Berechnung Daten geändert werden

Ablaufintegrität (2)

- Sicherung durch das Konzept der **Transaktion**
- **Transaktion:** Zusammenfassung zusammengehörender Datenbankoperationen, deren Ausführung **durch das DBMS** gesteuert und überwacht wird
- **Concurrency Control** (Mehrbenutzerkontrolle): sichert die Korrektheit der nebenläufigen Ausführung vieler Transaktionen

Definition Transaktion

- **Folge von Operationen** (Aktionen), die die Datenbank von einem **konsistenten** in einen **konsistenten**, veränderten Zustand überführt, wobei das **ACID-Prinzip** eingehalten wird

ACID-Eigenschaften (1)

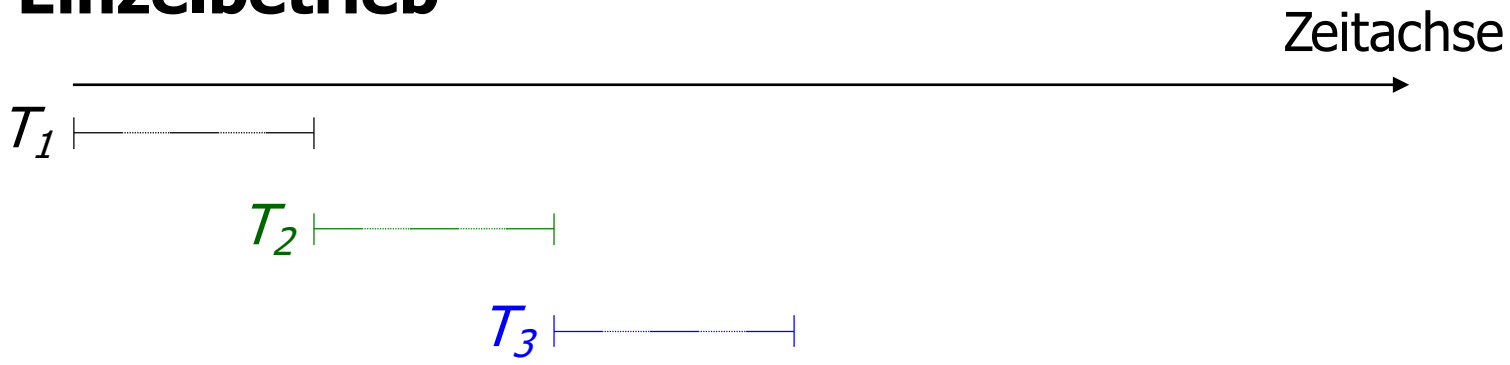
- **Atomicity** (Atomarität)
  - Transaktion wird entweder ganz oder gar nicht ausgeführt
  - nach Abbruch einer Transaktion:
    - keine Zwischenergebnisse in der Datenbank
    - Rollback: Zustand der Datenbank vor Beginn der Ausführung der Transaktion wird wieder hergestellt
- **Consistency** (Konsistenz oder Integritätserhaltung)
  - Nach Abschluss der Transaktion liegt wieder ein konsistenter Zustand vor, während der Transaktion sind Inkonsistenzen erlaubt.

ACID-Eigenschaften (2)

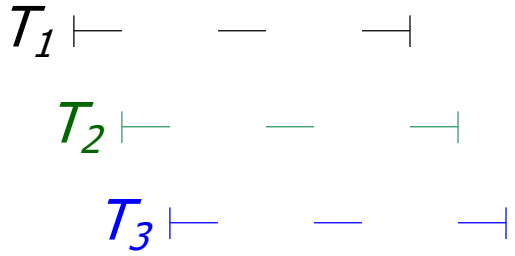
- **Isolation** (Isolation)
  - Nebenläufig ausgeführte Transaktionen dürfen sich nicht beeinflussen, d. h. jede Transaktion hat den Effekt, den sie verursacht hätte, als wäre sie allein im System.
- **Durability** (Dauerhaftigkeit/Persistenz)
  - Die Wirkung einer erfolgreich abgeschlossenen Transaktion bleibt dauerhaft in der Datenbank (auch nach einem späteren Systemfehler).

Ausführung der Transaktionen T_1, T_2 und T_3:

(a) im Einzelbetrieb



(b) im (verzahnten) Mehrbenutzerbetrieb (gestrichelte Linien repräsentieren Wartezeiten)



DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

… und in der Praxis …

- Transaktionen nicht immer atomar und isoliert:
  - parallel angestoßene Transaktionen **überlappen aus Effizienzgründen:** Isolation verletzt
  - Transaktion wird abgebrochen durch **Hardware- bzw Laufzeitfehler** oder durch **das Datenbanksystem (z.B Scheduler):** Atomicity verletzt

Dirty Read (1) (Abhängigkeiten von nicht freigegebenen Daten)

- Situation: Transaktionen wurden abgebrochen, bevor sie die endgültigen Daten geschrieben haben
- Problem: andere Transaktionen könnten diese nicht freigegebenen Daten lesen

→ **„dirty read“**

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Dirty Read (2) (Abhängigkeiten von nicht freigegebenen Daten)

Schritt	T_1	T_2
1.		
2.	$a_1 := a_1 - 300$	
3.	write(A,a_1)	
4.		read(A,a_1)
5.		$a_1 := a_1 * 1.03$
6.		write(A,a_1)
7.	read(B,b_1)	
8.	. . .	
9.	abort	

- Transaktion T_2 schreibt die Zinsen gut anhand eines Betrages, der nicht in einem konsistenten Zustand der Datenbasis vorkommt, da Transaktion T_1 später durch ein **abort** zurückgesetzt wird.

Das Phantom-Problem (1)

- **Phantome**: **Datenbankobjekte**, die **existieren können**, aber während einer **Transaktion zeitweise nicht existieren**
- bei Datenbankoperationen
  - Transaktion ermittelt einen Wert (z.B. count)
  - eine andere Transaktion erzeugt oder löscht Datenbankobjekte, bevor der Wert verarbeitet wurde

Das Phantom-Problem (2)

T_1	T_2
X = **SELECT** COUNT(*) **FROM** Kunde;	
	INSERT INTO Kunde **VALUES** (*6789*,‘Lilo‘,...)
UPDATE Kunde **SET** Bonus = Bonus + 1000/X	

Während der Abarbeitung der Transaktion T_1 fügt Transaktion T_2 ein Datum ein, welches T_2 später liest. Dadurch verteilt Transaktion T_2 mehr Geld als verfügbar ist.

Werden während der Transaktion veränderte Daten ausgegeben, bezeichnet man dies als Phantom-Problem, da diese Daten somit aus dem Zusammenhang gerissen werden.

Lost Update (1) (Verloren gegangene Änderungen)

- Situation: zwei Transaktionen **ändern „quasi gleichzeitig“** einen Wert und schreiben ihn dann nacheinander in die Datenbank zurück
- Problem: zwischen dem Lesen und Schreiben einer Transaktion kann eine andere Transaktion die Daten lesen, die momentan ungültig sind

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Lost Update (2) (Verloren gegangene Änderungen)

Schritt	T_1	T_2
1.	read(A,a_1)	
2.	$a_1 := a_1 - 300$	
3.		read(A,a_1)
4.		$a_1 := a_1 * 1.03$
5.		write(A,a_1)
6.	write(A,a_1)	
7.	read(B,b_1)	
8.	$b_1 := b_1 + 300$	
9.	write(B,b_1)	

Transaktion T_1 transferiert 300 € von Konto A nach Konto B, Transaktion T_2 schreibt Konto A die 3 % Zinseinkünfte gut.

Die im Schritt 5 von Transaktion T_2 gutgeschriebenen Zinsen gehen verloren, da sie in Schritt 6 von Transaktion T_1 wieder überschrieben werden.

Mehrbenutzer-Anomalie (1)

- Situation**: mehrere Transaktionen** greifen **gleichzeitig** auf dieselben Daten zu und verändern diese
- Problem **Inkonsistenzen:** Transaktion liest **Daten von einer anderen Transaktion**, die noch **nicht freigegeben** sind und diese verändert

Mehrbenutzer-Anomalie (2)

- Beispiel
  - Es soll gelten A = B
  - Integritätsbedingung verletzt, da nicht mehr *A* = *B* gilt

T_1	T_2	A	B
read(A);		10	10
$A := A + 10$;			
write(A);		20	
	read(A);		
	$A := A \cdot 1.1$;		
	write(A);	22	
	read(B);		
	$B := B \cdot 1.1$;		
	write(B);		11
read(B);			
$B := B + 10$;			
write(B);		22	21

Rollback-Segmente

- Datenblöcke, die beim Start der Abfrage aktuell waren, werden bevor sie verändert werden, in sogenannte Rollback-Segmente geschrieben.
- Wenn während einer Abfrage Daten verändert werden, nimmt das System bei Feldern, deren Änderungszeitpunkt nach dem Abfragezeitpunkt liegt, einen Rollback auf den letzten Zeitpunkt vor dem Abfragezeitpunkt vor.

Lock-Verfahren

- Sperren von Datenfeldern, Zeilen oder Tabellen

Serialisierbarkeit - Beispiele (1)

- Gegeben:

  T_1: **`read`** A; A:=A-10; **`write`** A; **`read`** B; B:=B+10; **`write`** B;

  T_2: **`read`** B; B:=B-20; **`write`** B; **`read`** C; C:=C+20; **`write`** C;

- **serielle Ausführung**
  - beide Transaktionen werden hintereinander ausgeführt
  - der Wert von $A + B + C$ bleibt unverändert, egal ob T_1 vor T_2 oder umgekehrt ausgeführt wird

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Serialisierbarkeit - Beispiele (2)

Variante 1: serielle Ausführung

Varianten 2 + 3: verschränkte Ausführungen (gemischt)

Ausführung 1		Ausführung 2		Ausführung 3	
T_1	T_2	T_1	T_2	T_1	T_2
read A		**read** A		**read** A	
$A - 10$			**read** B	$A - 10$	
write A		$A - 10$			**read** B
read B			$B - 20$	**write** A	
$B + 10$		**write** A			$B - 20$
write B			**write** B	**read** B	
	read B	**read** B			**write** B
	$B - 20$		**read** C	$B + 10$	
	write B	$B + 10$			**read** C
	read C		$C + 20$	**write** B	
	$C + 20$	**write** B			$C + 20$
	write C		**write** C		**write** C

Serialisierbarkeit - Beispiele (3)

- Ergebnisse der Abläufe unterscheiden sich

	A	B	C	$A+B+C$
initialer Wert	10	10	10	30
nach Ausführung 1	0	0	30	30
nach Ausführung 2	0	0	30	30
nach Ausführung 3	0	20	30	50

- Ausführungsfolgen 1 und 2 zeigen denselben Effekt
- Ausführung 3: Transaktion T_1 liest den alten Wert von B bevor T_2 den aktualisierten Wert schreibt

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Serialisierbarkeit – Ein einfaches Modell (1)

- man betrachtet **nur Lese- und Schreiboperationen**
- jede lesende Transaktion **schreibt** den Wert in die Datenbank **zurück** (Schreiben ohne vorheriges Lesen ist verboten)
- **einfaches Sperrmodell**: Datenobjekte werden zwischen Lese- und Schreibzeitpunkt für andere Transaktionen gesperrt (**Lock-Unlock-Modell**)

DHBW
Duale Hochschule
Baden-Württemberg
Lörrach

Serialisierbarkeit – Ein einfaches Modell (1)

- Beispiel: Transaktion von oben

  T_1: **lock** A; **unlock** A; **lock** B; **unlock** B;

  T_2: **lock** B; **unlock** B; **lock** C; **unlock** C;

Ausführung 1		Ausführung 2		verbotene Ausführung	
T_1	T_2	T_1	T_2	T_1	T_2
lock A		lock A		lock A	
unlock A			lock B		lock B
lock B		unlock A		unlock A	
unlock B			unlock B	lock B	
	lock B	lock B			unlock B
	unlock B		lock C		lock C
	lock C	unlock B		unlock B	
	unlock C		unlock C		unlock C

Deadlock

- wenn zwei Transaktionen zur gleichen Zeit versuchen, während ihres Abarbeitens auf die gleichen Adressbereiche/ Ressourcen zuzugreifen
- Folge: Keine der Transaktionen kann abgeschlossen werden, sie sperren sich gegenseitig, da eine Transaktion erst mit erfolgreichem Abschluss aller Statements als abgeschlossen gilt → unvollendete Transaktionen.
- System muss über „rollback“-Mechanismen den Zustand vor den jeweiligen Transaktionen wieder zurücksetzten.

8.2 Sicherung der Integrität

8.2.1 Sicherung der Zugriffsintegrität

- Data Security (Datenschutz) bezieht sich auf die Verwendung der Daten
- Privilegien und Rollen (privileges and roles)
- Benutzer und Benutzergruppen
- Vererbung von Rechten von einem User zum anderen
- Quotas → Plattenplatzbegrenzungen für einzelne User
- Selective Auditing -> Benutzerüberwachung seitens des DB-Administrators

- Data Safety (Datensicherheit) bezieht sich auf die physische Existenz der Daten auf Speichermedien
  - Möglicher Fehler
    - Statement Error: Fehlermeldung
    - Process Error (User-, Server, Backgroundprozesse): Process- und System-Monitor
    - Network Error: System Monitor
    - Medien Disk Error: Backups und Rede-log Files (online Archiv)
    - User Error: Schulung

- NOT NULL (Mandatory) -> Entity Integrität
- UNIQUE (Primary Key) -> Schlüsseleindeutigkeit
- Domänen (m und w oder 0 und 1)
- Dead Links: ein Fremdschlüssel muss als Primär-schlüssel in einer anderen Tabelle enthalten sein

→ Referentielle Integrität und referentielle Aktionen

Gewährleistung durch

- **Primäre Konsistenzbedingungen**
  - Erfolgreiche Datenmodellierung und Implementierung (Normalisierung usw.)
- **Sekundäre Konsistenzbedingungen**
  - Prüfroutinen (Plausibilitätsprüfungen)
  - Benutzerschulungen
  - Überwachung durch DB-Administrator