

8. Datenkontrolle

- ACID und Datenkontrolle
- Zugriffskontrolle/Autorisierung in SQL
 - GRANT (with GRANT OPTION)
 - REVOKE
- Synchronisation
 - Anomalien
 - Isolation Level
- Recovery

ACID und Datenkontrolle

- Transaktionskonzept (ACID-Eigenschaften)
 - im SQL-Standard: COMMIT WORK, ROLLBACK WORK, Beginn einer Transaktion implizit
 - Einhaltung der logischen DB-Konsistenz (Consistency)
 - Verdeckung der Nebenläufigkeit (concurrency isolation)
 - Verdeckung von (erwarteten) Fehlerfällen -> Logging und Recovery
- Integritätskontrolle
 - *semantische Integritätskontrolle*: möglichst hohe Übereinstimmung von DB-Inhalt und Miniwelt (Datenqualität)
 - bei COMMIT müssen alle semantischen Integritätsbedingungen erfüllt sein (Transaktionskonsistenz)
 - Einhaltung von *physischer Integrität* und *Ablaufintegrität* (operationale Integrität)
- Zugriffskontrolle
 - Maßnahmen zur Datensicherheit und zum Datenschutz
 - Sichtkonzept
 - Vergabe und Kontrolle von Zugriffsrechten

ACID und Datenkontrolle (2)

Art der Integrität	Transaktions-eigenschaft	realisierende DBS-Komponente
semantische Integrität	C (Consistency, Konsistenz)	(semantische) Integritätskontrolle
physische Integrität	D: Dauerhaftigkeit A: Atomarität	Logging, Recovery (+ korrekte Implementierung der DB-Operationen)
Ablaufintegrität	I (Isolation)	Synchronisation (z. B. Sperrverwaltung)

Zugriffskontrolle in SQL

- Sicht-Konzept: wertabhängiger Zugriffsschutz
 - Untermengenbildung
 - Verknüpfung von Relationen
 - Verwendung von Aggregatfunktionen
- **GRANT**-Operation: Vergabe von Rechten auf Relationen bzw. Sichten

```
GRANT {privileges-commalist | ALL PRIVILEGES}
ON accessible-object TO grantee-commalist [WITH GRANT OPTION]
```
- Zugriffsrechte: SELECT, INSERT, UPDATE, DELETE, REFERENCES, USAGE
 - Erzeugung einer „abhängigen“ Relation erfordert REFERENCES-Recht auf von Fremdschlüsseln referenzierten Relationen
 - USAGE erlaubt Nutzung spezieller Wertebereiche (character sets)
 - Attributeinschränkung bei INSERT, UPDATE und REFERENCES möglich
 - dynamische Weitergabe von Zugriffsrechten: WITH GRANT OPTION (dezentrale Autorisierung)
- Empfänger: Liste von Benutzern bzw. PUBLIC

Vergabe von Zugriffsrechten: Beispiele

- GRANT SELECT ON DEPT TO PUBLIC
- GRANT INSERT, DELETE ON DEPT TO Mueller, Weber WITH GRANT OPTION
- GRANT UPDATE (GEHALT) ON PERS TO Schulz
- GRANT REFERENCES (PRONR) ON PROJECT TO PUBLIC

Rücknahme von Zugriffsrechten: Revoke

```
REVOKE      [GRANT OPTION FOR] privileges-commalist
ON accessible-object
FROM grantee-commalist {RESTRICT | CASCADE }
```

- ggf. fortgesetztes Zurücknehmen von Zugriffsrechten

Beispiel: REVOKE SELECT
ON DEPT
FROM Weber CASCADE

- wünschenswerte Entzugssemantik:
 - der Entzug eines Rechtes ergibt einen Zustand der Zugriffsberechtigungen, als wenn das Recht nie erteilt worden wäre
- Probleme:
 - Rechteempfang aus verschiedenen Quellen
 - Zeitabhängigkeiten

Revoke (2)

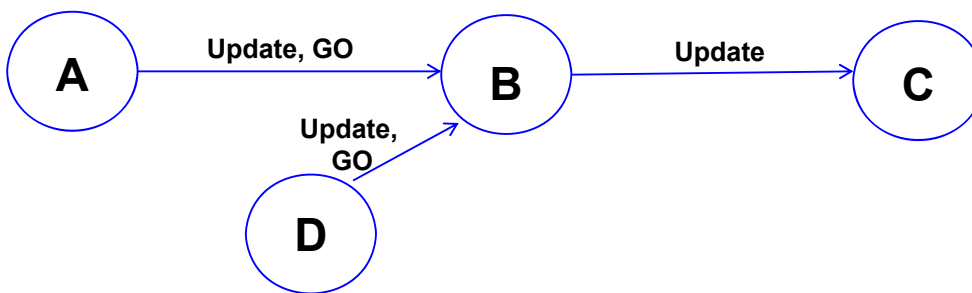
- Führen der Abhängigkeiten in einem *Autorisierungsgraphen* pro Objekt (z.B. Tabelle)

- *Knoten: User-Ids*
- *gerichtete Kanten: Weitergabe von Zugriffsrechten (Recht, ggf. Zeitpunkt)*

- Beispiel für Tabelle R

- User A: GRANT UPDATE ON R TO B WITH GRANT OPTION
- User B: GRANT UPDATE ON R TO C
- User D: GRANT UPDATE ON R TO B WITH GRANT OPTION
- User A: REVOKE UPDATE ON R FROM B CASCADE

- Autorisierungsgraph für R:



Synchronisation

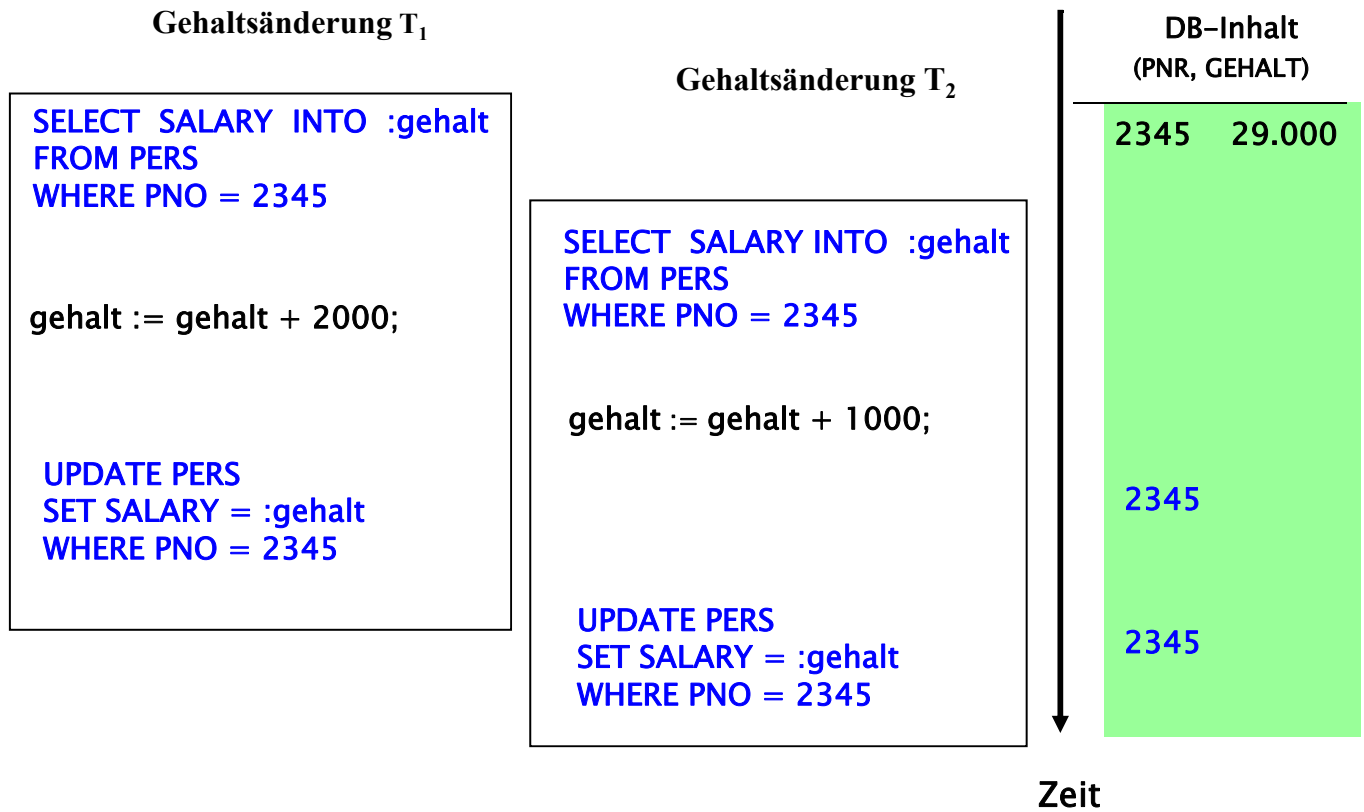
- DBS müssen Mehrbenutzerbetrieb unterstützen

- ohne Synchronisation kommt es zu so genannten **Mehrbenutzer-Anomalien**

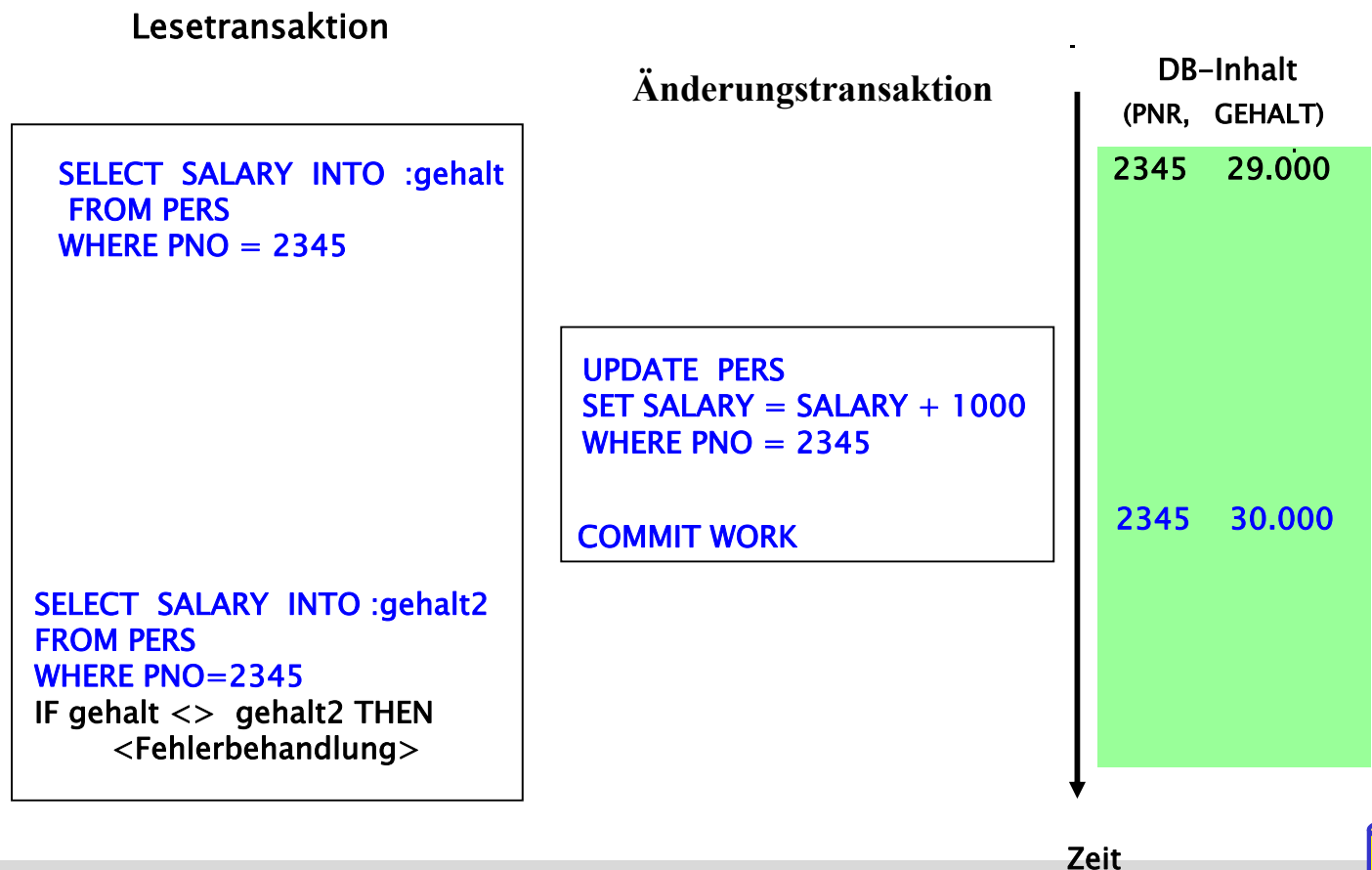
- verlorengangene Änderungen (lost updates)
- Zugriffe auf nicht freigegebene Änderungen (dirty read, read uncommitted)
- inkonsistente Analyse (non-repeatable read)
- Phantom-Probleme

- Anomalien sind nur durch Änderungen verursacht

Verloren gegangene Änderung (Lost Update)



Non-repeatable Read



Phantom-Problem

Lesetransaktion
(Gehaltssumme überprüfen)

```
SELECT SUM (SALARY) INTO :summe  
FROM PERS  
WHERE DNO = 17
```

```
SELECT SUM (SALARY) INTO :summe2  
FROM PERS  
WHERE DNO = 17
```

```
IF summe <> summe2 THEN  
  <Fehlerbehandlung>
```

Änderungstransaktion
(Einfügen eines neuen Angestellten)

```
INSERT INTO PERS (PNO, DNO, SALARY)  
VALUES (4567, 17, 55.000)
```

```
COMMIT WORK
```

Zeit

Synchronisation

- Synchronisation erfolgt automatisch durch DBS, z.B. durch
 - Setzen von (Lese/Schreib-) Sperren vor Datenzugriff
 - Freigabe der Sperren am Transaktionsende
- idealerweise werden alle Anomalien beseitigt (logischer Einbenutzerbetrieb)
 - Synchronisation gewährleistet „*Serialisierbarkeit*“
 - gleichzeitige Ausführung von Transaktionen hat den gleichen Effekt auf die DB wie eine serielle Ausführung dieser Transaktionen
- Inkaufnahme einiger Anomalien verbessert jedoch i.a. Leistungsfähigkeit
 - weniger Sperren, Blockierungen (Warteverzögerungen auf frei werdende Sperren) und Deadlocks

Konsistenzstufen in SQL92

- vier Konsistenzebenen (Isolation Level) zur Synchronisation
 - Default ist Serialisierbarkeit (serializable)
 - Lost-Update muss generell vermieden werden
 - READ UNCOMMITTED für Änderungstransaktionen unzulässig

Isolation Level	Anomalie		
	Dirty Read	Non-Repeatable Read	Phantome
Read Uncommitted	+	+	+
Read Committed	-	+	+
Repeatable Read	-	-	+
Serializable	-	-	-

- SQL-Anweisung zum Setzen der Konsistenzebene:

```
SET TRANSACTION [READ WRITE | READ ONLY] ISOLATION LEVEL <level>
```

Beispiel: SET TRANSACTION Read Only ISOLATION LEVEL Read Committed



Recovery-Unterstützung

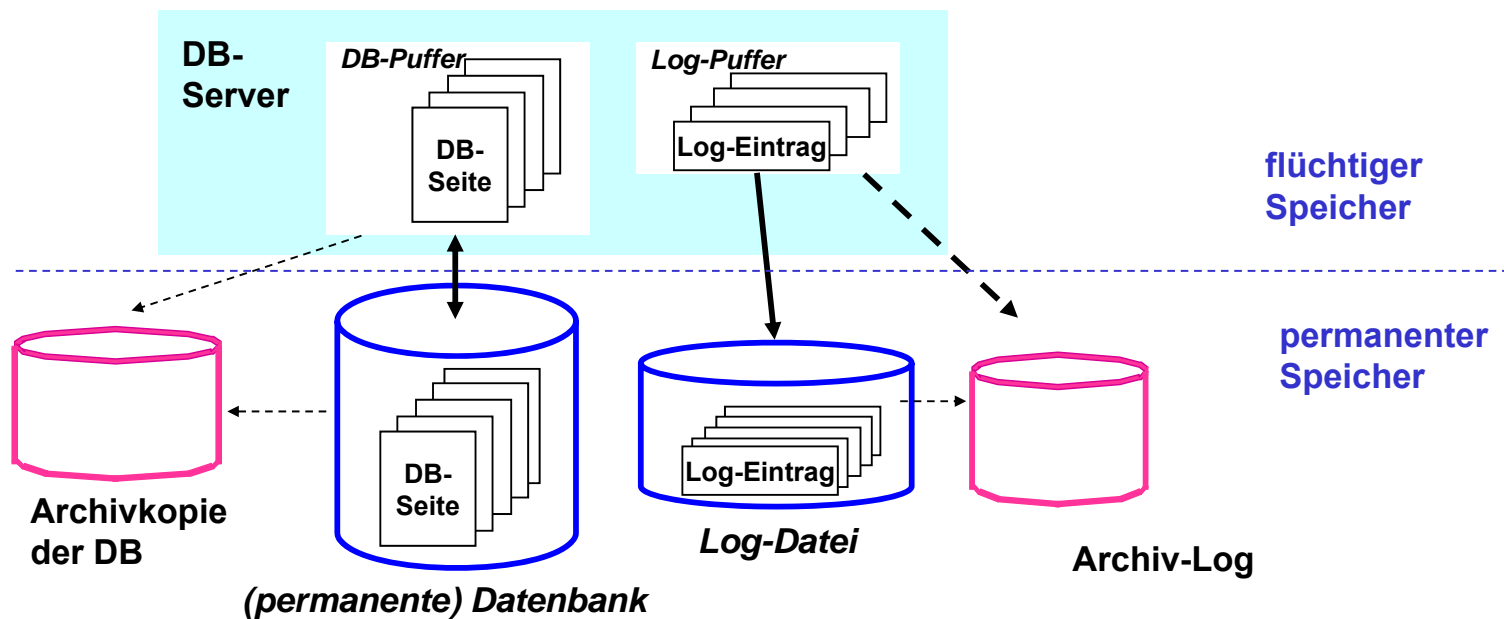
- automatische Behandlung aller erwarteten Fehler durch das DBS
- Transaktionsparadigma verlangt:
 - Alles-oder-Nichts-Eigenschaft von Transaktionen
 - Dauerhaftigkeit erfolgreicher Änderungen
- Zielzustand nach Recovery: jüngster, transaktionskonsistenter Zustand vor Erkennen des Fehlers
- Voraussetzung: Sammeln redundanter Informationen während Normalbetrieb (Logging)



Fehlerarten

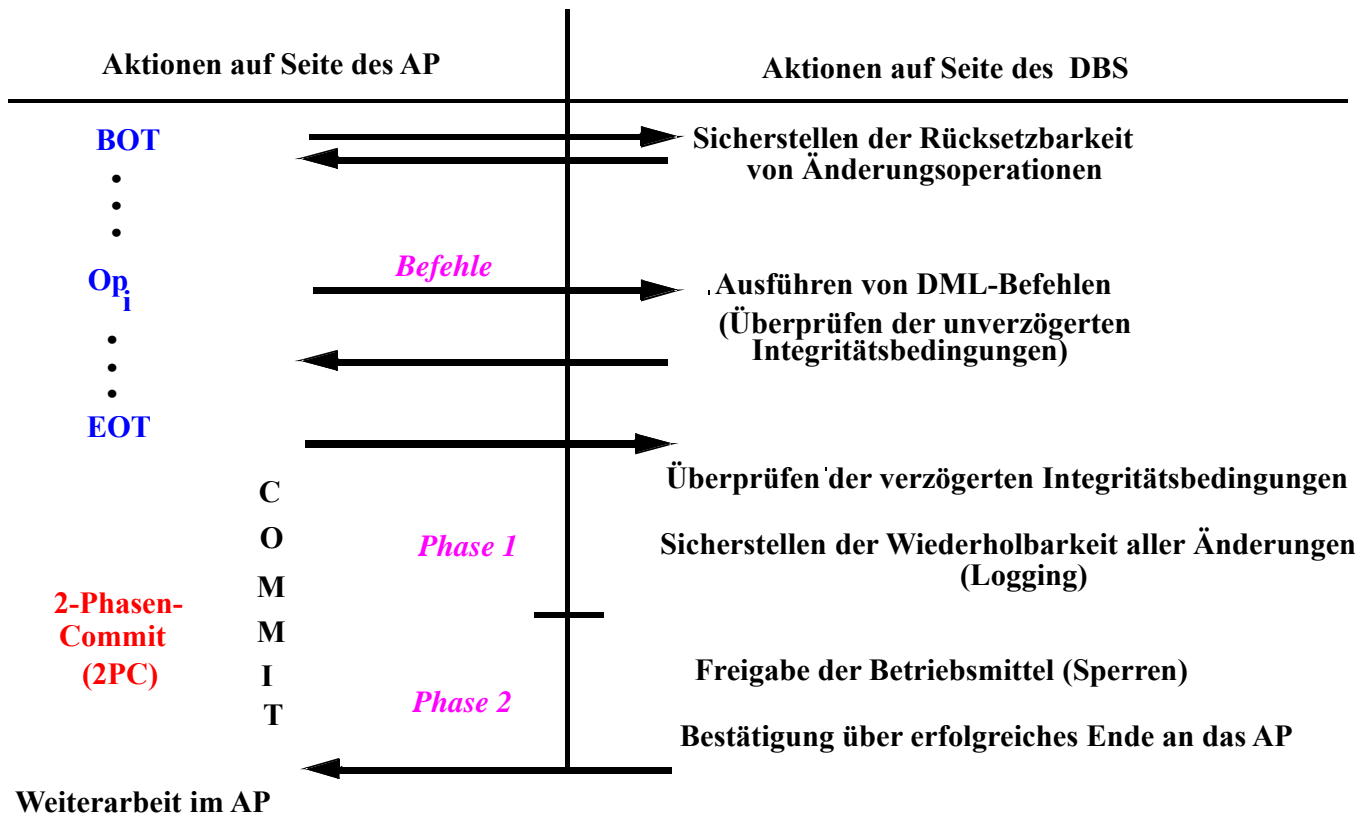
- **Transaktionsfehler**: vollständiges Zurücksetzen auf Transaktionsbeginn (Undo)
- **Systemfehler** (Rechnerausfall, DBVS-Absturz)
 - REDO für erfolgreiche Transaktionen (Wiederholung verlorengangener Änderungen)
 - UNDO aller durch Ausfall unterbrochenen Transaktionen (Entfernen derer Änderungen aus der permanenten DB)
- **Gerätefehler** (Plattenausfall):
 - vollständiges Wiederholen (REDO) aller Änderungen auf einer Archivkopie
 - oder: Spiegelplatten bzw. RAID-Disk-Arrays
- **Katastrophen** (Komplettausfall Rechenzentrum durch Überschwemmung, Erdbeben, etc.)
 - verteilte Datensicherung auf geographisch separierten Systemen

Systemkomponenten zur Recovery



- Pufferung von Log-Daten im Hauptspeicher (Log-Puffer)
 - Ausschreiben spätestens am Transaktionsende ("Commit")
- **Log-Datei** zur Behandlung von Transaktions- und Systemfehler:
DB + Log => DB
- Behandlung von Gerätefehlern: **Archivkopie + Archiv-Log** => DB

Die Transaktion als Schnittstelle zwischen Anwendungsprogramm und DBS



Zusammenfassung

- dezentrales Autorisierungskonzept
 - Grant (with Grant Option)
 - Revoke
- Synchronisation, z.B. durch Sperrverfahren
 - Umgehung von Mehrbenutzeranomalien
 - unterschiedliche Konsistenzstufen
- Recovery: automatische Behandlung erwarteter Fehler
 - Transaktions-, System- und Gerätefehlern mit Log-Daten
 - ggf. geographisch verteilte Replikation der Daten