8. Datenkontrolle

- ACID und Datenkontrolle
- Integritätskontrolle
  - Klassifikation von Integritätsbedingungen
  - Integritätsbedingungen in SQL
  - Integritätsregeln / Trigger
- Zugriffskontrolle/Autorisierung in SQL
  - GRANT (with GRANT OPTION)
  - REVOKE
- Synchronisation
  - Anomalien
  - Isolation Level
- Recovery

ACID und Datenkontrolle

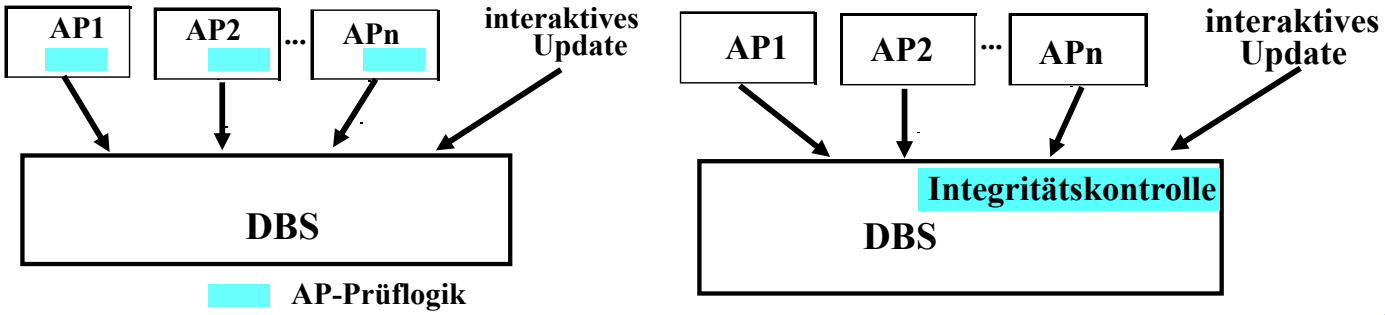
- Transaktionskonzept (ACID-Eigenschaften)
  - im SQL-Standard: COMMIT WORK, ROLLBACK WORK, Beginn einer Transaktion implizit
  - Einhaltung der logischen DB-Konsistenz (Consistency)
  - Verdeckung der Nebenläufigkeit (concurrency isolation)
  - Verdeckung von (erwarteten) Fehlerfällen -> Logging und 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)

Semantische Integritätskontrolle

- Logische DB-Konsistenz: Überwachung von semantischen Integritätsbedingungen durch Anwendungen oder DBS
- DBS-basierte Integritätskontrolle
  - größere Sicherheit
  - vereinfachte Anwendungserstellung
  - Unterstützung von interaktiven sowie programmierten DB-Änderungen
  - leichtere Änderbarkeit von Integritätsbedingungen
- Integritätsbedingungen der Miniwelt sind explizit bekannt zu machen, um automatische Überwachung zu ermöglichen



Klassifikation von Integritätsbedingungen

1. modellinhärente Integritätsbedingungen (vs. anwendungsspezifische IB)
   - Primärschlüsseleigenschaft
   - referentielle Integrität für Fremdschlüssel
   - Definitionsbereiche (Domains) für Attribute
2. Reichweite

Reichweite	Beispiele
Attribut	GEB-JAHR ist numerisch, 4-stellig
Satzausprägung	ABT.GEHALTSSUMME < ABT.JAHRESETAT
Satztyp	PNR ist eindeutig
mehrere Satztypen	ABT.GEHALTSSUMME ist Summe aller Angestelltengehälter in PERS

3. Zeitpunkt der Überprüfbarkeit
   - unverzögert (sofort bei Änderungsoperation)
   - verzögert (am Transaktionsende)
4. Art der Überprüfbarkeit
   - Zustandsbedingungen (statische Integritätsbedingungen)
   - dynamische Integritätsbedingungen

Dynamische Integritätsbedingungen

- Beziehen sich im Gegensatz zu statischen IB auf Änderungen selbst und damit auf mehrere Datenbankzustände
- Zwei Varianten
  - *Übergangsbedingungen*: Änderung von altem zu neuem DB-Zustand wird eingeschränkt
  - *temporale Bedingungen*: Änderungen in bestimmtem zeitlichen Fenster werden eingeschränkt
- Beispiele dynamischer Integritätsbedingungen
  - Übergang von FAM-STAND von 'ledig' nach 'geschieden' ist unzulässig
  - Gehalt darf nicht kleiner werden
  - Gehalt darf innerhalb von 3 Jahren nicht um mehr als 25% wachsen

Integritätsbedingungen in SQL

- Eindeutigkeit von Attributwerten
  - UNIQUE bzw. PRIMARY KEY bei CREATE TABLE
  - Satztypbedingungen

  Bsp.: CREATE TABLE PERS ...
  PNR INT UNIQUE (bzw. PRIMARY KEY)

- Fremdschlüsselbedingungen
  - FOREIGN-KEY-Klausel
  - Satztyp- bzw. satztypübergreifende Bedingung

- Wertebereichsbeschränkungen von Attributen
  - CREATE DOMAIN
  - NOT NULL
  - DEFAULT
  - Attribut- und Satztyp-Bedingungen

Integritätsbedingungen in SQL (2)

- Allgemeine Integritätsbedingungen
  - CHECK-Constraints bei CREATE TABLE
  - allgemeine Assertions, z. B. für satztypübergreifende Bedingungen

CHECK-Constraints bei CREATE TABLE

```
CREATE TABLE PERS ....
   GEB-JAHR  INT
   CHECK (VALUE BETWEEN 1900 AND 2100)
CREATE TABLE ABT .....
   CHECK (GEHALTSSUMME <  JAHRESETAT)
```

Anweisung CREATE ASSERTION

```
CREATE ASSERTION A1
   CHECK (NOT EXISTS
   (SELECT * FROM ABT A
    WHERE GEHALTSSUMME <>
   (SELECT SUM (P.GEHALT) FROM PERS P
        WHERE P.ANR = A.ANR)))
   DEFERRED
```

- Festlegung des Überprüfungszeitpunktes:
  - IMMEDIATE: am Ende der Änderungsoperation (Default)
  - DEFERRED: am Transaktionsende (COMMIT)
- Unterstützung für dynamische Integritätsbedingungen durch Trigger (ab SQL:1999)

Integritätsregeln

- Standardreaktion auf verletzte Integritätsbedingung: ROLLBACK
- Integritätsregeln erlauben Spezifikation von Folgeaktionen, z. B. um Einhaltung von Integritätsbedingungen zu erreichen
  - SQL92: deklarative Festlegung referentieller Folgeaktionen (CASCADE, SET NULL, ...)
  - SQL99: Trigger
- Trigger: Festlegung von Folgeaktionen für Änderungsoperationen
  - INSERT
  - UPDATE oder
  - DELETE
- Trigger wesentlicher Mechanismus von ***aktiven DBS***
- Verallgemeinerung durch sogenannte ECA-Regeln (Event / Condition / Action)

Integritätsregeln (2)

- Beispiel: Wartung der referentiellen Integrität
  - **deklarativ**

```
CREATE TABLE PERS
 (PNR INT PRIMARY KEY,
    ANR INT FOREIGN KEY
     REFERENCES ABT
     ON DELETE CASCADE
   ... );
```

  - **durch Trigger**

```
CREATE TRIGGER MITARBEITERLÖSCHEN
BEFORE DELETE ON ABT
REFERENCING OLD AS A
DELETE FROM PERS P
WHERE P.ANR = A.ANR;
```

Trigger

- ausführbares, benanntes DB-Objekt, das implizit durch bestimmte Ereignisse ("triggering event") aufgerufen werden kann
- Triggerspezifikation besteht aus
  - auslösendem Ereignis (Event)
  - Ausführungszeitpunkt
  - optionaler Zusatzbedingung
  - Aktion(en)
- zahlreiche Einsatzmöglichkeiten
  - Überwachung nahezu aller Integritätsbedingungen, inkl. dynamischer Integritätsbedingungen
  - Validierung von Eingabedaten
  - automatische Erzeugung von Werten für neu eingefügten Satz
  - Wartung replizierter Datenbestände
  - Protokollieren von Änderungsbefehlen (Audit Trail)
  - • • •

Trigger (2)

- SQL99-Syntax

```
CREATE TRIGGER <trigger name>
 {BEFORE|AFTER}{INSERT|DELETE|
   UPDATE [OF <column list>]}
  ON <table name>
 [ORDER <order value>]
 [REFERENCING <old or new alias list>]
 [FOR EACH {ROW|STATEMENT}]
 [WHEN (<search condition>)]
   <triggered SQL statement>
```

```
<old or new alias> ::=
OLD [AS]<old values correlation name>|
NEW [AS]<new values correlation name>|
OLD_TABLE [AS]<old values table alias>|
NEW_TABLE [AS]<new values table alias>
```

  - Trigger-Events: INSERT, DELETE, UPDATE
  - Zeitpunkt: BEFORE oder AFTER
    - mehrere Trigger pro Event/Zeitpunkt möglich (benutzerdefinierte Aktivierungsreihenfolge)
  - Bedingung: beliebiges SQL-Prädikat (z. B. mit komplexen Subqueries)
  - Aktion: beliebige SQL-Anweisung (z. B. auch neue prozedurale Anweisungen)
    - Trigger-Bedingung und -Aktion können sich sowohl auf alte als auch neue Tupelwerte der betroffenen Tupel beziehen
  - Trigger-Ausführung für jedes betroffene Tupel einzeln (FOR EACH ROW) oder nur einmal für auslösende Anweisung (FOR EACH STATEMENT)

Trigger-Beispiele

- Realisierung einer dynamischen Integritätsbedingung (*Gehalt darf nicht kleiner werden*):

```
CREATE TRIGGER GEHALTSTEST
AFTER UPDATE OF GEHALT ON PERS AS
```

- Wartung einer materialisierten Sicht ARME_PROGR_MV

```
CREATE TRIGGER AP-INSERT
AFTER INSERT ON PERS
FOR EACH ROW
REFERENCING NEW AS N
WHEN N.BERUF = „Programmierer“ AND N.GEHALT < 30000
  INSERT INTO ARME_PROGR_MV (PNR, NAME, BERUF, GEHALT)
  VALUES (N.PNR, N.NAME, N.BERUF, N.GEHALT)
```


Probleme von Triggern

- teilweise prozedurale Semantik (Zeitpunkte, Verwendung alter/neuer Werte, Aktionsteil im Detail festzulegen)
- Trigger derzeit beschränkt auf Änderungsoperationen einer Tabelle (UPDATE, INSERT, DELETE)
- i. a. keine verzögerte Auswertung von Triggern
- Gefahr zyklischer, nicht-terminierender Aktivierungen
- Korrektheit des DB-/Trigger-Entwurfes (Regelabhängigkeiten, parallele Regelausführung, ...)

dennoch sind Trigger mächtiges und wertvolles Konstrukt

- „aktive“ DBS
- DBS-interne Nutzungsmöglichkeiten, z.B. zur Realisierung von Integritätskontrolle und Aktualisierung von materialisierten Sichten / Replikaten

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

Rücknahme von Zugriffsrechten: Revoke

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

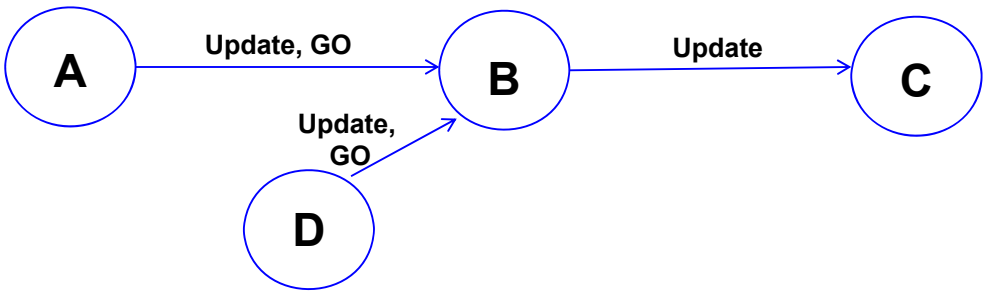
- ggf. fortgesetztes Zurücknehmen von Zugriffsrechten

Beispiel: REVOKE SELECT
ON ABT
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)

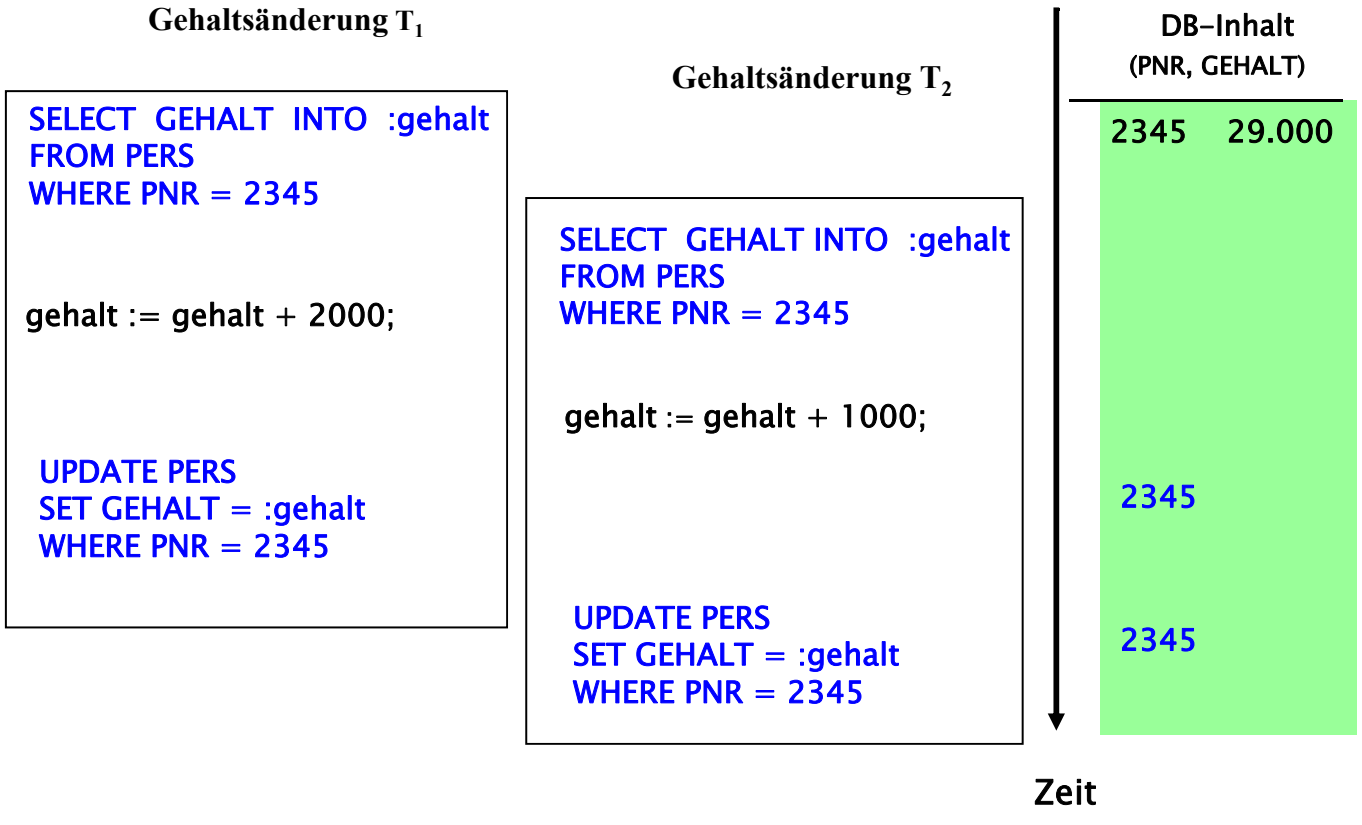
- Führen der Abhängigkeiten in einem *Autorisierungsgraphen* pro Objekt (z.B. Tabelle)
  - *Knoten: User-Ids*
  - *gerichtete Kanten: Weitergabe von Zugriffsrechten (Recht, ggf. Zeitpunkt)*
- Beispiel für Tabelle R
  - User A: GRANT UPDATE ON R TO B WITH GRANT OPTION
  - User B: GRANT UPDATE ON R TO 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
  - verlorengegangene 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

Lesetransaktion

```
SELECT  GEHALT  INTO  :gehalt
 FROM PERS
WHERE PNR = 2345

SELECT  GEHALT  INTO :gehalt2
FROM PERS
WHERE PNR=2345
IF gehalt <>  gehalt2 THEN
      <Fehlerbehandlung>
```

Änderungstransaktion

```
UPDATE  PERS
SET GEHALT = GEHALT + 1000
WHERE PNR = 2345

 COMMIT WORK
```

DB-Inhalt

(PNR, GEHALT)

2345 29.000

2345 30.000

Zeit

Phantom-Problem

Lesetransaktion

(Gehaltssumme überprüfen)

```
SELECT  SUM (GEHALT)   INTO  :summe
FROM PERS
WHERE ANR = 17

SELECT SUM (GEHALT) INTO :summe2
FROM PERS
WHERE ANR = 17

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

Änderungstransaktion

(Einfügen eines neuen Angestellten)

```
INSERT  INTO PERS (PNR, ANR, GEHALT)
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 TRANSACTON Read Only ISOLATION LEVEL Read Committed

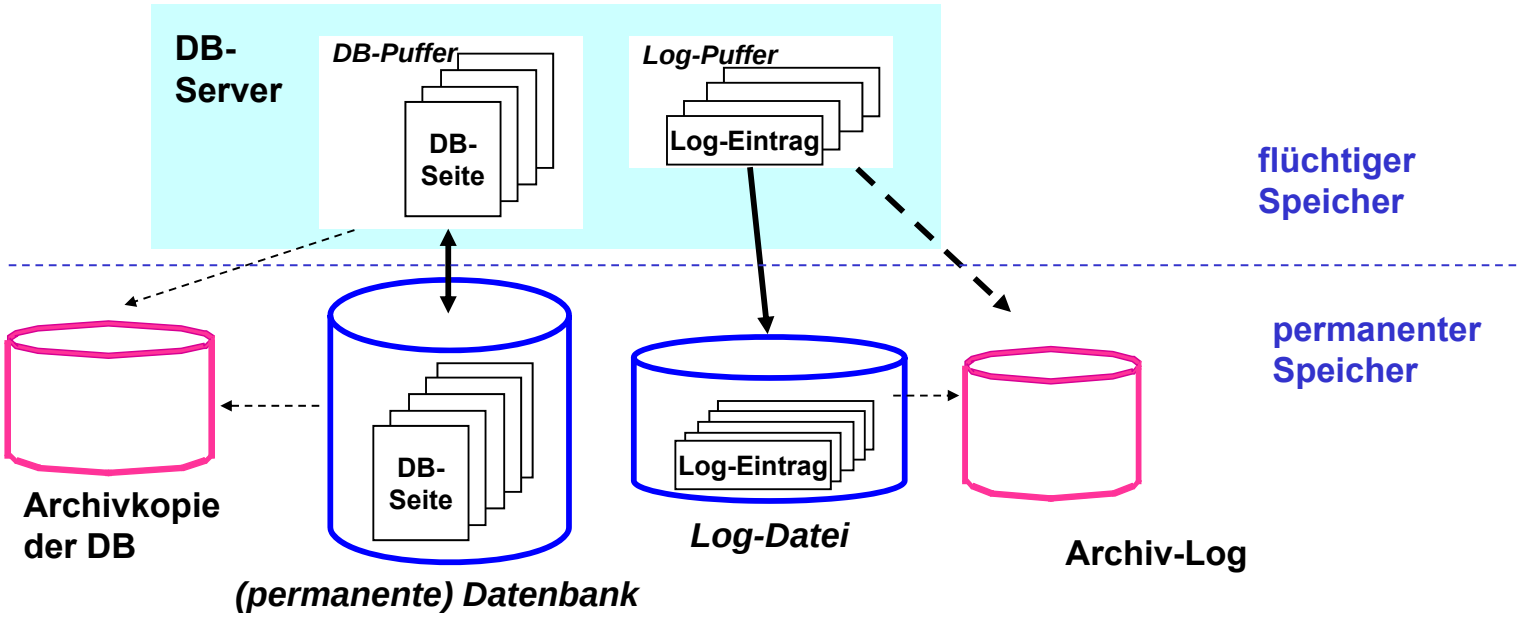
Recovery-Unterstützung

- 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

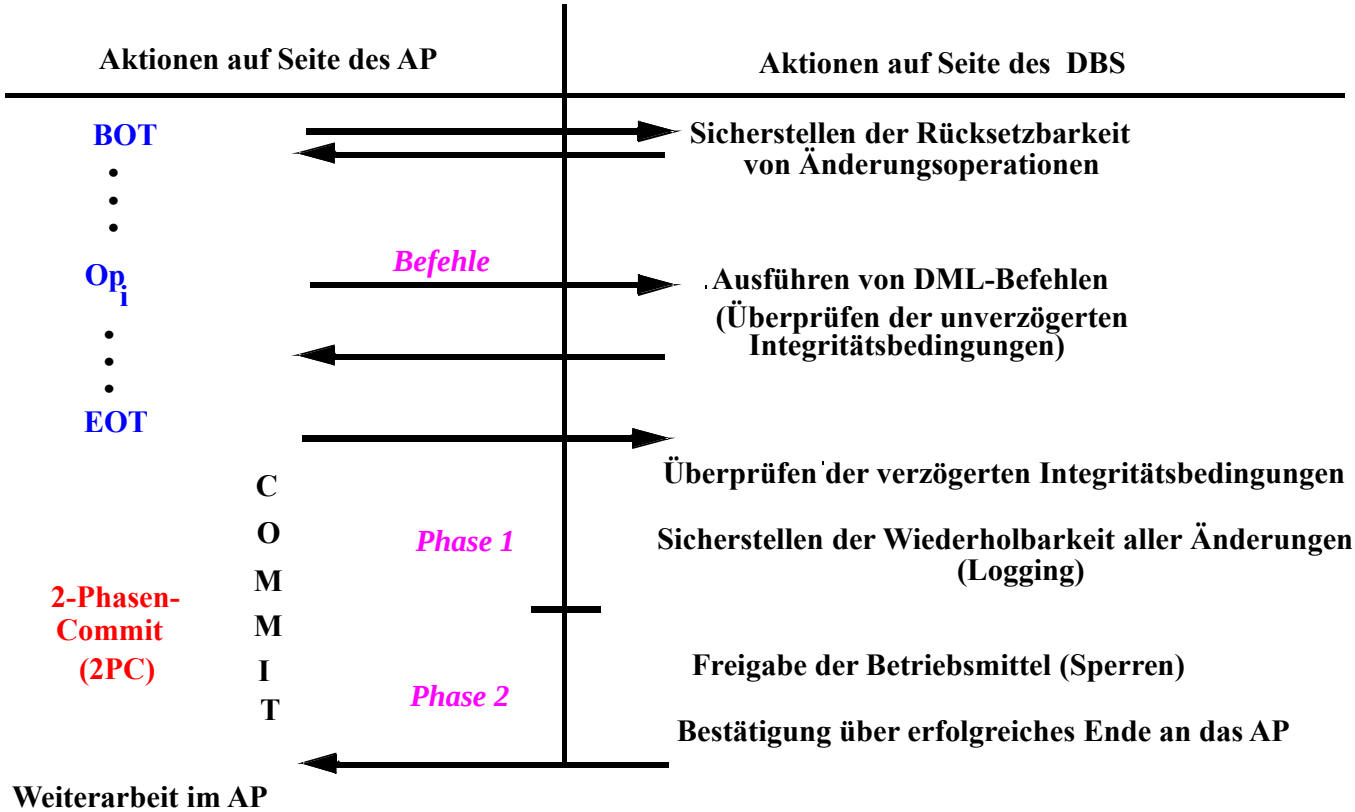
- *Transaktionsfehler*: vollständiges Zurücksetzen auf Transaktionsbeginn (Undo)
- *Systemfehler* (Rechnerausfall, DBVS-Absturz)
  - REDO für erfolgreiche Transaktionen (Wiederholung verlorengegangener Änderungen)
  - UNDO aller durch Ausfall unterbrochenen Transaktionen (Entfernen derer Änderungen aus 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

- Integritätsbedingungen
  - Klassifikation gemäß 4 Kriterien
  - umfassende Unterstützung in SQL
- Trigger
  - automatische Reaktion bei DB-Änderungen (-> „aktive DBS“)
  - zahlreiche Nutzungsmöglichkeiten: Integritätskontrolle, mat. Sichten ...
- dezentrales Autorisierungskonzept
  - Grant (with Grant Option)
  - Revoke
- Synchronisation, z.B. durch Sperrverfahren
  - Umgehung von Mehrbenutzeranomalien
  - unterschiedliche Konsistenzstufen
- Recovery: automatische Behandlung erwarteter Fehler von Transaktions-, System- und Gerätefehlern mit Log-Daten
  - ggf. geographisch verteilte Replikation der Daten