1. Integritätsbedingungen und Trigger

- ACID und Datenkontrolle
  - Synchronisation zur Ablaufintegrität
  - Transaktionsablauf
- Integritätsbedingungen
  - Klassifikation: Statische vs. Dynamische Integritätsbedingungen, ...
  - Integritätsbedingungen in SQL92
- Integritätsregeln / Trigger
  - Konzept
  - CREATE TRIGGER
  - Einsatzformen von Trigger

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)
  - nur 'sinnvolle' und 'zulässige' Änderungen der DB: bei COMMIT müssen alle semantischen Integritätsbedingungen erfüllt sein (Transaktionskonsistenz)
  - Einhaltung der *physischen Integrität* sowie der *Ablaufintegrität* (operationale Integrität)
- Zugriffskontrolle
  - Maßnahmen zur Datensicherheit und zum Datenschutz
  - Sichtkonzept
  - Vergabe und Kontrolle von Zugriffsrechten

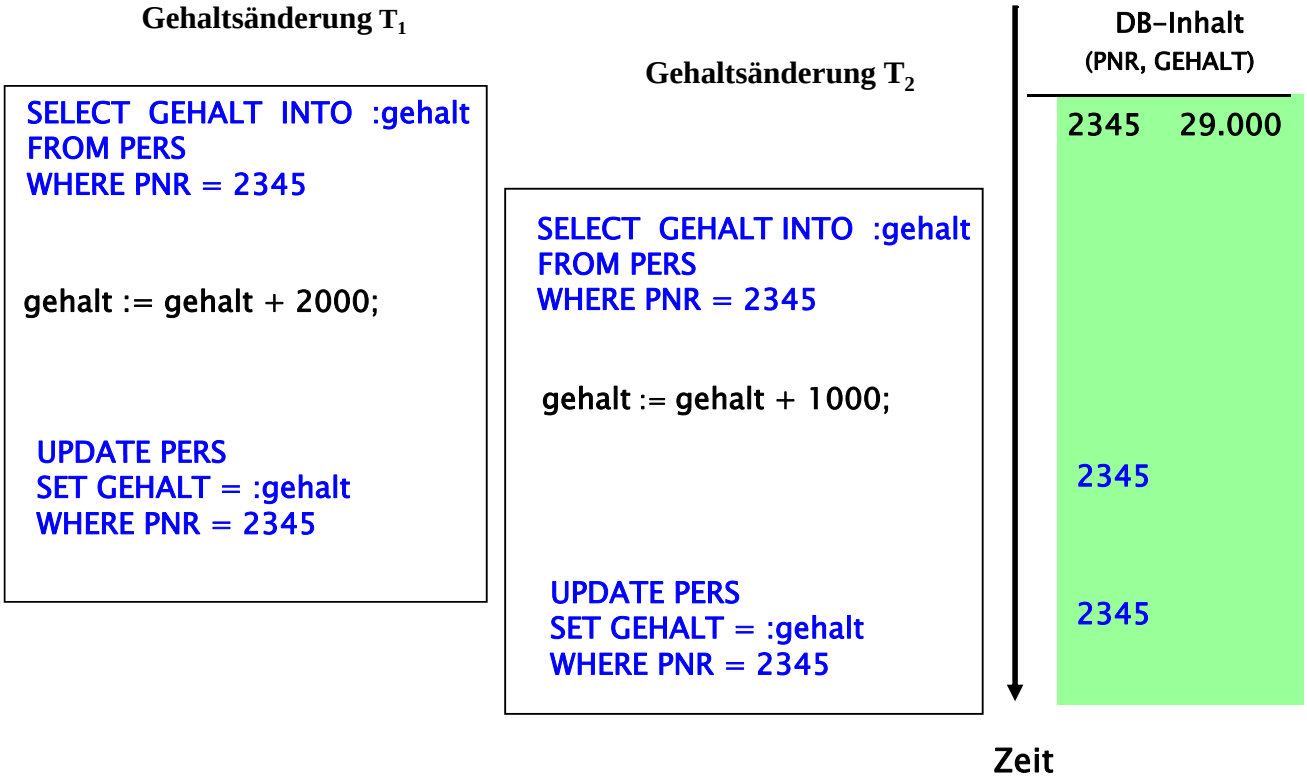
ACID und Datenkontrolle (2)

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)

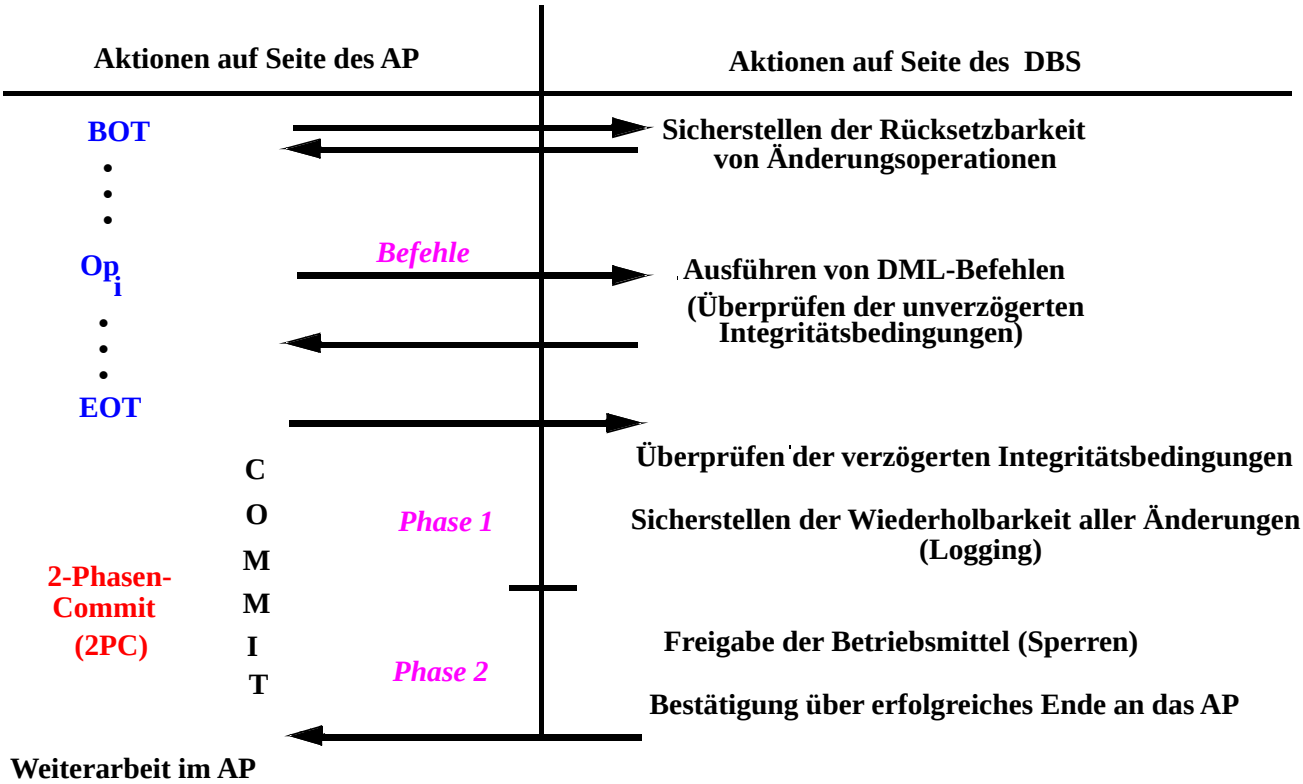
Synchronisation

- DBS müssen Mehrbenutzerbetrieb unterstützen
- ohne Synchronisation kommt es zu so genannten Mehrbenutzer-Anomalien
  - Verlorengegangene Änderungen (lost updates)
  - Abhängigkeiten von nicht freigegeben Änderungen (dirty read, dirty overwrite)
  - inkonsistente Analyse (non-repeatable read)
  - Phantom-Probleme
- Anomalien sind nur durch Änderungen verursacht
- Synchronisation erfolgt automatisch durch das DBS, z.B. durch Setzen von Sperren vor Datenzugriff (Freigabe der Sperren am Transaktionsende)

Verloren gegangene Änderung (Lost Update)

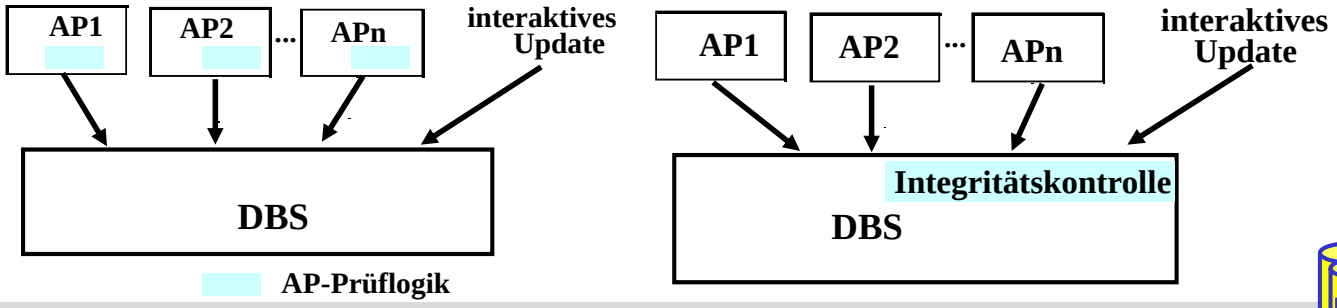


Die Transaktion als Schnittstelle zwischen Anwendungsprogramm und DBS



(Semant.) Integritätskontrolle

- Wahrung der logischen DB-Konsistenz
- Überwachung von semantischen Integritätsbedingungen durch Anwendungen oder durch 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

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

```
CREATE TRIGGER GEHALTSTEST
AFTER UPDATE OF GEHALT ON PERS
REFERENCING OLD AS AltesGehalt,
NEW AS NeuesGehalt
WHEN (NeuesGehalt < AltesGehalt)
        ROLLBACK;
```

- Wartung einer materialisierten Sicht ARME_PROGRAMMIERER

```
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_PROGRAMMIERER
  VALUES (N.PNR, N.NAME, N.BERUF, …)
```

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)
- derzeit i. a. keine verzögerte Auswertung von Triggern
- Gefahr zyklischer, nicht-terminierender Aktivierungen
- Korrektheit des DB-/Trigger-Entwurfes (Regelabhängigkeiten, parallele Regelausführung, ...)

Dennoch sind Trigger sehr mächtiges und wertvolles Konstrukt, auch zur DBS-internen Nutzung

Zusammenfassung

- Datenkontrolle
  - Semantische Integritätskontrolle
  - Synchronisation (Ablaufintegrität)
  - Logging und Recovery (Voraussetzung für physische Datenkonsistenz)
  - Zugriffskontrolle
  - ACID-Klammer
- Integritätsbedingungen
  - Klassifikation gemäß 4 Kriterien
  - Umfassende Unterstützung in SQL92
- Trigger
  - automatische Reaktion bei DB-Änderungen (-> „aktive DBS“)
  - zahlreiche Anwendungsmöglichkeiten: Integritätskontrolle, materialisierte Sichten, ...