8. Integritätskontrolle

- **Semantische Integritätsbedingungen**
  - Klassifikation
  - neu: Assertions
  - sprachliche Mittel und Beispiele
- **Trigger**
  - Konzept
  - aktives Verhalten
  - Reihenfolge- und Terminierungsprobleme
  - Trigger-Granulate
  - Trigger-Einsatz

Semantische Integritätsbedingungen (1)

- **ZIEL**[1]
  - Nur DB-Änderungen zulassen, die allen definierten *Constraints* entsprechen (offensichtlich ‘falsche’ Änderungen zurückweisen!)
  - Möglichst hohe Übereinstimmung von DB-Inhalt und Miniwelt (Datenqualität)

➥ *Integritätsbedingungen der Miniwelt sind explizit bekannt zu machen, um automatische Überwachung zu ermöglichen.*

- **Klassifikation**

  Unterscheidung nach

  1. Ebenen der Abbildungshierarchie eines DBS
     (Blöcke, Seiten, Tupel, ...)
  2. Reichweite
     (Attribut, Relation, mehrere Relationen)
  3. Zeitpunkt der Überprüfbarkeit
     (sofort, erst nach mehreren Operationen)
  4. Art der Überprüfbarkeit
     (Zustand, Übergang)
  5. Anlaß für Überprüfung
     (Datenänderung, ...)

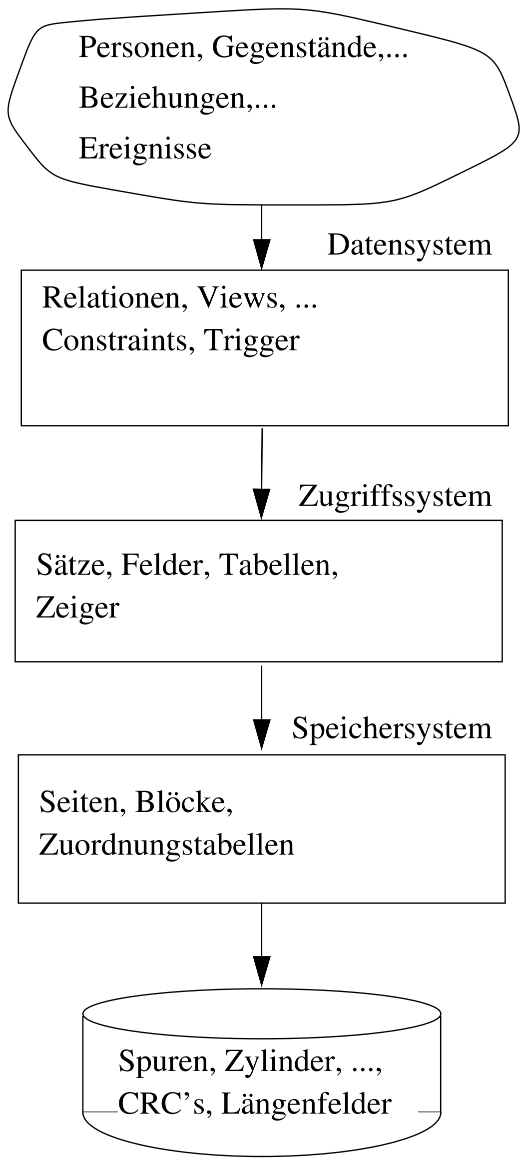
- **Konsistenz der Transaktionsverarbeitung**
  - Bei COMMIT müssen alle semantischen Integritätsbedingungen erfüllt sein.
  - Zentrale Spezifikation/Überwachung im DBS: *„system enforced integrity“*

---

1. **Golden Rule** nach C. J. Date: No update operation must ever be allowed to leave any relation or view (relvar) in a state that violates its own predicate. Likewise no update transaction must ever be allowed to leave the database in a state that violates its own predicate.

Semantische Integritätsbedingungen (2)

- **Ebenen der Abbildungshierarchie**



Anwendungsbereich („Miniwelt") des Benutzers

Semantische Integritätsbedingungen

Datenmodell zur formalisierten Abbildung der Miniwelt

Einhaltung der Relationale Invarianten, benutzerdef. Integritätsbedingungen

Repräsentation in einem linearen Adreßraum

Konsistenzbedingungen von Zugriffspfaden, Tabellen, Zeigern, usw.

Repräsentation in Dateien

Konsistenzbedingungen und Abbildungsvorschriften bei Blöcken, Seiten, usw.

Physische Repräsentation auf Externspeichermedien

Paritäts-, Längenbedingungen, usw.

- **Physische Konsistenz** der DB ist Voraussetzung für logische Konsistenz
  - Gerätekonsistenz
  - Dateikonsistenz
  - Speicherkonsistenz (Speicherungsstrukturen/Zugriffspfade/Zeiger sind konsistent)

- **Logische Konsistenz**
  - modellinhärente Bedingungen (z. B. Relationale Invarianten)
  - benutzerdefinierte Bedingungen aus der Miniwelt

Semantische Integritätsbedingungen (3)

- **Reichweite**

  Art und Anzahl der von einer Integritätsbedingung (genauer: des die Bedingung ausdrückenden Prädikats) betroffenen **Objekte**

  - **ein Attribut**
    (PNR vierstellige Zahl,
    NAME nur Buchstaben und Leerzeichen)

  - **mehrere Attribute eines Tupels**
    (GEHALTS-SUMME einer Abteilung muß kleiner sein als JAHRES-ETAT)

  - **mehrere Tupel derselben Relation**
    (kein GEHALT mehr als 20 % über dem Gehaltsdurchschnitt aller Angestellten derselben Abteilung, PNR ist Primärschlüssel)

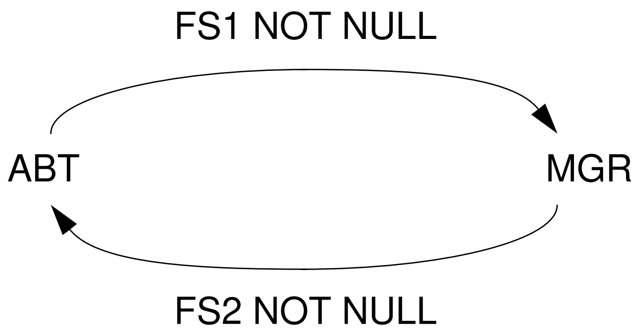
  - **mehrere Tupel aus verschiedenen Relationen**
    (GEHALTS-SUMME einer Abteilung muß gleich der Summe der Attributwerte in GEHALT der zugeordneten Angestellten sein)

➥ **geringere Reichweite = einfachere Überprüfung**

Semantische Integritätsbedingungen (4)

- **Zeitpunkt der Überprüfbarkeit**
  - **Unverzögerte** Bedingungen
    - müssen immer erfüllt sein, unabhängig davon, was in der DB passiert
    - können sofort nach Auftauchen des Objektes überprüft werden (typisch: solche, die sich auf ein Attribut beziehen)

  - **Verzögerte** Bedingungen



zyklische Fremdschlüsselbedingungen

    - lassen sich nur durch eine Folge von Änderungen erfüllen (typisch: mehrere Tupel, mehrere Relationen)
    - benötigen Transaktionsschutz (als zusammengehörige Änderungssequenzen)

Semantische Integritätsbedingungen (5)

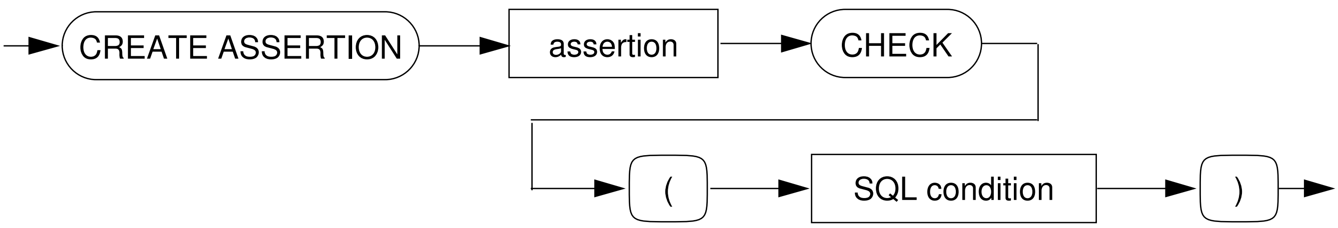
- **Art der Überprüfbarkeit**
  - **Zustandsbedingungen**
    - betreffen den zu einem bestimmten Zeitpunkt in der DB abgebildeten Zustand der Objekte
  - **Übergangsbedingungen**
    - Einschränkungen der Art und Richtung von Wertänderungen einzelner oder mehrerer Attribute
    - Beispiele: GEHALT eines Angestellten darf niemals sinken, FAM-STAND darf nicht von „ledig“ nach „geschieden“ oder von „verheiratet“ nach „ledig“ geändert werden
    - sind am Zustand nicht prüfbar - entweder sofort bei Änderung oder später durch Vergleich von altem und neuem Wert (Versionen)

Integritätsbedingungen in SQL

- **Bereits eingeführt** (siehe Datendefinition)
  - CHECK-Bedingungen bei CREATE DOMAIN, CREATE TABLE, Attributdefinition
  - Verbot von Nullwerten, UNIQUE, PRIMARY KEY
  - Fremdschlüsselbedingungen (FOREIGN-KEY-Klausel)

    ➥ Diese Integritätsbedingungen sind an die betreffenden DB-Objekte gebunden.

- **Allgemeine Integritätsbedingungen**
  - beziehen sich typischerweise auf mehrere Relationen
  - lassen sich als eigenständige DB-Objekte definieren
  - erlauben die Verschiebung ihres Überprüfungszeitpunktes
  - Syntax der Assertion-Anweisung



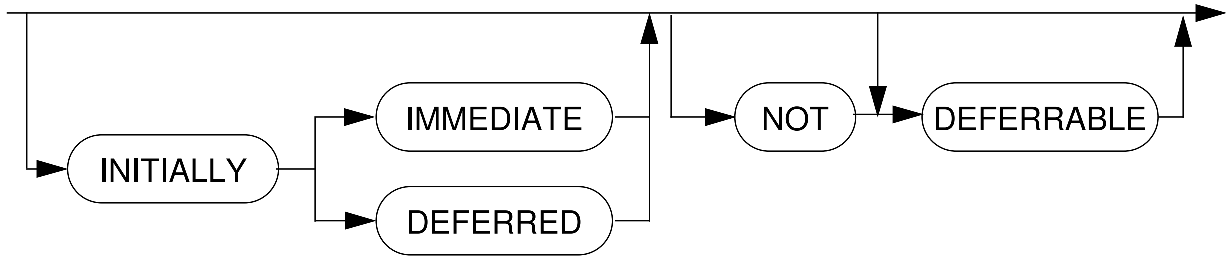
- **Beispiel**
  Die Relation Abt enthält ein Attribut, in dem (redundant) die Anzahl der Angestellten einer Abteilung geführt wird. Es gilt folgende Zusicherung:

```
CREATE ASSERTION A1
    CHECK (NOT EXISTS
              (SELECT * FROM Abt A
               WHERE A.Anzahl_Angest <>
                  (SELECT  COUNT (*) FROM  Pers P
                   WHERE P.Anr = A.Anr)));
```

➥ Bei welchen Operationen und wann muß überprüft werden?

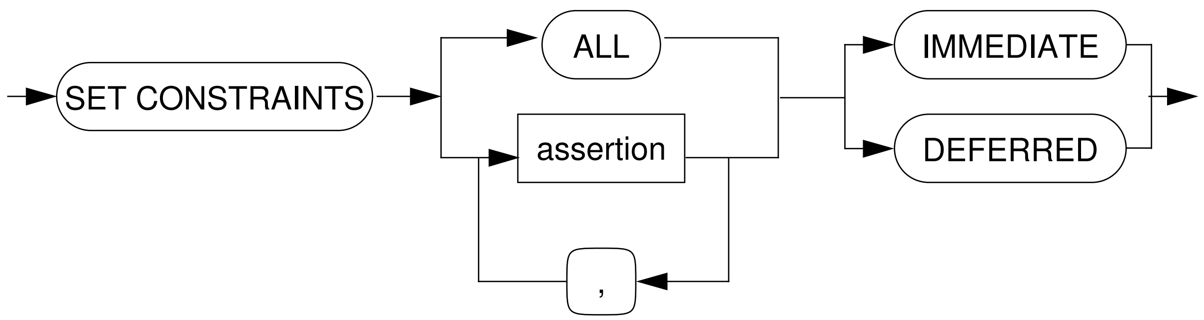
Integritätsbedingungen in SQL (2)

- **Festlegung des Überprüfungszeitpunktes:**



  - IMMEDIATE: am Ende der Änderungsoperation (Default)
  - DEFERRED: am Transaktionsende (COMMIT)

- **Überprüfung** kann durch **Constraint-Modus gesteuert** werden



  - Zuordnung gilt für die aktuelle Transaktion
  - Bei benamten Constraints ist eine selektive Steuerung der Überprüfung möglich; so können gezielt Zeitpunkte vor COMMIT ausgewählt werden.

Beispiel-DB

Abt	Anr	Aname	Ort	Anzahl_Angest
	K51	PLANUNG	KL	1
	K53	EINKAUF	F	1
	K55	VERTRIEB	F	2

Pers	Pnr	Name	Alter	Gehalt	Anr	Mnr
	406	COY	47	50 000	K55	123
	123	MÜLLER	32	43 500	K51	-
	829	SCHMID	36	45 200	K53	777
	574	ABEL	28	30 000	K55	123

- Gehaltssumme an Abt anhängen

- Gehaltssumme mit Werten füllen

- Einfügen eines neuen Angestellten

Wann wird Constraint A2 überprüft?

```
CREATE ASSERTION A2
    CHECK (NOT EXISTS
              (SELECT * FROM Abt A
               WHERE A.Geh_Summe <>
                  (SELECT  SUM (P.Gehalt) FROM  Pers P
                   WHERE P.Anr = A.Anr)))
     INITIALLY DEFERRED;
```

Trigger

- **Integritätsbedingungen beschreiben, was innerhalb der DB gültig und zulässig ist.**

- **Neue Idee:**
  <u>Spezifikation und Durchführung von Reaktionen</u>
  auf bestimmte Situationen oder Ereignisse in der DB[1]

  ➥ Oft synonyme Nutzung der Begriffe Produktionsregel, Regel, Aktive Regel, Trigger, Alerter

- **Neue Anforderung:**
  Wirkungsweise kann nicht durch ein **statisches Prädikat** spezifiziert werden, sondern als „**Zusammenhangsregel**“

  ➥ Wenn Pers.Gehalt um mehr als 10% erhöht wird, benachrichtige Top-Level-Manager

  ➥ Wenn eine Abteilung auf weniger als 5 Angestellte verkleinert wird, kürze ihr Budget um 25%

  - Zusammenhang durch kausale, logische oder „beliebige“ Verknüpfung

- **Wer ist für die Triggerausführung verantwortlich?**
  - Kopplung mit Transaktionen
  - Events unabhängig von Transaktionen?

---

1. Je mehr Semantik des modellierten Systems explizit repräsentiert ist, umso mehr kann das DBS „aktiv“ werden!

Trigger (2)

- **Trigger**
  - „triggernde“ Operation wird als *Event* bezeichnet
  - Reaktion auf Event besteht aus Folge von DB-Operationen

    AFTER <Event> ON <Table>

    DO <DML-Operation>, ...
  - In relationalen Systemen sind i. allg. nur Modifikationsoperationen als Events vorgesehen.
  - In objektorientierten Systemen kann i. allg. jeder Aufruf einer Methode ein Event sein.

    ➥ DBS muß das Auftreten der Events erkennen und die zugehörigen Operationen durchführen.

- **Ausführung von Triggern**
  - mehrere Trigger für ein Event
  - rekursive Auflösung von Triggern

    ➥ zentrale Probleme:
    Terminierung und Reihenfolge der Regelausführung

Terminierung bei Triggern

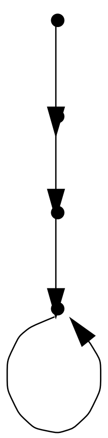
- **Kontrolle der Regelausführung**
  Datenänderungen triggern Regeln, die Daten ändern, die Regeln triggern, die Daten ändern, ...
  - hier: rekursive Auflösung von Triggern

Beispiel zur Terminierung:

Create Trigger V
After Update (Gehalt) ON Pers A } Event

Update Pers P
Set P.Gehalt = 1.02 * P.Gehalt } Action
Where P.Pnr = A.Mnr

Op1: Update Pers P
Set P.Gehalt = 1.05 * P.Gehalt
Where P.Pnr = P1



	Pers (Pnr,	Mnr,	Gehalt)	Update
t1	P1	P2	100	
t2	P2	P3	120	
t3	P3	P4	150	
t4	P4	P4	200	

Auswertungsreihenfolge bei Triggern

- **Kontrolle der Regelausführung**
  Datenänderungen triggern Regeln, die Daten ändern, die Regeln triggern, die Daten ändern, ...
  - hier: mehrere Trigger für ein Event

    Erweiterung:

```
Create Trigger NHV
   After Update (Gehalt) ON Pers A
         Update Pers P
         Set P.Gehalt = 1.01 * P.Gehalt
         Where P.Pnr =
            ( Select X.Mnr
              From Pers X
              Where X.Pnr = A.Mnr)
```

➥ Es existieren V und NHV! !

Pers	(Pnr,	Mnr,	Gehalt)
t1:	P1	P2	100
t2:	P2	P3	120
t3:	P3	P4	150
t4:	P4	---	200

Trigger-Konzept

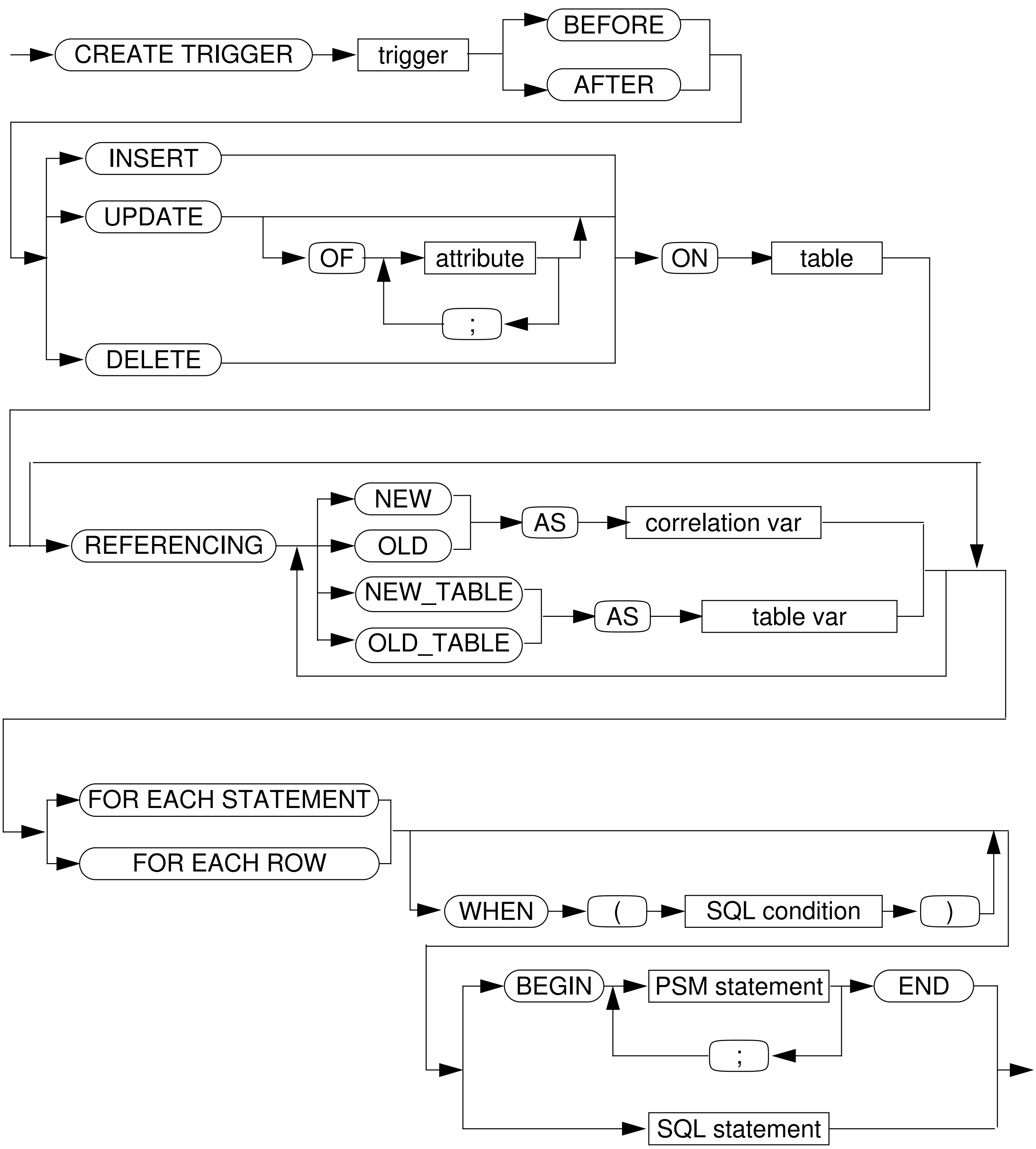
- **Idee:** automatische Korrektur des DB-Zustandes:
  Starten von Folgeänderungen zur Wahrung der DB-Integrität

➥ Trigger werden schon seit ~1985 in relationalen DBS eingesetzt.
Ihre Standardisierung wurde jedoch erst in SQL99 vorgenommen.

- **Trigger-Konzept**
  - Wann soll ein Trigger ausgelöst werden?
    - Zeitpunkte: BEFORE / AFTER
    - auslösende Operation: INSERT / DELETE / UPDATE
  - bei Übergangsbedingungen
    - Bezug auf verschiedene DB-Zustände erforderlich
    - OLD/NEW erlaubt Referenz von alten/neuen Werten
  - Ist die Trigger-Ausführung vom DB-Zustand abhängig?
    (WHEN-Bedingung optional)
  - Was soll wie verändert werden?
    - pro Tupel oder pro DB-Operation (Trigger-Granulat)
    - mit einer SQL-Anweisung oder mit einer Prozedur aus
      PSM-Anweisungen (persistent stored module, stored procedure)
  - mehrere Trigger-Definitionen pro Relation (Tabelle) sowie
    mehrere Trigger-Auslösungen pro Ereignis möglich

Trigger-Konzept (2)

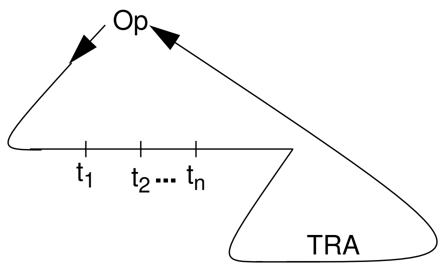
- **Trigger-Syntax in SQL99**

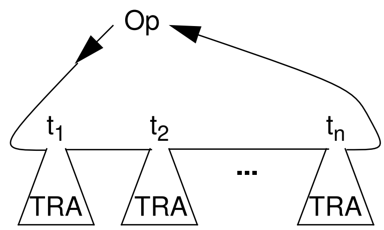


Trigger-Konzept (3)

- **Übergangstabellen und -variablen**
  - Sie vermerken Einfügungen (bei INSERT), Löschungen (bei DELETE) und die alten und neuen Zustände (bei UPDATE).
  - Übergangstabellen (transition tables) enthalten mengenorientierte Änderungen, während Übergangsvariablen (transition variables) die tupelweisen Änderungen aufnehmen.

- **Trigger-Granulat**
  - FOR EACH STATEMENT: mengenorientiertes Verarbeitungsmodell
  - FOR EACH ROW: tupelorientiertes Verarbeitungsmodell
  - TRA: Trigger-Aktion





- **Einsatzbeispiel**
  - Die Gehaltsumme in Abt soll bei Änderungen in Pers, die „Gehälter“ betreffen, automatisch aktualisiert werden.
  - Es sind Trigger für INSERT/DELETE/UPDATE erforderlich. Sie werden bei Auftreten der spezifizierten Änderungsoperationen sofort ausgeführt.

Trigger-Einsatz

Abt	Anr	Aname	Ort	Geh_Summe
	K51	PLANUNG	KAISERSLAUTERN	43500
	K53	EINKAUF	FRANKFURT	45200
	K55	VERTRIEB	FRANKFURT	80000

Pers	Pnr	Name	Alter	Gehalt	Anr	Mnr
	406	COY	47	50 000	K55	123
	123	MÜLLER	32	43 500	K51	-
	829	SCHMID	36	45 200	K53	777
	574	ABEL	28	30 000	K55	123

- Einfügen eines Angestellten:

Einsatz von Übergangsvariablen: NEW : NP.Anr . . . NP.Gehalt

- **Wie wird Trigger T1 ausgeführt?**

```
CREATE TRIGGER T1
AFTER  INSERT ON Pers                        (* Ereignis *)
REFERENCING NEW AS NP
FOR EACH ROW
        UPDATE Abt A                         (* Aktion *)
        SET A.Geh_Summe =
            A.Geh_Summe + NP.Gehalt
        WHERE A.Anr = NP.Anr;
```

Trigger-Einsatz (2)

Abt	Anr	Aname	Ort	Geh_Summe
	K51	PLANUNG	KAISERSLAUTERN	43500
	K53	EINKAUF	FRANKFURT	45200
	K55	VERTRIEB	FRANKFURT	80000

Pers	Pnr	Name	Alter	Gehalt	Anr	Mnr
	406	COY	47	50 000	K55	123
	123	MÜLLER	32	43 500	K51	-
	829	SCHMID	36	45 200	K53	777
	574	ABEL	28	30 000	K55	123

- Gehaltserhöhung von K55-Angestellten um 10 %:

Einsatz von Übergangsvariablen: OLD : OP.Anr . . . OP.Gehalt

NEW : NP.Anr . . . NP.Gehalt

- **Wie wird Trigger T2 ausgeführt?**

```
CREATE TRIGGER T2
AFTER  UPDATE OF Gehalt ON Pers                 (* Ereignis *)
REFERENCING OLD AS OP NEW AS NP
FOR EACH ROW
        UPDATE Abt A                            (* Aktion *)
        SET A.Geh_Summe =
            A.Geh_Summe + (NP.Gehalt - OP.Gehalt)
        WHERE A.Anr = NP.Anr;
```

Trigger-Einsatz (3)

Abt	Anr	Aname	Ort	Geh_Summe
	K51	PLANUNG	KAISERSLAUTERN	43500
	K53	EINKAUF	FRANKFURT	45200
	K55	VERTRIEB	FRANKFURT	80000

Pers	Pnr	Name	Alter	Gehalt	Anr	Mnr
	406	COY	47	50 000	K55	123
	123	MÜLLER	32	43 500	K51	-
	829	SCHMID	36	45 200	K53	777
	574	ABEL	28	30 000	K55	123

- Gehaltserhöhung von K55-Angestellten um 10 %:

  Einsatz von Übergangstabellen: OLD_TABLE: OT.Anr . . . OT.Gehalt

  NEW_TABLE: NT.Anr . . . NT.Gehalt

- **Wie wird Trigger T3 ausgeführt?**

```
CREATE TRIGGER T3
AFTER  UPDATE OF Gehalt ON Pers                    (* Ereignis *)
REFERENCING OLD_TABLE AS OT NEW_TABLE AS NT
FOR EACH STATEMENT
        UPDATE Abt A                               (* Aktion *)
        SET A.Geh_Summe = A.Geh_Summe +
             (SELECT SUM (Gehalt) FROM NT WHERE Anr = A.Anr) -
             (SELECT SUM (Gehalt) FROM OT WHERE Anr = A.Anr)
        WHERE A.Anr IN (SELECT Anr FROM NT);
```

Trigger-Einsatz (4)

- **Gültige Kombinationen**
  für Trigger-Granulate und Übergangstabellen und -variablen

Granularität	Aktivierungs zeit	Triggernde Operation	Übergangs-variablen erlaubt	Übergangstabellen erlaubt
ROW	BEFORE	INSERT	NEW	NONE
		UPDATE	OLD, NEW	
		DELETE	OLD	
	AFTER	INSERT	NEW	NEW_TABLE
		UPDATE	OLD, NEW	OLD_TABLE, NEW_TABLE
		DELETE	OLD	OLD_TABLE
STATE-MENT	BEFORE	INSERT	NONE	NONE
		UPDATE		
		DELETE		
	AFTER	INSERT	NONE	NEW_TABLE
		UPDATE		OLD_TABLE, NEW_TABLE
		DELETE		OLD_TABLE

Trigger-Beispiel (2)

- **Auch das ist ein Trigger!**

Triggername	Beschreibung (Trigger-Einsatz in einem Workflow-System)
ResultToS	Wenn das Resultat in den Status „S“ (abgeschlossen) wechselt, dann ist der Vorgang abgeschlossen. In der Tabelle CurrentFlow wird der Status des Vorgangs auf „S“ (abgeschlossen) geändert und das Attribut FlowTimeEnd auf die aktuelle Zeitmarke gesetzt. Danach wird der gesamte Vorgang in die History-Tabellen kopiert und aus den Current-Tabellen gelöscht.

```
CREATE TRIGGER ResultToS
AFTER UPDATE OF AcStatus ON CurrentAc
REFERENCING NEW AS n
FOR EACH ROW MODE DB2SQL
WHEN (n.AcChar = 'R' AND n.AcStatus = 'S')
BEGIN ATOMIC
    UPDATE CurrentFlow
    SET     FlowStatus = 'S',
            FlowTimeEnd =   CASE
                                    WHEN    FlowTimeEnd IS NULL
                                    THEN    CURRENT TIMESTAMP
                                    ELSE    FlowTimeEnd
                            END
    WHERE FlowId = n.FlowId;
    INSERT INTO HistoryFlow SELECT * FROM CurrentFlow WHERE FlowId = n.FlowId;
    INSERT INTO HistoryAc SELECT * FROM CurrentAc WHERE FlowId = n.FlowId;
    INSERT INTO HistoryTr SELECT * FROM CurrentTr WHERE FlowId = n.FlowId;
    INSERT INTO HistoryTrAfterAc    SELECT *
                                    FROM CurrentTrAfterAc
                                    WHERE FlowId = n.FlowId;
    INSERT INTO HistoryTrBeforeAc   SELECT *
                                    FROM CurrentTrBeforeAc
                                    WHERE FlowId = n.FlowId;
    DELETE FROM CurrentTrBeforeAc WHERE FlowId = n.FlowId;
    DELETE FROM CurrentTrAfterAc WHERE FlowId = n.FlowId;
    DELETE FROM CurrentTr WHERE FlowId = n.FlowId;
    DELETE FROM CurrenAc WHERE FlowId = n.FlowId;
    DELETE FROM CurrentFlow WHERE FlowId = n.FlowId
END;
```

Zusammenfassung

- **Semantische Integritätskontrolle**
  - Relationale Invarianten, referentielle Integrität und Aktionen
  - Benutzerdefinierte Integritätsbedingungen (*assertions*)
    - ➥ zentrale Spezifikation/Überwachung im DBS wird immer wichtiger
- **Aktives DB-Verhalten zur**
  - Integritätssicherung
  - automatische Korrekturen durch Trigger
  - Verantwortung des Benutzers (Reihenfolge/Terminierung)
- **Triggerkonzept in SQL99 standardisiert**