we focus on students

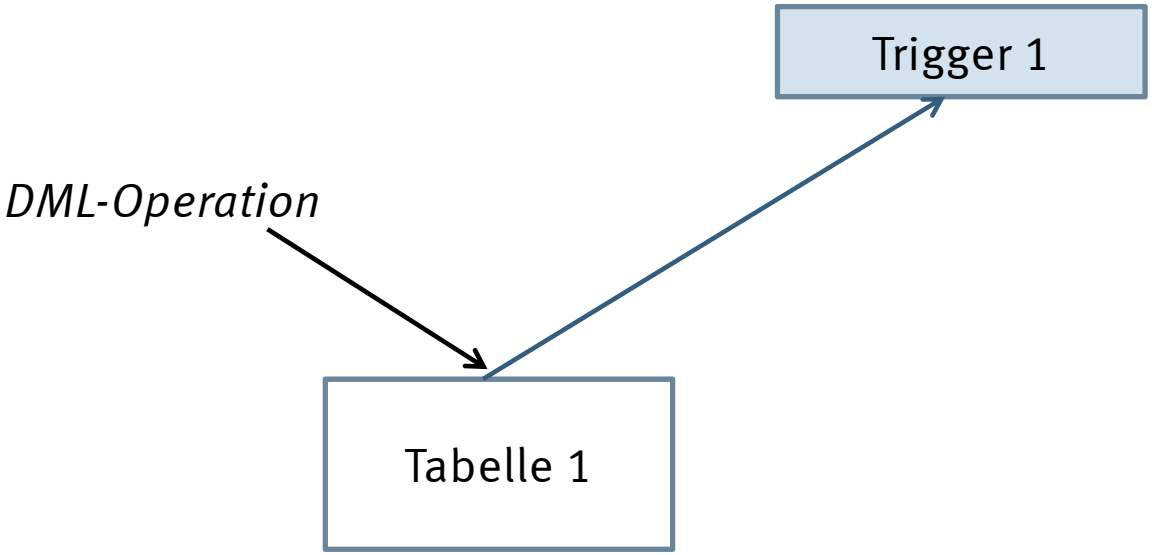
Datenbankprogrammierung

Trigger (Konzept)

Fachhochschule Dortmund
University of Applied Sciences

Trigger

Ein **Trigger** ist ein ausführbares Programm, welches durch ein Ereignis ausgelöst (=getriggert) wird.

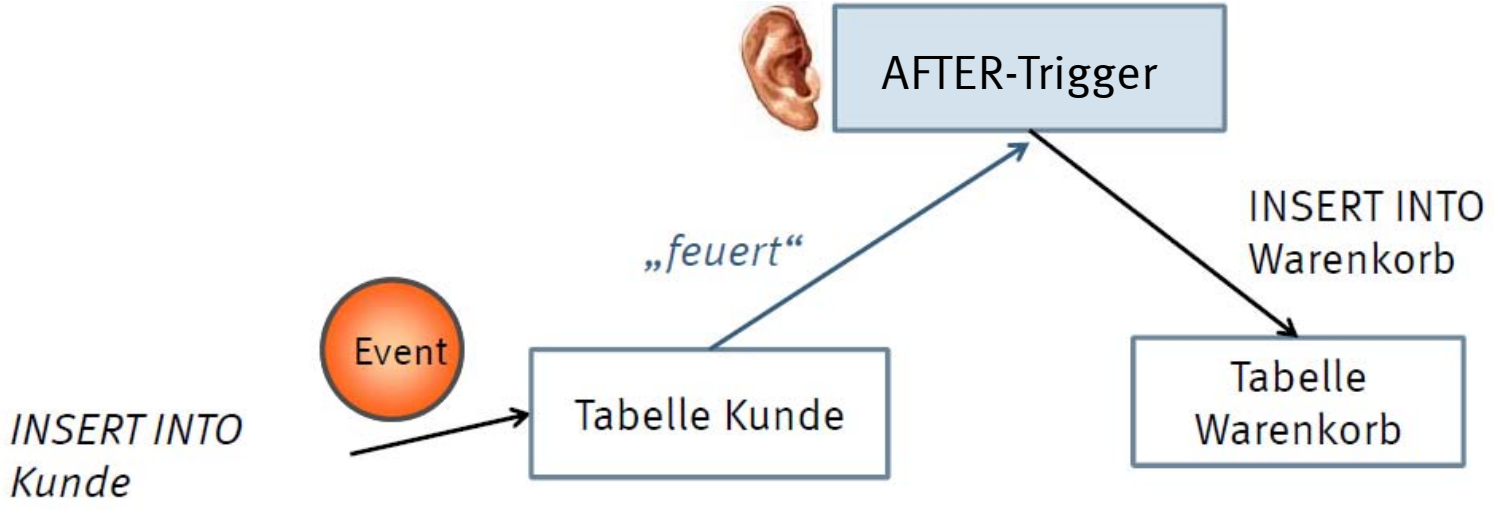


AFTER-Trigger

Nachverarbeitung von Daten in der Datenbank

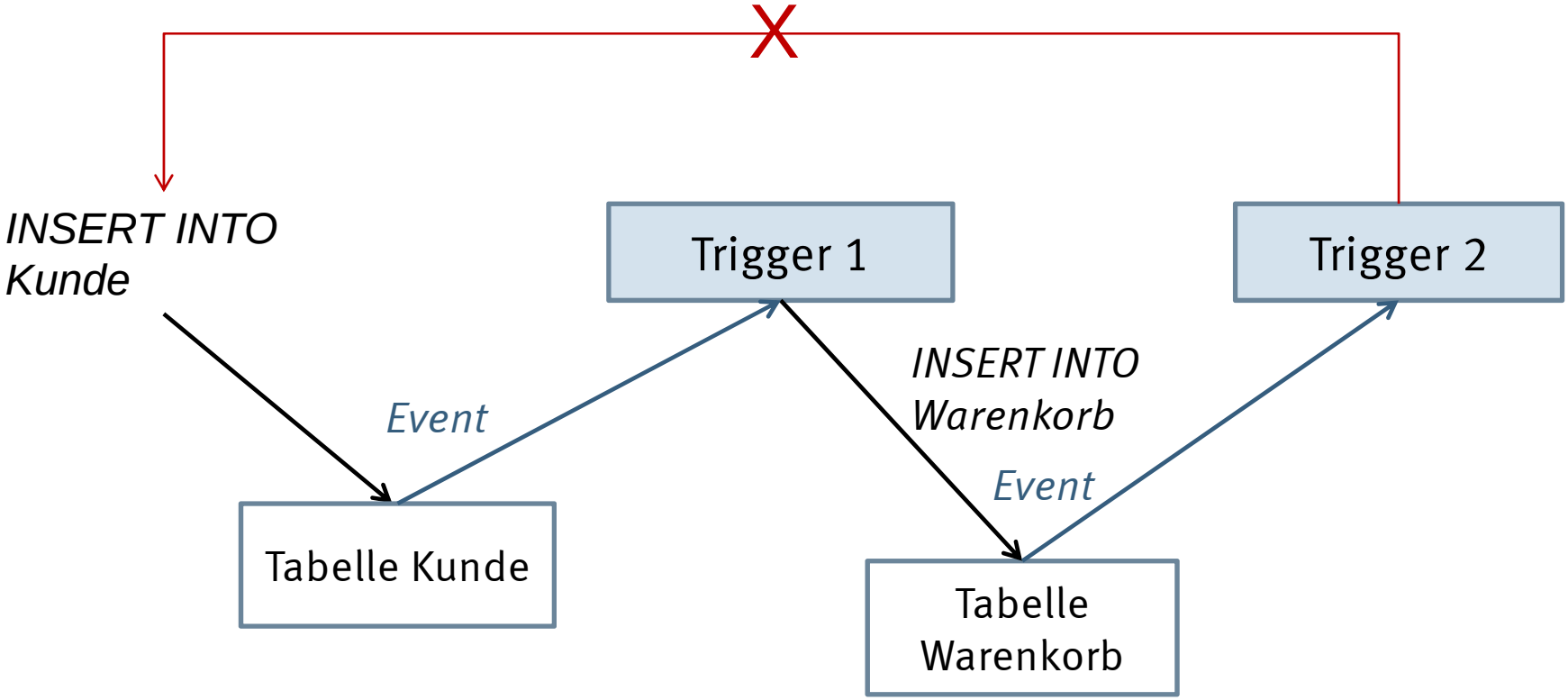
Beispiel:

Einfügen eines Willkommenspräsents in den Warenkorb eines neuen Kunden

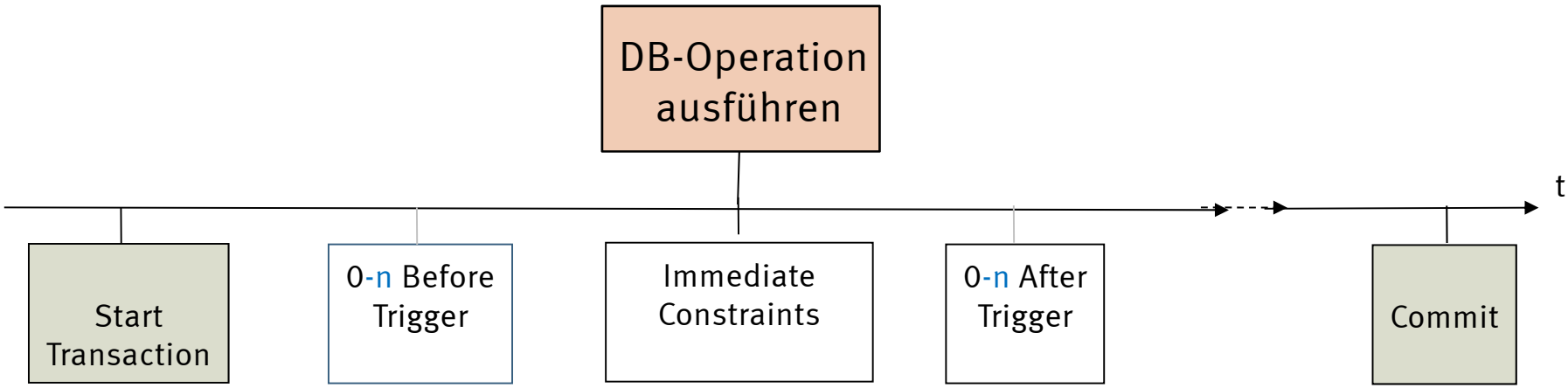


Eventsteuerung in SQL - Trigger

Eine Schleifenbildung durch Trigger ausgelöste DB-Operationen darf nicht erfolgen. Daher kann ein Trigger nur DML-Operation ausführen, die sich **nicht** auf die zugehörende Tabelle beziehen.



Zeitpunkt der Triggerausführung

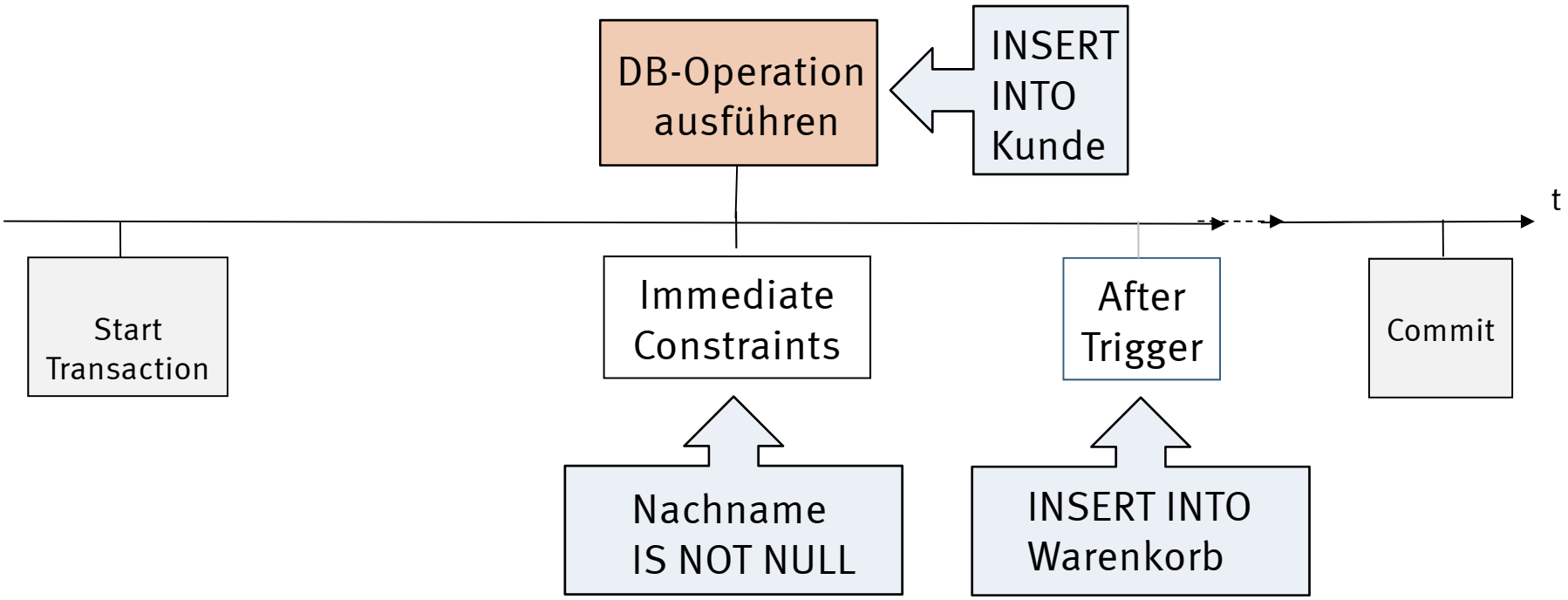


AFTER-Trigger

Nachverarbeitung von Daten in der Datenbank

Beispiel:

Jeder neue Kunde erhält ein Willkommenspräsent.

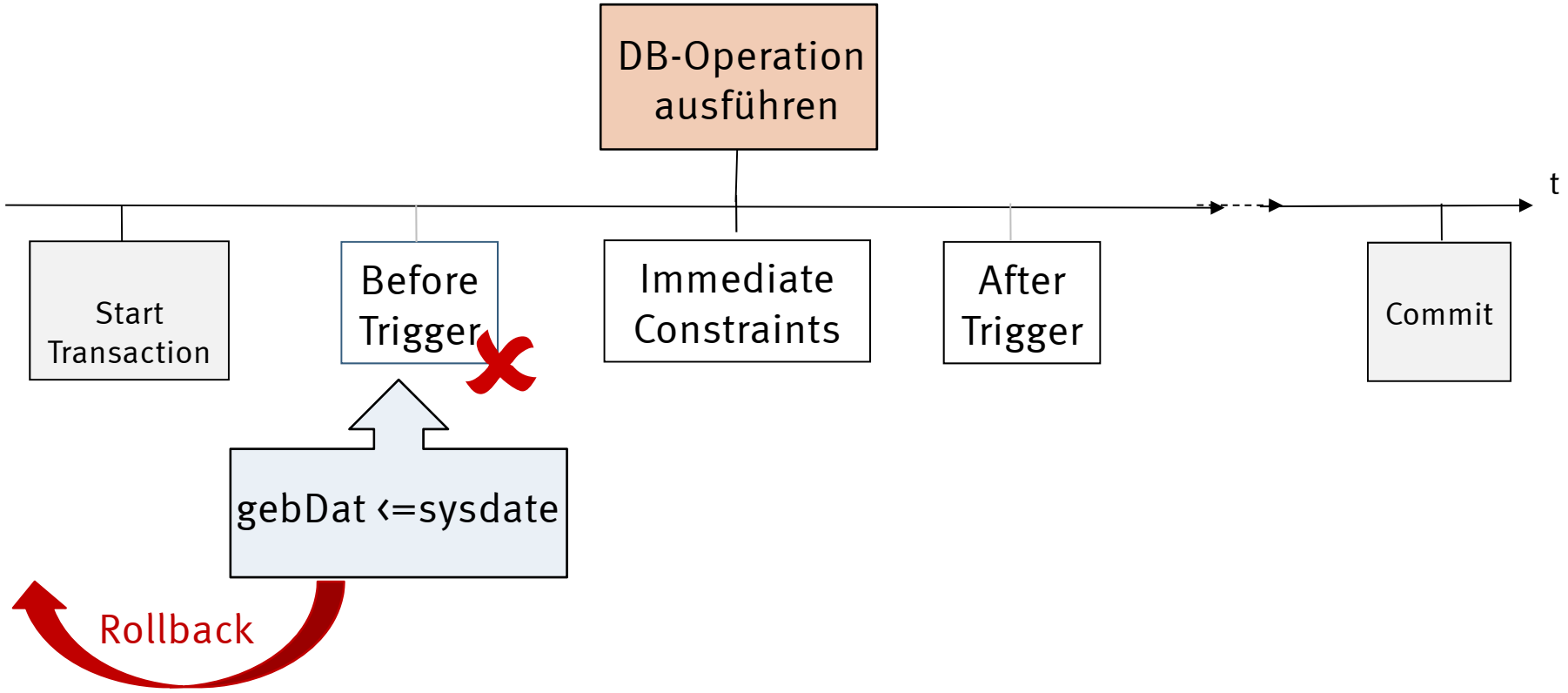


BEFORE-Trigger

Integritätsprüfung **vor** der Ausführung einer Datenänderung

Beispiel:

Das Geburtsdatum eines Kunden darf nicht in der Zukunft liegen.

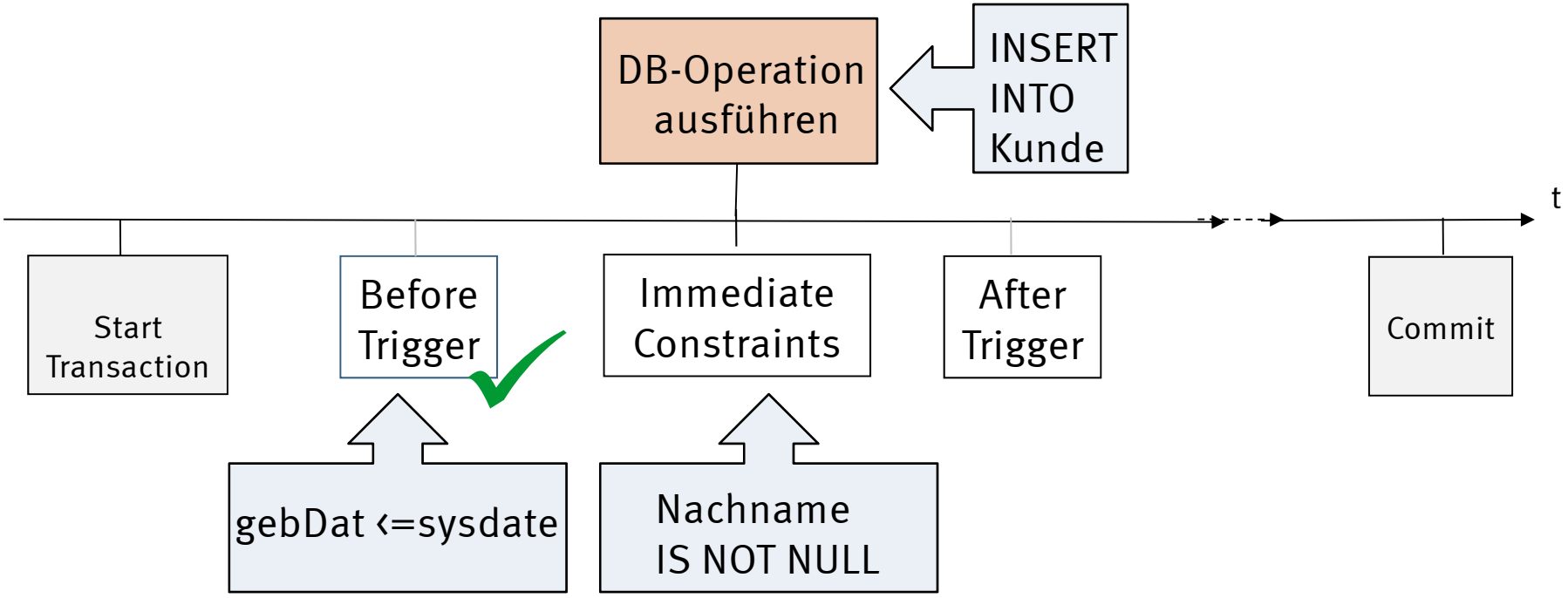


BEFORE-Trigger

Integritätsprüfung **vor** der Ausführung einer Datenänderung

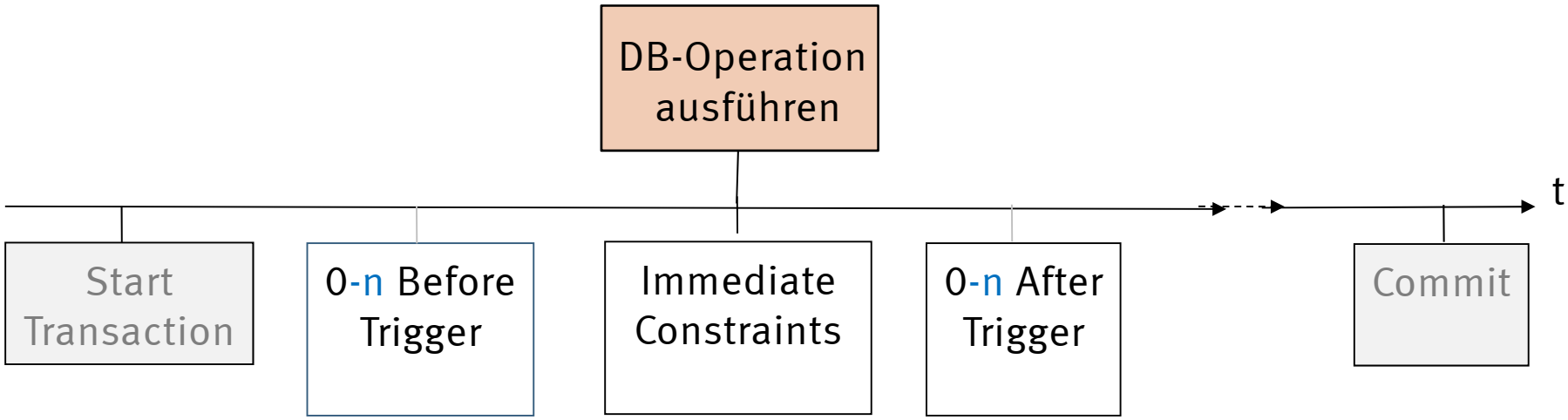
Beispiel:

Das Geburtsdatum eines Kunden darf nicht in der Zukunft liegen.



Zeitpunkt der Triggerausführung

- Bevor eine Datenbankoperation ausgeführt wird, werden die zugehörenden Before-Trigger ausgeführt.
- Nach der Ausführung der Datenbankoperation werden die zugehörenden After-Trigger ausgeführt.
- Alle Änderungsoperationen finden in einem Transaktionskontext statt, so dass im Fehlerfall die gesamte Bearbeitung abgebrochen und zurückgesetzt wird.



Anwendungsfälle für Datenbank-Trigger

Trigger werden zur Konsistenzüberwachung und zur Herstellung der Datenintegrität (Wahrung der Korrektheit von Daten) genutzt.

- Anwendungsfälle sind …
  - Nachverarbeitung von Daten in der Datenbank

    Beispiel:

    Jeder neue Kunde erhält ein Willkommenspräsent
  - Integritätsprüfung vor der Ausführung der Datenänderung

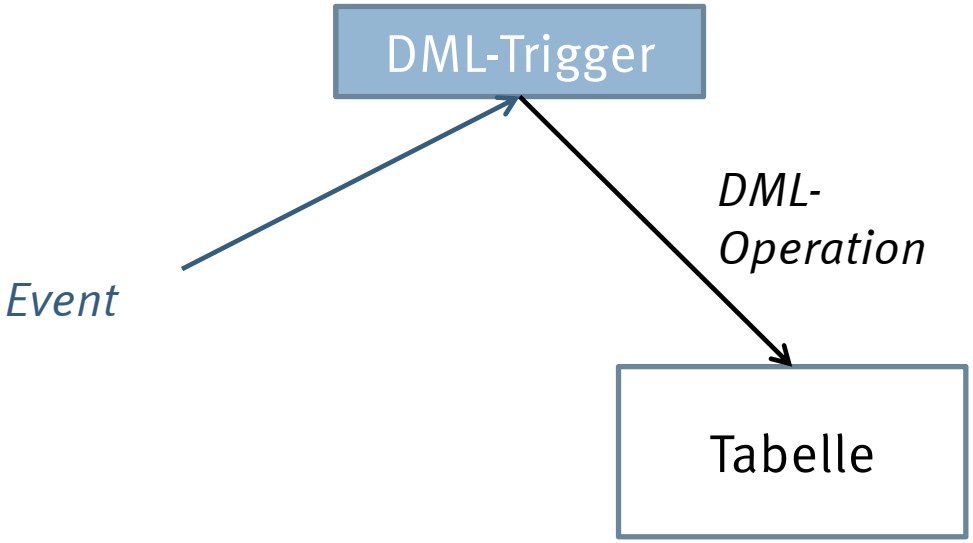
    Beispiel:

    Das Geburtsdatum eines Kunden darf nicht in der Zukunft liegen
  - Integritätsherstellung

    Beispiel:

    Ein Artikel kann nur gelöscht werden, wenn kein Lagerbestand vorhanden ist.

Triggerarten



Event	MySQL	Oracle
DML-Operation (auf Tabelle)	Ja	DML-Trigger
DML-Operation (auf View)	Nein	Instead-Of-Trigger
DDL-Operation	Nein	DDL-Trigger
DB-Operation (Startup, Shutdown, Logon, Logoff, Servererror)	Nein	Database-Event-Trigger
Abbruch eines Statements (z.B. Überschreitung des Tablespace-Quota)	Nein	After-Suspend Trigger

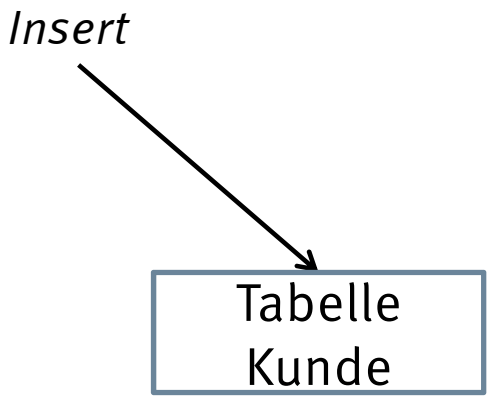
we
focus
on
students

Datenbankprogrammierung

Trigger implementieren

Fachhochschule
Dortmund
University of Applied Sciences

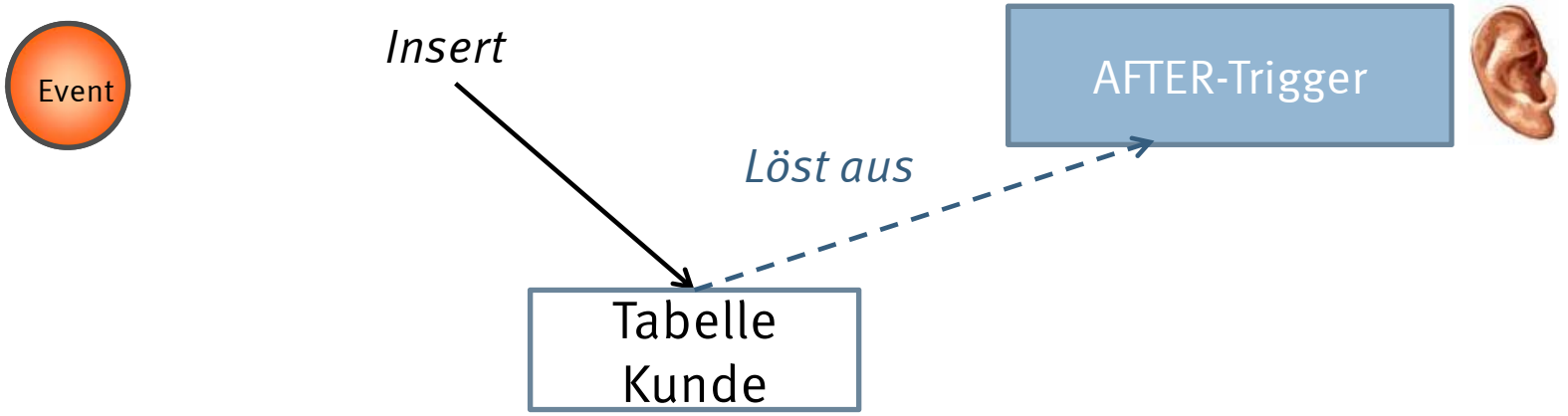
Fragestellung



Logtabelle

Zeitstempel	User	Action

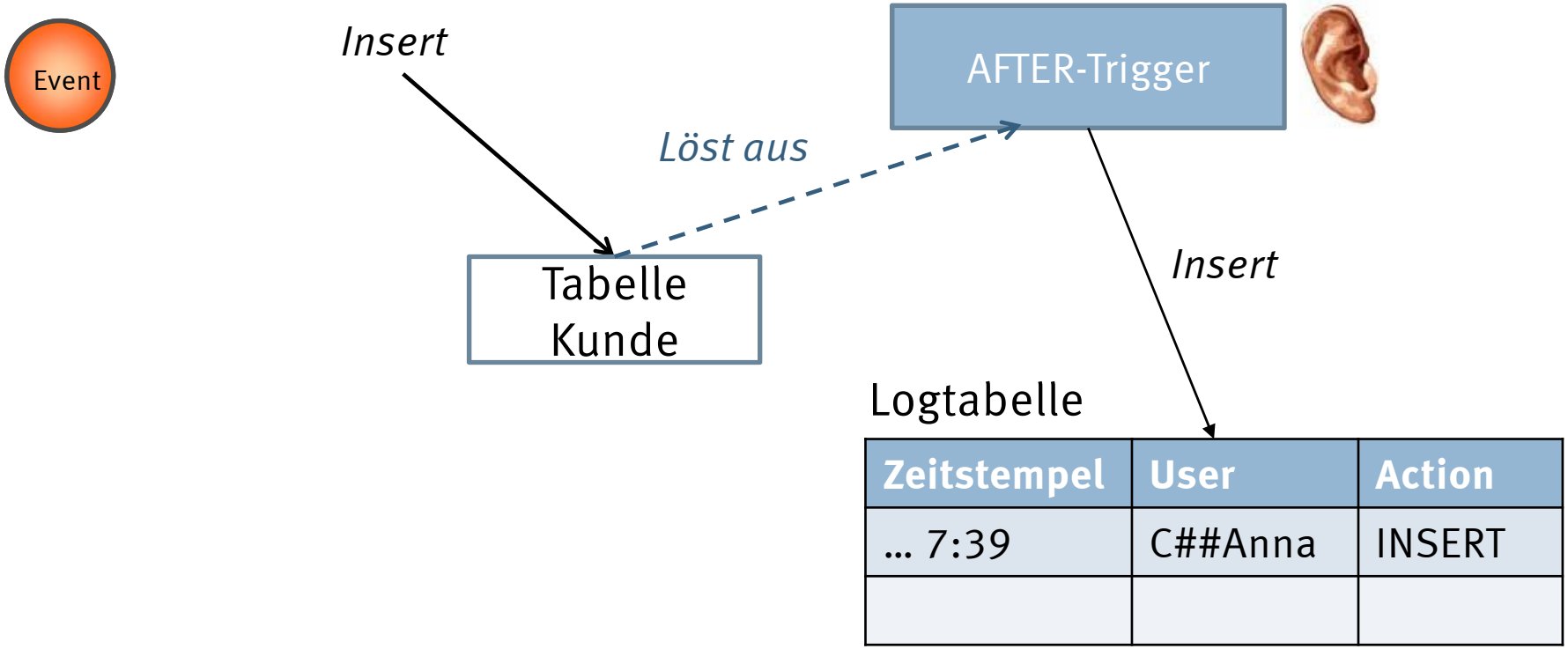
Fragestellung



Logtabelle

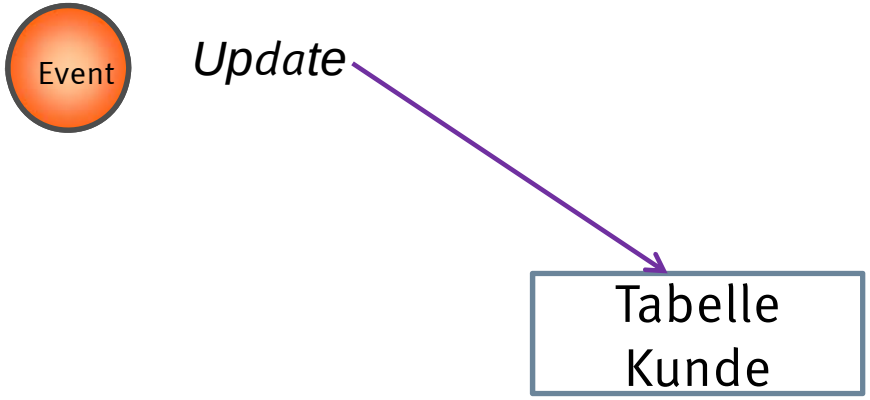
Zeitstempel	User	Action

Fragestellung



Zeitstempel	User	Action
… 7:39	C##Anna	INSERT

Fragestellung

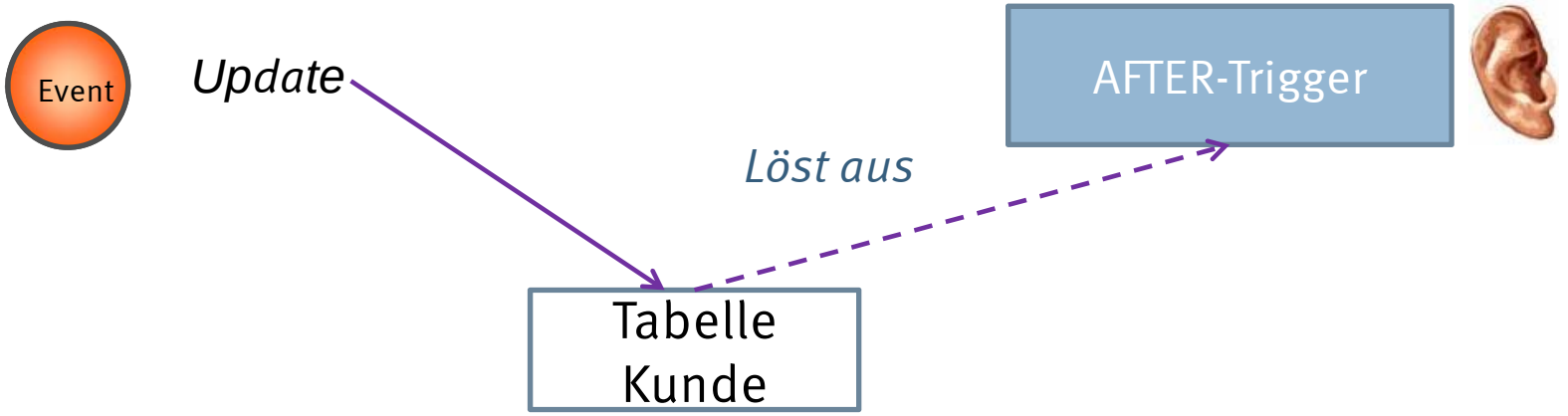


AFTER-Trigger

Logtabelle

Zeitstempel	User	Action
… 7:39	C##Anna	INSERT

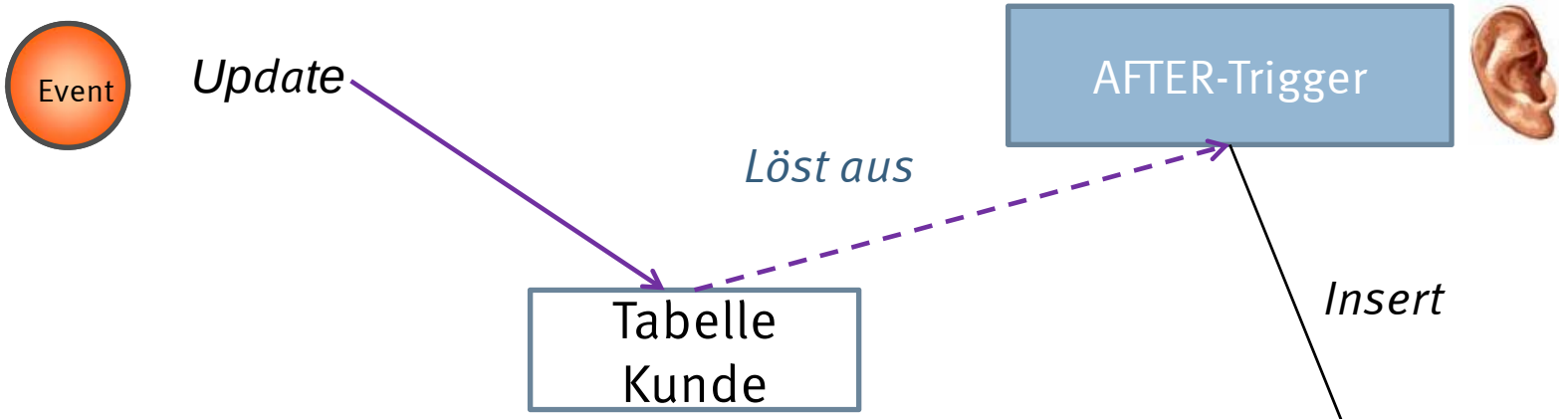
Fragestellung



Logtabelle

Zeitstempel	User	Action
… 7:39	C##Anna	INSERT

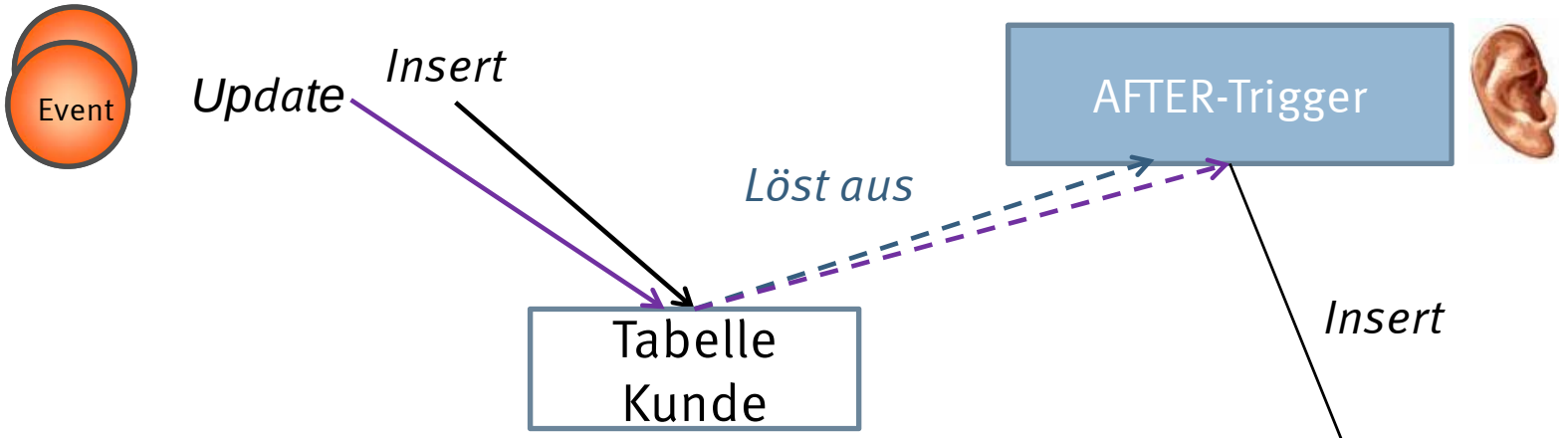
Fragestellung



Logtabelle

Zeitstempel	User	Action
… 7:39	C##Anna	INSERT
… 7:45	C##Jan	UPDATE

Fragestellung

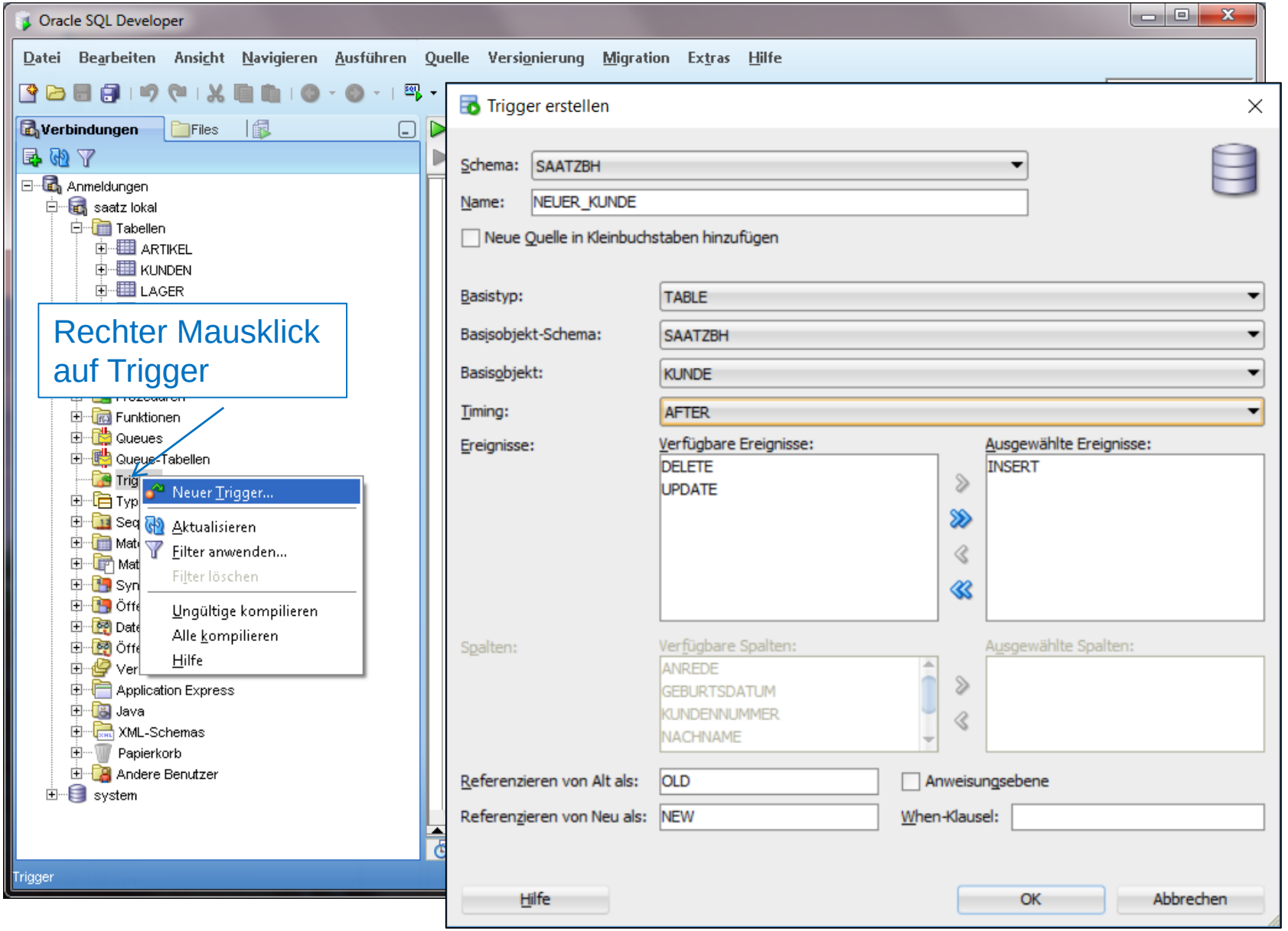


Logtabelle

Zeitstempel	User	Action
… 7:39	C##Anna	INSERT
… 7:45	C##Jan	UPDATE

Wie kann dieser Trigger implementiert werden?

Trigger in Oracle anlegen



Befehlsorientierter Trigger

- Beispiel:
  Es sollen in der Tabelle aenderungen die ausgeführten DML-Operationen mit dem zugehörenden Login auf der Tabelle Kunde dokumentiert werden.

```
CREATE TABLE aenderungen(
   log_datum TIMESTAMP PRIMARY KEY,
   login     VARCHAR(20),
   aenderung VARCHAR(10));
```

```
CREATE OR REPLACE
TRIGGER KUNDENLOG_TRIGGER
AFTER INSERT OR DELETE OR UPDATE ON KUNDE
DECLARE  action VARCHAR(10);
         userlogin VARCHAR(20);
BEGIN
         select user INTO userlogin from dual;
         IF INSERTING THEN action := 'Insert';
         ELSIF UPDATING THEN action := 'Update';
         ELSIF DELETING THEN action := 'Delete';
         END IF;
         INSERT INTO aenderungen (log_datum, login, aenderung)
         VALUES (SYSDATE, userlogin, action);
END;
```

Event abfragen

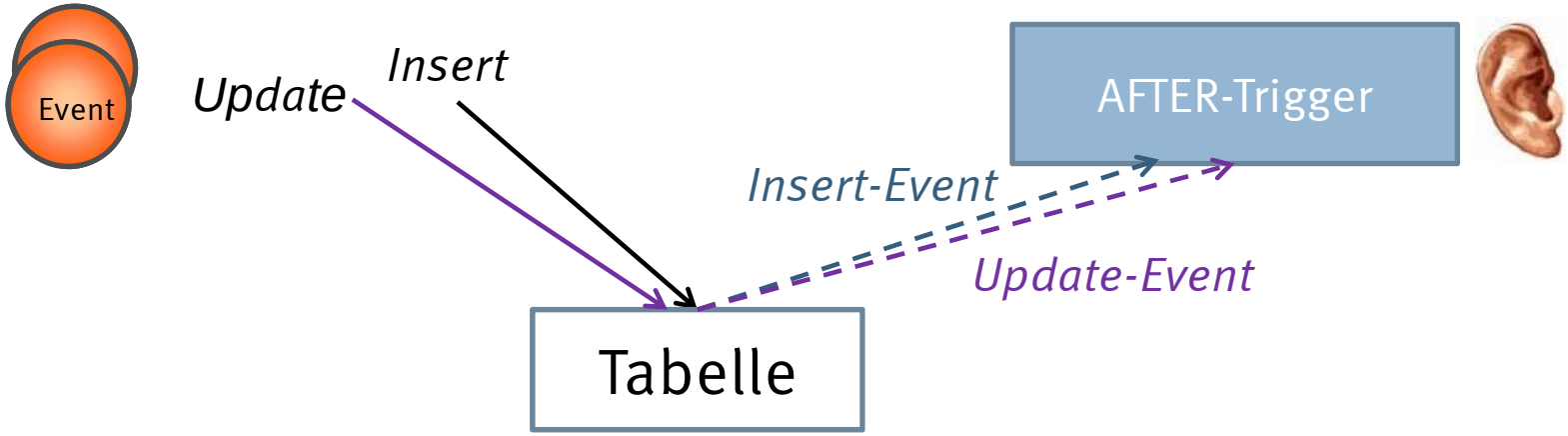
Im Trigger Events erkennen (Oracle)

In Oracle können mehrere Events denselben Trigger auslösen. Zur Unterscheidung, welches Event den Trigger ausgelöst hat, können die booleschen Systemvariablen

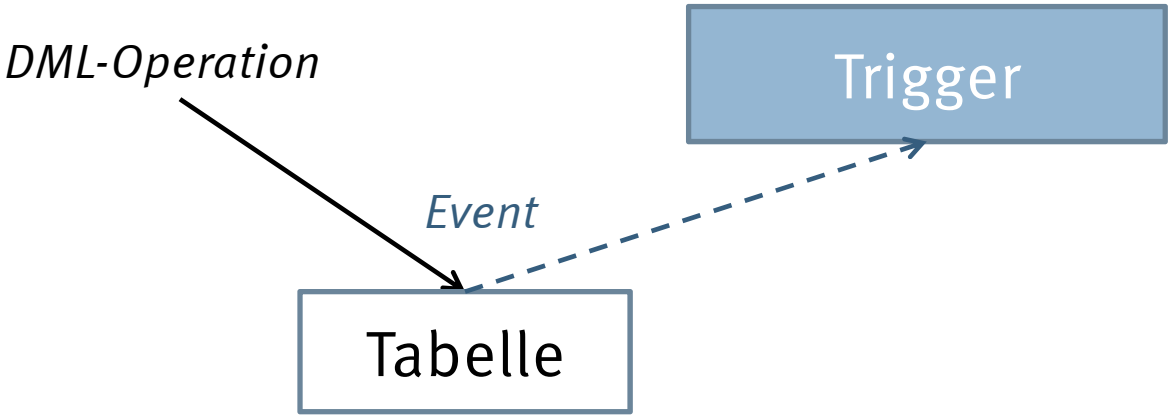
- Inserting
- Deleting
- Updating
- Updating(‹Spalte›)

ausgewertet.

```
IF deleting THEN … END IF;
```



Syntax eines DML-Triggers (Oracle)



ECA-Regeln

EVENT →

CONDITION →

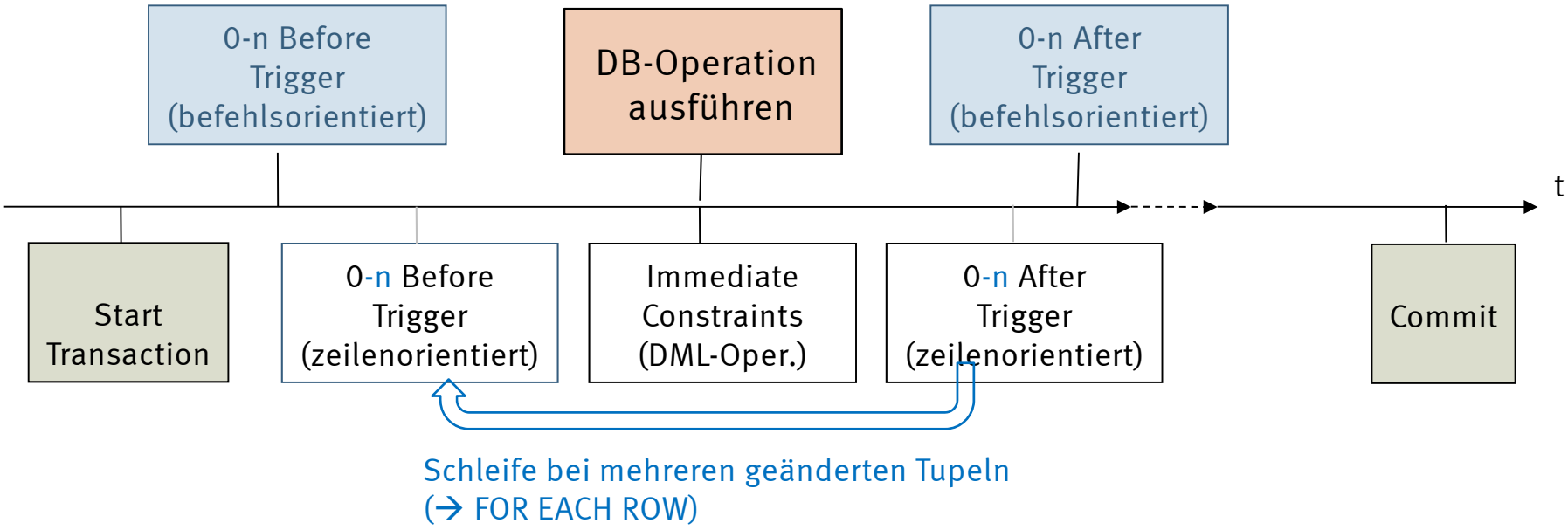
ACTION →

```
CREATE OR REPLACE TRIGGER ‹trigger_name›

{ BEFORE|AFTER}

{INSERT OR UPDATE [of ‹spalte1, spalte2, ...›]
        OR DELETE} ON ‹tabellenname›

[FOR EACH ROW]
[WHEN ‹ausführungsbedingung›]

‹anweisungen›
```

Einschränkung des Events, z.B. nur auf Änderung des Attributs Wohnort

Weitere Einschränkung des Events, z.B. auf einen Insert von Kunden aus Dortmund

Befehls- und Zeilenorientierte Trigger



Datenbankprogrammierung

Dynamische Integritätsbedingungen

Fachhochschule
Dortmund
University of Applied Sciences

Fragestellung

Dynamische Integritätsbedingung

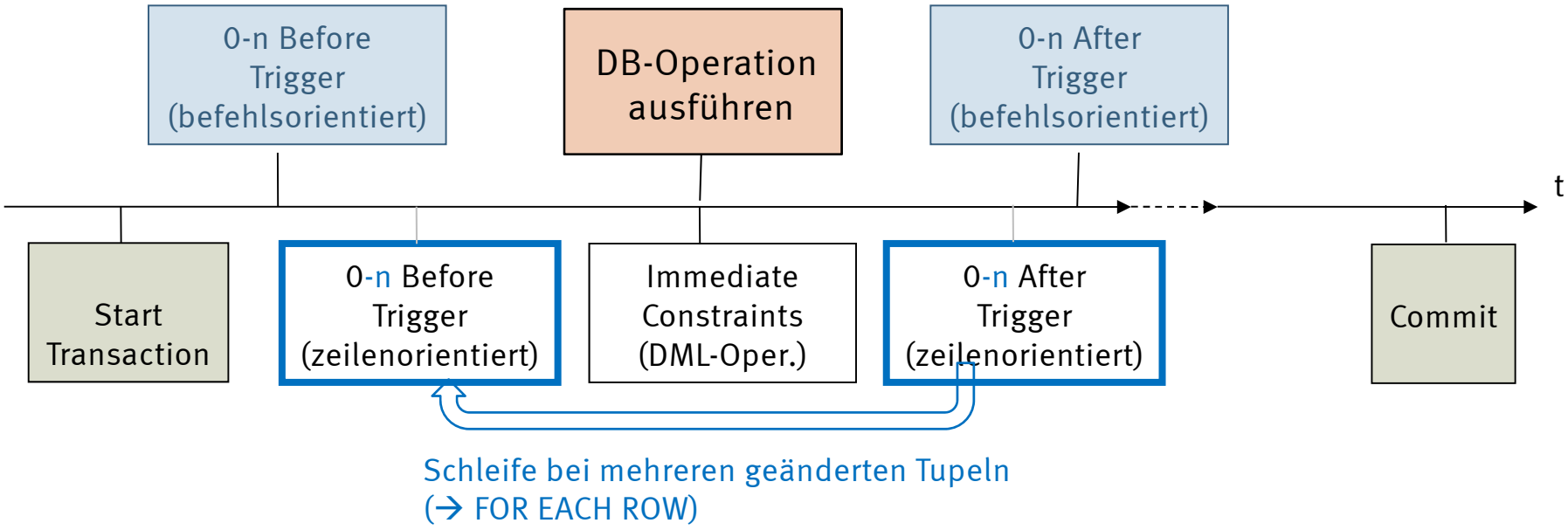
Beispiel:

Eine Bestellung kann nur innerhalb von 14 Tagen widerrufen werden.

Welcher Trigger ist zu implementieren?

Zugriff auf Attributwerte

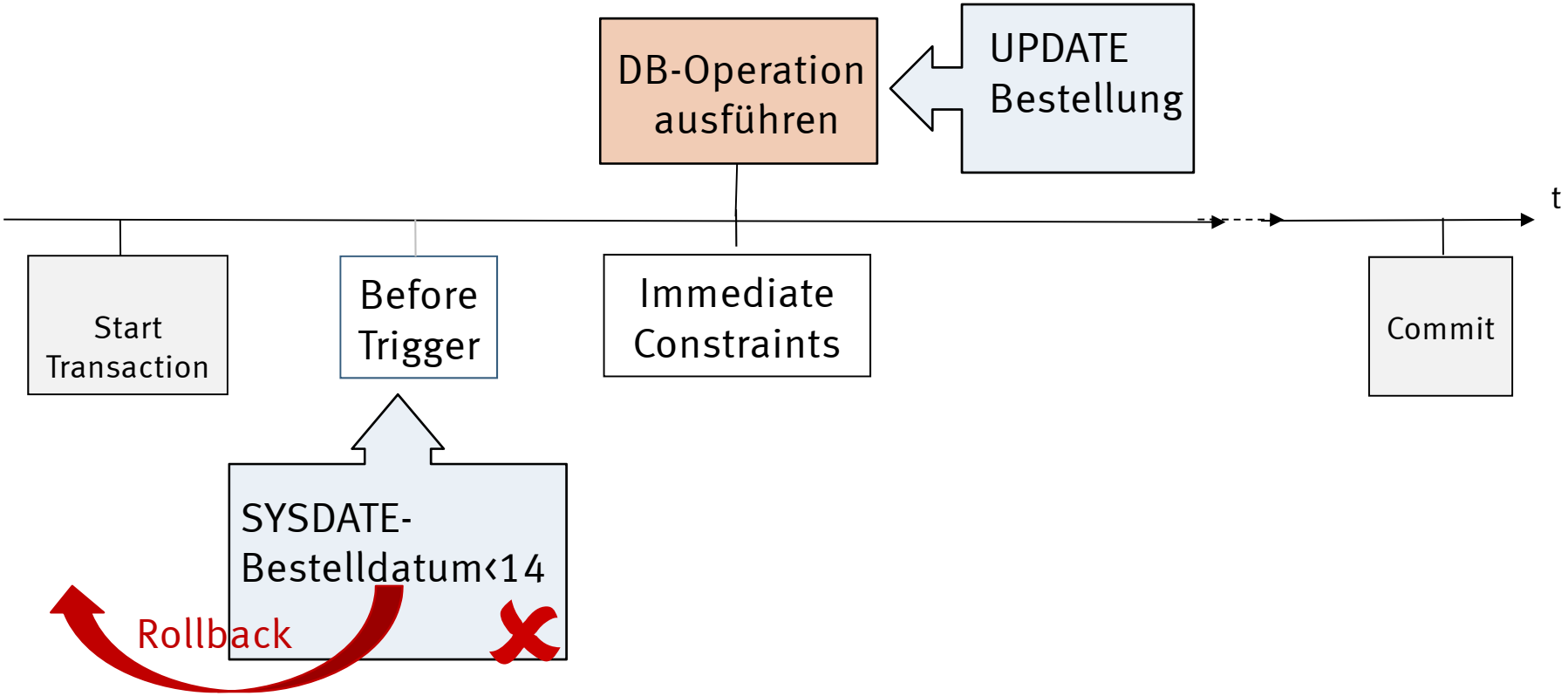
Nur in einem zeilenorientierten Trigger kann auf die Attribute des bearbeiteten Tupels (Tupelvariable) zugegriffen werden.



Integritätsprüfung

Beispiel:

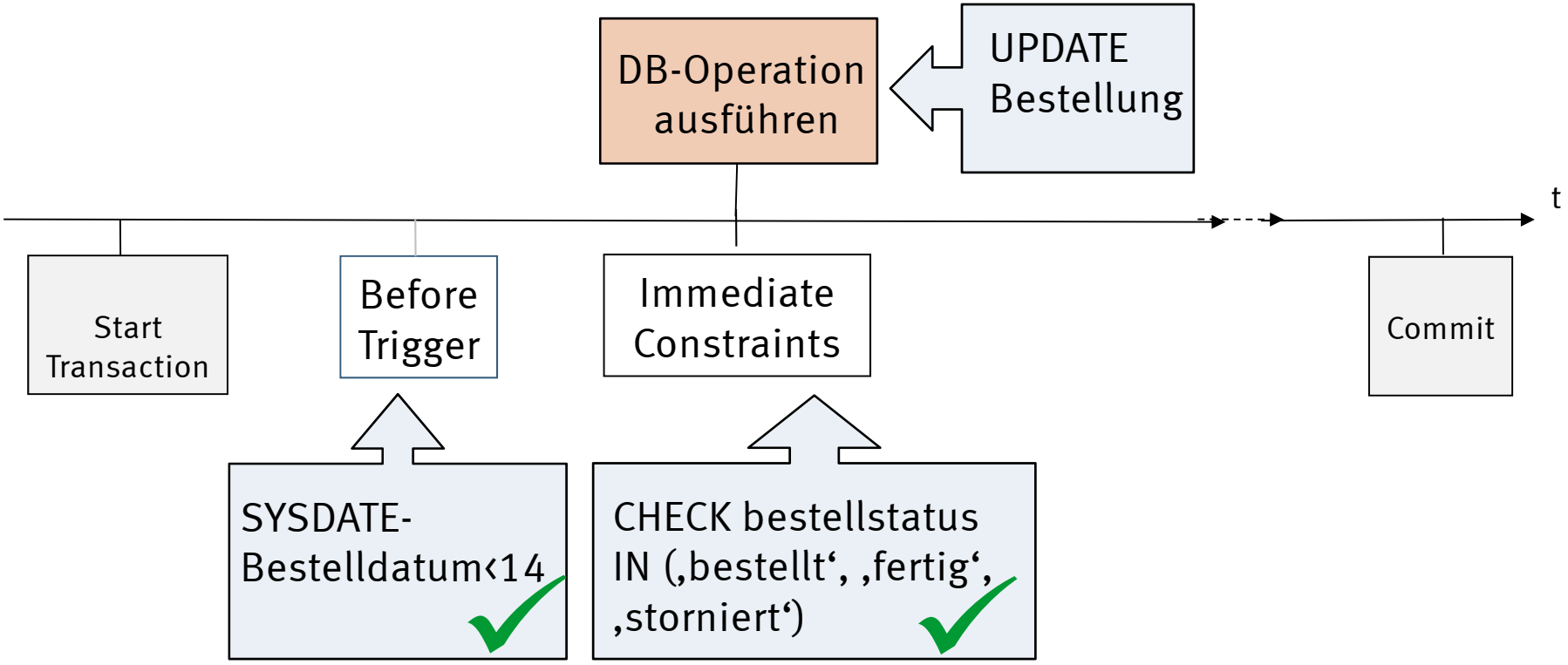
Eine Bestellung kann nur innerhalb von 14 Tagen widerrufen werden.



Integritätsprüfung

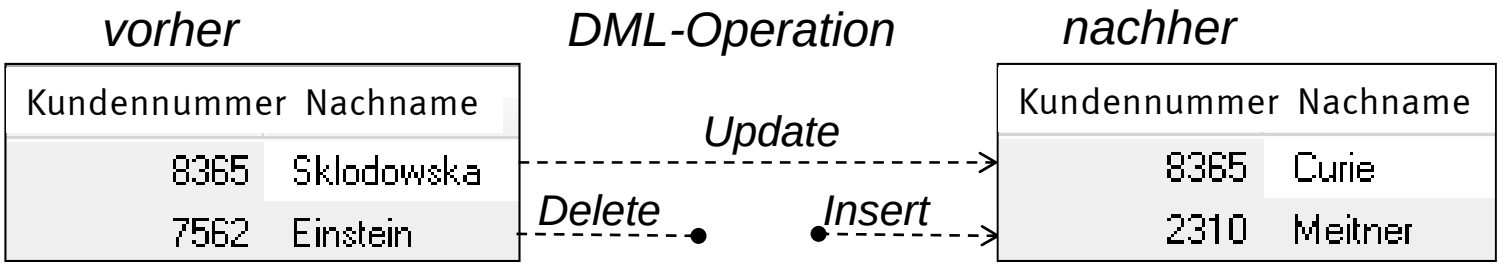
Beispiel:

Eine Bestellung kann nur innerhalb von 14 Tagen widerrufen werden.



Tupelzugriff im zeilenorientierten Trigger

Der Zugriff auf Tupelvariablen erfolgt mittels der old- und new-Präfixe. Old wird für den vorherigen Attributwert und new für den geänderten Wert verwendet.



Im Trigger zur DML-Operation	old. (alter Wert)	new. (neuer Wert)	Beispiel
INSERT	null	neuer Wert	:new.Nachname='Meitner'
UPDATE	vorheriger Wert	neuer Wert	:old.Nachname='Sklodowska' :new.Nachname='Curie'
DELETE	vorheriger Wert	null	:old.Nachname='Einstein'

Zusammenfassung

Integritätsregel:
Eine Bestellung darf nur innerhalb von 14 Tagen widerrufen werden.

```
CREATE OR REPLACE TRIGGER Trigger_Bestellstatus
BEFORE UPDATE(Bestellstatus) ON Bestellung
FOR EACH ROW
WHEN (new.Bestellstatus ='storniert')
DECLARE
   bestelldatumException   EXCEPTION;
BEGIN
   IF (SYSDATE - :old.Bestelldatum ›14) THEN
            RAISE bestelldatumException;
   END IF;
EXCEPTION
            WHEN bestelldatumException
            THEN raise_application_error (-20500, 'Stornierung nicht mehr möglich');
END;
```

Achtung: In WHEN-Klausel kein : vor new!

Fehler deklarieren

Fehler werfen

Fehler behandeln

we focus on students

Datenbankprogrammierung

Instead of Trigger

Fachhochschule Dortmund
University of Applied Sciences

Beispiel Instead-of-Trigger (Oracle)

View:

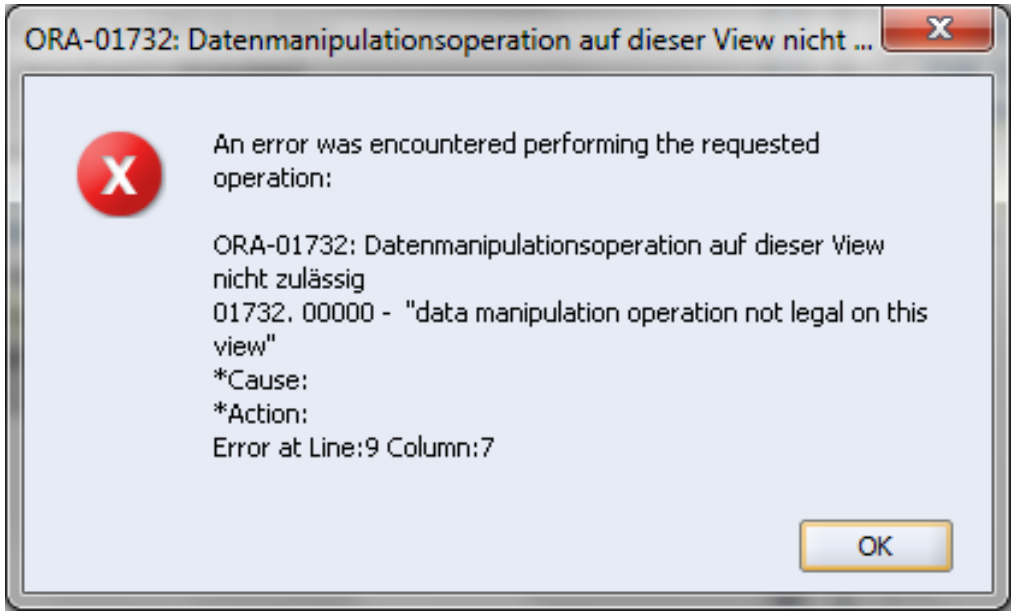
```
CREATE VIEW view_geschaeftskunde
AS
SELECT Kundennummer, Nachname, Ort, COUNT(Anzahl) Anzahl
FROM       Kunde NATURAL JOIN Warenkorb
WHERE ANREDE IS NULL
GROUP BY Kundennummer, Name, Ort
```

Änderung:

```
UPDATE view_geschaeftskunde
SET Ort = 'Bochum'
WHERE  Kundennummer=8523
```

Funktioniert nicht !!

Weshalb?



Beispiel Instead-of-Trigger (Oracle)

Die Anweisungen des Instead-of-Trigger werden anstelle der Änderungsoperation auf dem nicht direkt änderbaren View ausgeführt. Im Beispiel wird so die Änderung des Wohnortes ermöglicht, jedoch nicht des Kundennamens.

Nicht direkt änderbares View:

```
CREATE VIEW view_geschaeftskunde
AS
SELECT Kundennummer, Nachname, Ort, COUNT(Anzahl) Anzahl
FROM    Kunde NATURAL JOIN Warenkorb
WHERE ANREDE IS NULL
GROUP BY Kundennummer, Nachname, Ort
```

Event:

```
UPDATE view_geschaeftskunde
SET Ort = 'Bochum'
WHERE  Kundennummer=8523
```

Trigger:

```
CREATE OR REPLACE TRIGGER geschaeftskunde_update
INSTEAD OF UPDATE ON view_geschaeftskunde
FOR EACH ROW
BEGIN
        UPDATE Kunde
        SET Ort = :NEW.ORT
        WHERE Kundennummer= :OLD.Kundennummer;
END;
```

Datenbankprogrammierung

Tabellenzugriff aus einem Trigger

Fachhochschule
Dortmund
University of Applied Sciences

Beispiel FHKennung

Tabelle FHKennung:

```
CREATE TABLE FHKennung(
Login          CHAR(8) PRIMARY KEY,
Vorname     VARCHAR(20),
Nachname  VARCHAR(20)
);
```

Beispiel

Vorname: Marie

Nachname: Sklodowska

FHKennung:

MaSkl1

MaSkl2, wenn MaSKl 1x vorhanden, …

Integritätsregeln:

1. Wenn ein neuer Nutzer eingefügt wird, dann soll die FHKennung anhand des bereits vorhandenen Datenbestandes dynamisch ermittelt werden.
2. Wenn sich der Nachname der Person ändert, dann soll auch die FHKennung entsprechend angepasst werden.

Triggerimplementierung

Trigger ermittelt zu einem Namen die passende FHKennung

```
CREATE OR REPLACE TRIGGER TriggerFHKennung
BEFORE INSERT OR UPDATE ON FHKennung
FOR EACH ROW
DECLARE
 anzahl INTEGER DEFAULT 0;
BEGIN
   SELECT count(*) INTO anzahl
   FROM FHKennung
   WHERE Nachname like Substr(:new.Nachname,1,3) || '%'
   AND   Vorname  like Substr(:new.Vorname,1,2)|| '%';
   :new.Login := Substr(:new.Vorname,1,2) || Substr(:new.Nachname,1,3)|| (anzahl+1);
END;
```

Einfügen einer Kennung funktioniert:

```
INSERT INTO FHKennung (Nachname, Vorname)
VALUES ('Sklodowska', 'Marie');
SELECT * FROM FHKennung;
```

kriptausgabe x | Abfrageergebnis x

SQL | Alle Zeilen abgerufen:1 in 0,005 Sekunden

	LOGIN	VORNAME	NACHNAME
1	MaSkl1	Marie	Sklodowska

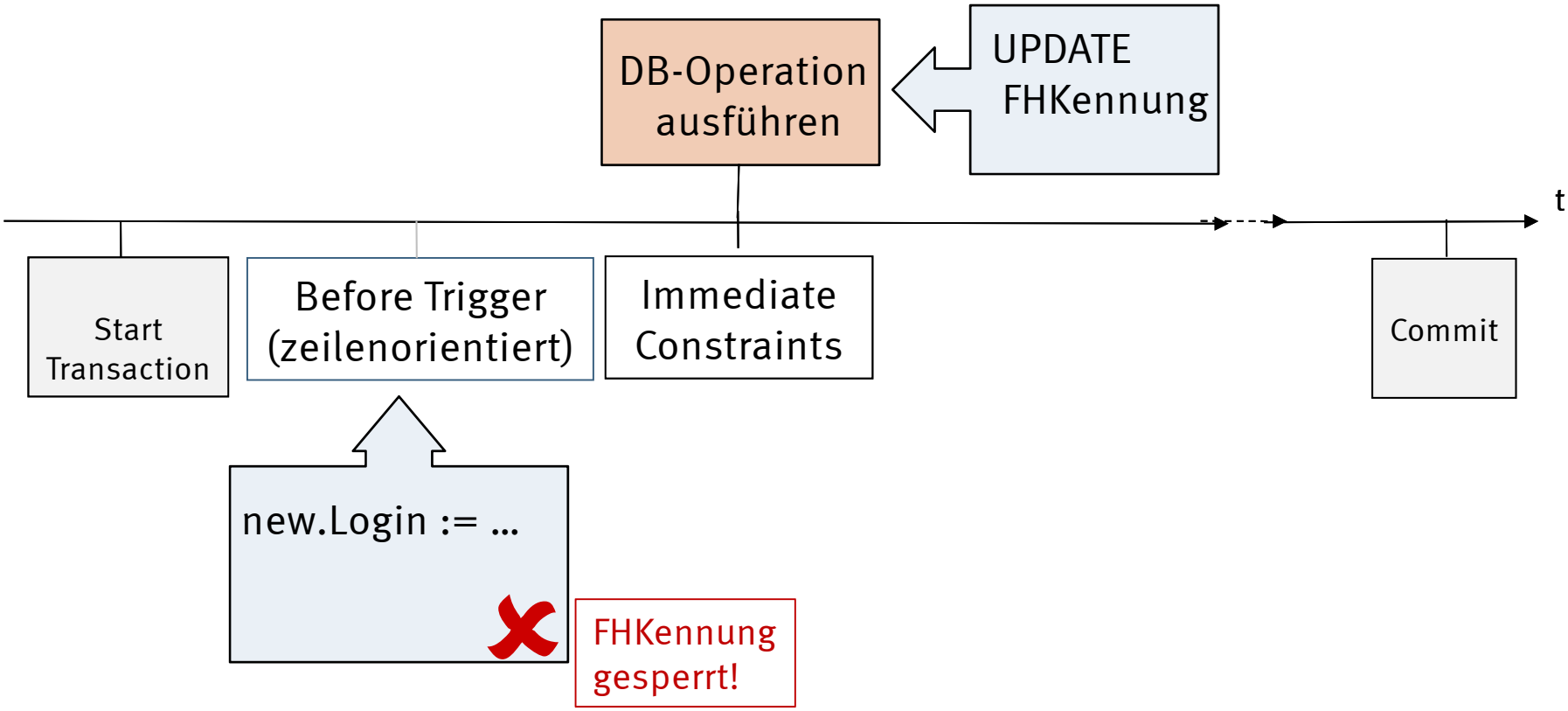
Test des Triggers (Oracle)

Das Ändern der Kennung funktioniert nicht:

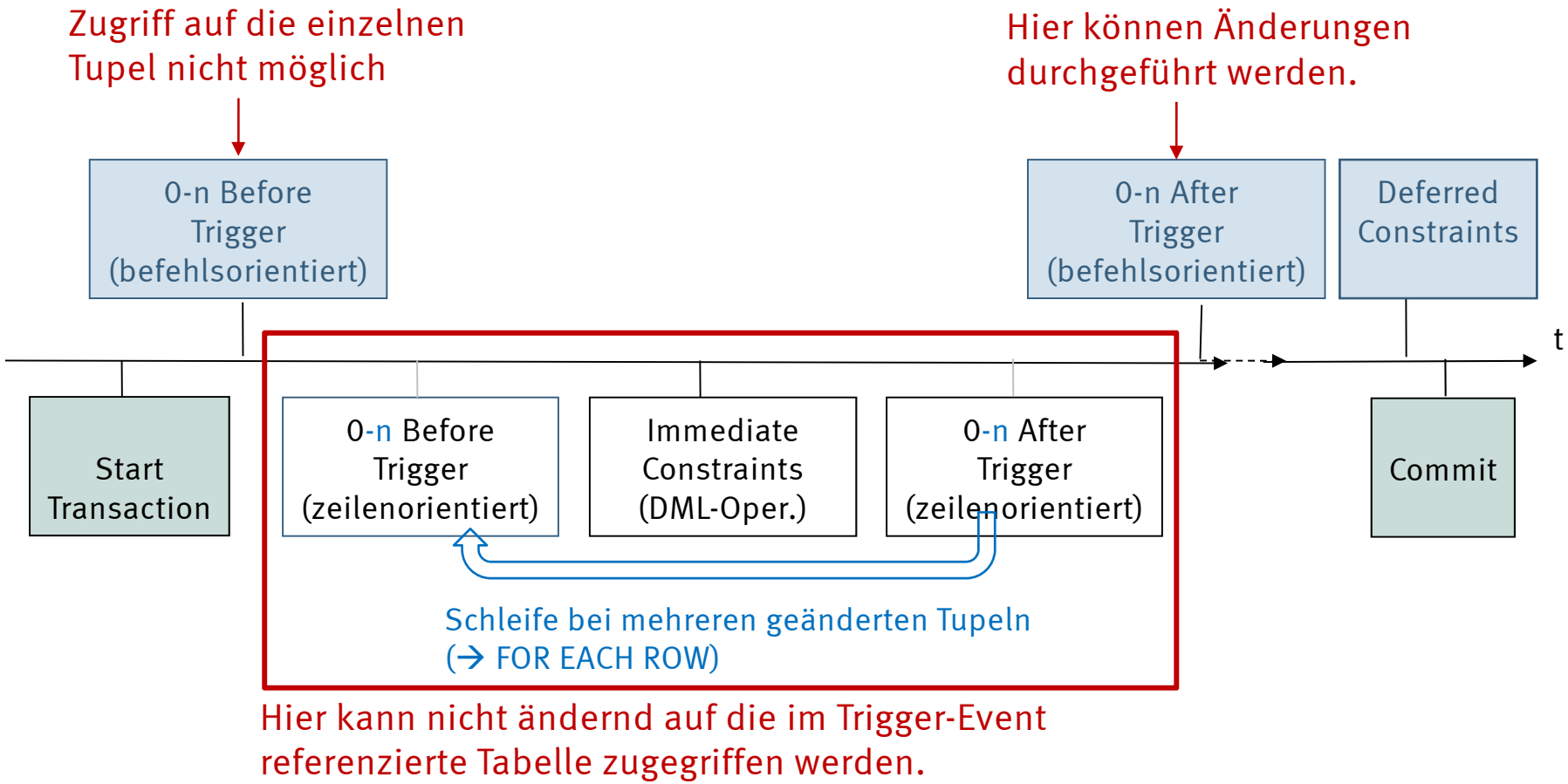
```
Fehler beim Start in Zeile : 8 in Befehl -
UPDATE FHKennung
SET Nachname='Curie'
WHERE Login='MaSkl1'
Fehlerbericht -
SQL-Fehler: ORA-04091: table SAATZ.FHKENNUNG is mutating, trigger/function may not see it
ORA-06512: at "SAATZ.INSERTFHKENNUNG", line 4
ORA-04088: error during execution of trigger 'SAATZ.INSERTFHKENNUNG'
04091. 00000 -  "table %s.%s is mutating, trigger/function may not see it"
*Cause:    A trigger (or a user defined plsql function that is referenced in
           this statement) attempted to look at (or modify) a table that was
           in the middle of being modified by the statement which fired it.
*Action:   Rewrite the trigger (or function) so it does not read that table.
```

Tabellenzugriff

Ein zeilenorientierter Trigger kann nicht auf die Tabelle selbst zugreifen, auf die er selbst definiert ist. Die Tupel ist aufgrund der auslösenden (DML-)Operation bereits für weitere Zugriffe gesperrt (exklusive Schreibsperre).



Ausführungsreihenfolge der Integritätsprüfung



Lösungsidee

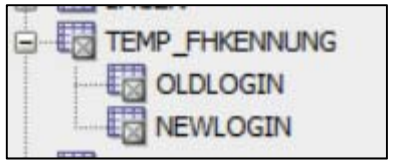
1. Schritt: In einem BEFORE Trigger die Änderungen in eine Hilfstabelle schreiben
2. Schritt: Die Hilfstabelle mit einem befehlsorientierter After-Trigger auswerten
3. Schritt: Die Inhalte der Hilfstabelle wieder löschen

Temporäre Tabellen

Bei einer temporären Tabelle werden die Daten automatisch vom Datenbanksystem gelöscht. Die Lebensdauer der Daten ist entweder bis zum Ende der laufenden Transaktion oder bis zum Ende der Session. Die Verwendung von temporären Tabellen bietet deutliche Performance-Vorteile gegenüber "normalen" Tabellen. Die Verwendung von Fremdschlüssel-Constraints ist mit temporärer Tabelle als Ziel nicht erlaubt.

- 1. Schritt: Änderungen in eine Hilfstabelle schreiben
  - Erstellen einer temporären Hilfstabelle

```
CREATE GLOBAL TEMPORARY TABLE
TEMP_FHKennung(
oldLogin  VARCHAR(8),
newLogin  VARCHAR(5)
)ON COMMIT DELETE ROWS;
```



Tabelleninhalte werden am Transaktionsende (Commit) gelöscht

Anm.: Oracle kennt nur globale temporäre Tabellen, daher ist keine dynamische Erstellung im Trigger möglich.

Tabellenzugriff im Trigger

2. Schritt: BEFORE-Trigger schreibt die Änderungen in die Hilfstabelle

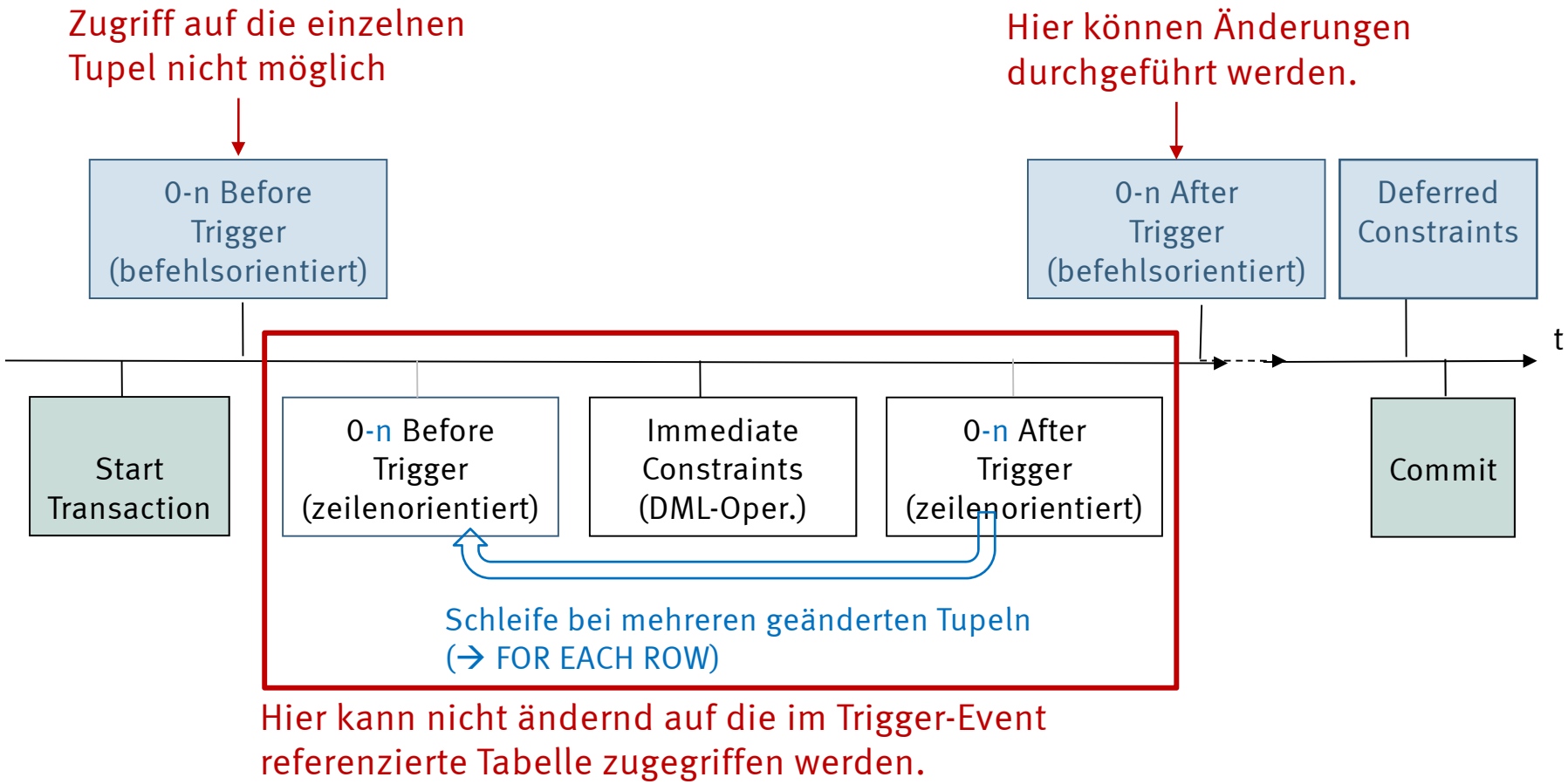
```
13 CREATE OR REPLACE TRIGGER TriggerFHKennung
14 BEFORE INSERT ON FHKennung
15 FOR EACH ROW
16 DECLARE
17     anzahl INTEGER DEFAULT 0;
18 BEGIN
19     SELECT count(*) INTO anzahl
20     FROM FHKennung
21     WHERE Login like Substr(:new.Vorname,1,2)||Substr(:new.Nachname,1,3)||'%';
22     :new.Login:= Substr(:new.Vorname,1,2)||Substr(:new.Nachname,1,3)||(anzahl+1);
23 END;
24 /
25 CREATE OR REPLACE TRIGGER Namensaenderung
26 BEFORE UPDATE OF Vorname, nachname ON FHKennung
27 FOR EACH ROW
28 BEGIN
29     INSERT INTO TEMP_FHKennung (oldLogin, newLogin)
30     VALUES (:old.Login, Substr(:new.Vorname,1,2)|| Substr(:new.Nachname,1,3));
31 END;
32 /
```

Tabellenzugriff im Trigger

3. Schritt: AFTER-Trigger führt die Änderungen aus

```
33  CREATE OR REPLACE TRIGGER AFTER_UPDATE_FHKennung
34  AFTER UPDATE OF Vorname, Nachname ON FHKennung
35  DECLARE anzahl INTEGER :=1;
36      CURSOR c_fhkennung IS
37          SELECT newLogin, oldLogin FROM TEMP_FHKennung;
38  BEGIN
39      FOR tupel in c_fhkennung
40      LOOP
41          SELECT count(*) INTO anzahl FROM FHKennung
42          WHERE Login like tupel.newLogin || '%';
43          UPDATE FHKennung
44          SET Login = tupel.newLogin || (anzahl+1)
45          WHERE Login = tupel.oldLogin;
46      END Loop;
47  END;
48  /
```

Zusammenfassung



Lösungsidee

1. Schritt: In einem BEFORE Trigger die Änderungen in eine Hilfstabelle schreiben
2. Schritt: Die Hilfstabelle mit einem befehlsorientierter After-Trigger auswerten
3. Schritt: Die Inhalte der Hilfstabelle wieder löschen