Kap. 3: DB-interne Programmierung

a) Stored Procedures mit PL/SQL

- "**Stored Procedures**" sind Codeblöcke, die in der Datenbank global einmal gespeichert werden (und verfügbar sind) und von der Datenbank ausgeführt werden
- **Prozedur** (ohne Rückgabetyp) **Funktion** (mit Rückgabebefehl)
- DECLARE -> AS ->Begin -> Exception -> End

```
CREATE [OR REPLACE] PROCEDURE proc_name [list of parameters]
AS
       Declaration section (DECLARE)
BEGIN
       Execution section
[EXCEPTION
       Exception section ]
END;
```

PL/SQL: Variablen und Datentypen

Variablen werden im DECLARE-Teil definiert

- Syntax: `variable_nametype [CONSTANT] [NOTNULL] [:=value];`
- Mit %TYPE kann der Datentyp einer Datenbankspalte referenziert werden z.B. `students.first_name%TYPE`

PL/SQL: Ablauf

1. Schritt: **Schreiben** der Prozedur bzw. der Funktion in PL/SQL
2. Schritt: **Speichern** der Prozedur bzw. der Funktion in der Datenbank
3. Schritt: **Ausführen**
   - Prozedur: als eigenständiger Aufruf
   - Funktion: als Teil eines Ausdrucks

```
create or replace
  function get_extra_hours
    (p_employee_id in employee.employee_id%type)
    return number is
  v_hours number;
begin
  select work_hours into v_hours
  from work_hour
  where employee_id = p_employee_id;

  return v_hours;
end;
```

b.) Aktive Datenbanksysteme

Aktives DBMS: Ein Datenbanksystem ist aktiv, wenn es auf (externe oder interne) Ereignisse durch (externe oder interne) Aktionen reagiert.

- Durch eine Regelbasis wird festgelegt, auf welche Ereignisse wie reagiert werden soll
- Regeln werden häufig auch in folgender Form spezifiziert:
  - ON Ereignis IF Bedingung DO Aktion
  - ON Ereignis DO (IF Bedingung THEN Aktion)
- Der effektive Programmablauf wird dadurch dynamisch verändert
- Der Programmablauf ist ggf. nichtdeterministisch

Einsatz

- Integritätsbedingungen (auch transitional, Bsp: Gehälter dürfen nicht sinken)
- Workflow-Steuerung (Benachrichtigung bei Fehlern)
- Protokollierung
- Anstoßen externer Aktionen

Vorteile

- Anwendungsprogrammierung wird entlastet
- Weniger Wissen im Code, mehr Wissen in der Datenbank

Nachteile:

- Nicht-Determinismus,
- Ablauf der Entwicklung bzgl. DB-Code (git),

Trigger

- Trigger können gerollbacked werden
- Wenn ein Ereignis eintritt, überprüfe ob die Bedingung erfüllt ist. Falls ja, dann führe die zugehörige Aktion aus.
  - Die Ausführungsreihenfolge ist im Allgemeinen nicht-deterministisch.
- Ereignistypen: Zeitereignisse, Datenbankereignisse, DBMS Ereignisse

```
CREATE [OR REPLACE] TRIGGER trig_name
Event z.B. AFTER UPDATE OF column ON table_name
[FOR EACH ROW]
[WHEN condition]
BEGIN
      Execution section
END:
```

Zeilenbasierter Trigger ("FOR EACH ROW")

- **Nur Zeilen-Trigger können auf :new & :old zugreifen, Statement-Trigger können das nicht**

```
create or replace TRIGGER trg_name
AFTER UPDATE ON personen
FOR EACH ROW
BEGIN
      -- Aktionen, kann z.B. :new.name verwenden
END;
```

Statement- Trigger (OHNE "FOR EACH ROW")

- **Nur Zeilen-Trigger können auf :new & :old zugreifen, Statement-Trigger können das nicht**

```
create or replace TRIGGER trg_name
AFTER UPDATE ON personen
BEGIN
      -- Aktionen, kann kein :new.name verwenden !!
END;
```

Mutation-Table-Problem (mutierender Trigger) *bei Zeilenbasierten Trigger!*

- Reihenfolgenunabhängigkeit wird verletzt
- Trigger löst sich selbst aus (mutiert)
  - Ein **zeilenbasierter Trigger** darf die Zieltabelle nicht verwenden