Datenbankprogramme – am Beispiel PL/SQL

Datenbanktechnologien

Prof. Dr. Ingo Claßen Prof. Dr. Martin Kempa

Hochschule für Technik und Wirtschaft Berlin

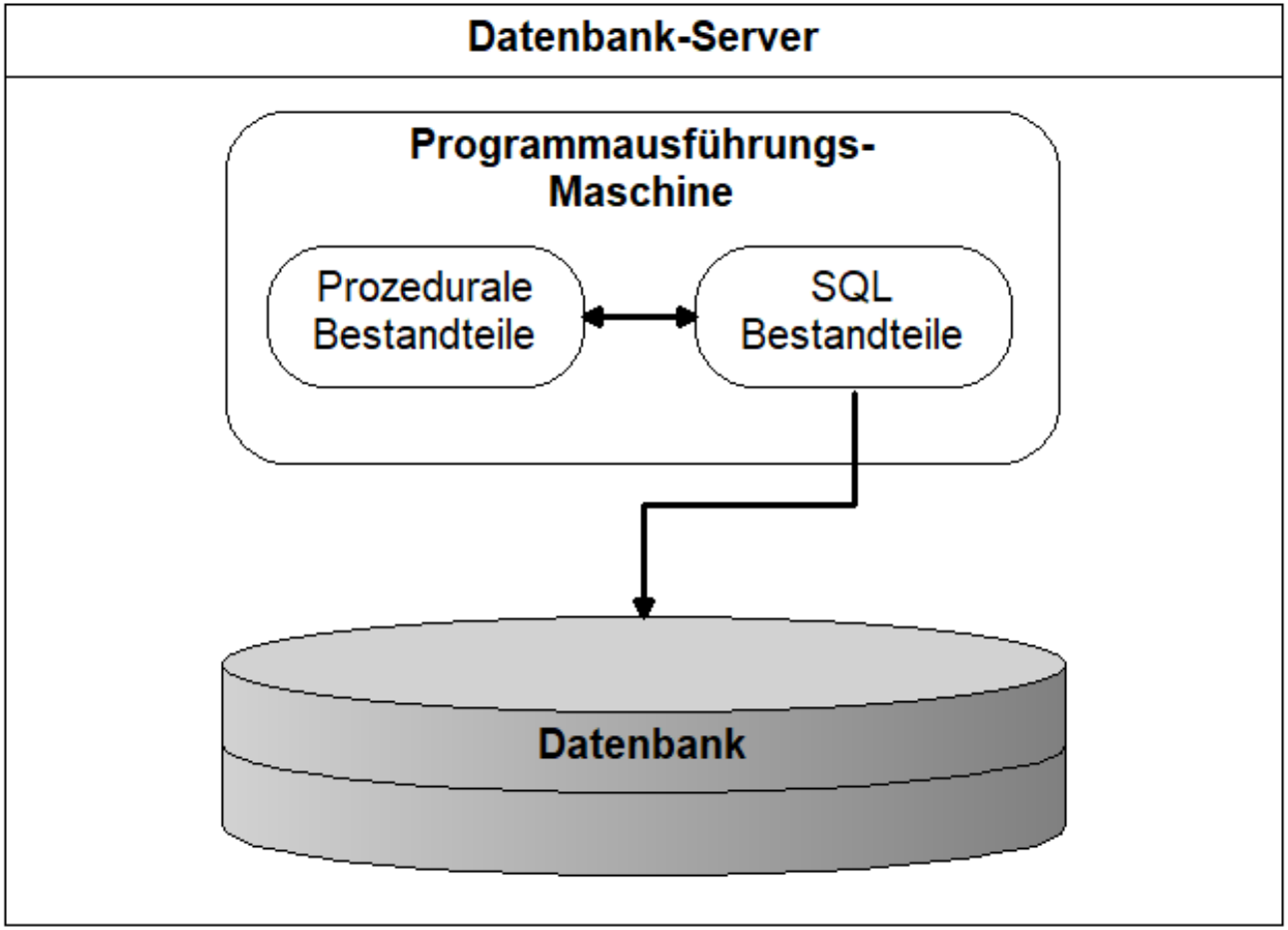
Einleitung

Elemente der Programmiersprache

Zugriff auf Datenbanktabellen

Anwendungsbeispiel

Architektur

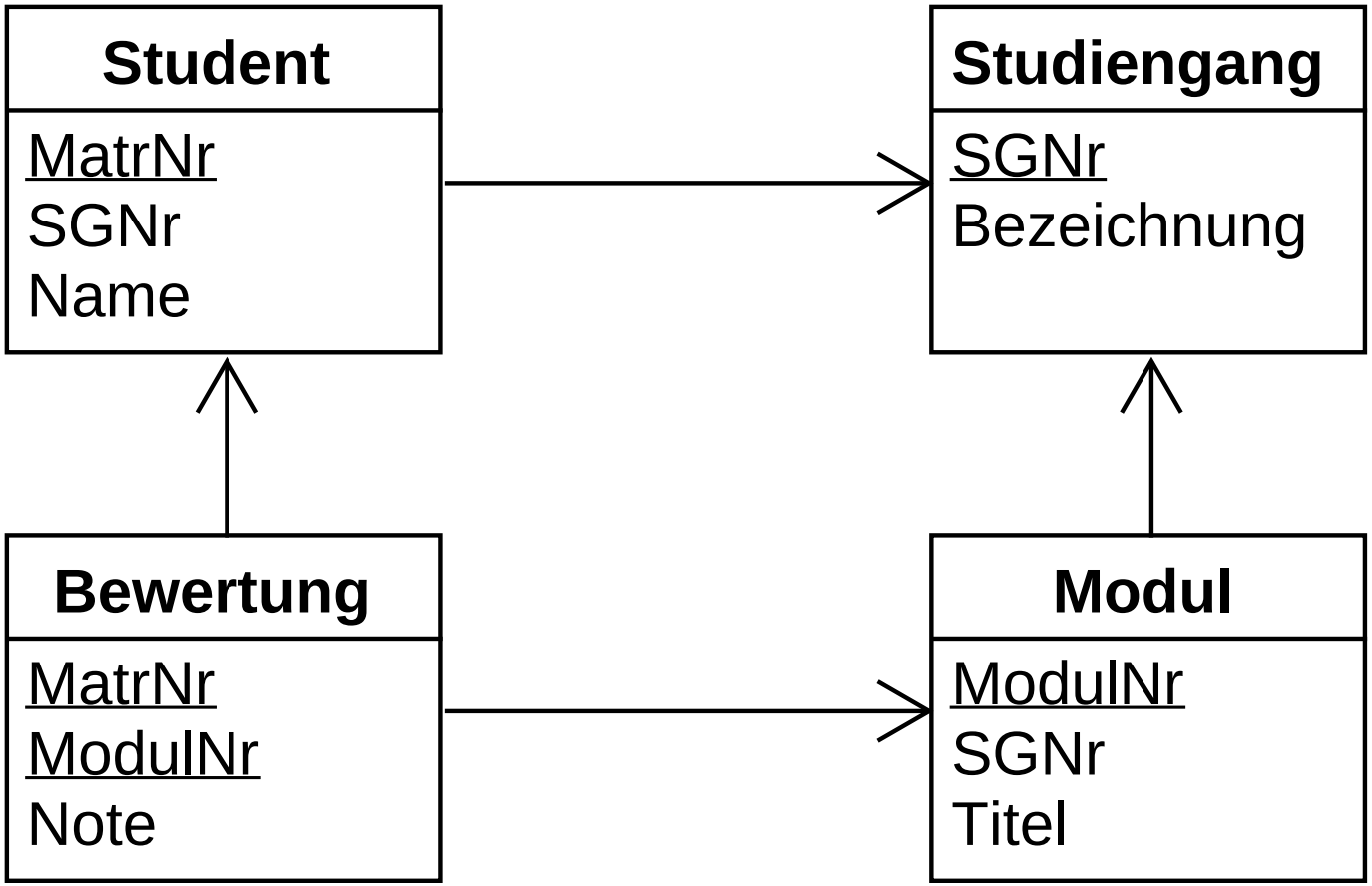


Eigenschaften

- Code läuft im Datenbank-Server
- Reduzierung der Netzwerk-Roundtrips durch Zusammenführung mehrerer SQL-Anweisungen in einem Code-Block
- Typsystem der Programmiersprache stimmt mit Typsystem der Datenbank überein
- Programmcode und Datenbankcode sind miteinander integriert, d.h. ein Namensraum für Programm- und Datenbankobjekte
  Konventionen: Präfixe für Namen

v_	Variablen
t_	Typen
p_	Parameter
cur_	Cursor

Datenmodell für die Beispiele



Beispielprogramm

- Wird an den Datenbankserver gesendet
- dort kompiliert
- und ausgeführt

```
declare
  v_matr_nr integer := 501011;
  v_name varchar(25) := 'Meier';
  v_grade decimal(2,1) := 1.3;
begin
  dbms_output.put_line(v_matr_nr);
  dbms_output.put_line(v_name);
  dbms_output.put_line(v_grade);
end;
```

Zuweisung vs Gleichheit

	Java	PL/SQL
Zuweisung	=	:=
Gleichheit	==	=

Fallunterscheidungen

```
declare
  v_x integer;
  v_y integer;
```

Varianten der Fallunterscheidung

```
begin
  if v_x = 0 then
    v_y := v_x + 10;
  end if;
end;
```

```
begin
  if v_x = 0 then
    v_y := v_x + 10;
  else
    v_y := v_x + 20;
  end if;
end;
```

```
begin
  if v_x = 0 then
    v_y := v_x + 10;
  elsif v_x < 0 then
    v_y := v_x - 20;
  else
    v_y := v_x + 20;
  end if;
end;
```

Schleifen

```
declare
  v_i integer := 1;
  v_x integer := 0;
```

Schleifenvarianten

```
begin
  while v_i <= 10 loop
    v_x := v_x + v_i;
    v_i := v_i + 1;
  end loop;
end;
```

```
begin
  for v_i in 1..10 loop
    v_x := v_x + v_i;
  end loop;
end;
```

```
begin
 loop
    v_x := v_x + v_i;
    v_i := v_i + 1;
    if v_i > 3 then
      exit;
    end if;
  end loop;
end;
```

Record Typen / Assoziierte Typen

- **Studiengang.Bezeichnung%type**
  Typ der Spalte Bezeichnung in Tabelle Studiengang
- **STUDENT%rowtype**
  Recordtyp eines gesamten Studentendatensatzes

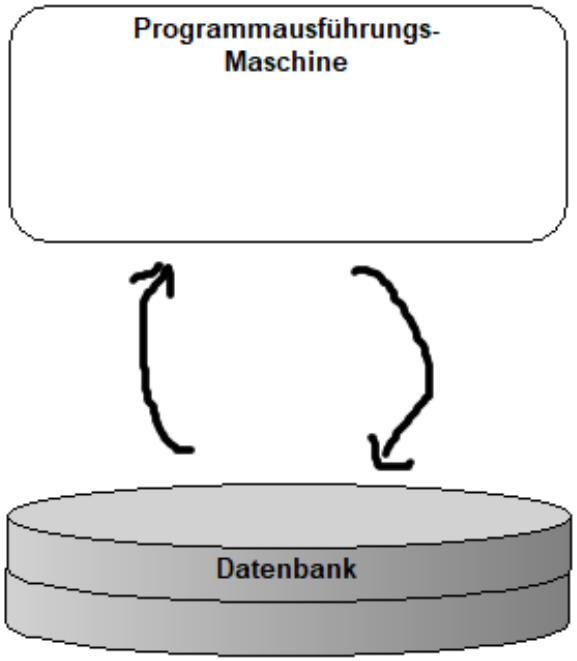
```
declare
  type t_sg is record (
    sgnr integer,
    bezeichnung Studiengang.Bezeichnung%type
  );
  v_sg t_sg;
  v_student STUDENT%rowtype;
begin
  v_sg.bezeichnung := 'WI';
  v_student.matrnr := 555123;
end;
```

Collection-Typen

- Array
  - Festgelegte Anzahl von Einträgen
  - ```
    type t_array is varray(3) of integer;
    v_array t_array := t_array(10, 20, 30);
    v_array(2) := 22;
    ```
- Table
  - Anzahl von Einträgen nicht festgelegt, kann Lücken haben
  - ```
    type t_table is table of integer;
    v_table t_table := t_table(10, 20, 30);
    v_table.delete(2);
    ```
- Associative Array (entspricht Map in Java)
  - Anzahl von Einträgen nicht festgelegt, erweiterbar
  - ```
    type t_map is table of integer index by varchar(10);
    v_map t_map := t_map;
    v_map('Berlin') := 3769143;
    ```
    ```

Zugriff auf Datenbanktabellen

- Die Daten der Datenbank müssen in die Programmausführungsmaschine transferiert werden
- Erst dort können sie im Programm verarbeitet werden



ein Datensatz	insert into
mehrere Datensätze	bulk collect
viele Datensätze	cursor

Insert Into

SQL-Anweisung darf nur eine Zeile liefern

Zugriff auf eine einzelne Spalte

```
declare
  v_matr_nr integer := 555123;
  v_name Student.Name%type;
begin
  select Name into v_name
  from Student
  where MatrNr = v_matr_nr;
end;
```

Zugriff auf alle Spalten mit einer Anweisung

```
declare
v_matr_nr integer := 555123;
v_student STUDENT%rowtype;
begin
  select * into v_student
  from Student
  where MatrNr = v_matr_nr;
end;
```

Bulk Collect (1)

```
declare
  type t_sg is table of STUDIENGANG%rowtype;
  v_sg_table t_sg;
begin
  select * bulk collect into v_sg_table
  from Studiengang;

  for v_i in 1 .. v_sg_table.count
  loop
    dbms_output.put_line(v_sg_table(v_i).bezeichnung);
  end loop;
end;
```

Bulk Collect (2)

```
declare
  type t_int_table is table of integer;
  v_deleted t_int_table;
begin
  delete from Student
  where SGNr = 4
  returning MatrNr bulk collect INTO v_deleted;

  for v_i in 1 .. v_deleted.count
  loop
    dbms_output.put_line(v_deleted(v_i));
  end loop;
end;
```

Cursor

```
declare
  v_matr_nr Student.MatrNr%type;
  v_name Student.Name%type;
  cursor cur_students is
    select MatrNr, Name from Student where SGNr = 1;
begin
  open cur_students;
  loop
    fetch cur_students into v_matr_nr, v_name;
    exit when cur_students%notfound;

    dbms_output.put_line(v_matr_nr || ':' || v_name);
  end loop;
  close cur_students;
end;
```

Cursor-Attribute

Attribut	Bedeutung
%FOUND	Datensatz wurde aus dem Coursor geholt
%NOTFOUND	Kein Datensatz mehr vorhanden
%ISOPEN	Cursor ist geöffnet
%ROWCOUNT	Anzahl bisher geholter Datensätze

Cursor mit Parameter

```
declare
  v_student Student%rowtype;
  cursor cur_students(p_sg_r Student.SGNr%type)
      return Student%rowtype is
    select * from Student where SGNr = p_sg_r;
begin
  open cur_students(1);
  loop
    fetch cur_students into v_student;
    exit when cur_students%notfound;

    dbms_output.put_line(
      v_student.MatrNr || ':' || v_student.Name
    );
  end loop;
  close cur_students;
end;
```

Füllen einer Datumstabelle

Ergänzen von Tagesdatensätzen

Tag	
TagNr	Datum
1	13.10.2020
2	14.10.2020

- Fallunterscheidung ob Tabelle leer oder nicht
- `sysdate` liefert das aktuelle Systemdatum
- Operator + addiert Tage zu einem Datum (z. B. `sysdate + 7`)

Implementierung (1)

```
declare
  v_anzahl_tage integer := 2;
  v_max_tag_nr integer;
  v_max_datum date;
  v_tag_nr integer;
  v_datum date;
  v_i integer;
begin
  -- weiter auf naechster Seite
```

Implementierung (2)

```
  select max(TagNr), max(Datum) into v_max_tag_nr, v_max_datum
  from Tag;

  if v_max_tag_nr is null then
    v_tag_nr := 1;
    v_datum := sysdate;
  else
    v_tag_nr := v_max_tag_nr + 1;
    v_datum := v_max_datum + 1;
  end if;
  v_i := 0;
  while v_i < v_anzahl_tage loop
    insert into Tag values (v_tag_nr + v_i, v_datum + v_i);
    v_i := v_i + 1;
  end loop;
end;
```