Datenbankprogramme – am Beispiel PL/SQL

Datenbanktechnologien

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

Hochschule für Technik und Wirtschaft Berlin

Gespeicherte Prozeduren / Funktionen

Pakete

Ausnahmebehandlung

Optimierung

Gespeicherte Prozeduren

```
create or replace procedure remove_student(p_matr_nr integer) as
begin
  delete from Student where MatrNr = p_matr_nr;
end remove_student;

declare
  v_matr_nr integer := 555123;
begin
  remove_student(v_matr_nr);
end;
```

Gespeicherte Funktionen

```
create or replace function get_grade(
  p_matr_nr integer, p_modul_nr integer) return decimal
as
  v_grade decimal(2,1);
begin
  select Note into v_grade
  from Bewertung
  where MatrNr = p_matr_nr and ModulNr = p_modul_nr;
  return(v_grade);
end get_grade;

declare
  v_matr_nr integer := 555123;
  v_modul_nr integer := 1;
  v_grade decimal(2,1);
begin
  v_grade := get_grade(v_matr_nr, v_modul_nr);
  dbms_output.put_line(v_grade);
end;
```

Default-Werte und benannte Parameter

```
create or replace function get_grade(
  p_matr_nr integer, p_modul_nr integer := 1) return decimal
as
  v_grade decimal(2,1);
begin
  ...
end get_grade;

declare
  v_matr_nr integer := 555123;
  v_modul_nr integer := 1;
  v_grade number(2,1);
begin
  v_grade := get_grade(p_matr_nr => v_matr_nr);
  v_grade := get_grade(
               p_modul_nr => v_modul_nr, p_matr_nr => v_matr_nr);
  v_grade := get_grade(v_matr_nr, p_modul_nr => v_modul_nr);
end;
```

Parameterdeklaration

	IN	OUT	INOUT
Funktion	Wertübergabe in Prozedur	Wertübergabe aus Prozedur	beides
Verhalten	wie eine Konstante	wie eine nicht initialisierte Variable	wie eine initialisierte Variable
Standardwert	möglich	nicht möglich	nicht möglich
Wertzuweisung in Prozedur	nicht möglich	erforderlich	möglich
Aufrufparameter	Initialisierte Variable, Konstante oder Ausdruck	muss Variable sein	muss Variable sein

Prozedur mit OUT -Parameter

```
create or replace procedure calc_avg(p_matr_nr in integer,
  p_avg_grade out number) as
begin
    select avg(Note) into p_avg_grade
    from Bewertung
    where MatrNr = p_matr_nr;
end calc_avg;

declare
  v_matr_nr integer := 555123;
  v_avg_grade number(2,1);
begin
  calc_avg(v_matr_nr, v_avg_grade);
  dbms_output.put_line(v_avg_grade);
end;
```

Pakete

- Pakete in PL/SQL entsprechen Schnittstellen in Java
- Definition von Prozedur- und Funktionssignaturen
- Nur Schnitstelle, noch kein Implementierungscode

```
create or replace package bewertung_service as

  procedure erstelle_bewertung(
    p_matr_nr integer, p_modul_nr integer, p_note integer);

  function finde_bewertung(
    p_matr_nr integer, p_modul_nr integer) return integer;

end bewertung_service;
```

Paketkörper

```
create or replace package body bewertung_service as
  procedure erstelle_bewertung(
    p_matr_nr integer, p_modul_nr integer, p_note integer)
  as
     ...
  begin
    ...
  end;

  function finde_bewertung(
    p_matr_nr integer, p_modul_nr integer) return integer
  as
    ...
  begin
    ...
  end;
end bewertung_service;
```

Fehlerbehandlung durch Ausnahmen

```
declare
  v_stud_sum integer := 973;
  v_course_sum integer := 0;
  v_course_avg integer;
begin
  -- Division durch 0
  v_course_avg := v_stud_sum / v_course_sum;
exception
  when zero_divide then
      dbms_output.put_line('zero_divide');
  when others then
      dbms_output.put_line('others');
end;
```

Vordefinierte Ausnahmen

Ausnahme	**Auslösung durch**
DUP_VAL_ON_INDEX	Einfügen eines Datensatzes mit einem Primärschlüssel der bereits existiert
NO_DATA_FOUND	Select-Anweisung liefert keine Daten
TOO_MANY_ROWS	Select-Anweisung, die nur eine Zeile liefern darf, liefert mehrere Zeilen
VALUE_ERROR	Fehler bei Datenkonversion, z. B. Umwandlung einer Zeichenkette, die Buchstaben enthält, in eine Zahl

Beispiel NO_DATA_FOUND

```
declare
  v_matr_nr integer := 999;
  v_name Student.Name%type;
begin
  select Name into v_name
  from Student
  where MatrNr = v_matr_nr;
exception
  when NO_DATA_FOUND then
    dbms_output.put_line('keine Daten vorhanden');
end;
```

Wiederauslösung behandelter Ausnahmen

```
declare
  v_matr_nr integer := 999;
  v_name Student.Name%type;
begin
  select Name into v_name
  from Student
  where MatrNr = v_matr_nr;
exception
  when NO_DATA_FOUND then
    dbms_output.put_line('keine Daten vorhanden');
    raise;
end;
```

Nicht behandelte Ausnahmen werden weitergeleitet

```
declare
  v_matr_nr integer := 999;
  v_name Student.Name%type;
begin
  select Name into v_name
  from Student
  where MatrNr = v_matr_nr;
end;
```

Selbstdefinierte Ausnahmen

```
declare
  exc_past_due exception;
  v_due_date date := sysdate - 1;
  v_todays_date date := sysdate;
begin
  if v_due_date < v_todays_date then
    raise exc_past_due;
  end if;
exception
  when exc_past_due then
    dbms_output.put_line('Datum ueberschritten');
end;
```

Selbstdefinierte Ausnahmen mit Fehlernummer verbinden

- Oracle liefert Fehlernummern an Aufrufer zurück
- z.B. an Java Programm
- Kann dort weiter verarbeitet werden

```
declare
  exc_past_due exception;
  pragma exception_init(exc_past_due, -20001);
  v_due_date date := sysdate - 1;
  v_todays_date date := sysdate;
begin
  if v_due_date < v_todays_date then
    raise exc_past_due;
  end if;
end;
```

Verwendung der Fehlernummer in Java

```
public void doSomething(Integer id) {
  try (CallableStatement cStmt =
          useConnection().prepareCall("{call doIt(?)}")) {
    cStmt.setInt(1, id);
  } catch (SQLException e) {
    if (e.getErrorCode() == 20001) {
      throw new PastDueException();
    } else {
      throw new DataException(e);
    }
  }
}
```

Bulk SQL

- Minimiert Kommunikations-Overhead zwischen PL/SQL and SQL
- Batch-Verarbeitung

```
delete from Modul where ModulNr=1;
delete from Modul where ModulNr=2;
delete from Modul where ModulNr=3;
```

versus

```
forall v_i in 1..3
  delete from Modul where ModulNr=v_i;
```

Tabelle für Bulk-Insert (siehe Folgefolie)

```
create table parts (
  pnum integer,
  pname varchar2(15)
);
```

Bulk-Insert

```
declare
  type numtab is table of parts.pnum%type index by integer;
  type nametab is table of parts.pname%type index by integer;
  pnums   numtab;
  pnames  nametab;
  iterations integer := 50000;
begin
  for j in 1..iterations loop  -- populate collections
    pnums(j) := j;
    pnames(j) := 'part no. ' || to_char(j);
  end loop;
```

Dauer: ca. 2,23 Sekunden

```
for i in 1..iterations loop
  insert into parts (pnum, pname)
  values (pnums(i), pnames(i));
end loop;
```

Dauer: ca. 0,06 Sekunden

```
forall i in 1..iterations
  insert into parts (pnum, pname)
  values (pnums(i), pnames(i));
```