Spezielles zu Funktionen, Prozeduren und Paketen

Exceptions standardisieren

Erzeugen eines Package, welches alle benannten und selbstdefinierten Exceptions der Applikation beinhaltet.

```
CREATE OR REPLACE PACKAGE error_pkg IS
  e_fk_err            EXCEPTION;
  e_seq_nbr_err EXCEPTION;
  PRAGMA EXCEPTION_INIT (e_fk_err, -2292);
  PRAGMA EXCEPTION_INIT (e_seq_nbr_err, -2277);
  ...
END error_pkg;
/
```

Fehlerbehandlung standardisieren

Einheitliche Funktionen und Prozeduren zur Behandlung und Anzeige von Fehlern:

- Anzeige von Fehlern auf Basis von `SQLCODE` und `SQLERRM`
- Parametrisierung des Codes zur Verfolgung von Laufzeitfehlern:
  - Prozedur, in der der Fehler auftritt
  - Lokalisierung (Zeilennummer) der Fehlerstelle

Lokale Unterprogramme

Prozeduren und/oder Funktionen, die am Ende der deklarativen Sektion eines Blocks definiert werden.

```
CREATE PROCEDURE employee_sal(p_id NUMBER) IS
   v_emp employees%ROWTYPE;
   FUNCTION tax(p_salary VARCHAR2) RETURN NUMBER IS
   BEGIN
     RETURN p_salary * 0.825;
   END tax;
BEGIN
   SELECT * INTO v_emp
   FROM EMPLOYEES WHERE employee_id = p_id;
   DBMS_OUTPUT.PUT_LINE('Tax: '|| tax(v_emp.salary));
END;
/
EXECUTE employee_sal(100)
```

Lokale Unterprogramme

Prozeduren und/oder Funktionen, die am Ende der deklarativen Sektion eines Blocks definiert werden.

```
CREATE PROCEDURE employee_sal(p_id NUMBER) IS
   v_emp employees%ROWTYPE;
   FUNCTION tax(p_salary VARCHAR2) RETURN NUMBER IS
   BEGIN
     RETURN p_salary * 0.825;
   END tax;
BEGIN
   SELECT * INTO v_emp
   FROM EMPLOYEES WHERE employee_id = p_id;
   DBMS_OUTPUT.PUT_LINE('Tax: '|| tax(v_emp.salary));
END;
/
EXECUTE employee_sal(100)
```

Welche Berechtigungen gelten bei Ausführung von Unterprogrammen

Berechtigung des Erstellers (Definer's rights):

- Vor Oracle8*i* benutzt
- Programme laufen unter der Kennung des Erstellers.
- Benutzer benötigen keine Privilegien bzgl. der durch die Programme verwendeten DB-Objekte, nur das Recht zur Ausführung des Programms.
- Ist der Standard (default) bis heute

Berechtigung des Aufrufers (Invoker's rights):

- Eingeführt in Oracle8*i*
- Programme laufen mit den Privilegien des aufrufenden Nutzers.
- Der Aufrufer benötigt entsprechende Berechtigungen bzgl. der DB-Objekte, die das Programm verwendet.

Spezifikation der Privilegien des Aufrufers

```
CREATE OR REPLACE PROCEDURE add_dept(
  p_id NUMBER, p_name VARCHAR2) AUTHID CURRENT_USER IS
BEGIN
  INSERT INTO departments
  VALUES (p_id, p_name, NULL, NULL);
END;
```

Kann bei Packages als auch Standalone Funktionen/Prozeduren verwendet werden. Bei Verwendung gilt:

- In Abfragen, DML-Anweisungen und dynamischen SQL verwendete Namen werden im Schema des Aufrufers aufgelöst.
- Aufrufe an andere Pakete, Funktionen oder Prozeduren werden im Schema des Erstellers aufgelöst.

Autonome Transaktionen

- Eine neue unabhängige Transaktion kann innerhalb einer laufenden Transaktion gestartet werden.
- Wird spezifiziert durch
  `PRAGMA AUTONOMOUS_TRANSACTION`

```
PROCEDURE proc1 IS
  emp_id  NUMBER;
BEGIN
  emp_id := 1234;
  COMMIT;
  INSERT ...
  proc2;
  DELETE ...
  COMMIT;
END proc1;
```

```
PROCEDURE proc2 IS
 PRAGMA
  AUTONOMOUS_TRANSACTION;
  dept_id  NUMBER := 90;
 BEGIN
   UPDATE ...
   INSERT ...
   COMMIT;   -- Required
 END proc2;
```

Merkmale autonomer Transaktionen

- Unabhängig von der laufenden Transaktion
- Die Transaktion des Aufrufers wird angehalten bis die autonome Transaktion bendet ist.
- Es erfolgt kein Rollback bei Rollback der Transaktion des Aufrufers.
- Werden gestartet und beendet durch Unterprogramme, nicht durch anonyme oder geschachtelte Blöcke.

Beispiel für autonome Transaktion

```
PROCEDURE bank_trans(p_cardnbr NUMBER, p_loc NUMBER) IS
BEGIN
  log_usage(p_cardnbr, p_loc);
  INSERT INTO txn VALUES (9001, 1000,...);
END bank_trans;
```

```
PROCEDURE log_usage (p_card_id NUMBER, p_loc NUMBER)
IS
  PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
  INSERT INTO usage    -- usage is an existing table
  VALUES (p_card_id, p_loc);
  COMMIT;
END log_usage;
```

Parameterrückgabe per Referenz

- `OUT` und `IN OUT` Parameter können per Referenz statt per Wert zurückgegeben werden.

```
DECLARE
  TYPE           emptabtype IS TABLE OF employees
   %ROWTYPE;
  rec_emp  emptabtype;
  PROCEDURE populate(p_tab IN OUT NOCOPY emptabtype)IS
    BEGIN
    . . .
    END;
BEGIN
  populate(rec_emp);
END;
```

Besonderheiten bei Verwendung des Hinweises NOCOPY

- Falls das Unterprogramm durch nicht gefangene Exception beendet wird:
  - Die Werte der aktuell übergebenen Parameter sind unsicher
  - Unvollständige Veränderungen werden nicht rückgängig gemacht (kein Rollback)

Der Compiler kann den NOCOPY Hinweis ignorieren

Der Hinweis wird ignoriert, falls

- der aktuelle Parameter:
  - ein Element eines index-by Table ist
  - eingeschränkt ist (z.B. bei Scale oder `NOT NULL`)
  - und die formalen Parameter sind Records, von denen mindestens einer via `%ROWTYPE` oder `%TYPE` definiert wurde, und die Constraints in den Feldern der Records differieren
  - erfordert eine implizite Typkonversion
- Das Unterprogramm ist Teil eines externen oder Remote Prozedur-Aufrufs

Nutzen des Cross-Session PL/SQL Function Result Cache

- Wenn eine PL/SQL Funktion, für die der Result Cache aktiviert ist, mit verschiedenen Parametern aufgerufen wird, werden die Parameter und das Ergebnis im Cache gespeichert.
- Der Cache ist Teil der Shared Global Area (SGA) und somit verfügbar für jede Session.
- Spätere Aufrufe mit der gleichen Parameterbelegung nutzen das Ergebnis aus dem Cache.
- Dieser Performanzvorteil empfiehlt sich bei Funktionen, die häufig aufgerufen werden und von Daten abhängen, die sich eher selten ändern.

Result-Caching für eine Funktion aktivieren

```
CREATE OR REPLACE FUNCTION emp_hire_date (p_emp_id
    NUMBER) RETURN VARCHAR
 RESULT_CACHE RELIES_ON (employees) IS
  v_date_hired DATE;
BEGIN
  SELECT hire_date INTO v_date_hired
  FROM HR.Employees
  WHERE Employee_ID = p_emp_ID;
  RETURN to_char(v_date_hired);
END;
```

Deterministische Funktionen

- Mit Hilfe der `DETERMINISTIC-Klausel` kann angezeigt werden, dass die Funktion das gleiche Ergebnis für gleiche Parameter liefert.
- Dadurch kann der Optimierer redundante Funktionsaufrufe vermeiden.
- Sollte nicht bei Funktionen verwendet werden, deren Ergebnisse vom Zustand der Session oder von Schema-Objekten abhängen.