Procedure Syntax

Create [or replace] procedure Procname
[(argument [IN|OUT | INOUT] datatype,....)]
AS
Variable declarations
  Begin
    statements
  End;

1
Function Syntax

Create [or replace] function funcname
[(argument [IN|OUT | INOUT] datatype,....)]
Return datatype IS variable length

Begin
  statements
  
  return(variable )
End;
Creating Database Triggers

• Code is similar to all PL/SQL program unit blocks with some additional syntax
• Database triggers cannot accept parameters (an important distinction from procedures)
• Trigger cannot use Commit, Rollback or Savepoint
Defining Triggers

• To define a trigger, you must specify:
  – Statement type that causes trigger to fire (Cause)
    • INSERT, UPDATE, DELETE
  – Timing
    • BEFORE or AFTER (the SQL statement)
    • E.g. when grade field is updated in a table or after the GPA is calculated
  – Level
    • STATEMENT or ROW
Trigger Timing

• **BEFORE:** trigger fires before statement executes
  – Example: for audit trail, records grade value before it is updated

• **AFTER:** trigger fires after statement executes
  – Example: update QOH (quantity on hand) after item is sold
Trigger Levels

• ROW: trigger fires once for each row that is affected
  – Example: when adding multiple order lines, update multiple inventory QOH values

• STATEMENT: trigger fires once, regardless of how many rows are updated
  – Example: for audit trail, you just want to record that someone updated a table, but you don’t care how many rows were updated
Creating a Trigger in SQL*Plus

CREATE OR REPLACE TRIGGER \textit{trigger\_name} \\
[BEFORE|AFTER] \ [INSERT|UPDATE|DELETE] \ ON \\
\textit{table\_name} \\
\hspace{1em} [FOR EACH ROW| statement] \ [WHEN \hspace{0.5em} (condition)] \\
BEGIN \\
\hspace{1em} \textit{trigger\_body} \\
END;
How Trigger Works

• Triggers stored in namespace where object identifiers for users’ objects
• Timing is specified – Before or After Event
• Values of before and after firing can be specified (old grade and new grade)
• When is optional;
• Trigger body is PL/SQL code block or program
• Audit table is created to save the values generated after firing the trigger
Trigger Example in Oracle 7

• This trigger initializes a counter variable before the execution of an INSERT statement that add tuples to student relation
• CREATE TRIGGER init_count BEFORE INSERT ON students
  DECLARE
    count INTEGER
  BEGIN
    count := 0
  END
• this trigger is executed just once per insert statement, regardless of the number of records inserted. (No FOR EACH ROW phrase)
• statement level trigger.
Trigger Example in Oracle 7

• This trigger increments the counter for each inserted tuple that satisfies the condition age < 18
• CREATE TRIGGER incr_count AFTER INSERT ON students
  WHEN (new.Age < 18)
  FOR EACH ROW
  BEGIN
    count := count + 1;
  END
• Row-level-trigger (FOR EACH ROW phrase)
• new is used to refer newly inserted tuple.
Trigger Example

• Row level before update trigger

• invoked when the new value of the Amount column is more than 10% greater than its old value

• create trigger ledger_bef_upd_row
  before update on LEDGER
  for each row
  when(new.Amount/old.Amount > 1.1)
  begin
  insert into LEDGER AUDIT
  values(:old.Action, :old.Item, :old.Amount, old.Person);
  end;