



Why Stored Procedures?

- So far, all data processing is done at the client
 - Lots of data may have to be transferred
 - Functionality (code) replicated at each client
 - Lots of state (e.g., locks, transaction data) at the DBMS
 While client processes the data
- Stored procedures execute in same process space as DBMS
 - Encapsulates application logic and is close to the data
 - ${\scriptstyle \succ}\,$ Reuse of common functionality by different clients
- Vendors introduced their own procedural extensions
 e.g., Oracle's PL/SQL

SQL/PSM

SQL Persistent Stored Modules

- > SQL standard for stored procedures, available in SQL:2003
- Commercial vendors may offer own extensions of PSM

Standard language for stored procedures

- Supports both procedures and functions
- > Functions can return results through RETURN statement
- > Procedures can return results in parameters
- In this course we focus on Oracle PL/SQL

PL/SQL

PL/SQL (Procedural Language SQL)

Procedural extension to SQL developed by Oracle

- Most prominent DBMS procedural language
- Another language is T-SQL from Microsoft (MS SQL)
- Only DML allowed in PL/SQL
- > DDL such as creating or dropping tables NOT allowed

Basic program structure is a block

- There can be nested blocks
- > PL/SQL syntax is not case sensitive (variable names as well)

PL/SQL Program Structure

DECLARE variable_declarations BEGIN procedural_code EXCEPTION error_handling END;

PL/SQL in SQL Plus

- Ensure output goes to screen SET SERVEROUTPUT ON
- Executing PL/SQL in command line BEGIN

DBMS_OUTPUT.PUT_LINE('Hello World'); END;

1

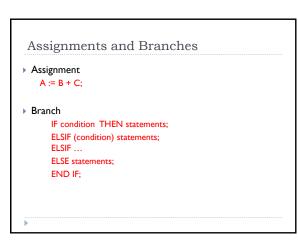
The / must be by itself on separate line

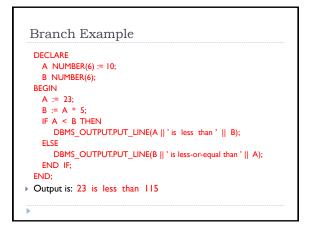
 DBMS_OUTPUT.PUT_LINE equivalent of printf() in C or System.out.println() in Java

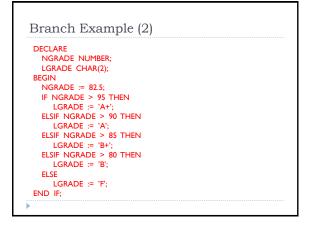
Data Types

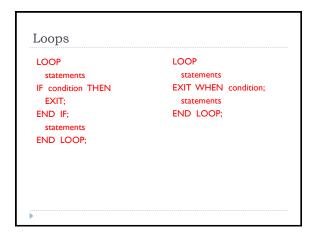
- It is possible to use ORACLE SQL types NUMBER, VARCHAR, etc
- PL/SQL allows directly referring to a column type tablename.columnname%TYPE
- e.g, SAILORS.SNAME%TYPE
- Also possible to define a row type (e.g., tuple) tablename%ROWTYPE
- Declaring a variable: <var_name> <TYPE>; sailor_rec SAILORS%ROWTYPE;
- Can later refer to individual fields using column names DBMS_OUTPUT.PUT_LINE('Name: ' || sailor_rec.name || 'Age:' || sailor_rec.age);

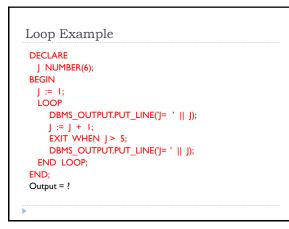
|| means string concatenation (like + in Java)

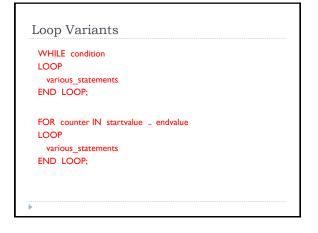


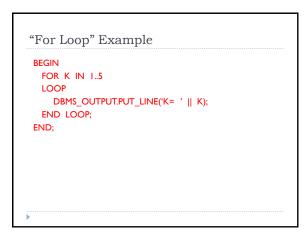


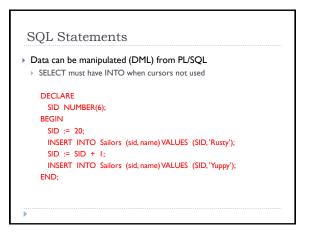












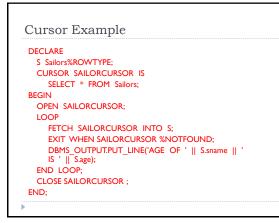
SQL Statements - retrieving data

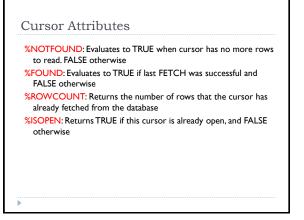
- As before, there are two cases
- Single-tuple result (the "easy" case) SELECT selectfields INTO declared_variables FROM table_list WHERE search_criteria;

DECLARE VAR_NAME Sailors.name%TYPE; VAR_AGE Sailors.age%TYPE; BEGIN SELECT name, age INTO VAR_NAME, VAR_AGE FROM Sailors WHERE SID = 10;

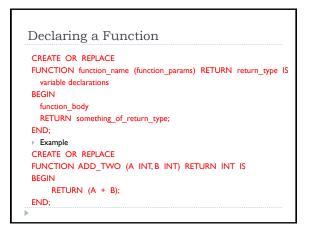
DBMS_OUTPUT.PUT_LINE('Age of ' || VAR_NAME || ' is ' || VAR_AGE); END;

SQL Statements – retrieving data Multiple-tuples result: *cursors* are needed CURSOR cursorname IS SELECT_statement; OPEN cursorname; FETCH cursorname INTO variable_list; CLOSE cursorname;





Declaring a Procedure CREATE OR REPLACE PROCEDURE procedure_name (parameters) IS variable declarations BEGIN procedure_body END; • Parameters can be IN, OUT or INOUT, default is IN CREATE OR REPLACE PROCEDURE SUM_AB (A INT, B INT, C OUT INT) IS BEGIN C := A + B; END;



Exceptions

- Exceptions defined per block (similar to Java)
 - Each BEGIN...END has its own exception handling
 - If blocks are nested, exceptions are handled in an "inside to outside" fashion
 - If no block in the nesting handles the exception, a runtime error occurs
- There are multiple types of exceptions
 - Named system exceptions (most frequent) we only cover these
 - Unnamed system exceptions
 - User-defined exceptions

ZERO_DIVIDE

Exceptions

DECLARE

BEGIN

EXCEPTION WHEN ex_name I THEN error handling statements WHEN ex_name2 THEN error handling statements

WHEN Others THEN

error handling statements

END;

Exception Name	Reason	Error Numbe
CURSOR_ALREADY_OPEN	When you open a cursor that is already open.	ORA-06511
INVALID_CURSOR	When you perform an invalid operation on a cursor like closing a cursor or fetch data from a cursor that is not opened.	ORA-01001
NO_DATA_FOUND	When a SELECTINTO clause does not return any row from a table.	ORA-01403
TOO_MANY_ROWS	When you SELECT or fetch more than one row into a record or variable.	ORA-01422

When you attempt to divide a number by zero.

ORA-01476