**PL/SQL Packages**

A package is a schema object that groups logically related PL/SQL types, variables, and subprograms.

Packages usually have two parts, a specification (spec) and a body sometimes the body is unnecessary.

The specification is the interface to the package. It declares the types, variables, constants, exceptions, cursors, and subprograms that can be referenced from outside the package. The body defines the queries for the cursors and the code for the subprograms.

A package always has a **specification**, which declares the **public items** that can be referenced from outside the package. If the public items include cursors or subprograms, then the package must also have a **body**

You can think of the spec as an interface and of the body as a *black box*. You can debug, enhance, or replace a package body without changing the package spec.

The **AUTHID clause** of the package specification determines whether the subprograms and cursors in the package run with the privileges of their definer (the default) or invoker, and whether their unqualified references to schema objects are resolved in the schema of the definer or invoker.

The **ACCESSIBLE BY clause** of the package specification lets you specify a white list of

PL/SQL units that can access the package. You use this clause in situations like these:

• You implement a PL/SQL application as several packages—one package that provides the application programming interface (API) and helper packages to do the work. You want clients to have access to the API, but not to the helper packages. Therefore, you omit the ACCESSIBLE BY clause from the API package specification and include it in each helper package specification, where you specify that only the API package can access the helper package.

• You create a utility package to provide services to some, but not all, PL/SQL units in the same schema. To restrict use of the package to the intended units, you list them in the ACCESSIBLE BY clause in the package specification.

**Reasons to Use Packages**

Packages support the development and maintenance of reliable, reusable code with the following features:

• **Modularity**

Packages let you encapsulate logically related types, variables, constants, subprograms, cursors, and exceptions in named PL/SQL modules. You can make each package easy to understand, and make the interfaces between packages simple, clear, and well defined. This practice aids application development.

• **Easier Application Design**

When designing an application, all you need initially is the interface information in the package specifications. You can code and compile specifications without their bodies. Next, you can compile standalone subprograms that reference the packages. You need not fully define the package bodies until you are ready to complete the application.

• **Hidden Implementation Details**

Packages let you share your interface information in the package specification, and hide the implementation details in the package body. Hiding the implementation details in the body has these advantages:

– You can change the implementation details without affecting the application interface.

– Application users cannot develop code that depends on implementation details that you might want to change.

• **Added Functionality**

Package public variables and cursors can persist for the life of a session. They can be shared by all subprograms that run in the environment. They let you maintain data across transactions without storing it in the database.

• **Better Performance**

The first time you invoke a package subprogram, Oracle Database loads the whole package into memory. Subsequent invocations of other subprograms in same the package require no disk I/O.

Packages prevent cascading dependencies and unnecessary recompiling. For example, if you change the body of a package function, Oracle Database does not recompile other subprograms that invoke the function, because these subprograms depend only on the parameters and return value that are declared in the specification.

• **Easier to Grant Roles**

You can grant roles on the package, instead of granting roles on each object in the package.

**What Goes In a PL/SQL Package?**

The following is contained in a PL/SQL package:

■ Get and Set methods for the package variables, if you want to avoid letting other procedures read and write them directly.

■ Cursor declarations with the text of SQL queries. Reusing exactly the same query text in multiple locations is faster than retyping the same query each time with slight differences. It is also easier to maintain if you need to change a query that is used in many places.

■ Declarations for exceptions. Typically, you need to be able to reference these from different procedures, so that you can handle exceptions within called subprograms.

■ Declarations for procedures and functions that call each other. You do not need to worry about compilation order for packaged procedures and functions, making them more convenient than standalone stored procedures and functions when they call back and forth to each other.

■ Declarations for overloaded procedures and functions. You can create multiple variations of a procedure or function, using the same names but different sets of parameters.

■ Variables that you want to remain available between procedure calls in the same session. You can treat variables in a package like global variables.

■ Type declarations for PL/SQL collection types. To pass a collection as a parameter between stored procedures or functions, you must declare the type in a package so that both the calling and called subprogram can refer to it.