In various programming languages, a pragma is generally a line of source code prescribing an action you want the compiler to take. It’s like an option that you give the compiler; it can result in different runtime behaviour for the program, but it doesn’t get translated directly into bytecode.

PL/SQL has a PRAGMA keyword with the following syntax:

PRAGMA <instruction\_to\_compiler>;

The PL/SQL compiler will accept such directives anywhere in the declaration section, but most of them have certain additional requirements regarding placement.

PL/SQL offers several pragmas:

AUTONOMOUS\_TRANSACTION

Tells the PL/SQL runtime engine to commit or roll back any changes made to the database inside the current block without affecting the main or outer transaction. See Chapter 14 for more information.

EXCEPTION\_INIT

Tells the compiler to associate a particular error number with an identifier you have declared as an exception in your program. Must follow the declaration of the exception.

RESTRICT\_REFERENCES

Tells the compiler the purity level (freedom from side effects) of a packaged program.

SERIALLY\_REUSABLE

Tells the PL/SQL runtime engine that package-level data should not persist between references to that data.

The following block demonstrates the use of the EXCEPTION\_INIT pragma to name a built-in exception that would otherwise have only a number:

DECLARE

no\_such\_sequence EXCEPTION;

PRAGMA EXCEPTION\_INIT (no\_such\_sequence, −2289);

BEGIN

...

EXCEPTION

WHEN no\_such\_sequence

THEN

q$error\_manager.raise\_error ('Sequence not defined');

END;

When you do not explicitly specify an ELSE clause of your own, PL/SQL implicitly uses the following:

ELSE

RAISE CASE\_NOT\_FOUND;