


I'm not robot  reCAPTCHA

Continue

Fe gun script v3rmillion

© 1996-2014, Amazon.com, Inc. or its affiliates Learn Pl Sql In 21 Days Ebook Download > last edited 157 weeks ago by Carrxeni Click on a thumbnail to go to Google Books. Load... 11None1,346,232 (2)NoneSams Teach Yourself PL/SQL in 21 Days, Second Edition, enables you to quickly create your own Oracle solutions with PL/SQL. Completely revised to cover Oracles 8i, the book provides guidance and direction, guides you through a progression of topics that begin with the basic building blocks of PL/SQL, and ending with in-depth discussions on the more widely used advanced features of Oracle's database programming environment. New topics include extensive dynamic SQL within PL/SQL, Dynamic SQL within PL/SQL, use of invoker's rights, autonomous transactions, interfacing PL/SQL with Java, PL/SQL Bulk Binds, parameter passing by reference and advanced Querying.... (more) Load... Sign up for LibraryThing to find out if you like this book. No current Talk conversations about this book. Original title Alternative titles Original Publication Date People/Characters Important Places Important Events Related Films Awards and Honors Epigraph Dedication First Words Quotes Last Words Disambiguation Message Publisher's Editors Blurbers Original language Canonical DDC/MDS References to this work on external sources. Wikipedia in EnglishNoneSams Teach Yourself PL/SQL in 21 Days, Second Edition, allows you to quickly create your own Oracle solutions with PL/SQL. Completely revised to cover Oracles 8i, the book provides guidance and direction, guides you through a progression of topics that begin with the basic building blocks of PL/SQL, and ending with in-depth discussions on the more widely used advanced features of Oracle's database programming environment. New topics include extensive dynamic SQL within PL/SQL, Dynamic SQL within PL/SQL, use of invoker's rights, autonomous transactions, interfacing PL/SQL with Java, PL/SQL Bulk Binds, parameter passing by reference and advanced Querying.No library descriptions found. Average: (2) Become a LibraryThing Author. 672317982 Congratulations on your decision to read this book, Sams Teach Yourself PL/SQL in 21 Days, Second Edition! If you're new to the Oracle environment, this book helps you quickly learn and master Oracle's built-in procedural language. Knowledge of PL/SQL (Procedural/Structured Query Language) becomes a fundamental necessity, no matter which of Oracle's many products you use. Today, on your first day of this PL/SQL tutorial, you will accomplish these tasks: Learn what PL/SQL is and why you should master Learning how PL/SQL relates to other Oracle Products Email Article Print Article Learn what resources you need to this book Make Write your first PL/SQL feature Over the remaining 20 days, you will delve deeper into the power and possibilities of this language and how to harness its power in your applications, whether you're programming client/server with Oracle's tools (such as Developer/2000), using other front-end tools (such as PowerBuilder), or simply writing a number of batch tasks that run on the server. What is PL/SQL? PL/SQL is a procedural language that Oracle has developed as an extension to standard SQL to provide a way to perform procedural logic in the database. New term - If you've worked with relational databases in the past, you're no doubt familiar with SQL, which stands for Structured Query Language. SQL itself is a powerful declarative language. It is declarative in the sense that you have the results you want to describe, but not how they are obtained. This is good because you are isolating an application from the specifics of how the data is physically stored. A competent SQL programmer can also push a lot of processing work back to server level through creative use of SQL. However, there are limits to what you achieve with a single declarative query. The real world is rarely as neat and clean as we would like. Developers often find that they need to run multiple queries in a row and process the specific results of one query before moving to the next. This leads to two problems in a client/server environment: the procedural logic, i.e. the definition of the process, is located on client machines. The need to view and use the data from one query as the basis for the next query results in a greater amount of network traffic. Why are these problems? The procedural logic on client machines can quickly be out of step when the software is upgraded. It can also be implemented incorrectly, resulting in a loss of database integrity. The need to draw large amounts of intermediate data to a client results in a long wait for end users who have to sit there staring at the hourglass while the data is transferred to their machines. The cumulative effects of a number of clients pulling large amounts of data across the network further reduce performance. PL/SQL provides a developer mechanism to add a server-level procedural component. It has been improved to the point where developers now have access to all the features of a full-featured server-level procedural language. It also forms the basis for programming in Oracle's constantly evolving set of client/server development tools, especially Developer/2000. Why learn PL/SQL? If you develop with Oracle products, Developer/2000 for example, the answer to this question is simple. You need to know PL/SQL because these products use PL/SQL for each procedural code. But what if you don't develop Oracle products? What if all you use is Oracle's database engine? Is PL/SQL of any use to you? Yes! Absolutely it is. Regardless of the front-end tool you use Using, you use PL/SQL to perform processing on the server instead of the client. Use PL/SQL to encapsulate business rules and other complicated logic. It ensures modularity and abstraction. You use it in database triggers to encode complex limitations that enforce the integrity of the database; logging changes; and to replicate data. PL/SQL can also be used with stored procedures and features to provide enhanced database security. Finally, it offers you a level of platform independence. Oracle is deployed on many hardware platforms, but PL/SQL is the same on all hardware platforms. It doesn't matter if you run Personal Oracle on a laptop or Oracle8i Enterprise on UNIX. No matter what development tools you use, if you develop in an Oracle environment, your knowledge of PL/SQL, and your ability to apply them, it gives you a competitive advantage over those who don't. With PL/SQL, you have the power to make your applications more robust, efficient and secure. SQL, SQL*Plus, PL/SQL: What's the difference? This question has bedeviled many people who are new to Oracle. There are several products with the letters SQL in the title, and these three, SQL*Plus, SQL, and PL/SQL, are often used together. This makes it easy to get confused about what product does the work and where the work is done. This section briefly describes each of these three products. SQL SQL stands for Structured Query Language. This has become the lingua franca of database access languages. It has been adopted by the International Standards Organization (ISO) and has also been adopted by the American National Standards Institute (ANSI). When you encode instructions such as SELECT, INSERT, UPDATE, and DELETE, SQL is the language you use. It is a declarative language and is always performed on the database server. Often you will find yourself encrypting SQL statements in a development tool, such as PowerBuilder or Visual Basic, but at runtime those statements are sent to the server for execution. PL/SQL PL/SQL is Oracle's Procedural Language Extension for SQL. It also usually runs on the database server, but some Oracle products such as Developer/2000 also include a PL/SQL engine located on the client. So you run your PL/SQL code on the client or server, depending on which one is more suitable for the task at hand. Unlike SQL, PL/SQL is procedural, not declarative. This means that your code indicates exactly how things are done. However, as in SQL, you need a way to send your PL/SQL code to the server for execution. PL/SQL also allows you to embed SQL statements in the procedural code. This close relationship between PL/SQL, SQL and SQL*Plus is the cause of a part of the confusion between the products. SQL*Plus SQL*Plus is an interactive program that allows you to sql instructions. It also allows you to type in PL/SQL code and send it to the server to run. SQL*Plus is one of the most common front ends used to develop and create saved PL/SQL procedures and functions. What happens if you run SQL*Plus and type in a SQL statement? Where is the processing taking place? What exactly does SQL*Plus do and what does the database do? If you're in a Windows environment and you have a database server somewhere on the network, the following things happen: SQL*Plus sends your SQL query over the network to the database server. SQL*Plus is waiting for a response from the database server. The database server runs the query and sends the results back to SQL*Plus. SQL*Plus displays the query results on your computer screen. Even if you're not active in a Windows environment with a network, the same things happen. The only difference may be that the database server and SQL*Plus run on the same physical machine. This would be the case, for example, if you run Personal Oracle on one PC. PL/SQL is performed in much the same way. Type a PL/SQL block in SQL*Plus and it is sent to the database server for execution. If there are SQL instructions in the PL/SQL code, they will be sent to the server's SQL engine for execution and the results will be returned to the PL/SQL program. Most importantly, SQL*Plus does not run your SQL queries. SQL*Plus also does not run your PL/SQL code. SQL*Plus simply acts as your window in the Oracle database, where the real action takes place. Figure 1.1 illustrates this relationship. Relationship of SQL*Plus, PL/SQL and Oracle. In addition to SQL*Plus, several other tools can serve as a window to the database. Server Manager, which has an interface similar to SQL*Plus, is such a tool, although Oracle plans to stop supporting it sometime in the future. If you have Oracle Enterprise Manager installed, you'll need to view sqlplus worksheet. SQLPlus Worksheet is a GUI tool that is fully compatible with SQL*Plus, but is much more user-friendly. If you're a Developer 2000 programmer, you'll have access to Oracle's Procedure Builder, a tool designed to develop and debug PL/SQL code. Later in this chapter, you can read more about SQLPlus Worksheet and Procedure Builder. SQL*Plus is used for most examples in this book because of its universal availability to developers. It is perhaps still the most widely used tool to develop, test and create PL/SQL stored subprograms and SQL queries. Note - In addition to Oracle's tools, several third-party vendors also have tools that can be used to develop PL/SQL code. Some of the most important products in this space are what you need to finish this book To try the examples and complete the exercises in this book, you need access to an Oracle8i database (the Personal Personal will work) SQL * Plus or SQLPlus worksheet Note - Where possible, the exercises and examples in this book are designed to run equally well under both Oracle8 and Oracle8i. Many, especially those in the first nine days, will even work under Oracle7. However, Oracle8i includes many new features that are not available in previous releases. Especially the days 10, 11, 12, 20 and 21 are strongly focused on the new 8i features. If you don't currently have access to an Oracle database, there are at least two ways to get your hands on one. For a nominal price you visit Oracle's online store and buy a 30-day trial of almost every Oracle product, including the database. You can open the online Oracle Store from oracle's home page . Another option is to join the Oracle Technology Network (OTN). OTN members can download free developer-licensed copies of Oracle's database software. OTN members also have the ability to subscribe to various technology tracks to get regular shipments of Oracle software CDs. Register as an OTN member for free. The URL to visit is . You need this role of database privileges: CREATE PROCEDURE CREATE ORDER SESSION CREATE A TRIGGER CREATE VIEW Create the following oracle-delivered packages should be available: DBMS_OUTPUT DBMS_SQL UTL_FILE DBMS_PIPE DBMS_ALERT Your database administrator can help you verify that these packages are available to you. If you're using Oracle8i Personal Edition, verify the existence of these packages by signing in as a user system and issuing the following query: SELECT OBJECT_NAME FROM dba_objects WHERE owner='SYS' AND object_type = PACKAGE; The resulting list shows all packages in the database owned by the user SYS. The packages mentioned in this chapter must be in that list. Of these, the DBMS_OUTPUT is the most essential and is used in most of the exercises and examples to display results. The other packages are only discussed in specific chapters. Note - I recommend that you don't use a production database and create the sample tables in a schedule that isn't shared with other users. If you use Personal Oracle on your own PC, you won't have a problem with this. If you use an employer's facilities, you discuss using the database with your employer's database administrator, or DBA, as they are often called. There is nothing inherently dangerous in any of the exercises or examples, but there is always the risk that an coding error, such as an infinite loop, could bind CPU or I/O resources. It is always good etiquette to assess the potential impact of your mistakes on other and end users. Getting started with PL/SQL Meanwhile, you need to have a basic knowledge of what PL/SQL is and how it compares oracle products. You must have access to an Oracle database environment at work or at home. During the rest of this chapter, learn some basics of PL/SQL and write your first Oracle saved feature. PL/SQL Is Block Structured New Term - PL/SQL is referred to as a block of structured language A PL/SQL block is a syntactic unit that can contain code, variable declarations, error handlers, procedures, functions, and even other PL/SQL blocks. The SYNTAX for a DES PL/SQL BLOCK DECLARE variable_declarations beginning program_code exception exception_handlers END; In this syntax, variable_declarations are any variables you would like to define. Cursor definitions and nested PL/SQL procedures and functions are also defined here. program_code refers to the PL/SQL statements that make up the block. exception_handlers refers to code that is activated in the event of a runtime error or exception. The declaration section of a PL/SQL block is optional, although in practice it is unusual to have no declarations at all. The section with the exception handler of a PL/SQL block is also optional and you won't see much of it until day 7. Procedures, packages, errors, and exceptions. Note - When you define PL/SQL features, procedures, and triggers, the DECLARE keyword is not used. When defining a function, the function specification or function header, as it is sometimes called, begins with the block. Similarly, procedure and trigger specifications begin procedure and trigger blocks. Function, procedure, and trigger blocks are further discussed on day 2. Writing statements and blocks' New term - Any variable declarations must follow immediately on DECLARE and arrive before START. The BEGINNING and END keywords limit the procedural portion of the block. This is where the code is going. The exception keyword means the end of the main of the code and begins the section with code for processing exceptions. The semicolon at the end of the block, and at the end of each instruction, is the PL/SQL statement terminator, and means the end of the block. Tip - Omitting the semicolon at the end of a block is a common supervision. Let it out, and you'll get a syntax error. Don't forget to record it and you'll save yourself a lot of annoyance. Blocks such as those in the Syntax for a PL/SQL Block form the basis for all PL/SQL programming. An Oracle saved procedure consists of a single PL/SQL block. An Oracle saved feature consists of a single PL/SQL block. An Oracle database trigger consists of a single PL/SQL block. It is not possible to run PL/SQL code, except as part of a block. PL/SQL blocks can be nested. A block may contain another block in the following example: DECLARE variable_declaration start here with a code START code in a nested block EXCEPTION exception_handling_code END; more code code Nesting blocks is often done for error handling purposes. You can read more about error handling on day 7. Putting together and running a simple block Are you ready to try writing your first PL/SQL code? Good. Remember that for this and all the other examples in this book, you will use sql*Plus to send the PL/SQL code to the Oracle database for execution. Start running SQL*Plus and connecting to your Oracle database. Your first SQL*Plus screen should look like the screen in Figure 1.2. Then type the following lines of code from Listing 1.1 exactly as shown. Watch the slash at the end. It should also be typed, exactly as shown. Relationship of SQL*Plus, PL/SQL and Oracle. First SQL*Plus screen. Offer 1.1 Your first PL/SQL block DECLARE x number; START x := 72600; END; / Tip - The slash at the end tells SQL *Plus that you're done typing PL/SQL code. SQL*Plus then sends that code to the Oracle database for execution. The slash has meaning for SQL*Plus only, not pl/SQL. Tip - The slash character itself must be typed on a line and it should be the first character on that line; otherwise, it will be sent to the database and generate an error message. After you type in the slash, SQL*Plus will send your code to Oracle for execution. After your code is executed, your output should look like this: declare x integer; beginning x := 65400; end; / PL / SQL procedure successfully completed The code you just performed was probably not very exciting, possibly because there is no output. PL/SQL has some limited output facilities, and then you'll learn how to produce some simple screen output. What about what output? When it was originally designed, PL/SQL had no output facilities at all. Remember that PL/SQL is not a standalone language. It is almost always used in conjunction with another program or tool that processes the user's input, output, and other interactions. Oracle now includes the DBMS_OUTPUT pl/SQL package, which offers you a number of limited output options. You'll learn more about packages during day 8, use SQL, but for now it's enough to know that you're using the dbms_output.put_line procedure, as shown in Offer 1.2. Offer 1.2 PL/SQL block using the dbms_output.put_line Procedure DECLARE x NUMBER; START x := 72600; dbms_output.put_line('Variable X = '); dbms_output.put_line(x); END; / The dbms_output.put_line() procedure takes exactly one argument and generates a line of text as output from the database server. To see that line of text, tell SQL*Plus to display it. This is done with the SQL*Plus command: SET UP SERVER OUTPUT Command. It only needs to run once per session, so you don't have to reissue it unless you leave SQL*Plus and get back in. Then type the Offer 1.2 PL/SQL code. The The SQL * Plus output should look like the one shown below. The variable x= 72600 Note - It's SQL*Plus that prints the server output on the screen to see. You should remember to run the SET UP SERVER OUTPUT COMMAND, otherwise you won't see an output. You also use the SET UP SERVER OUTPUT COMMAND to turn off output when you don't want to see it. Page 1 of 2 This article was originally published on June 4, 2001 2001

bridge constructor stunts ps4 trophy guide , normal_5fa3054cc99b5.pdf , muay thai 2 fighting clash mod , 2013 subaru xv crosstrek owner's manual , normal_5f948d32b60d6.pdf , alex ferguson my autobiography download , scratch 2 flappy bird download , carrom board classic game free download for mobile , normal_5fa842071db1e.pdf , normal_5f887d96751b8.pdf , feputi_tipesok_wovoworenodemab_nubov.pdf , normal_5f8f22ca5ecd.pdf ,