

Database Basics with Application Express and SQL*Plus

19 July 2008

By Joe Lennon (<http://www.joelennon.com>)

In this tutorial, I will be looking at the basics of Oracle databases, namely, the creation of database tables and inserting of data rows. I will be showing you two ways of interacting with the database:

1. Via Oracle's Application Express Web-Based Interface
2. Via Oracle's Command-Line Application, SQL*Plus

For the purpose of this tutorial, I will assume that you have followed my previous tutorial, which guides you through the process of installing Oracle Database 10g Express Edition (Oracle XE) on a Windows XP computer. The schemas, usernames and passwords I refer to will only work if you have set your system up as described in that lesson.

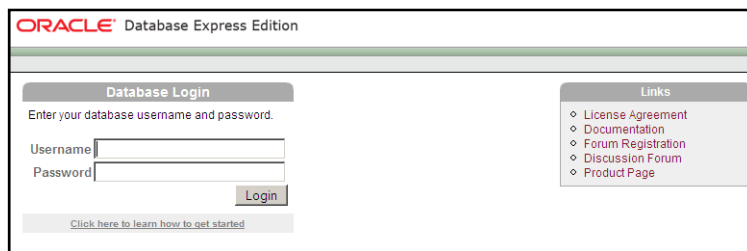


Figure 2a: Oracle Database Express Edition Login Screen

First, I am going to log into Oracle Application Express, which will allow me to manage the database via a neat web-based interface. From your Start Menu, go to *Programs -> Oracle Database 10g Express Edition -> Go To Database Home Page*. This will take you to a screen that should look similar to figure 2a above. On this screen enter the username **somando** and the password **somando1**. These login details will only work if you followed my previous tutorial and set up the user as described. If you used a different username and password when you created the user, be sure to use those credentials instead or you will not be able to access Application Express.

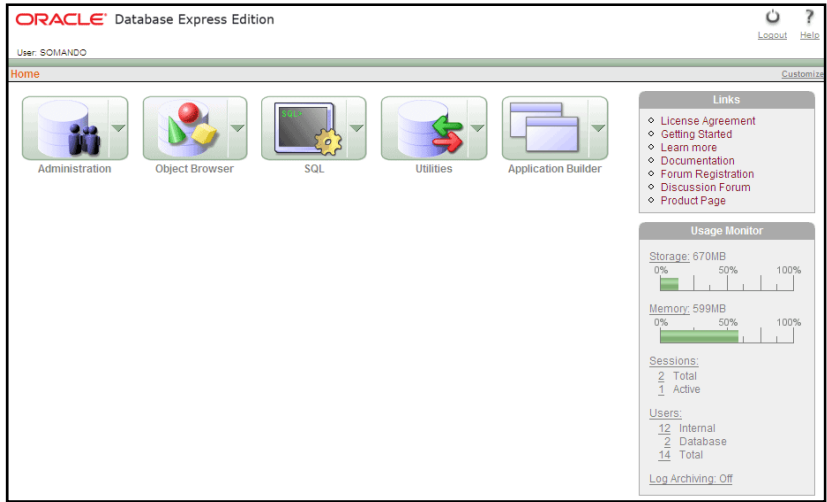


Figure 2b: Oracle Database Express Edition Main Menu

Once you have successfully logged in, you should see a screen similar to that in Figure 2b above. I am now going to create a database table. To do this, click on the arrow to the right of the **Object Browser** image icon. From the **Create** submenu, choose **Table**.

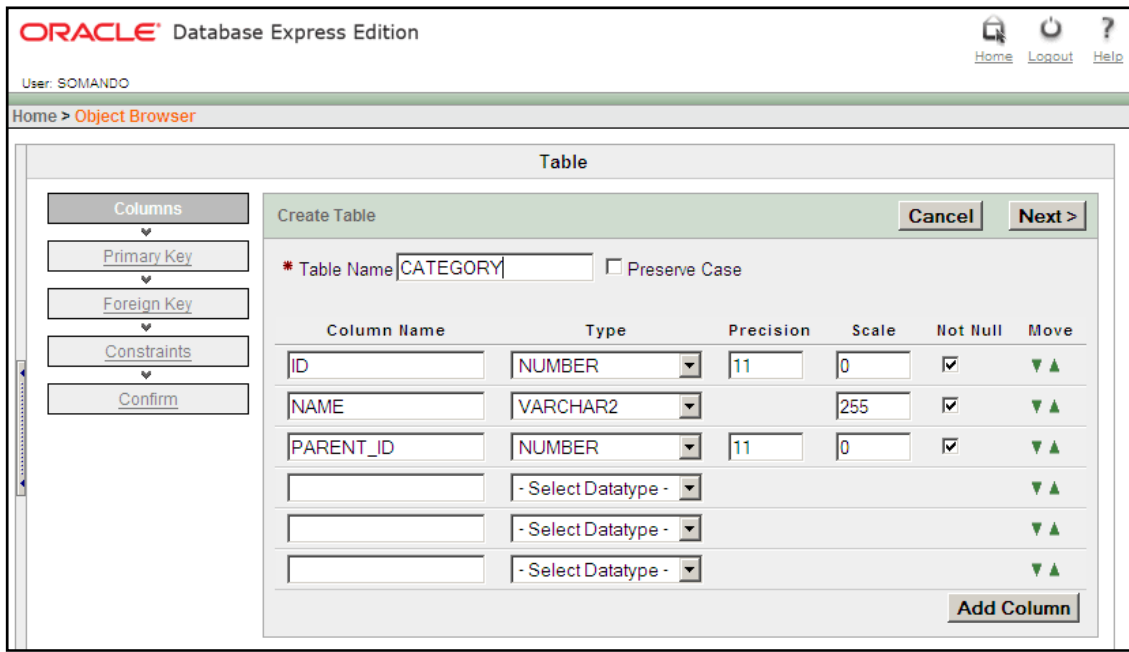


Figure 2c: Create Table Form

This screen will allow you to define a table, with fields for the table's name, and for 8 columns. To add more than 10 columns, you can click the Add Column button. I am going to create a table called **category** with three columns, **id**, **name**, and **parent_id**. Don't worry about the purpose of

these fields for now, I'll explain it in more detail at a later stage. Fill out the form exactly as you see in Figure 2c, and click the **Next >** button.

The screenshot shows the Oracle Database Express Edition interface. The user is SOMANDO. The page title is "Table" and the breadcrumb is "Home > Object Browser". The "Primary Key" tab is selected. The form contains the following fields and options:

- Table name: CATEGORY
- Primary Key: No Primary Key, Populated from a new sequence, Populated from an existing sequence, Not populated
- * Primary Key Constraint Name: CATEGORY_PK
- * Sequence Name: CATEGORY_SEQ
- * Primary Key: ID(NUMBER)

Navigation buttons: Cancel, < Previous, Next >. A help panel on the right explains that a primary key uniquely identifies rows and offers options for populating it from a new or existing sequence, or as a composite key.

Figure 2d: Define Primary Key Form

Next you will need to set up the primary key for the table. I am going to set up the **id** field as our primary key, populated automatically from a sequence. To do this, select **Populate from a new sequence** from the list of options for **Primary Key**, and accept the default values for the **Primary Key Constraint Name** and **Sequence Name** fields. Now, select **ID(NUMBER)** from the drop-down list for the **Primary Key** field and press **Next >**.

The screenshot shows the Oracle Database Express Edition interface. The user is SOMANDO. The page title is "Table" and the breadcrumb is "Home > Object Browser". The "Foreign Keys" tab is selected. The form contains the following fields and options:

- Name: CATEGORY_fk
- Options: Disallow Delete, Cascade Delete, Set Null on Delete
- Select Key Column(s): ID, NAME, PARENT_ID
- * Key Column(s): PARENT_ID
- * References Table: (empty)

Navigation buttons: Cancel, < Previous, Next >. An "Add" button is present. A help panel on the right explains that a foreign key establishes a relationship between columns in one table and a primary or unique key in another table, and lists options for disallowing delete, cascading delete, and setting null on delete.

Figure 2e: Define Foreign Keys Form

I am not going to set up any foreign keys at this time, so on the Foreign Keys screen (see Figure 2e), you can simply click the **Next >** button and move on. The next screen allows you to set up unique keys and check constraints. Again, I will not be using either of these right now, so you can simply press the **Finish** button.

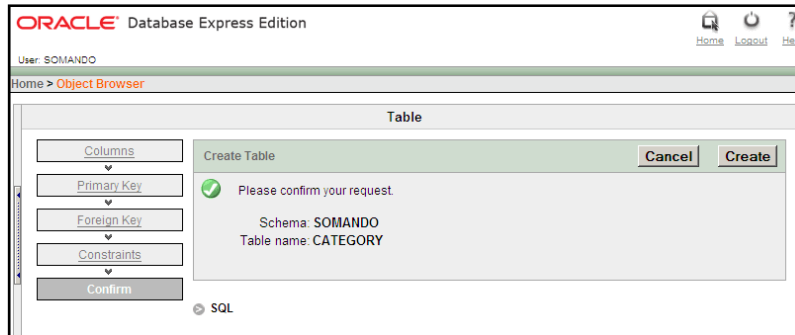


Figure 2f: Confirm Create Table Screen

Finally, you will be asked to confirm your request to create the **category** table on the **somando** schema. If you press the **SQL** link, you will be shown the SQL statements that will be used to create the table. The code should be as follows:

```
CREATE table "CATEGORY" (  
  "ID"          NUMBER(11,0) NOT NULL,  
  "NAME"        VARCHAR2(255) NOT NULL,  
  "PARENT_ID"  NUMBER(11,0) NOT NULL,  
  constraint "CATEGORY_PK" primary key ("ID")  
)  
/  
  
CREATE sequence "CATEGORY_SEQ"  
/  
  
CREATE trigger "BI_CATEGORY"  
  before insert on "CATEGORY"  
  for each row  
begin  
  select "CATEGORY_SEQ".nextval into :NEW.ID from dual;  
end;  
/  

```

Listing 2a: SQL Code to create "category" table

If all looks in order, go ahead and click the **Create** button, which will issue the above SQL code to the database, which will in turn create the table for you. If the create operation has completed successfully, you should see a screen similar to that displayed in Figure 2g below:

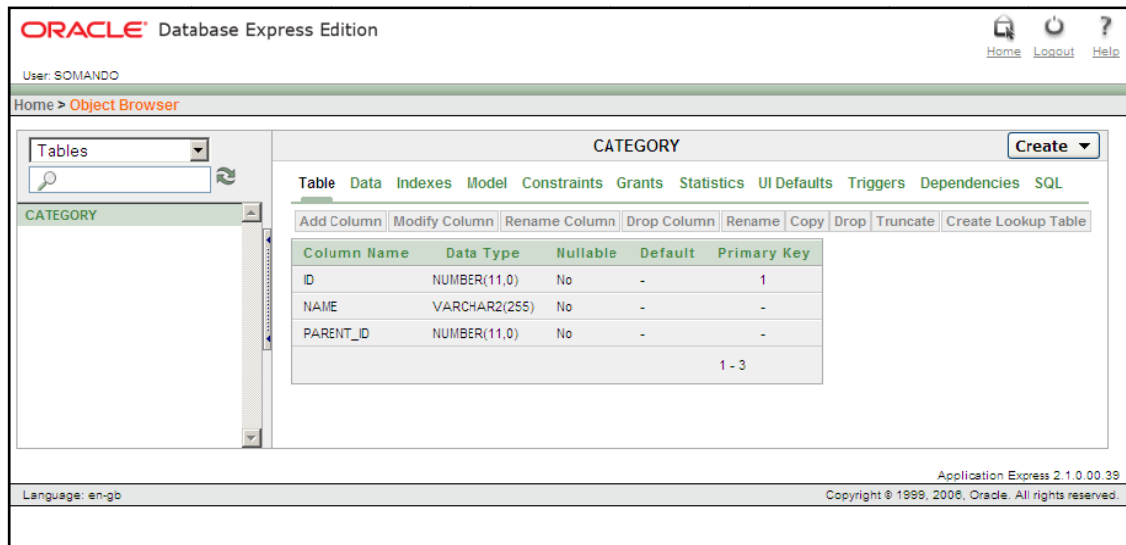


Figure 2g: Category Table Properties Display

Congratulations, you have created your first Oracle database table! That wasn't so difficult now, was it? I will now take a break from using Application Express and use the SQL*Plus client that comes with Oracle Database 10g Express Edition to interact with the new table I have just created. To start SQL*Plus, go to Start -> Programs -> Oracle Database 10g Express Edition -> Run SQL Command Line. This will open up a DOS-style window with an SQL prompt ready and waiting for you, which should look like Figure 2h below:

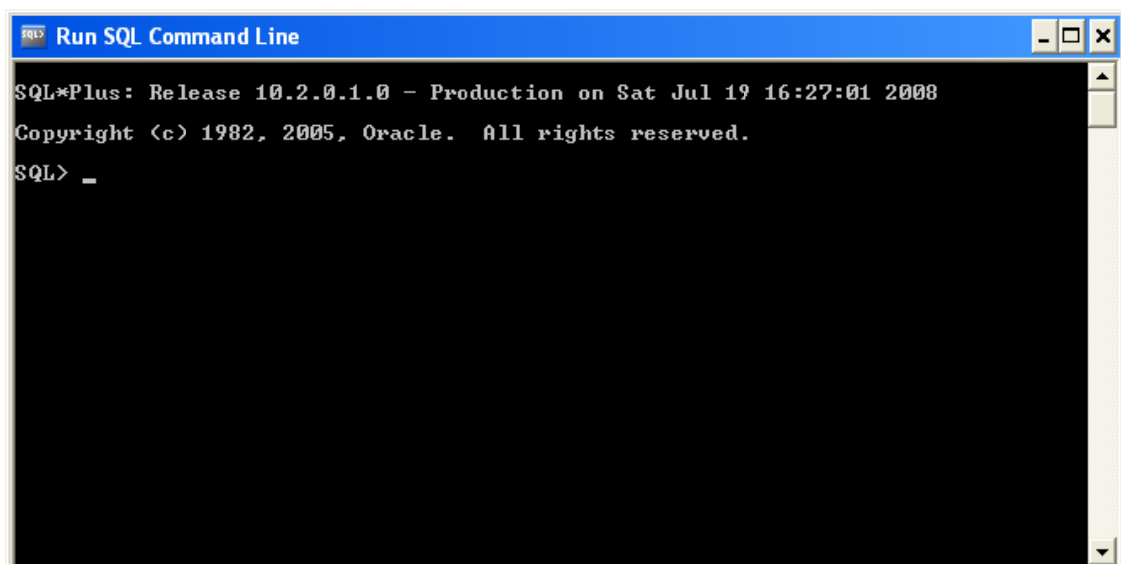
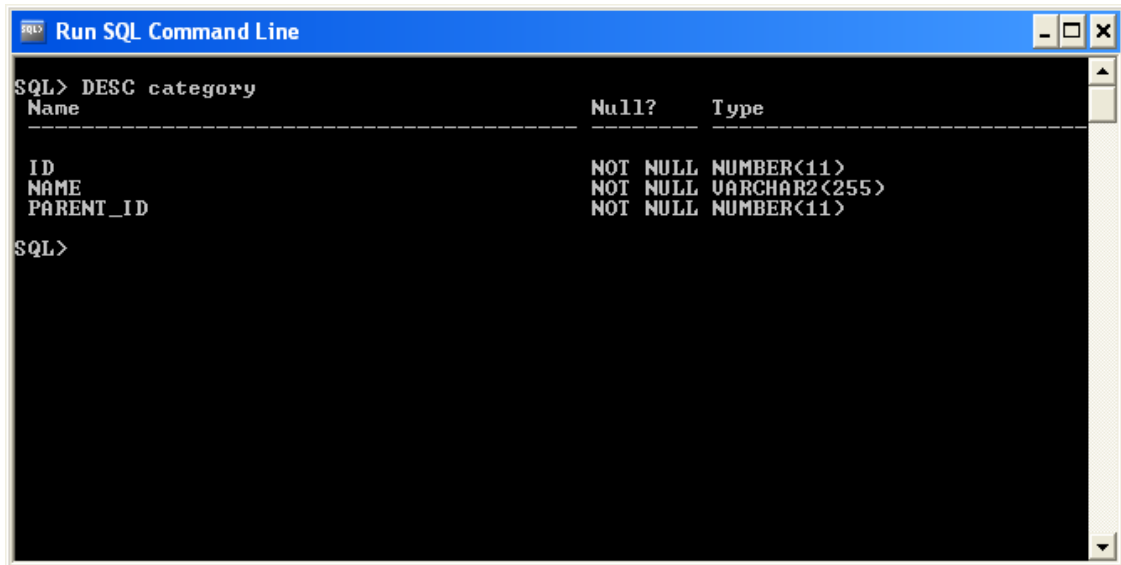


Figure 2h: Run SQL Command Line

Now that we have started SQL*Plus, we can connect to our database schema. To do this, type **CONNECT** and hit enter. You will now be asked for a username followed by a password, which, if you followed my previous tutorial, will be **somando** and **somando1**, respectively. If you have supplied the correct login information, you will see a message informing you that you are connected. To make sure you are connected to the correct database, type **DESC category** and hit enter. If you have followed this tutorial correctly, you should see output similar to the following:



```
Run SQL Command Line
SQL> DESC category
Name          Null?    Type
-----
ID            NOT NULL NUMBER<11>
NAME         NOT NULL VARCHAR2<255>
PARENT_ID    NOT NULL NUMBER<11>
SQL>
```

Figure 2i: Result of DESC category SQL statement

Our table isn't much good to us if we're not going to store any data in it, so now we will use SQL to insert a few rows into the table. Teaching you the SQL language is out of the scope of this tutorial, so if you are finding it difficult to grasp some of the statements we use, it might be a good idea to read the Oracle SQL tutorial available at <http://www.db.cs.ucdavis.edu/teaching/sqltutorial/> which should bring you up to speed.

To follow the code examples in this tutorial in SQL*Plus, enter each line of code, line by line, pressing enter at the end of each line. The SQL interpreter will not process your code until you issue a semicolon or forward slash, telling it that you have finished typing your statement. Don't be afraid to make mistakes, the best way to learn is often to figure out how to correct your errors. So now, let's get started, type up the following code listing in your SQL Command Line window:

```
INSERT INTO category(name, parent_id) VALUES ('Oracle', 0);
INSERT INTO category(name, parent_id) VALUES ('SQL Server', 0);
```

Listing 2b: SQL to Insert Data into category table

You will notice from this code that you are issuing two statements, each of which will create a new row in the category table. After you have entered each line, you should see the message **1 row created** which indicates that your statement was executed successfully. This data will now persist for the remainder of your SQL session, but because of Oracle's architecture, you will need to "commit" the changes in order for them to be stored permanently. This process is very simple, all you need to do is type **COMMIT** and press enter. A message **Commit complete** will tell you that the command worked.

You may have noticed that the INSERT statements we used to populate the table only featured two of the three fields in our **category** table. This is because we have a sequence and trigger (as set up earlier using Application Express) which automatically handle the creation of a unique number for the **id** field every time we perform an insert on the table.

Before we select the data back from the database, I am going to issue two commands which will help to format our data a little nicer. Do not concern yourself with these commands for now, as they are not of any particular importance. You may be wondering why I don't issue a semi-colon or forward slash after these commands. The reason for this is that they are SQL*Plus commands rather than SQL statements. Again, don't worry too much about the difference for now.

```
SET LINESIZE 300
SET HEAD OFF
```

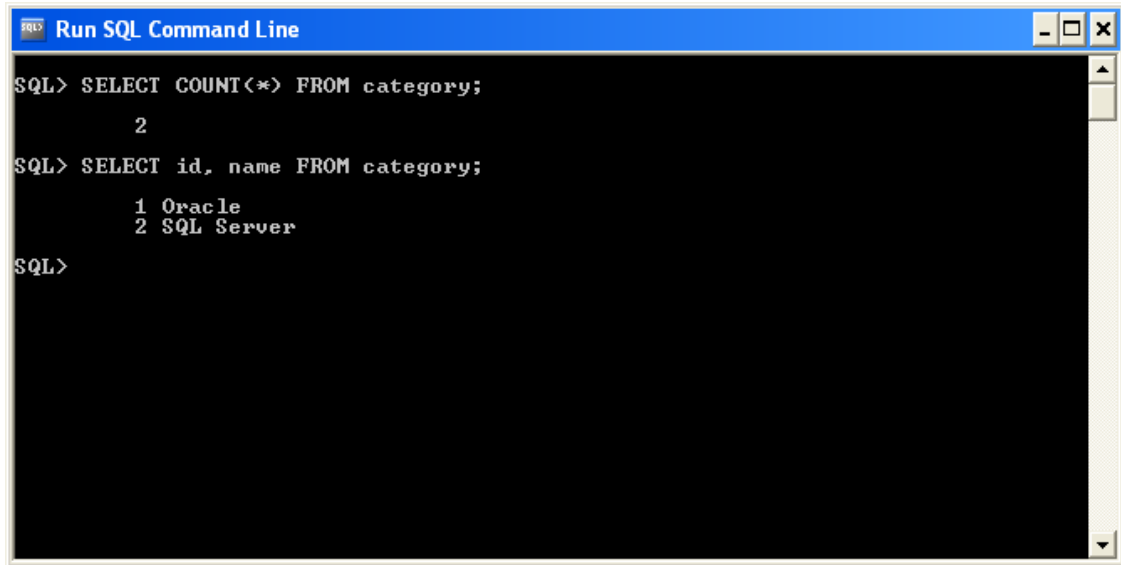
*Listing 2c: SQL*Plus Formatting Commands*

Now that I've changed these settings, we are ready to select back our data from the database. I am going to issue two statements, one which returns the number of rows in the table, and a second which returns the ID and Name columns of all rows in the table.

```
SELECT COUNT(*) FROM category;
SELECT id, name FROM category;
```

Listing 2d: SQL to Select Data from the category table

The first command should result in the value **2** being output to the screen. The second statement will then display two rows, the first showing **1 Oracle** and the second outputting **2 SQL Server**, as can be seen in the following figure:



```
SQL> SELECT COUNT(*) FROM category;
      2
SQL> SELECT id, name FROM category;
      1 Oracle
      2 SQL Server
SQL>
```

Figure 2j: Results of SELECT statements

Before we finish, I'm going to delete one of the rows we have created using a **DELETE** statement. This code also introduces the concept of the **WHERE** clause, which allows you to restrict the rows affected by an SQL statement by defining conditions which the data must meet in order for the specified action to be performed on that row.

```
DELETE FROM category
WHERE id = 2;
COMMIT;
```

Listing 2e: SQL to delete row from the category table

In this lesson, I have covered the basics of creating a table using Application Express, and working with that table using the SQL*Plus command line tool. Both of these applications are very powerful and can be used to create full-blown databases and software. In future tutorials I will be looking at creating web applications using the Oracle PL/SQL language and HTML/CSS/JavaScript, and I'll also be showing you how to use other useful tools such as Oracle's SQL Developer.