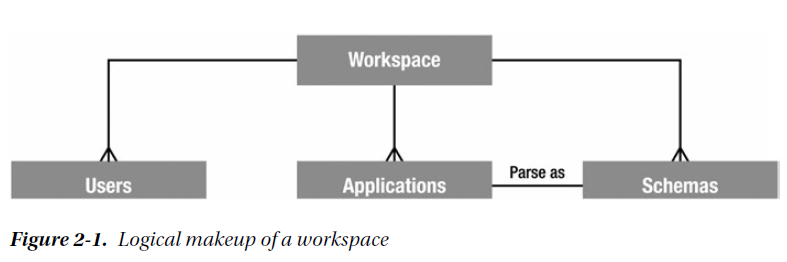
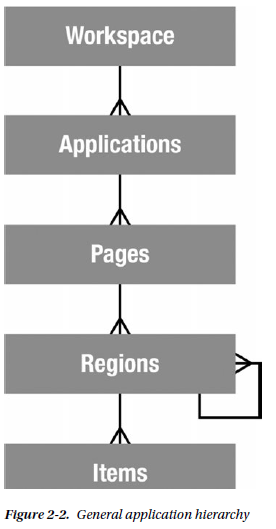
**Oracle APEX.**

**Libro: Begining Oracle Aplication Express 5**



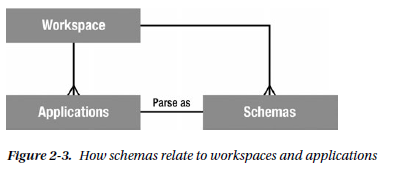
APEX 5.0 introduces the ability to use an external repository, such as Single Sign-on or LDAP, as a source

to assign and validate APEX users, meaning that a single user could have access to multiple workspaces.



*Regions* are UI items that serve as content containers. You can have any number of regions on a page, and regions can be nested in other regions. This gives you the opportunity to create things like dashboards,where you might nest a data report region and a graph region in a single parent HTML region.

***Items***are the HTML form elements that are used to present the UI to the user. These include things such as buttons, select lists, text fields, check boxes, radio groups, and so on. There are two categories of items: page-level items and **application-level items**. The difference is that the latter are defined at the application level and aren’t rendered directly on the page. You can think of these as **global variables**. Page-level items are defined on a specific page and are assigned to a region in order to control where and how they display to the user.



When a workspace is created, it’s linked with at least one, and possibly many, underlying database schemas. This provides access to database objects such as tables, views, stored PL/SQL program units, and so on.

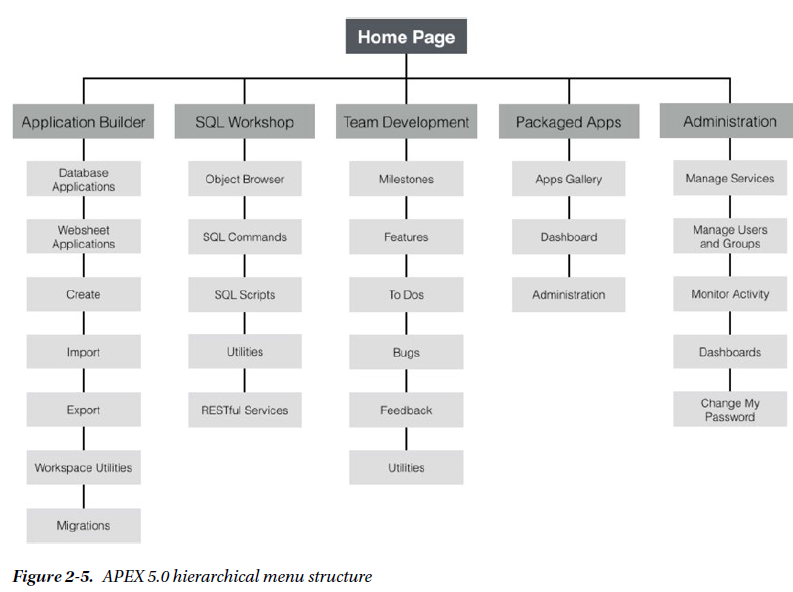
When an application is created, it’s assigned a single “parse as” schema from the list of schemas associated with the workspace. A *“parse as” schema* is the Oracle database user in which all SQL queries and PL/SQL calls run by that application are executed. So, if your application was defined with a “parse as” **schema of DOUG**, a query such as

**SELECT \* FROM EMP**

would execute in the database as if it were written

**SELECT \* FROM DOUG.EMP**

The **#OWNER#** replacement variable is substituted for the actual “parse as” schema for the application at runtime. So the statement



**SELECT \* FROM #OWNER#.EMP**

**The Page Designer**

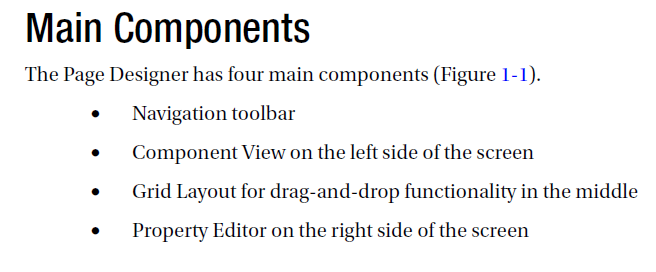
The Page Designer is where you’ll be spending most of your time as a developer creating and editing pages,

regions, and items.

**SQL Workshop**

The SQL Workshop is a suite of tools that provides developers with the ability to view and manage database

objects in the underlying schema(s) assigned to the workspace.



Variable que hace referencia al schema #OWNER#

Las consultas se hacen apuntando al schema.

Select \* from #OWNER#.emp

Lo cual se resuelve como

Select \* from BANESCO\_KPI.emp (nombre del schema)

BANESCO\_KPI es el “PARSE AS” SCHEMA de la aplicación.(solo puede ser uno)

IR: Interactive Report.

By default, all SQL statements executed via SQL Commands interface are automatically committed.

Pag. 37 Identifying the Problem and Desingning the Solution.

In APEX 4, the ability to use ROWIDs in place of primary keys was introduced to help solve the problem of multicolumn primary keys.

As a general rule of thumb, logic that controls or manipulates the UI is best placed in APEX, and logic that implements business rules or controls the data is best placed in stored program units in the database.

The APEX wizards make heavy use of database metadata for the objects in the “parse as” schema.

There are dozens of ways to draw ERDs, from pen and paper to high-end database-design tools.

However, I tend to take the middle ground and use Oracle’s SQL Developer Data Modeler.

**SQL Workshop / Object Browser : Crear las tablas del modelo.**

trigger named BI\_TICKETS (BI quiere decir Before Insert)

SQL Workshop’s Data Workshop utility to load and unload data from an Oracle schema in a number of ways.

Loading Data with the Data Workshop Utility

Lookup Table: Tablas de búsqueda (ejemplo caso del estatus:ABIERTO, CERRADO, PENDIENTE)

APEX can create a lookup table—complete with its own sequence, trigger, and foreign key—and modify the original table so it points to the new lookup table.

Pag. 60

The **SQL Scripts** tool of **SQL Workshop** allows you to create, upload, manage, and run SQL scripts.

**CREATE OR REPLACE FUNCTION** get\_status

(p\_status IN VARCHAR2)

**RETURN NUMBER**

**IS**

l\_status\_id status\_lookup.status\_id%TYPE;

**BEGIN**

SELECT status\_id INTO l\_status\_id FROM status\_lookup

WHERE status = p\_status;

RETURN l\_status\_id;

**END** get\_status;

/

Understanding User Interface Defaults

The Attribute Dictionary allows you to create more-generic UI Defaults based on attribute names.

Consider this a more macro-level definition. Attribute Dictionary definitions can also be assigned synonyms, allowing more than one attribute name to share the same actual definition.

Here are some things to note about UI Defaults:

Table Dictionary defaults always override Attribute Dictionary defaults.

When an item is created using UI Defaults, no relationship is established with the UI Default. Therefore, if you later change the definition of the UI Default, the changes aren’t propagated to previously created items.

Items created before UI Defaults have been established don’t inherit properties of the UI Default.

Developers can choose not to use UI Defaults, and even if they’re used, can override them after the component is created.

**UI Defaults** can be managed either from SQL Workshop’s Object Browser or from SQL Workshop’s Utilities

page.

The first step in **creating UI Defaults** is to synchronize the Table Dictionary with the database so it knows what tables are in your schema.

**Applications and Navigation**

Lists of Values (LOVs)

Applications in APEX are created through application imports, by copying an existing application, or

by running the Create Application wizard.

Las aplicacones “sample” vienen desbloquedas, miestras que las que no dicen “sample” están bloquedas y hay que desbloquearlas via el botón “Manage”.

Pag. 73 Applications from Scratch

Pag. 90 Public Pages

If any of the pages in an application require authentication, an appropriate authentication scheme must be applied to the whole application.

APEX lets you define individual pages as Public or Requires Authentication using a defining property of the page.

Public pages are useful for introductory landing pages, login pages, and information pages.

Navigation Bar Entries.

Placement of navigation bars is dictated by the page template substitution variable #NAVIGATION\_BAR#.

Global Pages

A *Global Page* is a special type of page that acts as a “master page” for your application and can be added

one per user- interface type (that is, you may have one Global Page for the Desktop UI and another for the

Mobile UI).

Lists

As briefly mentioned earlier in this chapter, the new Universal Theme uses static lists instead of Tabs for

navigation.

A list, as a shared component (unless it is designated as the default navigation list for the UI), doesn’t

display in an application directly. Normally, a list region must be configured on a page in order for the list to

be seen by the user. **APEX has a template type defined specifically to support lists.**

Lists of Values pag. 102

STATIC: Entries are automatically alphabetized.

STATIC2: Entries render in the order in which they’re entered.

The syntax for specifying a static LOV is as follows:

TYPE:DISPLAY;RETURN,DISPLAY;RETURN,...

The TYPE may be either STATIC or STATIC2.

Ejemplo: STATIC:C;1,A;2,D;3,B;4,

Form on a Table

Modify a form on a Table. Pag 115

oracle apex (en la nube)

workspace name: banesco\_kpi

user: ejimenmail@gmail.com

pass: salvador1208

Base de Datos Oracle:

user:SYSTEM

pass:salud

puerto:1521

SID:XE

Crear en las variables de ambiente:

ORACLE\_SID: XE

**Usuario y password Base de Datos Oracle en localhost.**

**User: SYSTEM**

**Pass: oracle**

INSTALACION DEL ORACLE APEX EN LOCALHOST.

Crear directorio:

C:\apex\_salud

SQLPLUS>

//crear el tablaspace para la base de datos.

create tablespace appx datafile 'c:\apex\_salud\apex.dbf' size 2000M extent management local segment space management auto;

create temporary tablespace temp0\_02 tempfile 'c:\apex\_salud\temp.dbf' size 2000M;

SQLPLUS> exit

>Copiar apex en C:\apex\_salud

“Entrar a sqlplus”

>sys as sysdba

oracle

SQLPLUS>@apexins.sql appx appx temp0\_02 /i/

“Entrar a sqlplus”

>sys as sysdba

oracle

SQLPLUS>@apex\_epg\_config.sql c:\apex\_salud (donde estan los dbf y lo que hace es crear un directorio de imágenes)

SQLPLUS>@apxldimg.sql c:\apex\_salud

SQLPLUS>@apxchpwd.sql (cambiar la contraseña)

user: ADMIN

password: Banesco#2017

http://localhost:8080/apex

Compañias de consultoria, hosting, libros, aplicaciones:

http://apex.oracle.com/community

blogs:

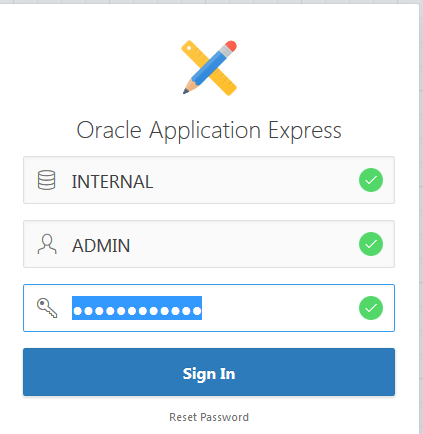
http://wwww.odtug.com/apex

Foro:

<https://community.oracle.com/community/database/developer-tools/application_express>

Grupos Meetups:

<http://apexmeetups.com>

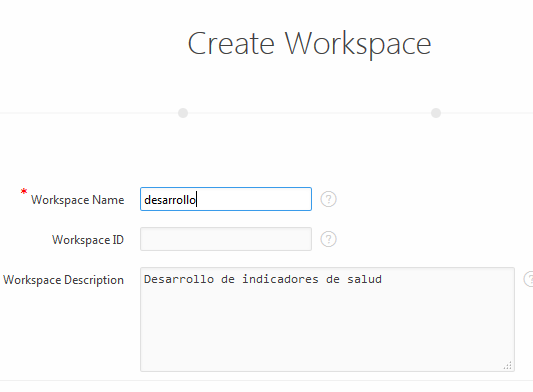


workspace defaul INTERNAL

usuario: ADMIN

passw: (por definir)

Crear workspaces:



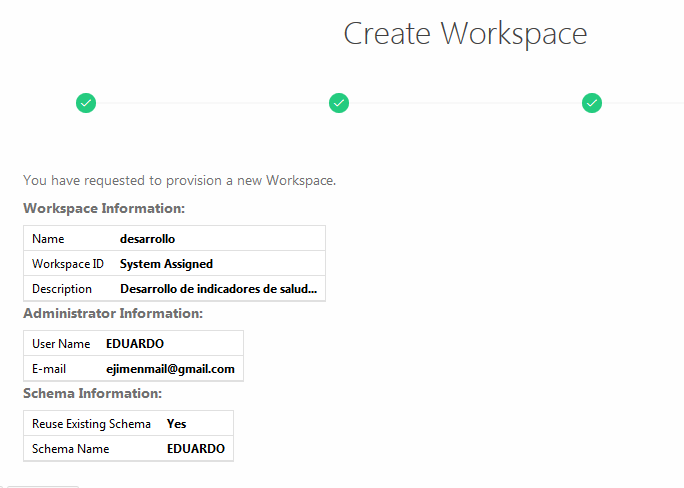
**CREAR UN ESQUEMA DE BD.**

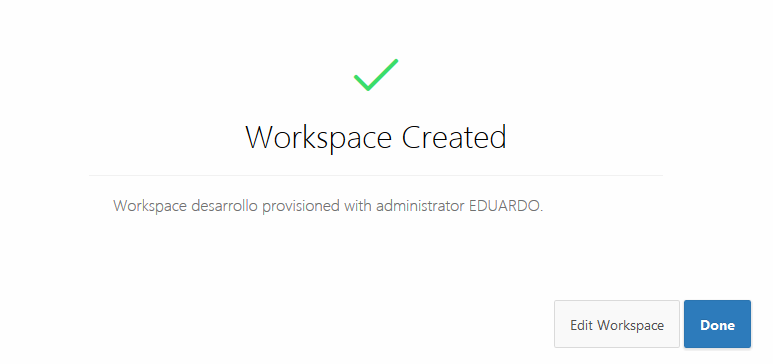
SQLPLUS> create tablespace APEX\_DEV datafile 'C:\oraclexe\app\oracle\oradata\XE\desarrollo.dbf' size 2000M extent management local segment space management auto;

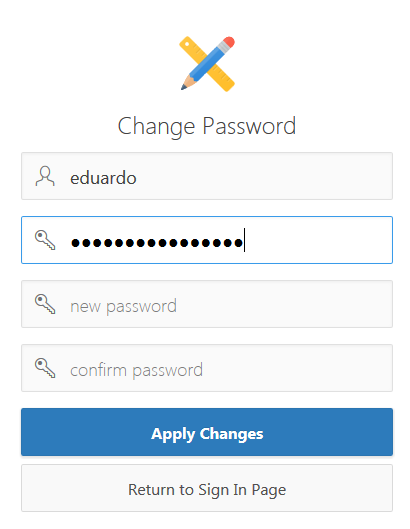
SQLPLUS> create user **eduardo** identified by **banescoSalud2017** default tablespace APEX\_DEV;

SQLPLUS> GRANT ALL PRIVILEGES TO EDUARDO;

**CREAR EL WORKSPACE EN APEX.**

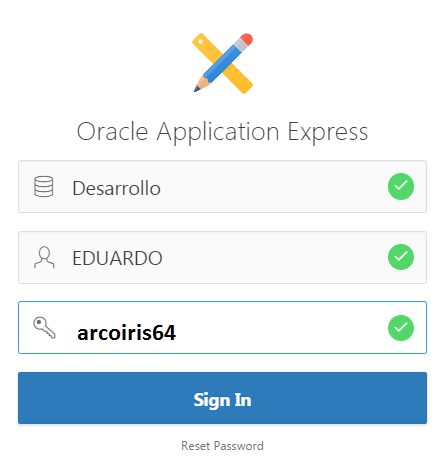






**NUEVO PASSWORD: arcoiris64**

**NUEVO PASSWORD: sti2017 (se cambio 13MAY2017)**

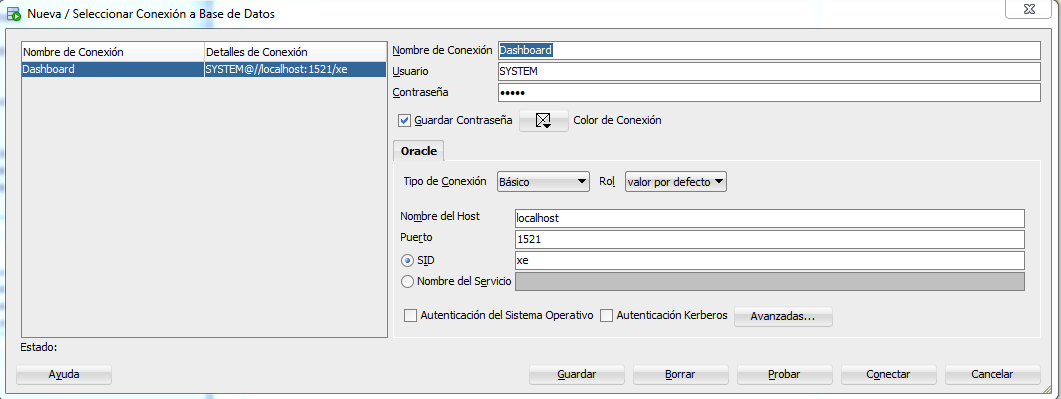


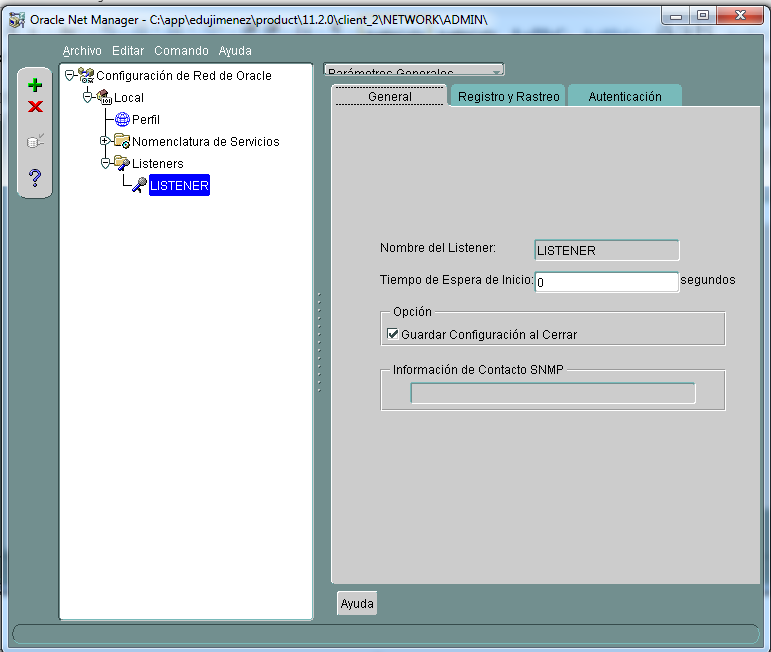
**oracle-apex-español.com (blog para las dudas)**

**Ubicaciones de los dbf en localhost: C:\oraclexe\app\oracle\oradata\XE**

create user salud identified by banescoSalud2017 Default tablespace APEX\_DEV;

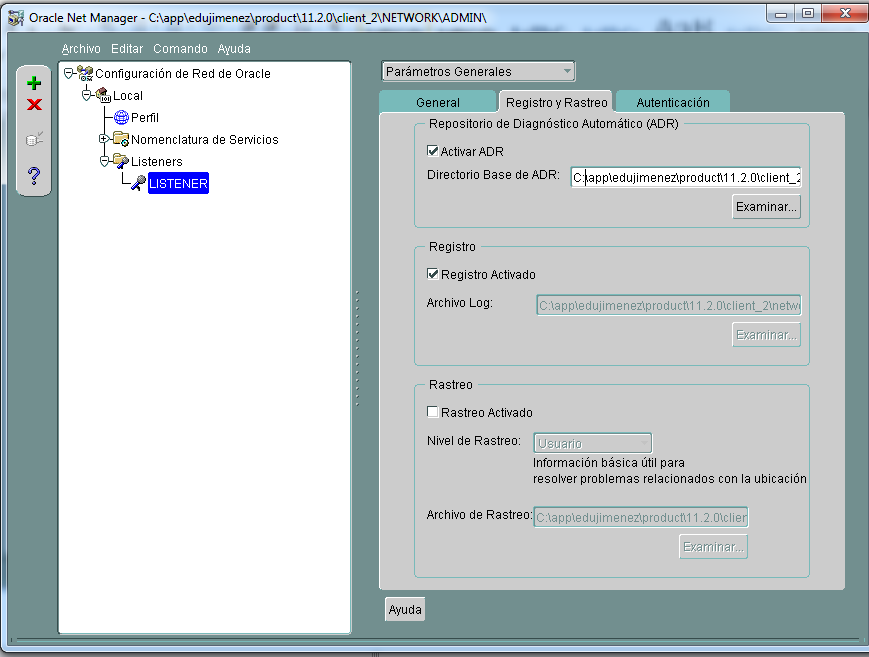
GRANT ALL PRIVILEGES TO EDUARDO;

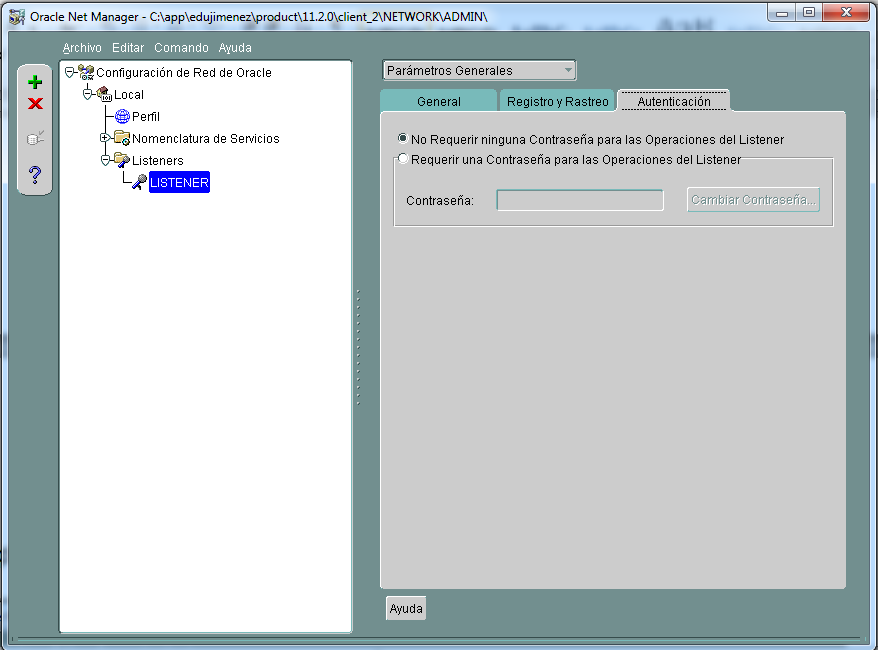


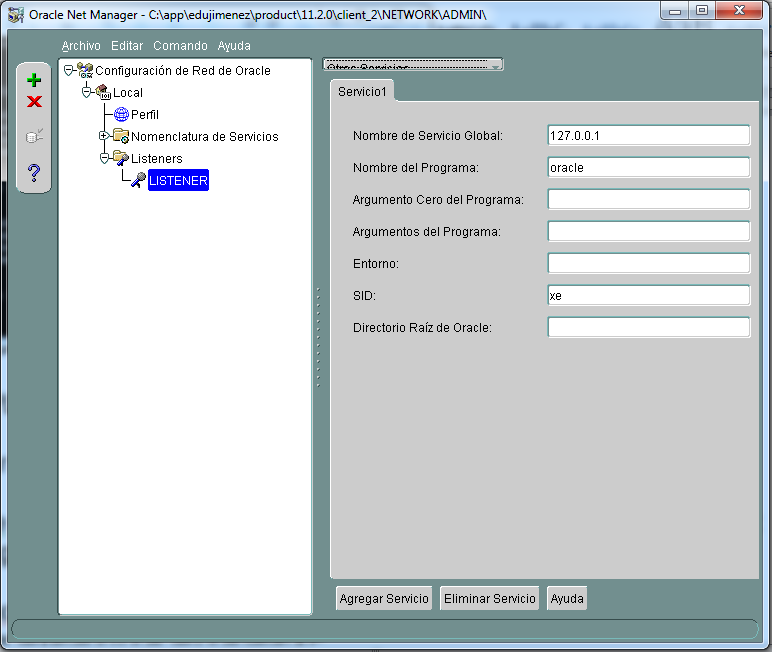


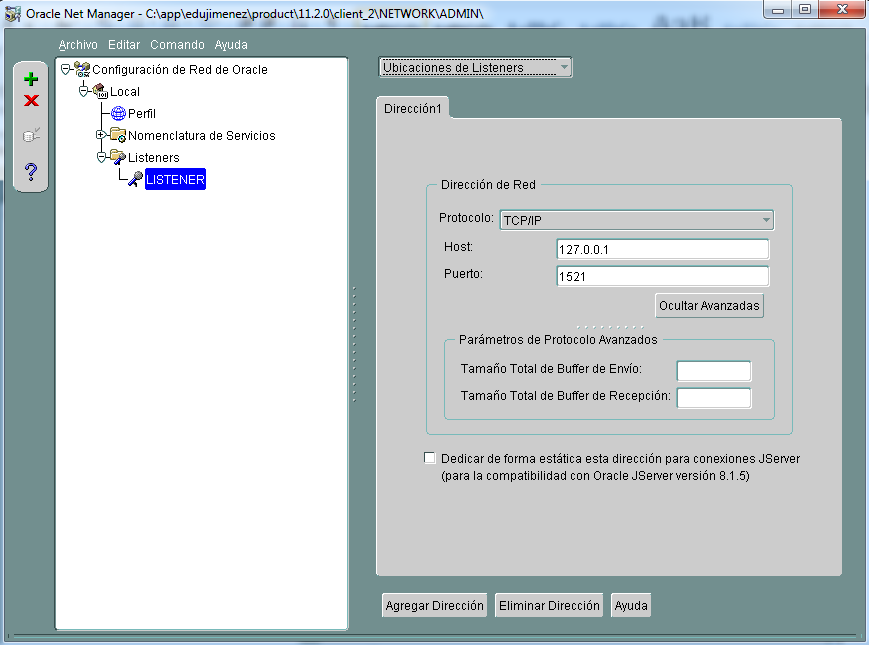
Cliente oracle: win64\_11gR2\_client.zip

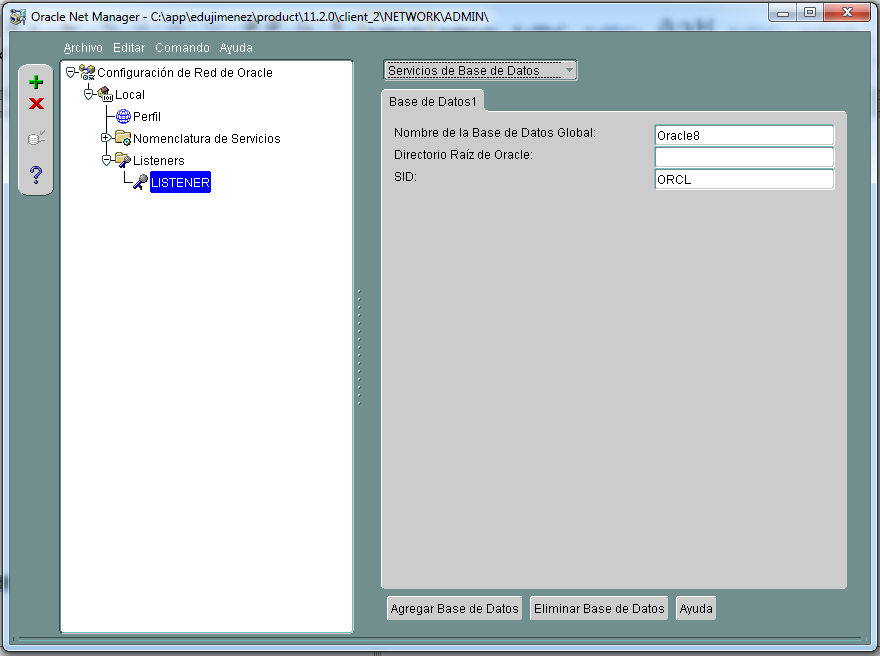
C:\app\edujimenez\product\11.2.0\client\_2\log

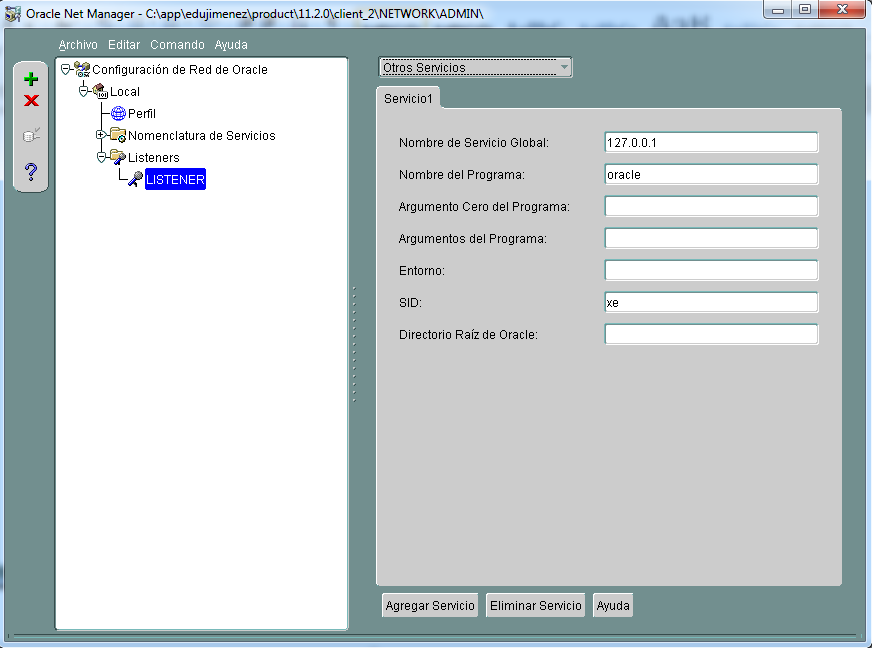


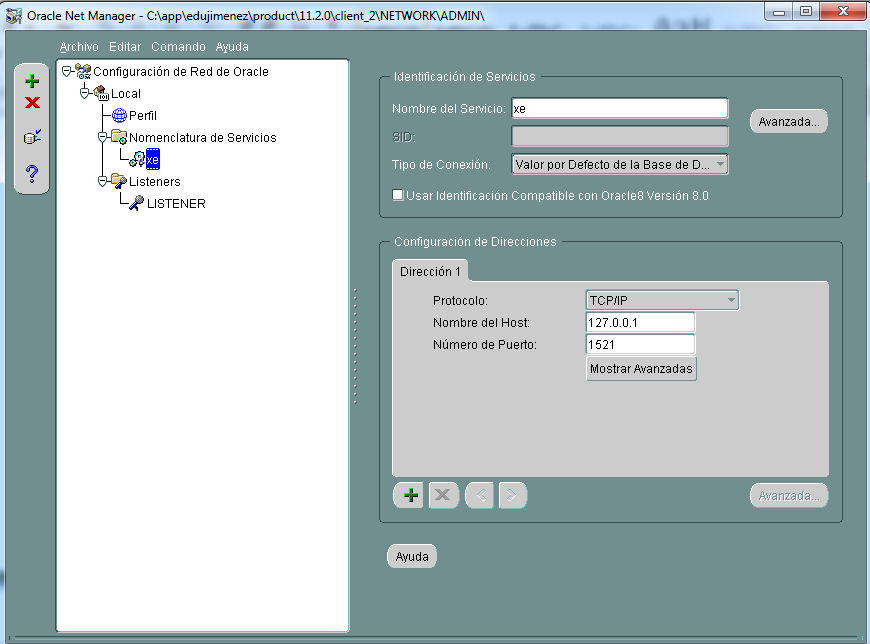








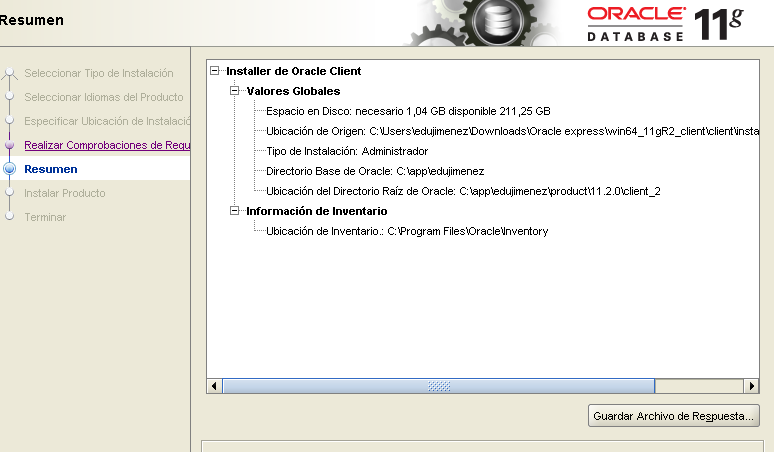




**Instalacion del Oracle 11g Express en localhost:**

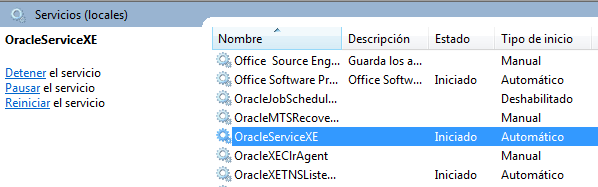
SYSTEM/SYS 🡪 passw: Oracle





**Para levantar APREX hay que levanta el servicio OracleServiceXE y levantar la base de datos.**

**Levantar los servicios Oracle:**



**SQL Scripts Interface Pag. 23**

When you upload a script file to the repository, the name of the script must be unique. You can’t

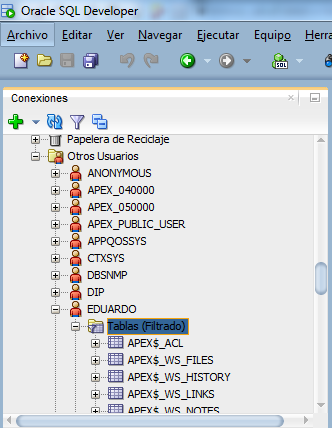
overwrite an existing script file of the same name with a new version without first deleting the existing script from the script repository.

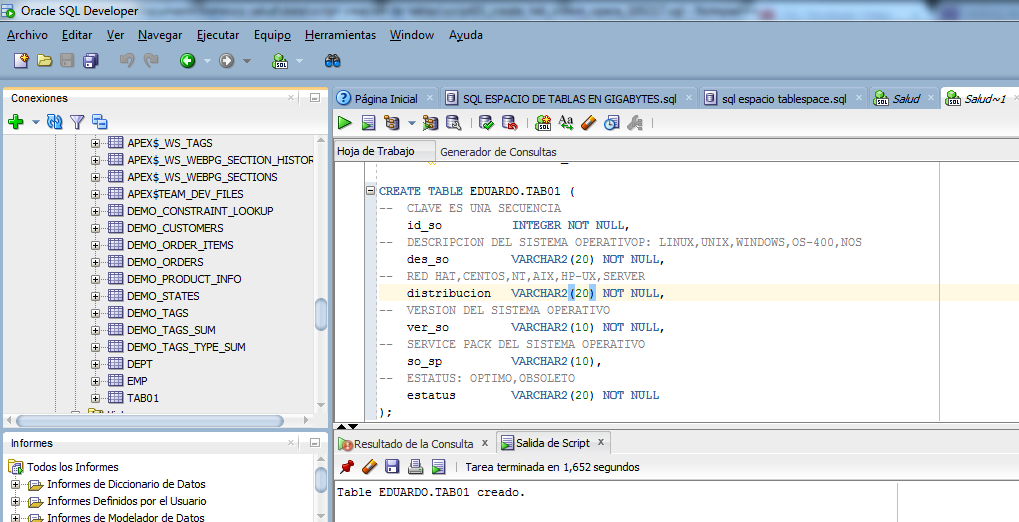
Actividad Pendiente: Instalar Oracle Datamodeler en Banesco.

The Query Builder Pag. 27

Ubicación de las tablas APEX dentro del SQL-Developer:

<https://www.youtube.com/watch?v=NYYbMDOsvzY>





Pag. 37

As a general rule of thumb, logic that controls or manipulates the UI is best placed in APEX, and logic that implements business rules or controls the data is best placed in stored program units in the database.

The best practice is to create the underlying database objects in the “parse as” schema for the application. This is how you will architect your help-desk system.

**Trigger Before Insert.**

create or replace trigger "BI\_TICKETS"

before insert on "TICKETS"

for each row

begin

if :NEW."TICKET\_ID" is null then

select "TICKETS\_SEQ".nextval into :NEW."TICKET\_ID" from sys.dual;

end if;

end;

**pag. 56**

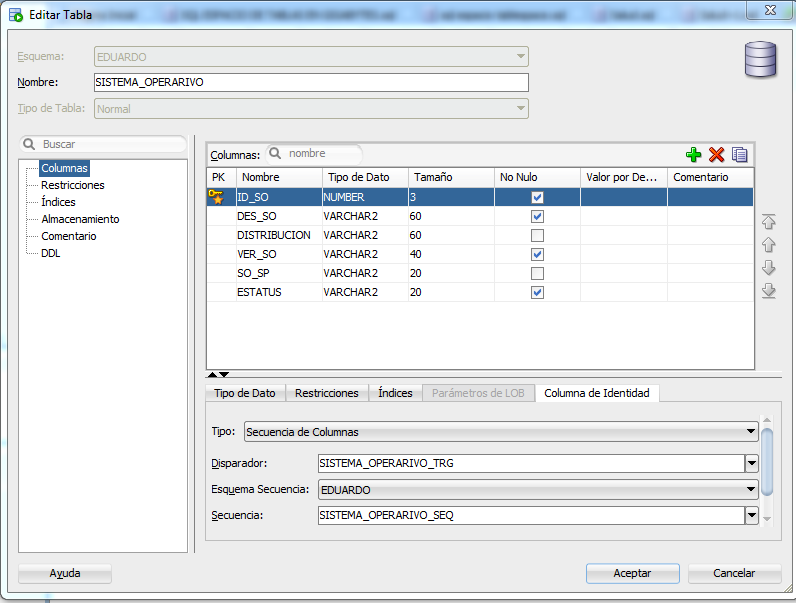
**Asignar secuencia a columna de clave primaria en tabla desde SQL Developer:**

MODIFICAR EL NOMBRE DE UN CAMPO:

ALTER TABLE COMPONENTE RENAME COLUMN ES\_VIRTUAL TO VIRTUAL;

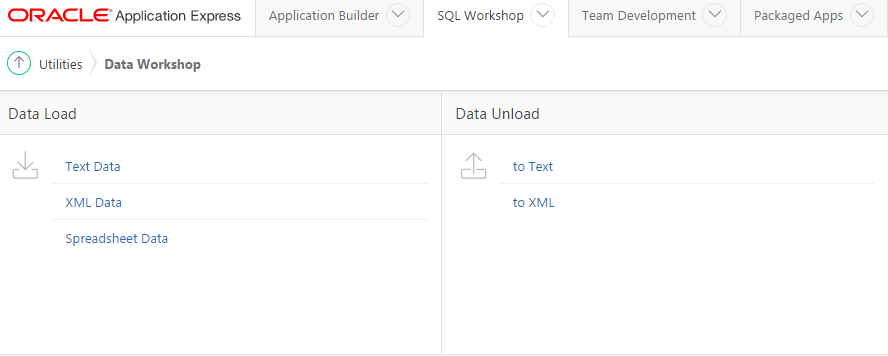
MODIFICAR EL TIPO DE UN CAMPO:

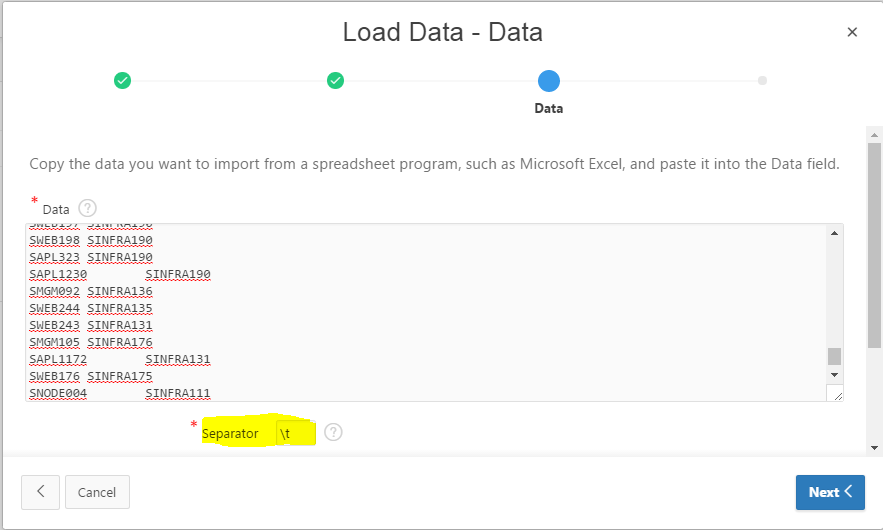
ALTER TABLE COMPONENTE MODIFY VIRTUAL VARCHAR2(20);



Subir la data desde Load Data de APEX:

Copiando y pegando directamente desde Excel.





Sobre la Función DECODE

DECODE (expression, search, result [, search, result]…[,default])

De una expresión dada, la función busca el valor y lo traduce a un resultado. Podemos tomar como ejemplo la siguiente consulta SQL

SELECT DECODE (categoria\_id,     1, 'Accesorios',

2, 'Mujer',

3, 'Hombre',

4, 'Niños',

5, 'Electrónicos',

'-') result

FROM demo\_categorias

**Lookup Table**: Se utiliza para crear una tabla ID, Desc, de un campo de una base de datos. Se crean desde la edición de la tabla en APEX.

Loading and Running SQL Scripts

APEX ignores any SQL\*PLUS-specific syntax.

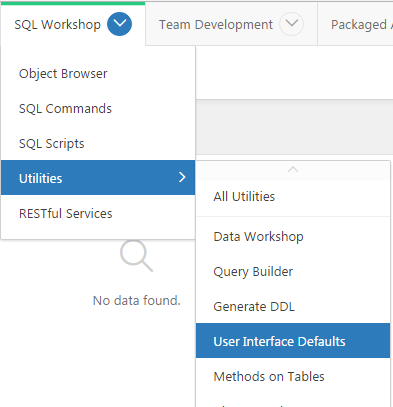
Let’s say you create an attribute-level default for any attribute named PHONE\_NUMBER. If a column named PHONE\_NUMBER appeared in a table and didn’t have a Table Dictionary default assigned, the Attribute Dictionary default would take effect.

The Table Dictionary allows you to define defaults for a specific table or column, and those defaults are only applied to APEX regions or items created for those specific items.

UI Defaults do help ensure consistency across your application and make your job much easier as a developer.

Defining UI Defaults for Tables

UI Defaults can be managed either from SQL Workshop’s Object Browser or from SQL Workshop’s Utilities page.



The first step in creating UI Defaults is to synchronize the Table Dictionary with the database so it knows what tables are in your schema.

1. Click the **Table Dictionary** **tab** along the top of the page, and then click the Synchronize button on the screen that appears.

2. Click the **Synchronize Defaults button** to begin the synchronization with the database. This may take a little time.

**Applications and Navigation**

After the example application has been created, you’ll add shared components to it. Shared components are items and structures that are common across all the pages in the application.

Static Content Regions

A Static Content region is a container that can have its own value, embedded JavaScript, or CSS definitions, or it can contain other page items. Any valid HTML entered in the source is rendered on an APEX page. Substitution-string syntax, such as &ITEM\_NAME., can also be used to display item values in the source text.

* From the Gallery in the lower center of the screen, select **Regions** for the component type.
* Click and drag the **Static Content** icon from the Gallery into the Grid Layout section of the screen and drop the component in the **Content Body** content area,

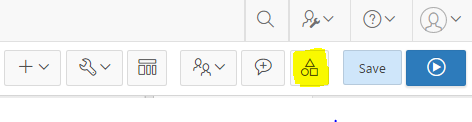
Public Pages

APEX lets you define individual pages as Public or Requires Authentication using a defining property of the page. Each page can have different security requirements (authorization), but only one authentication mechanism can be applied to an application. Public pages are useful for introductory landing pages, login pages, and information pages.

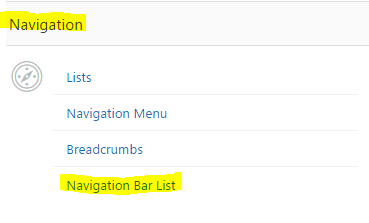
Navigation Bar Entries

Placement of navigation bars is dictated by the page template substitution variable **#NAVIGATION\_BAR#.** In most applications, the navigation bar is placed either at the upper right or upper left of the page.

Navigation bars are part of an application’s shared components, so they’re created and maintained from the **Shared Components** section of the **Application Builder**.



Shared Components

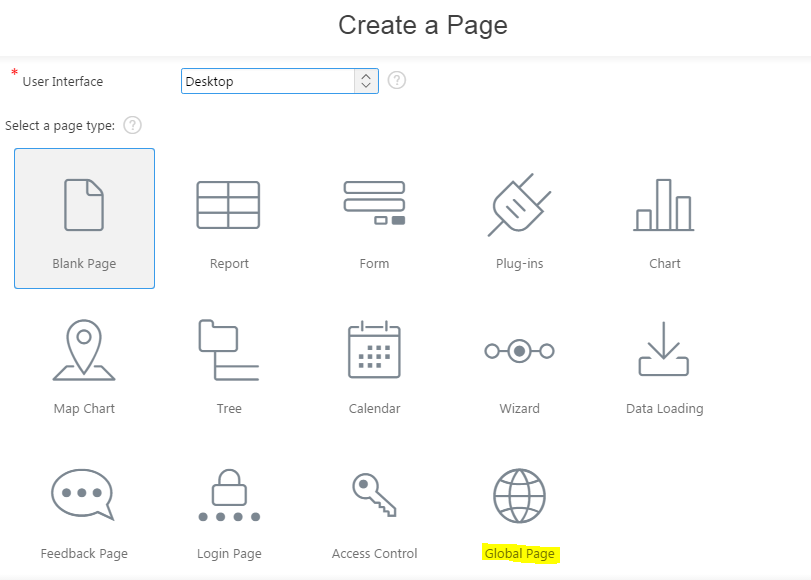


Global Pages

A *Global Page* is a special type of page that acts as a “master page” for your application and can be added one per user- interface type (that is, you may have one Global Page for the Desktop UI and another for the Mobile UI).

Items placed on a Global Page are rendered on every page in its related UI for that application unless conditionally told to do otherwise. This is particularly useful when you identify the need to display the same region on multiple pages or even on all pages in your application. Simply move a region to your Global Page, and it’s rendered with every page.

Although you can assign any page number to a Global Page, the default page number for a Global Page related to a desktop interface is zero (0).



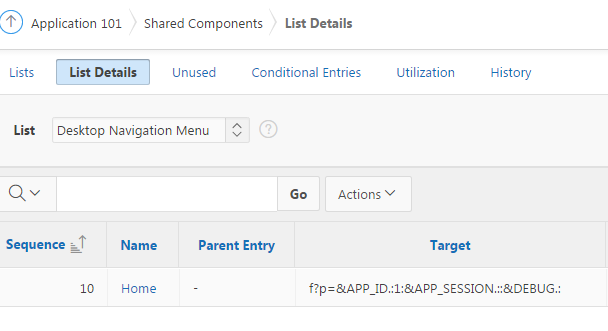
Global Page: se crean con el numero: 0

Breadcrumb Regions

Lists

As the name implies, a *list* is a structure that APEX uses to keep a collection of data for links. The list structure allows menus to be displayed consistently across numerous application pages, with easy maintenance performed in the Shared Components area of an application. Don’t confuse navigation lists, which we are discussing here, with lists of values (LOVs).

As briefly mentioned earlier in this chapter, the new Universal Theme uses static lists instead of Tabs for navigation.



Lists of Values

One of the fundamental benefits of writing an application on top of a database architecture is the ability to enforce data quality. LOVs are an APEX component that can be mapped to different item types, including Select Lists, Multiple Select Lists, Checkboxes, and Radio Groups.



**Static List of Values**

The syntax for specifying a static LOV is as follows:

TYPE:DISPLAY;RETURN,DISPLAY;RETURN,...

The TYPE may be either STATIC or STATIC2.

If you wish the display value and the return value for a given entry to be the same, omit the semicolon and specify only one value.

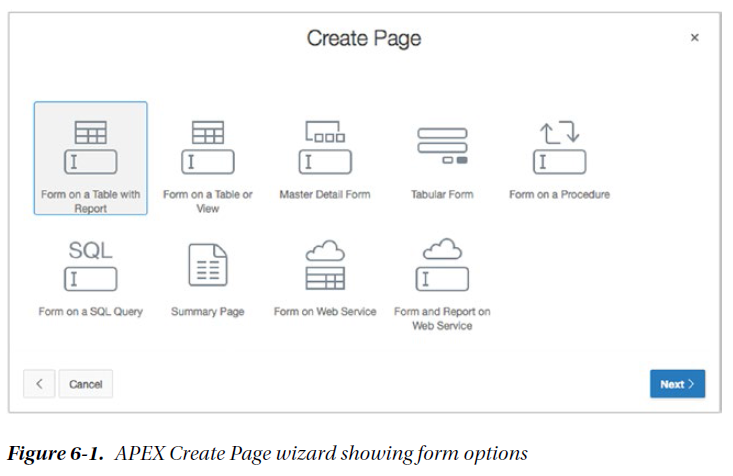
Ejemplo: STATIC:C;1,A;2,D;3,B;4,

STATIC2:C;1,A;2,D;3,B;4, (El STATIC2 respeta el orden, en STATIC muestra en orden alfabético)

**Dynamic List of Values**

As with static LOVs, dynamic LOVs have a display and return value pair requirement. The difference is that the values are obtained through an SQL query. The SQL query you write must return two columns. If the columns are the same, you need to use aliases to distinguish a display value and a return value.

APEX Forms



*Form on a Table with Report*: A form built on the columns of a table or view, having one item for each table column and processing a single row of data at a time, plus a report on the contents of the table or view, with navigational elements between the report and form pages.

*Form on a Table or View*: A form built on the columns of a table or view, having one item for each table column and processing a single row of data at a time.

*Master Detail Form*: A form on a pair of tables having a master–detail relationship. The APEX Master Detail Form Wizard creates all the data, processing, and navigational elements required for managing master–detail data.

*Tabular Form*: A multi-row, multi-column form (like a spreadsheet) that allows the editing of multiple rows and columns of data at once.

*Form on a Procedure*: A form based on the parameters of a procedure, typically to collect values for passing in to a procedure for subsequent processing.

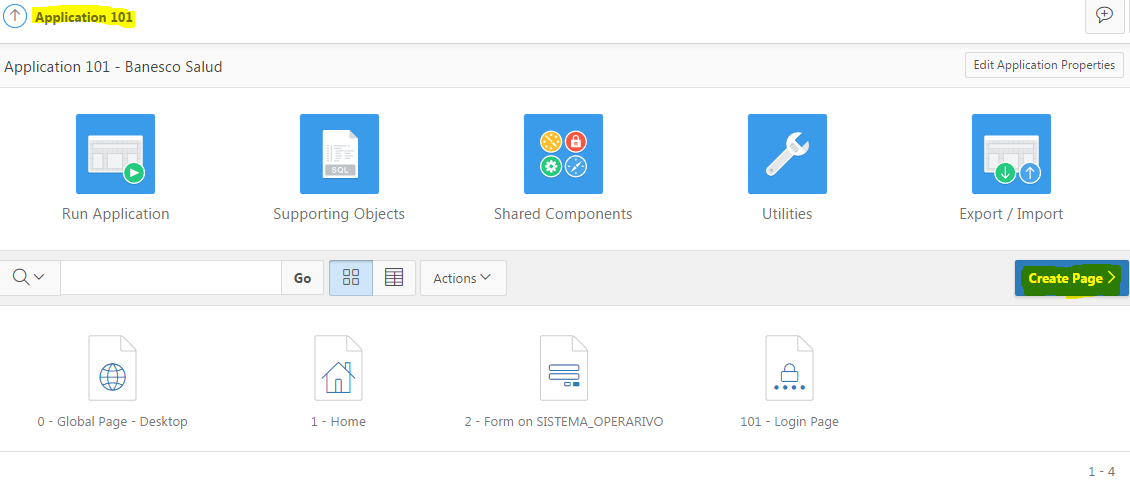
*Form on a SQL Query*: A form built on the results of a SQL query. This is a very powerful form construct due to its flexibility.

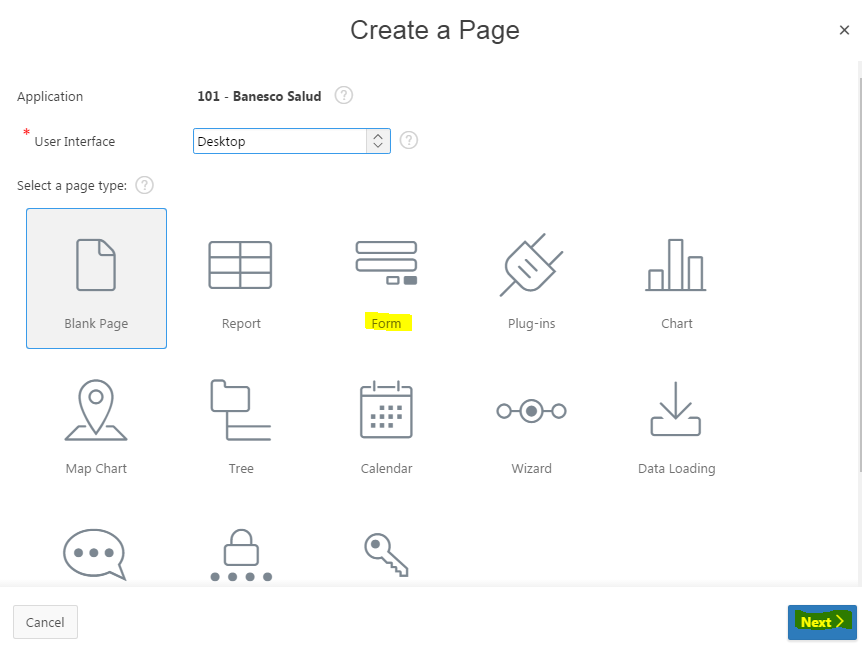
*Summary Page*: A display-only form showing selected items from an existing input form page. A summary page is often used in building a confirmation page for a wizard.

*Form on Web Service*: A form on the arguments of a web service.

*Form and Report on Web Service*: A single-row form on the arguments of a web service with a corresponding report of all rows of

**Creating a Form on a Table**





Pag 113

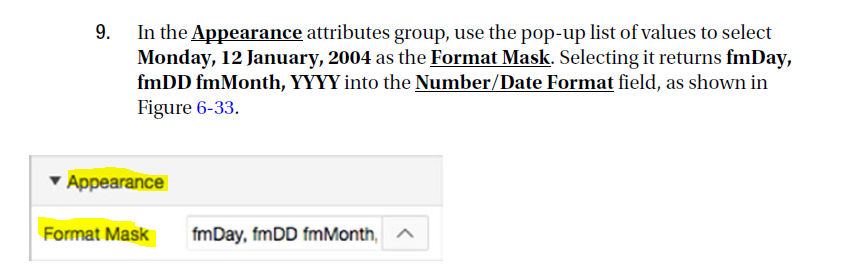
Forma Maestro Detalle.

<https://www.youtube.com/watch?v=a2l1r3RbHgg>

Instale el **web developer** en CHROME, falta instalarlo en FireFox:

<http://chrispederick.com/work/web-developer/help/>

Form on a Procedure pag 122



**Understanding Session State**

*Session state* is what allows APEX to keep track of all the values that belong in a particular user’s APEX session. Session state is particularly useful for keeping track of values as a user moves from page to page in the application. Unlike a stateful database application, where a connection is maintained continuously and all values are retained until changed or removed or until the session ends, an APEX application doesn’t maintain a continuous connection to the database. APEX is a *stateless* system—the APEX engine generates HTML pages based on directives stored in the APEX repository. Each page-rendering is a stateless transaction. An APEX session ties the stateless HTML pages together.

**Setting and Retrieving Session State**

In PL/SQL, when in a stored procedure, you can use the apex\_util.set\_session\_state procedure to set a value in session state, as follows:

**apex\_util.set\_session\_state( 'P1\_ITEM\_NAME', 'some value');**

The syntax to retrieve session state for an item varies according to where you’re referencing the item.

In templates or regions, tabs, menus, or lists, use the following substitution-string syntax (and don’t forget the trailing dot!):

**&P1\_ITEM\_NAME.**

Use the following syntax in SQL statements:

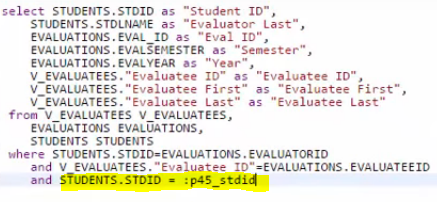
**:P1\_ITEM\_NAME**

’re in:

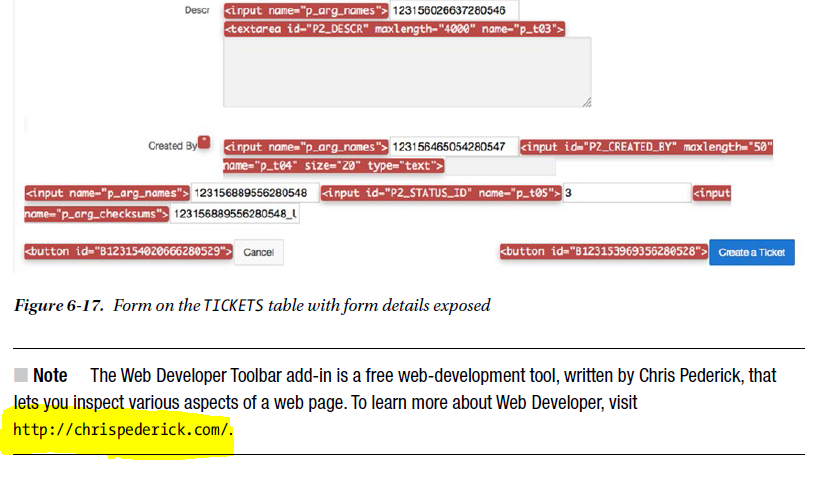
Anonymous PL/SQL block: **:P1\_ITEM\_NAME**.

PL/SQL Unit Called from APEX: **V('P1\_ITEM\_NAME')**

Within conditions, use this syntax: **P1\_ITEM\_NAME**



Modifying a Tabular Form Pag. 172



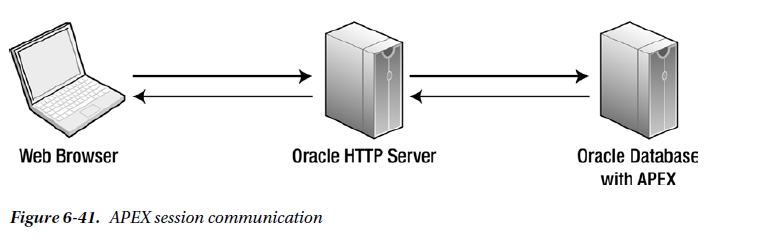
Session State

Next, let’s add a Search field to the report to allow users to **filter** for a specific ticket they may be interested in. Before we do, here’s a brief explanation of session state to help you understand how APEX keeps track of the values associated with a user’s session.

Unlike a stateful database application, where a connection is maintained continuously and all values are retained until changed or removed or until the session ends, an APEX application doesn’t maintain a continuous connection to the database. APEX is a stateless system—the APEX engine generates HTML pages based on directives stored in the APEX repository. Each page-rendering is a stateless transaction. An APEX

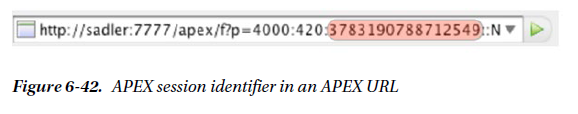
session ties the stateless HTML pages together.

An APEX session is logically and physically distinct from the underlying database session. A database session is stateful, and an APEX session is stateless.



**Sharing Database Connections**

Multiple APEX users can share the same database connection. There is a one-to-many relationship between APEX users and database sessions. This is why APEX can scale as well as it does—it doesn’t need dedicated database sessions, only a database session to use to process a request from a user.



**Setting and Retrieving Session State**

Session state is set by user-input items, computations, processes, and PL/SQL code. In PL/SQL, when within an APEX process, you can set an item to be equal to a value, like so:

**:P1\_ITEM\_NAME := 'some value';**

In PL/SQL, when in a stored procedure, you can use the apex\_util.set\_session\_state procedure to set a value in session state, as follows:

**apex\_util.set\_session\_state( 'P1\_ITEM\_NAME', 'some value');**

The syntax to retrieve session state for an item varies according to where you’re referencing the item.

In templates or regions, tabs, menus, or lists, use the following substitution-string syntax (and don’t forget the trailing dot!):

**&P1\_ITEM\_NAME.**

Use the following syntax in SQL statements:

**:P1\_ITEM\_NAME**

From PL/SQL, use one of the following two options, depending on what type of block or program unit you’re in:

Anonymous PL/SQL block: **:P1\_ITEM\_NAME.**

PL/SQL Unit Called from APEX: **V('P1\_ITEM\_NAME')**

Within conditions, use this syntax: **P1\_ITEM\_NAME**

**Viewing Session State**

To view session state, click the Session link on the Developer toolbar.

**The Importance of Bind Variables**

When referencing APEX item values, particularly in SQL queries in your APEX application, it’s important to think about SQL security basics, including SQL injection. Consider the example of an online form that allows a user to sign on with a username and password, which ultimately executes this query:

**SELECT COUNT(\*) FROM users**

**WHERE username = '&username'**

**AND password = '&password'**

If you enter this password

**I\_dont\_know OR 'x' = 'x**

the resulting SQL is

**SELECT COUNT(\*) FROM users**

**WHERE username = 'SCOTT'**

**AND password = 'I\_dont\_know' OR 'x' = 'x'**

This SQL statement erroneously returns 1, indicating True, rather than No data found. The user is allowed in! Not good. To prevent the injection of unintended SQL, use bind variables in the SQL query, like so:

**SELECT COUNT(\*) FROM users**

**WHERE username = :USERNAME**

**AND password = :PASSWORD**

**Built-In Items**

APEX includes several built-in items for referencing key APEX application-wide session-state values. These are set automatically by APEX and are available for reference by the developer throughout APEX. The most common of these are as follows:

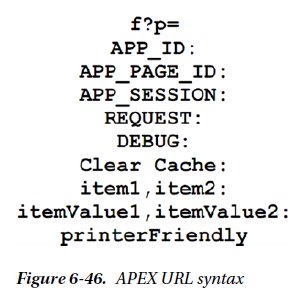
• APP\_ID: The application identifier of the currently running application

• APP\_ALIAS: The application alias of the currently running application

• APP\_USER: The currently signed-on user

• APP\_SESSION: The session identifier of the currently signed-on user

• APP\_PAGE\_ID: The currently running page identifier



**f?p** is the call to the f PL/SQL procedure passing the argument p. The argument is actually a concatenation of nine arguments combined into one, delimited by a colon. The nine elements of the p argument are the same for all APEX page requests. You may omit one or more of the arguments, but you must include the colon delimiters as placeholders.

The elements that form the p argument are as follows:

• APP\_ID: The application number or alias

• APP\_PAGE\_ID: The page number or alias

• APP\_SESSION: The APEX session identifier

• REQUEST: The HTML request

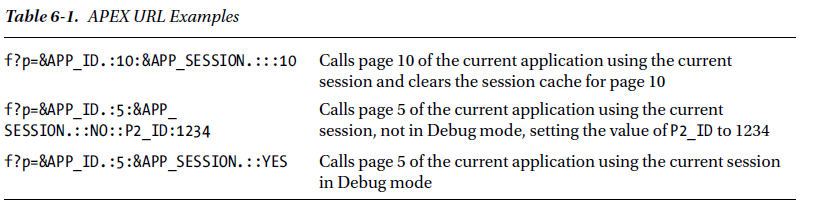
• DEBUG: A debug flag, set to YES or NO or omitted to use the current value of the debug flag

• Clear Cache: A list of pages for which to clear the cache

• Item names: A list of APEX item names, separated by commas

• Item values: A list of APEX item values, separated by commas, that correspond in order to the items specified in the list of item names

• Printer Friendly: A flag that determines whether the page is rendered in Printer Friendly mode



Pag 145

**PROCESS**

Processes can be defined at the individual page level or at the application level as part of the shared components. Functionally, page processes and application processes behave the same way. The difference is found in where business logic is contained. For processes that need to run on all pages, you can define an application process. Also, just as with regions, you can use Global Pages to define processes that run on every page, but only for page rendering.

**Process Types**

Each different process type has a different use depending on the requirements. The types and their uses are as follows:

• Automatic Row Fetch: Retrieves records from a single database table or view

• Automatic Row Processing (DML): Process to insert, update, or delete a record from a single database table or view

• Clear Session State: Clears session state values; also referred to as cache

• Close Dialog: Process to close the current modal or non-modal dialog

• Form Pagination: Process to retrieve the previous or next record from a database table or view. Most often used in master-detail forms

• Load Uploaded Data: Process to load the parsed spreadsheet data into an existing table or view

• Parse Uploaded Data: Process to parse the prepared spreadsheet data in preparation for loading into an existing table

• PL/SQL Code: Generally use for utilizing database PL/SQL logic

• Prepare Uploaded Data: Process to prepare spreadsheet data for uploading into an existing table

• Reset Pagination: Resets pagination for a report

• Send Email: Declarative interface to easily send email

• Tabular Form – Add Rows: Process to add a row into a tabular form region.

• Tabular Form – Multi-Row Delete: Process to delete multiple rows from a tabular form region

• Tabular Form – Multi-Row Update: Process to update multiple rows from a tabular form region

• User Preference: Process to set user preferences for the end user.

• Web Services: Submits a request to a web-service provider

• Plug-ins: Processes functionality provided by plug-ins

**PAGINA. 248**

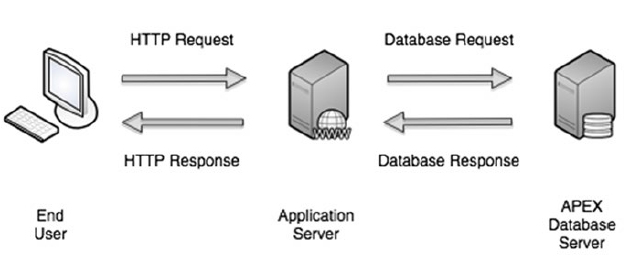
**Page Processing and Rendering**

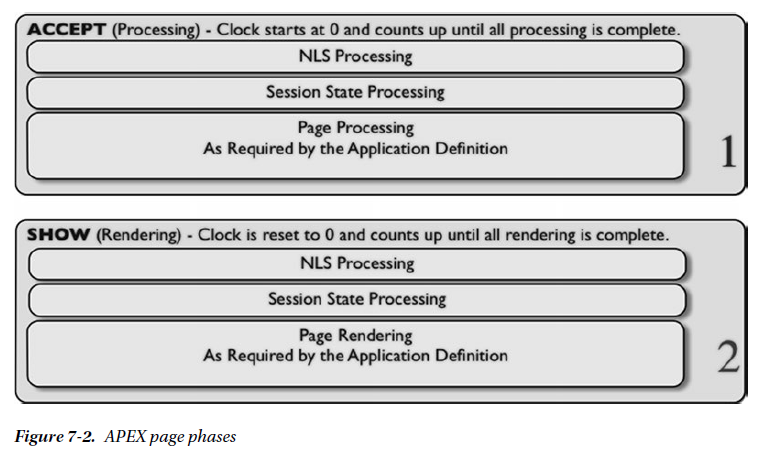
An APEX application is basically a group of pages that are linked together via buttons, hyperlinks, tabs, and so on. When a user navigates through the application, submits data, or requests to view an APEX page, there are two phases that the APEX engine goes through to provide the correct information to the user.

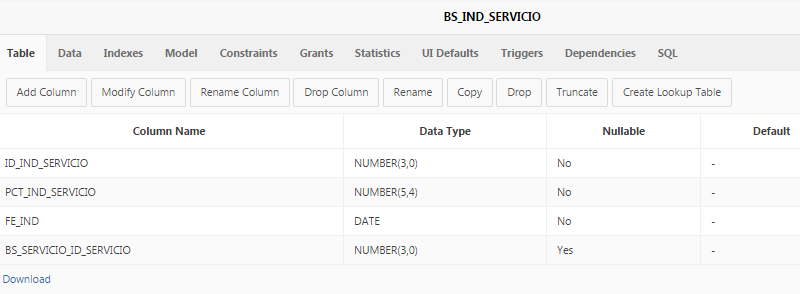
• ACCEPT (processing): This phase acts upon the request made by the user and, if appropriate, runs any defined validations, computations, processes, and branches. This includes setting session state and manipulating the underlying database tables.

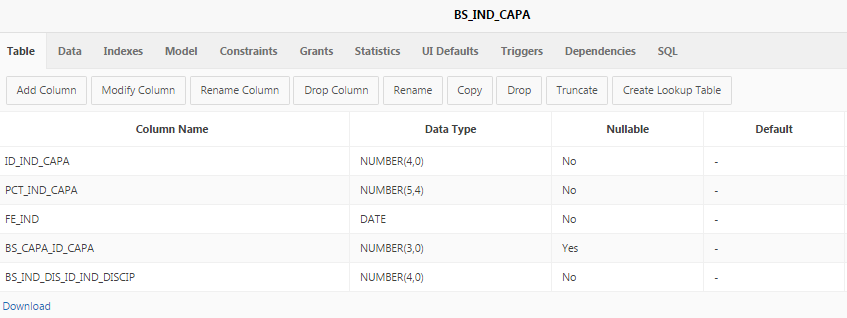
• SHOW (rendering): This phase executes all appropriate code to render the page that was identified by the branch taken in the processing phase. Page rendering may also contain computations, processes, and branches, as well as the visual components that make up the page to be rendered.

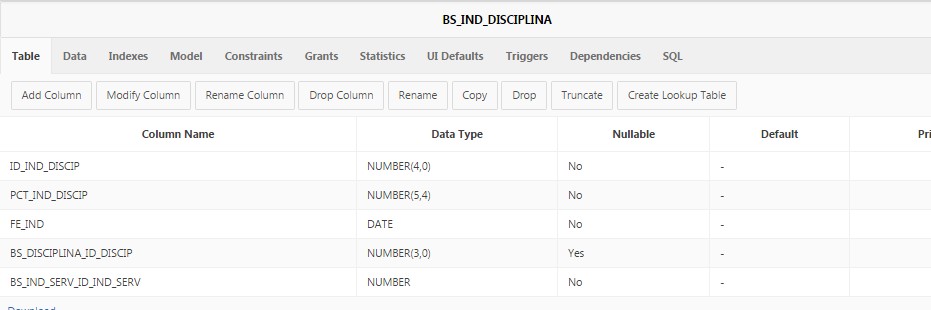
The reason these phases are important to debugging is that APEX breaks down the logging of debug information into these same component parts. When the application server tier connects to the Oracle database that contains the APEX installation, it grabs a connection from the database connection pool and uses that to service the user’s APEX request. You can see this sequence in Figure 7-1.

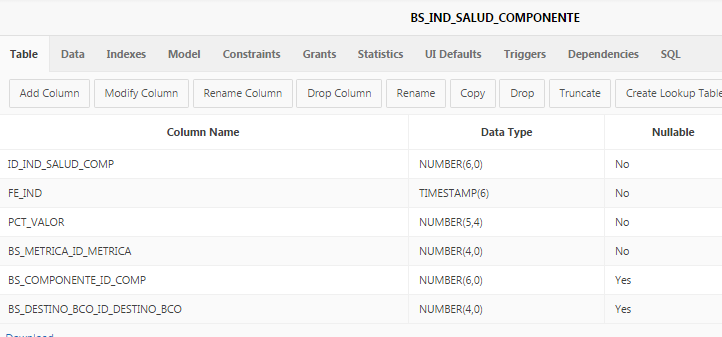


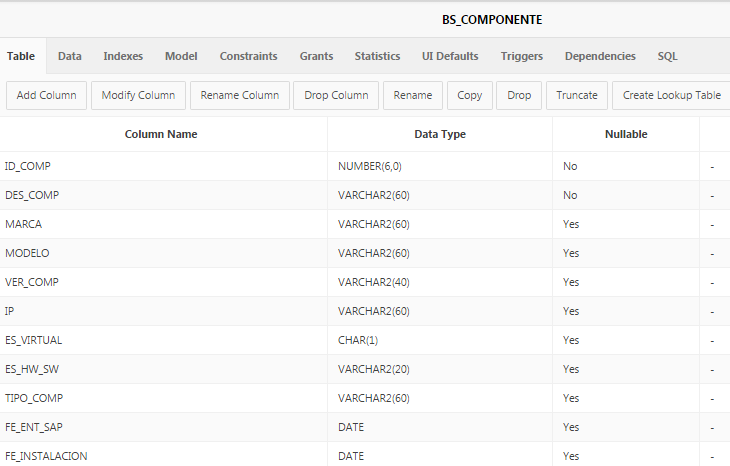












**APEX Tree implementation**

**jsTree region**

Like the legacy jsTree region, this new tree implementation is a Javascript-based, cross browser tree component that features optional keyboard navigation, and optional state saving

You can continue to edit existing jsTree regions, but no longer have the ability to create them.

You can create a Tree from a query that specifies a hierarchical relationship by identifying an ID and parent ID column in a table or view. The tree query utilizes a START WITH .. CONNECT BY clause to generate the hierarchical query.

Understanding Oracle APEX 5 Application Development, 2nd Edition

Pag 147.