UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

FACULTAD DE INFORMÁTICA

Ingeniería en Computación

Gestión de DDL en Oracle 11g

QUE COMO PARTE DE LOS REQUISITOS PARA OBTENER EL TITULO DE:

INGENIERO EN COMPUTACIÓN

PRESENTA

Gabriela López Mejía

Expediente. 127121

INC07

DIRIGIDO POR:

M. en A. Jabel Resendiz González

QUERÉTARO, QRO. 2012

GESTION DE DDL EN ORACLE 11g

Marco teórico:

Definición de DDL con ORALCE 2

Lenguaje de definición de datos (ddl) [ORACLE] 2
Creación de tablas y administración de tablas 2
Creación de tablas con CREATE TABLE 2
Restricciones de integridad 6
Anidamiento 12
Particionamiento 12
ENABLE y DISABLE en restricciones de integridad 12
Modificación de tablas con ALTER TABLE 15
Modificación de tabla 16
Modificación de columnas 19
Modificación de restricciones de integridad 20
Modificación de propiedades de columnas 20
Modificación de tablas externas 21
Movimiento de tablas 21
Cláusulas ENABLE/DISABLE 22
Modificación del particionamiento 22
Borrado de tablas con DROP TABLE 25
Recuperar tabla 26
Renombrar tabla 26
Truncamiento 27
Creación de índices 27

Sentencias de DDL con ORACLE
Tipos de datos 29
Alfanuméricos 30
Numéricos 30
Por fechas 30
Para objetos grandes 30
Rowid 31
Tipos más utilizados 31

Constrains 31
Tipos de Constrains 31

Comentarios 32

Crear y administrar secuencias 32

DDL en SQL 33

Tipos de objetos en SQL 34

Marco metodológico 34

Fuentes 71

Definición de DDL

Lenguaje de Definición de Datos.

Es un conjunto de instrucciones SQL que definen o eliminan objetos de base de datos como son las tablas o las vistas. Ejemplos de DDL son cualquier instrucción SQL que comience con create, alter y drop.

Esquemas, tablas, columnas y tipos de datos:

Las bases de datos organizan objetos relacionados dentro de un *esquema de base de datos.* Es normal organizar dentro de un solo esquema de base de datos todas las tablas y los demás objetos de base de datos necesarios para soportar una aplicación.

En Oracle, el concepto de esquema de base de datos está relacionado directamente con el concepto de usuario de base de datos, de modo que, un esquema de base de datos de Oracle se corresponde unívocamente con una cuenta de usuario y el esquema asociado tiene el mismo nombre.

Las tablas son las estructuras de datos básicas en cualquier base de datos relacional. Una tabla es una colección organizada de registros (o filas), todas ellas con los mismos atributos (columnas o campos) Las columnas de la tabla describen la estructura de la misma y las restricciones de *integridad de la tabla* describen los datos que son válidos dentro de la misma. Cuando se crea una tabla para una base de datos de Oracle, se establece la estructura de la tabla mediante la identificación de las columnas que se describen los atributos de la tabla.

Lenguaje de definición de datos (DDL) [ORACLE]

CREANDO TABLAS

La creación y administración de tablas de una de datos se realiza básicamente a partir de las sentencias de Oracle 10g SQL CREATE TABLE, ALTER TABLE Y DROP TABLE.

La sentencia CREATE TABLE tiene la siguiente sintaxis básica:

```
CREATE TABLE [GLOBAL TEMPORARY] TABLE [ usuario.] tabla
                    [ON COMMIT { DELETE | PRESERVE } ROWS]
               ( {columna tipo_dato [DEFAULT expresón]
                    [restricción_columna] | restricción_tabla }
```

[, {columna tipo_dato [DEFAULT expression]

[restricción_columna] | restricción_tabla}]…)

[CLUSTER cluster (columna [, columna]…] }

[INITRANS entero]

[MAXTRANS entero]

[PCTFREE entero]

[PCTUSED entero]

[STORAGE ([INITIAL entero]

[NEXT entero]

[PCTINCREASE entero]

[MINEXTENTS entero]

[MAXEXTENTS entero])

[TABLESPACE espacio_tabla]

[ENABLE active | DISABLE desactiva]…

[PARALLEL | NOPARALLEL]

[ORGANIZATION EXTERNAL]

[NESTED TABLE]

[PARTITION]

[AS consulta]

GLOBAL TEMPORARY indica que la tabla es temporal y su definición es visible para todas las sesiones. ON COMMIT específica cuándo los datos temporales persisten. DELETE ROWS borrará todas las filas de la tabla después de COMMIT (PERSERVE hace lo contrario). *Usuario* es el esquema o propietario de la tabla y si se omite, toma como valor por defecto al usuario que ejecuta la orden. *Tabla* es el nombre de la tabla que se crea, *Columna* es el nombre de las columnas que se van creando, tipo_dato es el tipo se los datos que componen la columna, *DEFAULT expresión* inidica el valor que se asignará a la columna por defecto cuando la sentencia INSERT omita el valor para ella, y *restricción_columna* y

restricción_tabla fija las restricciones de integridad de tabla y columna que serán analizadas.

Cluster incluye la tabla en el clúster nombrado, INITRANS indica el número inicial de transacciones que pueden utilizarse concurrentemente en un bloque de datos, MAXTRANS indica el número máximo de transacciones que pueden actualizarse concurrentemente en un bloque de datos, PCTFREE indica el porcentaje libre de espacio disponible en el bloque actual para insertar filas y PCTSUDED indica el porcentaje mínimo de espacio disponible en el bloque actual para insertar filas. STORAGE regula el almacenamiento, de modo que INITIAL asigna la primera extensión de espacio al objeto que está creando, NEXT es el tamaño de la extensión que se asigna después de haber la extensión inicial, PCTINCREASE controla la tasa de crecimiento de las extensiones posteriores a la segunda, MINEXTENTS indica que cuando se crea un objeto únicamente se asignan la extensión inicial y MAXEXTENTS establece el número total de extensiones que se pueden asignar.

PARALLEL se utiliza para que Oracle seleccione un tipo de paralelismo igual al número de CPU posibles en todas las instancias participantes a la vez. NOPARALLEL invalida el paralelismo.

EXTERNAL indica que la tabla es una tabla se sólo lectura localizada fuera de la base de datos.

TABLESPACE es el nombre del espacio de tabla al que se asigna la tabla actual y las cláusulas ENABLE y DISABLE activan o desactivan propiedades.

NESTED TABLE indica el anidamiento y PARTITION el particionamiento.

La cláusula AS crea las filas de la nueva tabla usando las filas obtenidas en la *consulta* específica.

Restricciones de Integridad

Para diseñar las tablas, es necesario identificar los valores válidos para cada columna y decidir cómo se debe exigir la integridad de los datos de la columna. SQL proporciona varios mecanismos para exigir la integridad de los datos de una columna, entre los que destacan:

- Restricciones PRIMARY KEY
- Restricciones FOREIGN KEY
- Restricciones UNIQUE
- Restricciones CHECK
- Restricciones DEFAULT

- La aceptación de NULL

Restricciones PRIMARY KEY:

Una tabla suele tener una columna o una combinación de columnas cuyos valores identifican de forma única cada fila de la tabla. Estas columnas se denominan claves principales de la tabla y exigen la integridad de entidad de la tabla.

Puede crear una tabla principal mediante la definición de una restricción PRIMARY KEY cuando cree o modifique una tabla. Una tabla sólo puede tener una restricción PRIMARY KEY, y ninguna columna a la que se le aplique una restricción PRIMARY KEY puede aceptar valores NULL. Dado que las restricciones PRIMARY KEY garantizan que los datos sean únicos, a menudo se definen para una columna de identidad.

Si especifica una restricción PRIMARY KEY en una tabla, SQL exige la exclusividad de los datos mediante la creación de un índice único para las columnas de clave principal. Este índice también permite un acceso rápido a los datos cuando se utiliza la clave principal en las consultas.

Si se define una restricción PRIMARY KEY para más de una columna, puede haber valores duplicados dentro de la misma columna pero cada combinación de valores de todas las columnas de la definición de la restricción PRIMARY KEY debe ser única. Tal como se muestra, las columnas *au_id* y *title_id* de la tabla *titleauthor* forma una restricción PRIMARY KEY compuesta para la tabla *titleauthor* que garantiza que la combinación de *au_id* y *title_id* sea única.

Restricciones FOREIGN KEY:

Una clave externa (FK) es una columna o combinación de columnas que se utiliza para establecer y exigir un vínculo entre los datos de dos tablas. Se crea un vínculo entre dos tablas al agregar en una tabla la columna o columnas que contienen los valores de la clave principal de la otra tabla. Esta columna se convierte en una clave externa para la segunda tabla.

Puede crear una clave externa mediante la definición de una restricción FOREIGN KEY cuando cree o modifique una tabla. Por ejemplo, la tabla *titles* de la base de datos *pubs* tiene un vínculo con la tabla *publishers* porque hay una relación lógica entre los libros y los editores. La columna *pub_id* de la tabla *titles* es la clave externa para la tabla *publishers*.

No es necesario que una restricción FOREIGN KEY está vinculada únicamente a una restricción PRIMARY KEY de otra tabla; también definirse para que hagan referencia las columnas de una restricción UNIQUE de otra tabla. Una restricción

FOREIGN KEY puede contener valores NULL, pero si alguna columna de una restricción FOREIGN KEY compuesta contiene valores NULL, se omitirá la comprobación de la restricción FOREIGN KEY.

Una restricción FOREIGN KEY puede hacer referencia a las columnas de tablas de la misma base de datos o columnas de una misma (tablas con referencia a sí mismas).

Aunque el propósito principal de una restricción FOREIGN KEY es el de controlar los datos que pueden almacenarse en la tabla de la clave externa, también controla los cambios realizados en los datos de tabla principal. Por ejemplo, si se elimina la fila de un publicador de la tabla *publishers* y el identificador de dicho publicador se utiliza para los libros que figura en la tabla *titles*, se ropmoe la integridad relacional entre ambas tablas: los libros del publicador eliminado quedarán sin correspondencia en la tabla *titles*, sin ningún vínculo con los datos de la tabla de la clave *publishers*. Con una restricción FOREIGN KEY se evita está situación. Está restricción exige la integridad referencial al asegurar que no se puedan realizar cambios en los datos e la tabla de la clave principal si esos cambios anulan el vínculo de los datos de la tabla de la clave externa. Si se intenta eliminar la fila de una tabla de la clave principal o cambia un valor de clave principal, la acción no progresará si el valor de la clave principal cambiado o eliminado corresponde a un valor de la restricción FOREIGN KEY de otra tabla. Para cambiar o eliminar una fila de restricción FOREIGN KEY, antes de eliminar los datos de la clave externa en la tabla de la clave externa o bien cambiar los datos de la clave externa en la tabla de la clave externa y vincular, de ese modo, la clave externa con otros datos de clave principal.

Una restricción FOREIGN KEY puede ser un índice adecuado por dos motivos:

- Los cambios en las restricciones PRIMARY KEY se comprueban con restricciones FOREIGN KEY en las tablas relacionadas.
- Las columnas de la clave externa suelen utilizarse en los criterios de combinación cuando los datos de las tablas relacionadas se combinan en consultas mediante la correspondencia de la columna o columnas de la restricción FOREIGN KEY de una tabla y la columna o columnas de la clave única o principal de la otra tabla. Un índice permite en Microsoft® SQL Server™ 2000 buscar rapidez datos relacionados en la tabla de clave externa. No obstante, no es necesario crear este índice. Pueden combinarse los datos de las dos tablas indica que éstas han sido optimizadas para su combinación en una consulta que utilice las claves como criterio.

Restricciones de integridad referencial en cascada:

Las restricciones de integridad referencial en cascada permiten definir las acciones que SQL lleva a cabo cuando un usuario intenta eliminar o actualizar una clave a la que apuntan las claves externas existentes. Las cláusulas REFERENCES de las instrucciones CREATE TABLE y ALTER TABLE admiten cláusulas ON DELETE CASCADE, CASCADE permite la eliminación o actualización de valores de clave en cascada en las tablas definidas para tener relaciones de claves externas que pueden volver a trazarse en la tabla en la que se realizan las modificaciones.

ON DELETE CASCADE específica que si se intenta eliminar una fila con una clave a la que hacen referencia claves externas de filas existentes en otras tablas, todas las filas que contienen dichas claves externas también se eliminan. Si las acciones referenciales en cascada se han definido también las tablas de destino, las acciones en cascada especificadas se toman también para ls filas eliminadas de dichas tablas.

Restricciones UNIQUE:

Puede utilizar restricciones UNIQUE para asegurar que no se escriban valores duplicados en columnas específicas que no formen parte de una clave principal. Aunque tanto una restricción UNIQUE como PRIMARY KEY exige que los elementos sean únicos, es preferible utilizar una restricción UNIQUE en vez de una restricción PRIMARY KEY cuando desee exigir la unidad de :

- Una columna o una combinación de columnas que no sean la clave principal en una tabla se pueden definir varias restricciones UNIQUE, pero sólo una restricción PRIMARY KEY.
- Una columna que acepte valores en NULL. Las restricciones UNIQUE pueden definirse en columnas que aceptan valores NULL, mientras que las restricciones PRIMARY KEY sólo pueden definirse en columnas que no aceptan valores NUL.

También es posible hacer referencia a una restricción UNIQUE con una restricción FOREIGN KEY.

Restricciones CHECK:

Las restricciones CHECK exigen la integridad del dominio mediante la limitación de los valores que pueden aceptar una columna. Son similares a las restricciones. FOREIGN KEY porque controlan los valores que se colocan en una columna. La diferencia escriba en la forma en que determinan los valores válidos: las restricciones FOREIGN KEY obtiene la lista de valores válidos de otra tabla, mientras que las restricciones CHECK determinan los valores válidos a partir de una expresión lógica que no se basa en datos de otra columna. Por ejemplo, es

posible limitar el intervalo de valores para una columna salary creando una restricción CHECK que sólo permite datos entre 15.000 y 100.000 dólares. De este modo se impide que se escriban salarios superiores al intervalo de salario normal. Puede crear una restricción CHECK con cualquier expresión lógica (booleana) que devuelva TRUE (verdadero) o FALSE (falso) basándose en operadores lógicos. Para el ejemplo anterior, la expresión lógica sería: salary >= 1500 AND salary <= 100000.

Es posible aplicar varias restricciones CHECK a una sola columna. Estas se evalúan en el orden en que fueron creadas. También es posible aplicar una sola restricción CHECK a varias columnas si se crea al nivel de la tabla. Por ejemplo, una restricción CHECK para varias columnas se puede utilizar para confirmar que cualquier fila con un valor USA en la columna *country* tiene también un valor de dos caracteres en la columna *state*. Así se pueden comprobar varias condiciones en un mismo sitio.

Definiciones DEFAULT:

Cada columna de un registro debe contener un valor, aunque sea un valor NULL. Puede haber situaciones en las que necesite cargar una fila de datos una tabla pero no conozca el valor de una columna o el valor de una columna o el valor ya no exista. Si la columna acepta valores NULL, puede cargar la fila con un valor NULL. Pero, dado que puede no resultar conveniente utilizar columnas que acepten valores NULL, una mejor solución podría ser establecer una definición DEFAULT para la columna siempre que sea necesario. Por ejemplo, es habitual especificar el valor cero como valor predeterminado para las columnas numéricas, o N/D (no disponibles) como valor predeterminado para las columnas de cadenas cuando no se especifica ningún valor. Al cargar una fila en una tabla con una definición DEFAULT para una columna, se indica implícitamente a SQL que cargue un valor predeterminado en la columna en la que no se haya especificado ningún valor. También puede indicarle explícitamente a SQL Server que inserte el valor predeterminado para la columna mediante la cláusula DEFAULT VALUES de la instrucción INSERT, DEFAULT ya ha sido analizada anteriormente.

Si una columna no acepta valores NULL y no tiene una definición DEFAULT, deberá especificar explícitamente un valor para la columna o SQL devolverá un error para indicar que la columna no permite valores NULL. El valor insertado en una columna que se define mediante la combinación de la definición DEFAULT, la aceptación de valores NULL de la columna y el valor insertado en la columna puede resumirse.

Permitir valores NULL

La aceptación de valores NULL de una columna determinada si las filas de una tabla pueden contener un valor NULL en esa columna. Un valor NULL, NULL, no es lo mismo que cero (0), en blanco o que una cadena de caracteres de longitud cero, como “”, NULL significa que no hay ninguna entrada. Su presencia suele implicar que el valor es desconocido o no definido. Por ejemplo, un valor NULL en la columna p*rice* que se encuentra en la tabla *titles* de la base de datos *pubs,* no significa que el libro no tenga precio; NULL significa que el precio se desconoce o que no ha sido especificado. En general, evite la aceptación de valores NULL dado que pueden implicar una mayor complejidad en las consultas y en las actualizaciones; para otra parte, hay otras opciones para las columnas, como las restricciones PRIMARY KEY, que no pueden utilizarse con columnas que aceptan valores NULL.

Si se inserta una fila pero no se incluye ningún valor para la columna que se permita valores NULL, SQL proporcionará el valor NULL (salvo que exista una definición u objeto DEFAULT). Una columna definida con la palabra clave NULL también acepta una entrada explícita de NULL por parte del usuario, independientemente del tipo de datos que se trate o de si tiene un valor predeterminado asociado. El valor NULL no debe ponerse entre comillas porque no sería interpretado como valor NULL, sino como la cadena de caracteres ‘NULL’.

Si se configura una columna de manera que no permita valores NULL, será más fácil mantener la integridad de los datos, ya que se asegura que una columna de una fila siempre contendrá datos. Si no se aceptan valores NULL, el usuario que escriba los datos en la tabla deberá especificar un valor para la columna, o la fila de la tabla no podrá ser aceptada n la base de datos.

Las columnas definidas con una restricción PRIMARY KEY o con una propiedad IDENTITY no puede aceptar valore NULL.

Todos los tipos de restricciones de integridad vistos hasta aquí son reglas que restringen el rango de valores válidos para una columna y que se colocan en una columna cuando se crea la tabla. Si la restricción de integridad afecta a una columna se denomina **restricción de columna**. Si la restricción de integridad afecta a varias columnas de la misma tabla se llama **restricción de tabla**. La sintaxis general de estos tipos de restricciones se presenta en los siguientes párrafos.

Restricción de columna

```
[CONSTRAINT restricción]
        {[NOT] NULL |
        [{UNIQUE | PRIMARY KEY}
```

```
  [USING INDEX
      [INITRANS entero]
      [MAXTRANS entero]
      [PCTFREE entero]
      [TABLESPACE entero]
      [STORAGE entero]
REFERENCES [usuario.] TABLA  [(columna ) ]  [ON DELETE CASCADE]
[CHECK (condición) ]
[EXCEPTIONS INTO [usuario.] TABLA ]
[DISABLE] )
```

La cláusula EXCEPTIONS INTO especifica una tabla en la que Oracle pone información sobre las filas que violan una restricción de integridad vigente. Esta tabla tiene que ser local.

La opción DISABLE permite desactivar la restricción de la integridad al crearla. Cuando la restricción está desactivada, Oracle no la mantiene automáticamente. Posteriormente, podemos activar la restricción con la cláusula ENABLE de CREATE o ALTER TABLE.

Restricción de tabla

```
[CONSTRAINT restricción]
    {[NOT] NULL |
    [{UNIQUE | PRIMARY KEY} (columna  [ ,columna] )
      [USING INDEX
          [INITRANS entero]
          [MAXTRANS entero]
          [PCTFREE entero]
          [TABLESPACE entero]
          [STORAGE entero]
    [FOREIGN KEY (columna [ ,columna])
      REFERENCES [usuario.] TABLA  [(columna  [ ,columna] ) ]
      [ON DELETE CASCADE] ]
    [CHECK (condición) ]
    [EXCEPTIONS INTO [usuario.] TABLA ]
    [DISABLE] )
```

Anidamiento:

La sentencia CREATE TABLE admite también condiciones de anidamiento mediante la sintaxis siguiente:

```
NESTED TABLE columna_de_anidamiento
[cláusula_columna_sustituible] STRORE AS tabla_de_almacenamiento
```

[((propiedades_obejto) [propiedades_físicas])]
[RETURN AS { LOCATOR | VALUE}]

La sintaxis cláusula_columna_sustituible puede tener la siguiente estructura:

[ELEMENT] IS OF [TYPE] ([ONLY] tipo)
[[NOT] SUBSTITUTABLE AT ALL LEVELS]

RETURN AS indica devolución de copia de la tabla anidada (con localizadores o sola).

Particionamiento:

La sentencia CREATE TABLE admite también condiciones de particionamiento en tablas. Una partición en una tabla se crea mediante las siguientes cláusulas:

{ rango de particionamiento
| situación del particionamiento
| composición del particionamiento]
| lista del particionamiento
}

La estructura del *rango* de particionamiento es la siguiente:

PARTITION BY RANGE (lista_columnas) (cláusula_valores_partición)
(cláusula_valores_subpartición)

La sintaxis *cláusula_valores_partición* puede tener la siguiente estructura:

PARTITION [partición] VALUES LESS THAN
({ value | MAXVALUE } [, { valor | MAXVALUE }] …)
descripción_tabla_partición
[, PARTITION [partition] VALUES LESS THAN
({ value | MAXVALUE } [, { valor | MAXVALUE }] …)
descripción_tabla_partición
] …

La *situación del particionamiento* tiene la siguiente estructura:

PARTITION BY HASH (columnn_list)
{ individual_hash_parititons | hash_partitions_by_quantity }

La sintaxis *individual_hash_partitions* tiene la siguiente estructura:

(**PARTITION** [partition hash_partitioning_storage_clause]
  [, **PARTITION** [partition hash_partitioning_storage_clause]] …
)

La sintaxis *hash_patitions_by_quantity* tiene la siguiente estructura:

PATITIONS cantidad [**STORE IN** (tablespace [, tablespace]…)]
[**OVERFLOW STORE IN** (TABLESPACE [, TABLESPACE] …)]

La *composición de particionamiento* tiene la siguiente estructura:

PARTITION BY RANGE (lista_columnas) [cláusula_subpartición]
(cláusula_valores__partición)
La *lista del particionamiento* tiene la siguiente estructura:

PARTITION BY LIST (columna)
(**PARTITION** [partición] **VALUES**
  ({ valor_partición | **NULL** } [, { valor_partición | **NULL**
}] …)
    descripción_tabla_partición
    [, **PARTITION** [partición] **VALUES**
      ({ valor_partición | **NULL** } [, { valor_partición | **NULL**
}] …)
    descripción_tabla_partición
  ] …
)

La sintaxis *descripción_tabla_partición* suele tener la siguiente estructura:

[segment_attributes_clause] [compression_clauses]
[**OVERFLOW** [segment_attributes_clause]]
[{ LOB_storage_clause | varray_storage_clause }
  [LOB_storage_clause | varray_storage_clause] …
]
 [partition_level_subpartitioning]

La sintaxis de *segment_atributes_clause* es la siguiente:

{ physical_atributes_clause
 | **TABLESPACE** TABLESPACE
 | { **LOGGIN** | **NOLOGGIN** }
 }
 [physical_atributes_clause
 | **TABLESPACE** tablespace

| { **LOGGIN** | **NOLOGGIN** }
] …

La sintaxis de *physical_attributes_clause* es la siguiente:

{ segment_attributes_clause
| **ORGANIZATION**
  { **HEAP** [segment_attributes_clause]
  | **INDEX** [segment_attributes_clause] index_org_table_clause
  | **EXTERNAL** external_table_clause
  }
| **CLUSTER** cluster (culmn [, column] …)
}
La sintaxis de LOB_*storage_clause* es la siguiente:

LOB
{ (LOB_item [, LOB_item] …) **STORE AS** (LOB_parameters)
| (LOB_item) **STORE AS**
  { LOB_segname (LOB_parameters)
  | LOB_segname
  | (LOB_parameters)
  }
}

La sintaxis de *compression_clauses* es la siguiente:

{ **COMPRESS** [integer] | **NOCOMPRESS** }
La sintaxis de *partition_level_subpartitioning* es la siguiente:
  SUBPARTITIONS quantity [**STORE IN** (TABLESPACE [, TABLESPACE] …)]
| (**SUBPARTITION** [subpartition hash_partitioning_storage_clause]
      [, **SUBPARTITION** [subpartition hash_partitioning_storagre_clause]
] …)

La sintaxis de *cláusula_partición* suele tener la siguiente estructura:

SUBPARTITION BY HASH (lista_columnas [, lista_columnas] …)
 [**SUBPARTITIONS** cantidad [**STORE IN** (tablespace [,
tablespace] …)]]

MODIFICACION DE TABLAS

Modificación de tablas con ALTER TABLE

Después de crear una tabla, puede cambiar muchas de las opciones que fueron definidas cuando se creó originalmente; por ejemplo, es posible:

- Agregar, modificar o eliminar columnas. Por ejemplo, se puede cambiar el nombre, la longitud, el tipo de datos, la precisión, la escala y la aceptación de valores NULL de la columna, aunque hay algunas restricciones.
- Agregar o eliminar restricciones PRIMARY KEY y FOREIGN KEY.
- Agregar o eliminar restricciones UNIQUE y CHECK, así como definiciones (y objetos) DEFAULT.
- Registrar una tabla y las columnas seleccionadas de una tabla para la indización de texto.
- Así mismo, puede cambiar el nombre o el propietario de una tabla. Cuando lo haga, también deberá cambiar el nombre de la tabla en todos los desencadenadores, procedimientos almacenados, secuencias de comandos de SQL u otros códigos de programación que utilicen el nombre o propietario anterior de la tabla.

Inicialmente la sintaxis de ALTER TABLE podría der la siguiente:

```
ALTER TABLE [usuario .] TABLA
{ Cláusulas_modificación_tabla
| Cláusulas_modificación_particionamiento
| Cláusulas_modificación_columnas
| Cláusulas_modificación_restricciones_integridad
| Propiedades_modificación_columna
| Cláusulas_modificación_tabla_externa
| Cláusulas_movimiento_tabla
}
[ { enable_disable_cláusula
  | { ENABLE | DISABLE } { TABLE LOCK | ALL TRIGGERS }
  }
  [ enable_disabler_clause
  | { ENABLE | DISABLE } { TABLE LOCK | ALL TRIGGERS }
  ] …
]
;
```

Modificación de tabla

Las cláusulas de modificación de tabla presentan la siguiente sintaxis general:

```
{ { cláusula_atributos_físicos
  | { LOGGING | NOLOGGING }
  | cláusula_grupos_redo_log_suplementarios
  | cláusula_asignación_extensión
  | cláusula_desasignación_no_uso
  | { CACHE | NOCACHE }
  | { MONITORING | NOMOTORING }
```

| **cláusula_actualizar_tabla**
| **cláusula_registros_por_bloque**
| **cláusula_paralelismo**
} …
| **RENAME TO nombre_nueva_tabla**
} [cláusula_cambio_índices_organizados_tabla]

CACHE sitúa los bloques de tabla recuperados en la posición de los usados más recientemente y NOCACHE hace lo contrario. MONOTORING se utiliza para activar estadísticas de la tabla y NOMONITORING hace lo contrario. RENAME TO cambia de nombre a la tabla.

La *cláusula_atributos_físicos* tienen una sintaxis similar a la que tenía en la sentencia CREATE TABLE:

[{ **PCTFREE** entero)
| **PCTUSED** entero
| **INITRANS** entero
| **MAXTRANS** entero
| cláusula_almacenamiento } …
]

La *cláusula_grupos_redo_log_suplementarios* sirve para añadir (ADD) o borrar (DROP) grupos redo log adicionales, y tiene las siguiente sintaxis:

{ **ADD SUPLEMENTAL LOG GROUP** log_group (column [, column] …) [**ALWAYS**]
| **DROP SUPPLEMENTAL LOG** GROUP log_group
}

La *cláusula_asignación_extensión* permite asignar un nuevo espacio adicional para la tabla, la partición o la subpartición, el segmento de desbordamiento de datos, el segmento de datos LOB o el índice LOB. El tamaño de la extensión se da en bytes, kilobytes (K) o megabytes (M). También se especifica el nombre del fichero y el número de la instancia de base de datos que recoge la extensión. La sintaxis es la siguiente:

ALLOCATE EXTENT
[({ **SIZE** bytes [K | M]
| **DATAFILE** 'fichero'
| **INSATANCE** entero
} …
)
]

La *cláusula_desagnación_no_uso* desasigna espacio que no se usa al final de la tabla, la partición o la subpartición, el segmento de desbordamiento de datos, el segmento de datos. LOB o el índice LOB, pudiendo mantener ocupado (KEEP) un cierto espacio. Su sintaxis es la siguiente:

DEALLOCATE UNUSED [**KEEP** integer [**K** | **M**]]

La *cláusula_actualizar_tabla* actualiza o no datos al tipo de su última versión. La sintaxis es la siguiente:

UPGRADE [[**NOT**] **INCLUDING DATA**]
[cláusula_almacenamiento_columna]
[cláusula_almacenamiento_partición]

La *cláusula_registros_por_bloque* se utiliza para manejar el número de registros que pueden ser almacenados en un bloque. La sintaxis es la siguiente:

{ **MINIMIZE** | **NOMINIMIZE** } **RECORDS_PER_BLOCK**

La *cláusula_paralelismo* se utiliza para habilitar o no el paralelismo. La sintaxis es la siguiente:

{ **NOPARALLEL** | **PARALLEL** [entero] }

La *cláusula_cambio_índicces_organizados_tabla* reúne las siguientes operaciones:

[{ cláusula_mapeado_tabla
  | **PCTHRESHOLD** entero
  | cláusula_comprensión
  } …
]
[cláusula_índices_organizados_desbordamiento]

La *cláusula_mapeado_tabla* tiene las siguiente sintaxis:

{ [{ { **MAPPING TABLE** | **NOMAPPING** }

La *cláusula_comprensión* tiene la siguiente sintaxis:

{ **COMPRESS** [entero] | **NOCOMPRESS** }
} …

La *cláusula_índices_organizados_desbordamiento* tiene la siguiente sintaxis:

[**INCLUDING** nombre_columna] **OVERFLOW** [cláusula_atributos_segmeto]

Modificación de columnas:

Las cláusulas de modificación de columna presenta la siguiente sintaxis general:

{ { ADD opciones_añadir_columna
| MODIFY opciones_modificar_columna
| cláusula_borrar_columna
} …
| MODIFY NESTED TABLE collection_item RETURN AS { LACATOR | VALUE }

La sintaxis *opciones_añadir_columna* tiene el siguiente desarrollo:

{ (tipo_datos_columna [**DEFAULT** expr] [columna_ref_restrcción]
[columna_restricción [columna_restricción] …])
| tabla_o_vista_restricción
| tabla_ref_restricción
} …
{ columna_propiedades [columna_propiedades] …
[(cláusula_almacenamiento_partición [, cláusula_almacenamiento_partición] …( ]
| modify_LOB_cláusula_almacenamiento [modify_LOB_cláusula_almacenamiento] …
| alter_varray_col_propiedades [alter_varray_col_propiedades] …
}

La sintaxis *opciones_modificar_columna* tiene el siguiente desarrollo:

(columna [tipo_datos] [**DEFAULT** expr] [columna_restricción
[columna_restricción] …]
)
[modify_LOB_cláusula_almacenamiento]

La sintaxis *cláusula_borrar_columna* tiene el siguiente desarrollo:

{ **SET UNUSED** { **COLUMN** columna | (columna [, columna] …)
[{ **CASCADE CONSTRAINTS** | **INVALIDATE** } [**CASCADE CONSTRAINTS** | **INVALIDATE**] …]
| **DROP** { **COLUMN** columna | (columna [, columna] …) }
[{ **CASCADE CONSTRAINTS** | **INVALIDATE** } [**CASCADE CONSTRAINTS** | **INVALIDATE**] …]
[**CHECHPOINT** entero]
| **DROP** { **UNUSED COLUMNS** | **COLUMNS CONTINUE** } [**CHECKPOINT** entero]
| **DROP** { **UNUSED COLUMNS** | **COLUMNS CONTINUE** } [**CHECKPOINT** entero]
}

Modificación de restricciones de integridad:

Las cláusulas de modificación de restricción de integridad presentan la siguiente sintaxis general:

MODIFY CONSTRAINT restricción restricción_estado
| **DROP**
{ { **PRIMARY KEY** | **UNIQUE** (column [, column] …) }
  [**CASCADE**] [{ **KEEP** | **DROP** } **INDEX**]
| **CRONSTAINT** restricción [**CASCADE**]
}
}

Modificación de propiedades de columnas:

Las cláusulas de modificación de propiedades de columnas presentan la siguiente sintaxis general:

columna_propiedades [columna_propiedades] …
[(cláusula_almacenamiento_partición
[, cláusula_almacenamiento_partición] …)]
| modify_LOB_storage_cláusula
[modify_LOB_storaage_cláusulas] …
| alter_varray_col_propiedades
[alter_varray_col_propiedades] …
}

La sintaxis *columna_propiedades* tiene el siguiente desarrollo:

{ **COLUMN** columna [**ELEMENT**] **IS OF** [**TYPE**] ([**ONLY**] tipo)
[[**NOT**] **SUBSTITUTABLE AT ALL LEVELS**]]
STORAGE AS LOB
{ LOB_segname (LOB_parámetros)
| LOB_segname
| (LOB_parámetros)
}
| **LOB**
{ (LOB_item [, LOB_item] …) **STORE AS** (LOB_parámetros)
| (LOB_item) **STORE AS**
  { LOB_segname (LOB_parámetros)
  | LOB_segname
  | (LOB_parámetros)
  }
}

| MXType_cláusula_almacenamiento

}

La sintaxis de *clásusula_almacenamiento_partición* se desarrolla como sigue:

PARTITION partición
{ LOB_storage_clause | varray_storage_clause }
[LOB_storage_clause | varray_storage_clause] …
[(**SUBPARTITION** subpartición
    { LOB_storage_clause | varray_storage_clause }
    [LOB_storage_clause | varray_storage_clause] …
)

]

La sintaxisde *modify_lob_storage_clause* se desarrolla como sigue:

MODIFY LOB (LOB_item) (modify_LOB_parámetros)

La sintaxis de *alter_varray_col_propiedades* se desarrolla como sigue:

MODIFY VARRAY varray_item (modify_LOB_storage_parámetros)

Modificación de tablas externas:

Las cláusulas de modificación de tablas externas presentan la siguiente sintaxis general:

ADD opciones_añadir_columna
| **MODIFY opciones_modificar_columna**
| **drop_columna_cláusula**
| **drop_restricción_cláusula**
| **parallel_cláusula**
| **external_data_propiedades**
| **REJECT LIMIT { entero** | **UNLIMITED }**
}

La sintaxis de *external_data_propiedades* se desarrolla como sigue:

DAFAULT DIRECTORY directorio
[**ACCESS PARAMETERS** { (opaque_format_spec) | **USING CLOB**
subquery }]
LOCATION ([directorio :] ‘location_specificador’ [,
[directorio :] ‘ location_specificador’] …)

Movimiento de tablas:

Las cláusulas de movimientos de tablas presentan la siguiente sintaxis general:

MOVE [ONLINE] [segment_attributes_cláusula]

[index_org_table_cláusula]
[{ LOB_storage_cláusula | varray_storage_cláusula }
[LOB_storage_cláusula | varray_storage_cláusula] …
]

La sintaxis *segment_attributes_cláusula* tiene el siguiente desarrollo:

{ phisycal_attributes_cláusula
| **TABLESPACE** tablespace
| { **LOGGING** | **NOLOGGING** }
}

La sintaxis *index_org_table_cláusule tiene el siguiente desarrollo:*

[{ mapping_table_cláusulas
| **PCTTHRESHOLD** entero
| compression_cláusulas
}
]
[index_organized_overflow_cláusula]

La sintaxis *LOB_storage_cláusula* tiene el siguiente desarrollo:

LOB
{ (LOB_item [, LOB_item] …) **STORE AS** (LOB_parámetros)
| (LOB_item) **STORE AS**
{ LOB_segname (LOB_parámetros)
| LOB_segname
| (LOB_parámetros)
}
}

Cáusulas ENABLE/DISABLE

Las cláusulas ENABLE/DISABLE presentan la siguiente sintaxis general:

ENABLE | DISABLE } [VALIDATE | NOVALIDATE]
{ UNIQUE (column [, column] …)
| PRIMARY KEY
| CONSTRAINT restricción
}
[using_index_cláusula] [exceptions_cláusula]
[CASCADE] [{ KEEP | DROP } INDEX]

Modificación del particionamiento:

Las cláusulas de modificación del particionamiento presentan la siguiente sintaxis general:

{ modify_table_default_atributos
| modify_table_partición
| modify_table_subpartición
| move_table_partción
| move_table_subpartición
| add _range_partición_cláusula
| add _hash_partición_cláusula
| add _list_partición_cláusula
| coalsece_table_partición
| drop_table_partición
| rename_table_partición
| truncate_table_partición
| split_table_partición
| merge_table_particiones
| exchange_table_partición
| row_movement_cláusula
}

La sintaxis de *modify_table_default_atributos* se desarrolla como sigue:

MODIFY DEFAILT ATTRIBUTES [**FOR PARTITION** partición]
[segment_attributes_cláusula]
[**PCTHRESHOLD** entero] [**COMPRESS** | **NOCOMPRESS**]
[overflow_atributos]
[{ **LOB**)LOB_item) | **VARRAY** varray } (LOB_paramentros)
  [{ **LOB**)LOB_item) | **VARRAY** varray } (LOB_paramentros)] …
]

La sintaxis de *modify_table_partición* se desarrolla como sigue:

MODIFY PARTITION partición
{ partición_attributos
| add_table_subpartición_cláusula
| **COALESCE** [**SUBPARITION**] [update_glabal_index_cláusula]
[parallel_cláusula]
| alter_table_mapping_cláusula
| { **ADD** | **DROP** } **VALUES** (partición_valor [,
Partición_valor] …)
| [**REBUILD**] **UNUSABLE LOCAL INDEXES**
}

La sintaxis de *add_table_subpartición* se desarrolla como sigue:

ADD SUPRATITION [subpartición [subpartición_atributos]]

[update_global_index_cláusula] [parallel_cláusula]

La sintaxis de *modify_table_subpartición* se desarrolla como sigue:

MODIFY SUBPARTITION subpartición
{ { { allocate_extenet_cláusula | deallocate_unused_cláusula }
  | { **LOB** LOB_item) | (**VARRAY** varray }
modify_LOB_storage_parámetros
      [**LOB** LOB_item) | (**VARRAY** varray }
modify_LOB_storage_parámetros] …
    }
| [**REBUILD**] **unusable local indexes**
}

La sintaxis de *move_table_partición* se desarrolla como sigue:

MODIFY PARTITION partición [**MAPPING TABLE**]
[table_partición_descripción]
[update_global_index_cláusula] [parallel_cláusula]

La sintaxis de *move_ttable_subpartición* se desarrolla como sigue:

MOVE SUBPARTITION subpartición subpartición_attributos
[update_global_index_cláusula] [parallel_cláusula]

La sintaxis de *add_range_partición* se desarrolla como sigue:

ADD PARTITION [partición] **VALUES LESS THAN**
({ valor | **MAXVALUE** } [, { valor | **MAXVALUE** }] …)
[table_partición_descripción]

La sintaxis de *add_hash_partición* se desarrolla como sigue:

ADD PARTITION [partición] hash_partitioning_storage_cláusula
[update_global_index_cláusula] [parallel_cláusula]

La sintaxis de *add_list_partición* se desarrolla como sigue:

ADD PARTITION [partición] **VALUES**
({ partition_value | **NULL** } [, { partition_value | **NULL** }] …
)
[table_partition_description]

La sintaxis de *drop_table_partición* se desarrolla como sigue:

DROP PARTITION partición [update_global_index_cláusula
[parallel_cláusula]]

La sintaxis de *rename_table_partición* se desarrolla como sigue:

RENAME { **PARTITION** | **SUBPARTITION** } current_name **TO** new_name

La sintaxis de *truncate_table_partición* se desarrolla como sigue:

TRUNCATE { **PARTITION** partició | **SUBPARTITION** subpartición }
[{ **DROP** | **REUSE** } STORAGE]
[update_global_index_cláusula [parallel_cláusula]]

La sintaxis de s*plit_table_partición* se desarrolla como sigue:

SPLIT PARTITION corrent_partición { **AT** | **VALUES** } (valor [, valor] …)
[**INTO** (**PARTITION** [partición] [table_partición_descripción]
, **PARTITION** [partición] [table_partición_descripción])]
[update_global_index_cláusula] [parallel_cláusula]

La sintaxis de *mega_table_partición* se desarrolla como sigue:

MEGA PARTITION partición_1 , partición_2
[**INTO** parición_spec] [update_global_index_cláusula]
[parallel_cláusula]

La sintaxis de *exchange_table_partición* se desarrolla como sigue:

EXCHANGE { **PARTITION** partición | **SUBPARTITION** subpartición }
WITH TALBE tabla [{ **INCLUDING** | **EXCLUDING** } **INDEXES**]
[{ **WITH** | **WITHOUT** } **VALIDATION**]
[exceptions_cláusula] [update_global_index_cláusula
[parellel_cláusula]]

La sintaxis de *coalesce_table_partición* se desarrolla como sigue:

COALESCE PARTITION [update_global_index_cláusula
[parallel_cláusula]]

BORRADO DE TABLAS

La sentencia DROP TALBE permite borrar tablas completas de la base de datos. Esta sentencia suprime la tabla valida los cambios pendientes en la base de datos. Únicamente un administrador de la base de datos (DBA) puede suprimir taclas de usuarios.

Al suprimir la tabla también se suprimen los índices y las concesiones asociadas a ella. Las visitas y sinónimos construidos sobre tablas suprimidas se marcan como inválidas y dejan de funcionar.

DROP TABLE [usuario .] tabla [**CASCADE CONSTRAINTS**] ;

La opción CASCADE CONSTRAINTS suprime todas las restricciones de integridad referencial que haga referencia a claves de la tabla suprimida.

Colocamos el comando DROP TABLE antes de que se cree cada tabla para que, en el caso de que encontremos un error en el script, podamos modificar el script de origen y volver a ejecutarlo de forma inmediata. Sabemos que la instrucción drop precedente suprimirá la tabla y todo su contenido.

- Un DROP TABLE es irreversible a nivel de transacción, pues ejecuta un COMMIT inmediato, además de forma similar a TRUNCATE no hace ninguna advertencia antes de ejecutarse.

RECUPERANDO UNA TABLA

- En Oracle 10g, podemos recuperar BD, y Tablas
- Cuando se borra una tabla, Oracle no reclama inmediatamente el espacio asociado a la misma, sino que coloca la tabla y cualquier objeto asociado a ella en la Papelera de Reciclaje.
- En caso de que la tabla se haya borrado por error, se podrá recuperar por medio del comando Flashback
- Si el espacio del área de resuperación es menor al tamaño de la tabla, no hace uso del área de recuperación y la tabla no será posible recuperarla.

FLASHBACK TABLE dept80 TO BEFORE DROP

RENOMBRAR

RENAME

Cambia el nombre a una tabla, vista, secuencia o sinónimo privado.
RENAME old TO new

Para columnas de tablas:

CREATE TABLE tmp AS SELECT MyColOld MyColNew, col2, col3 FROM MY_TABLE;
DROP TABLE MY_TABLE;

RENAME tmp TO MY_TABLE;

También se puede:

ALTER TABLE MODIFY COLUMN...
ALTER TABLE DROP COLUMN...
ALTER TABLE SET UNUSED COLUMN...

TRUNCAMIENTO

El truncamiento de una tabla hace que sus datos de fila dejen de estar disponibles y opcionalmente libera el espacio utilizado.

Los índices correspondientes se truncan.

TRUNCATE TABLE

- Elimina todas las filas de una tabla.
- Libera el espacio de almacenamiento utilizado por dicha tabla.
- No puede realizar rollback de la eliminación de filas si utiliza TRUNCATE.
- También puede eliminar filas utilizando la sentencia DELETE.
- No dispara los triggersDELETE.
- Si tienes llaves foráneas que hacen referencia a la tabla a truncar, marcara error y habrá que desactivar esos constraints.

CREACION DE INDICES

La sentencia CREATE TABLE admite también condiciones de utilización de índices mediante la sintaxis siguiente:

USING INDEX
[[esquema .] [indice)
| sentencia_creación_índice
| { **LOCAL** | índice_global_particiondo }
| { **PCTFREE** entero
| **INITRANS** entero
| **MAXTRANS** entero
| **TABLESPACE** tablespace
| storage_ clause
| **NOSORT**
| { **LOGGING** | **NOLOGGING** }
}

[**PCTFREE** entero
| **INITRANS** entero
| **MAXTRANS** entero
| **TABLESPACE** tablespace
| cláusula almacenamiento
| **NOSORT**
| { **LOGGING** | **NOLOGGING** }
] ….
]

La sintaxis *índice_global_particionado* tiene la siguiente estructura:

GLOBAL PARTITION BY RANGE (lista_columnas)
(cláusula_particionamiento_global)

La sintaxis *cláusula_particionamiento_global* tiene la siguiente estructura:

PARTITION [partición] **VALUES LESS THAN** (valor [, valor] …)
[cláusula_atributos_segmento]

ENABLE Y DISABLE en restricciones de integridad:

La cláusula ENABLE activa una restricción de integridad o un trigger (disparador) que se encuentra desactivado. Cuando se activa una restricción UNIQUE o PRIMARY KEY, Oracle construye un índice sobre las columna de la clave. La cláusula USING INDEX especifica los parámetros para el índice. Si activamos una restricción de integridad referencial, la clave externa también tiene que ser activada.

La cláusula EXCEPTIONS INTO especifica el nombre de una tabla de excepciones ya existente que tienen que almacenarse en modo local. Cuando Oracle detecta una violación de restricción de integridad, almacena la información en esa tabla.

La cláusula ALL TRIGGERS sólo puede introducirse en una sentencia ALTER TABLE y activa todos los disparadores asociados con la tabla.

La sintaxis es la siguiente:

ENABLE { { UNIQUE (columna [, columna] … |
PRIMARY KEY |
CONSTRAINT restricción }
[USING INDEX [INITRANS entero]

[MAXTRANS entero]
[PCTFREE entero]
[TABLESPACE entero]
[STORAGE entero]
[EXCEPTIONS INTO [usuario.] TABLA |
ALL TRIGGERS }

La cláusula DISABLE desactiva una restricción de integridad. Su sintaxis es la siguiente:

DISABLE { { UNIQUE (columna [, columna] … |
PRIMARY KEY |
CONSTRAINT restricción } [CASCADE] } |
ALL TRIGGERS }

UNIQUE desactiva una restricción de integridad predeterminada, PRIMARY KEY desactiva la restricción de clave primaria, CONSTRAINT desactiva la restricción de integridad nombrada. CASCADE desactiva también cualquier restricción de integridad que dependa de las restricciones especificadas.

La cláusula ALL TRIGGERS sólo puede introducirse en una sentencia ALTER TABLE y desactiva todos los disparadores asociados con la tabla.

TIPOS DE DATOS

Un tipo de dato es un atributo de una parte de los datos que indican algo sobre la clase de datos sobre los que se va a procesar.

Esto incluye imponer restricciones en los datos, como qué valores pueden tomar y qué operaciones se pueden realizar.

El tipo de datos indica a la base de datos qué clase de información puede contener una determinada columna.

Cada tipo de datos tiene asociado un conjunto de reglas y el motor de la base de datos de Oracle.

Valor Not null

En términos sencillos, un valor nulo es una columna que no contiene datos; en otras palabras, que no contiene nada.

Es recomendable crear columnas de la clave primaria con la cláusula not null. Es obligatorio especificar, antes de introducir una fila de datos, una entrada válida para todas las columnas de la tabla así etiquetadas o, de lo contrario, la base de datos rechazará el registro.

ALFANUMERICOS

- VARCHAR2
- NVARCHAR2
    - Igual varchar2 pero permite caracteres unicade
    - Almacena datos de tipo carácter de longitud variable
- CHAR
    - Almacena datos de tipo carácter de longitud fija

NUMERICOS

- NUMBER (l,d)
    - Almacena datos numéricos, siendo *l* la longitud y *d* el número de dígitos decimales
- FLOAT
    - Es la versión ANSI de number
- INTEGER
    - Equivale a number con cero decimales

FECHAS

- DATE
    - Almacena fechas
    - Siglo, año, mes, día, hora, minutos y segundos
- TIMESTAMP
    - Igual a date pero una precesión de 9 decimales en los segundos
- TIMESTAMP WITH TIMEZONE
- Interval Year to Month
- Interval Day to Second

OBJETOS GRANDES

- CLOB
    - Tamaño prácticamente ilimitado de caracteres
- NCLOB
- BLOB
    - Similar a Clob pero con datos binarios

    - Almacena hasta 4 GB de datos binarios(como textos, imágenes gráficas, clips de video y sonido) en la base de datos
    - Oracle recomienda utilizar siempre tipos de datos LOB(BLOB, CLOB, NCLOB y BFILE), en lugar de tipos de datos LONG
- BFILE
    - Puntero a un archivo almacenado en el SO
- LONG
    - Almacena datos de tipo carácter de longitud variable
    - La columnas definidas como LONG puede almacenar datos de tipo carácter de longitud variable que contenga hasta 2 GB de información
    - Obsoleto, sustituido por CLOB

ROWID

Un valor codificado de 64 bits, que es un puntero para localizar una fila en una tabla.

Es propiedad de Oracle, y no es normalmente visible.

TIPOS MÁS UTILIZADOS

- VARCHAR2
    - Debe indicarse la longitud máxima
- NUMBER
    - Puede indicarse opcionalmente la precisión (número max de dígitos) y la escala (max de decimales)
- DATE
    - Almacena fechas

CONSTRAINS

Limitaciones o medios por el cual podemos aplicar reglas de negocio a las tablas.

TIPOS DE CONSTRAINTS

- UNIQUE
    - Establece un conjunto de campos de forma única.
- NOT NULL
    - Obliga a que dicho campo contenga un valor.
- PRIMARY KEY
    - Es la mejor de las llaves candidatas y nos sirve para identificar de forma única una tulpa.
- FOREING KEY

- Es definida en la tabla hija, y representa una relación padre-hijo, los campos deben ser del mismo tipo en conjunto. Se aplican reglas de Eliminación y Actualización.

- CHECK
  - Sirve para aplicar una regla simple a un campo, le regla se verifica antes de la inserción o actualización de un valor.

COMENTARIOS

Pone un comentario en le diccionario de datos.

Sintaxis para tablas y vistas:

```
COMMENT ON TABLE [esquema.]tabla IS 'comentario';
COMMENT ON TABLE [esquema.]vista IS 'comentario';
COMMENT ON TABLE [esquema.]vista_materilizada IS 'comentario';
```

Los comentarios se pueden visualizar a través de las vistas del diccionario de datos:
– ALL_COL_COMMENTS
– USER_COL_COMMENTS
– ALL_TAB_COMMENTS
– USER_TAB_COMMENTS

Sintaxis para columnas:

```
COMMENT ON COLUMN [esquema.]tabla.columna IS 'comentario';
COMMENT ON COLUMN [esquema.]vista.columna IS 'comentario';
COMMENT ON COLUMN [esquema.]vista_materilizada.columna IS 'comentario';
```

CREAR Y ADMINISTRAR SENTENCIAS

Una secuencia es un objeto que genera una serie de enteros únicos y que es apropiada para tablas que usan columnas simples numéricas como claves. Cuando una aplicación inserta un fila nueva en una tabla, la aplicación simplemente solicita una secuencia de base de datos para proporcionar el siguiente valor disponible en la secuencia para el valor de la clave principal de la nueva fila.

Para crear un secuencia se utiliza el comando SQL siguiente:

```
CREATE SEQUENCE [usuario . ] sequence
[ { { INCREMENT BY | START WITH } entero
  | { MAXVALUE integer | NOMAXVALUE }
```

```
  | { MINVALUE integer | NOMINVALUE }
  | { CYCLE | NOCYCLE }
  | { CACHE integer | NOCACHE }
  | { ORDER | NOORDER }
  }
]
;
```

INCREMENT BY indica la amplitud de las secuencia, START WITH indica el número con que comienza la secuencia. MINVALUE y MAXVALUE son los números más alto y más bajo que genera la secuencia. Para reanudar una secuencia donde comenzó se utiliza CYCLE. CAHCE permite guardar en memoria un conjunto presagiado de números de secuencia. ORDER asigna los números de secuencia en orden de peticiones.

DDL en SQL

Las sentencias SQL pertenecen a dos categorías principales: Lenguaje de Definición de Datos, DDL y Lenguaje de Manipulación de Datos, DML. Estos dos lenguajes no son lenguajes en sí mismos, sino que es una forma de clasificar las sentencias de lenguaje SQL en función de su cometido. La diferencia principal reside en que el DDL crea objetos en la base de datos y sus efectos se pueden ver en el diccionario de la base de datos; mientras que el DML es el que permite consultar, insertar, modificar y eliminar la información almacenada en los objetos de la base de datos.

Cuando se ejecutan las sentencias DDL de SQL, el SGBD confirma la transacción actual antes y después de cada una de las sentencias DDL. En cambio, las sentencias DML no llevan implícito el commit y se pueden deshacer. Existe pues un problema al mezclar sentencias DML con DDL, ya que estas últimas pueden confirmar las primeras de manera involuntaria e implicita, lo que en ocasiones puede ser un problema.

TIPOS DE OBJETOS EN SQL

A continuación se presenta una tabla con las sentencias SQL más comunes, clasificadas según el lenguaje al que pertenecen.

Sentencia DDL	***Objetivo***
Alter procedure	Recompilar un procedimiento almacenado.
Alter Table	Añadir o redefinir una columna, modificar la asignación de almacenamiento.

Analyze	Recoger estadísticas de rendimiento sobre los objetos de la BD para utilizarlas en el optimizador basado en costes.
Create Table	Crear una tabla.
Create Index	Crear un índice.
Drop Table	Eliminar una tabla.
Drop Index	Eliminar un índice.
Grant	Conceder privilegios o papeles, roles, a un usuario o a otro rol.
Truncate	Eliminar todas las filas de una tabla.
Revoke	Retirar los privilegios de un usuario o rol de la base de datos.

Marco Metodológico

En esta parte se podrá observar todas las pruebas que se realizaron de acuerdo a lo mencionado anteriormente:

Como primer ejemplo muy sencillo creamos la tabla OFICINAS como sigue, con sus respectivos tipos de datos de acuerdo a los que necesite, también se puede observar los datos nulos. En la figura 2, se puede realizar un *des* para comprobar que campos contiene la tabla creada y así verificar de esta forma la forma en que fueron creados los campos.

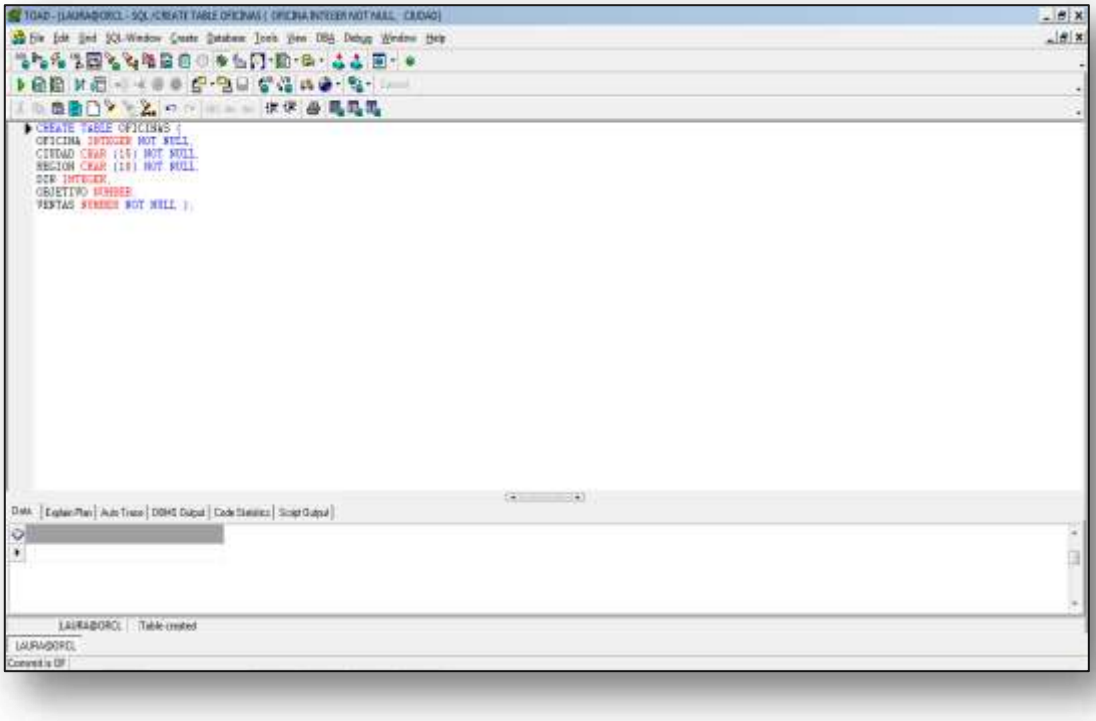


Figura 1. Creación de Tabla Oficinas

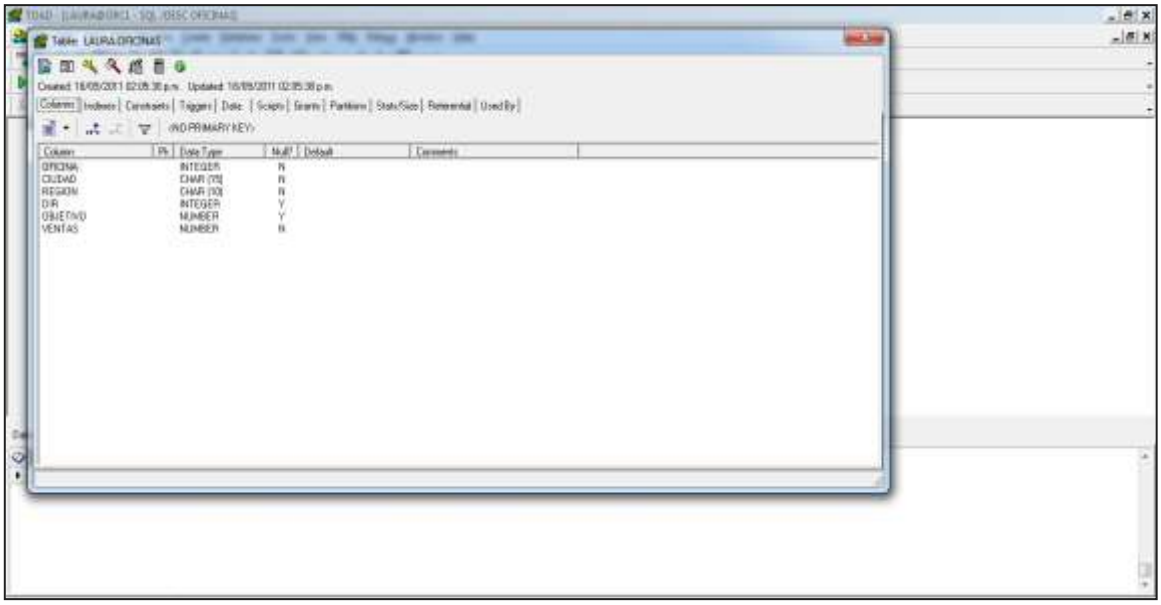


Figura 2. Describiendo tabla de Oficinas

En el ejemplo siguiente se crea la tabla *employees* utilizando distintas restricciones de integridad, como se puede observar que esta la restricción de PRIMARY KEY, UNIQUE. Confirmar campos de tabla EMPLOYEES.

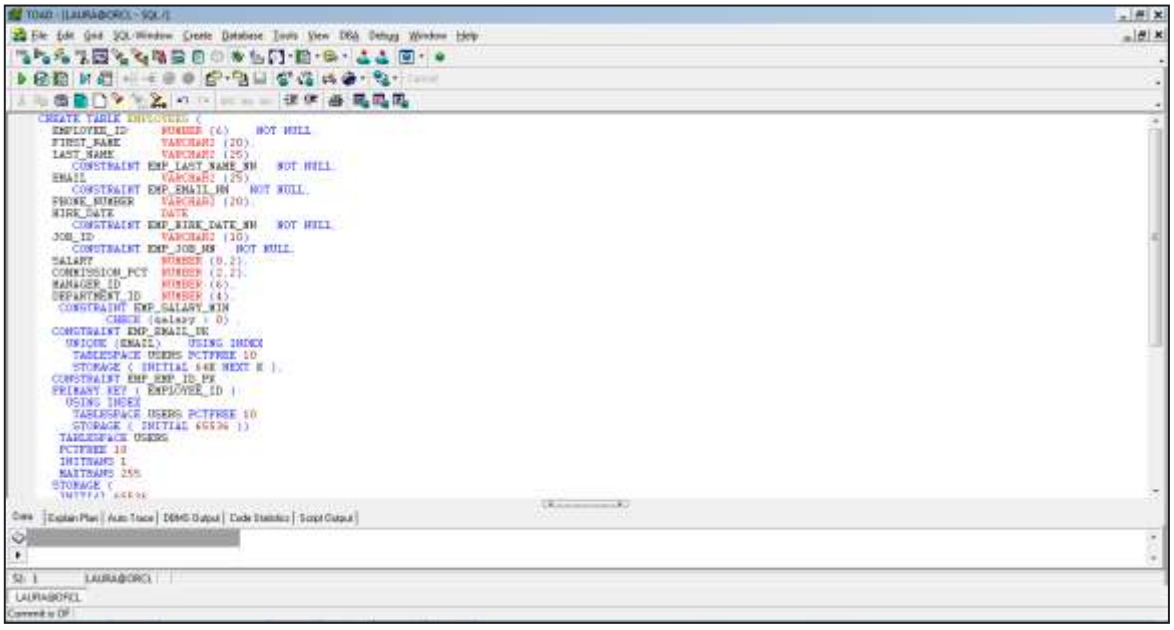


Figura 3. Creación de tabla *employees*.

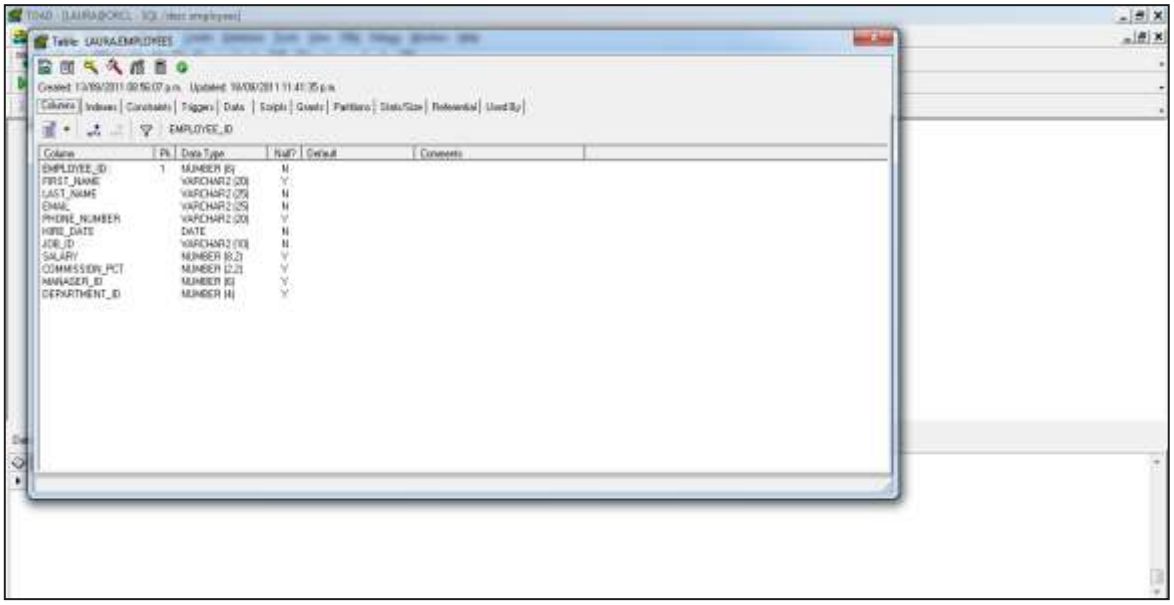


Figura 4. Descripción de tabla *employees*.

En este ejemplo se crea la tabla DEPT1 con algunas restricciones.

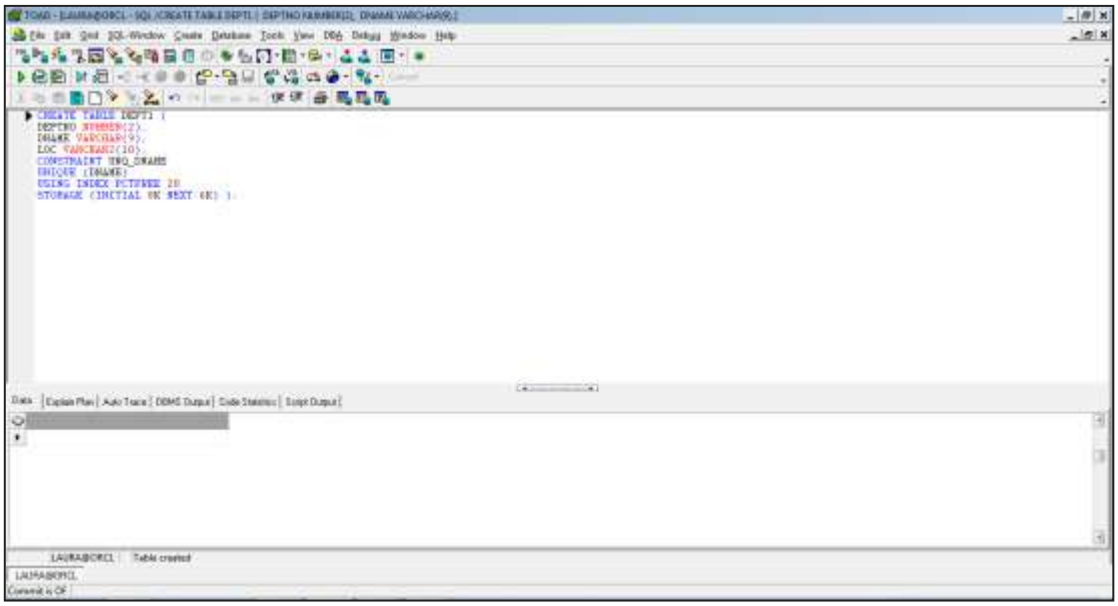


Figura 5. Creación de tabla de Dept1

En el ejemplo se crea una tabla DEPT2 con la restricción PRIMARY KEY.

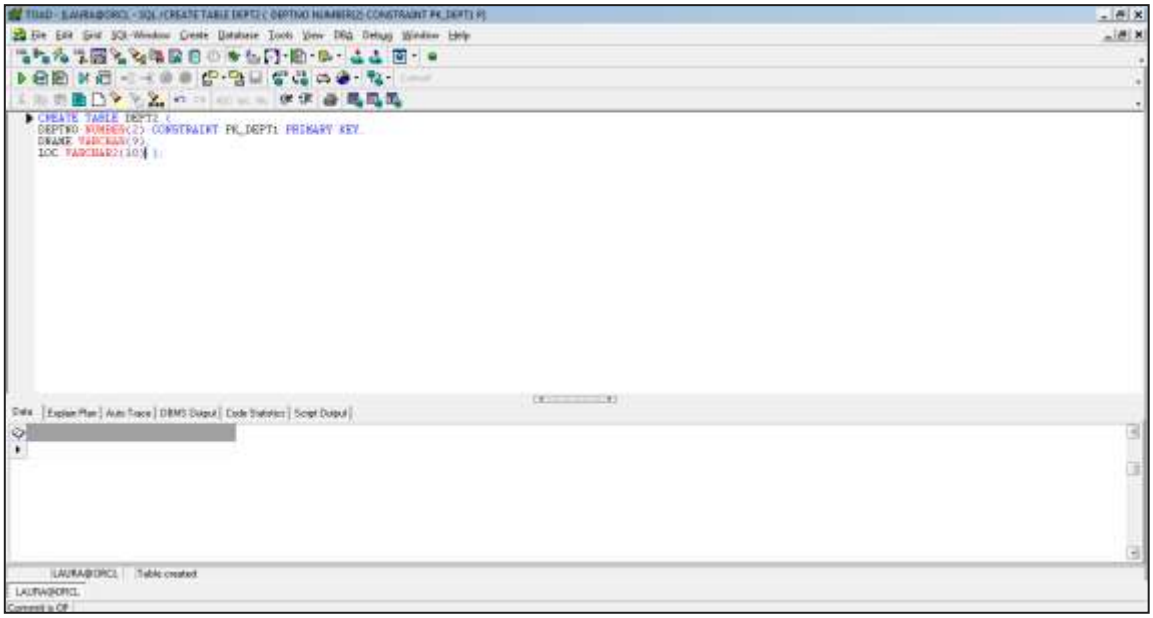


Figura 6. Creación de tabla Dept2.

Confirmación de campos agregados a la tabla DEPT2.

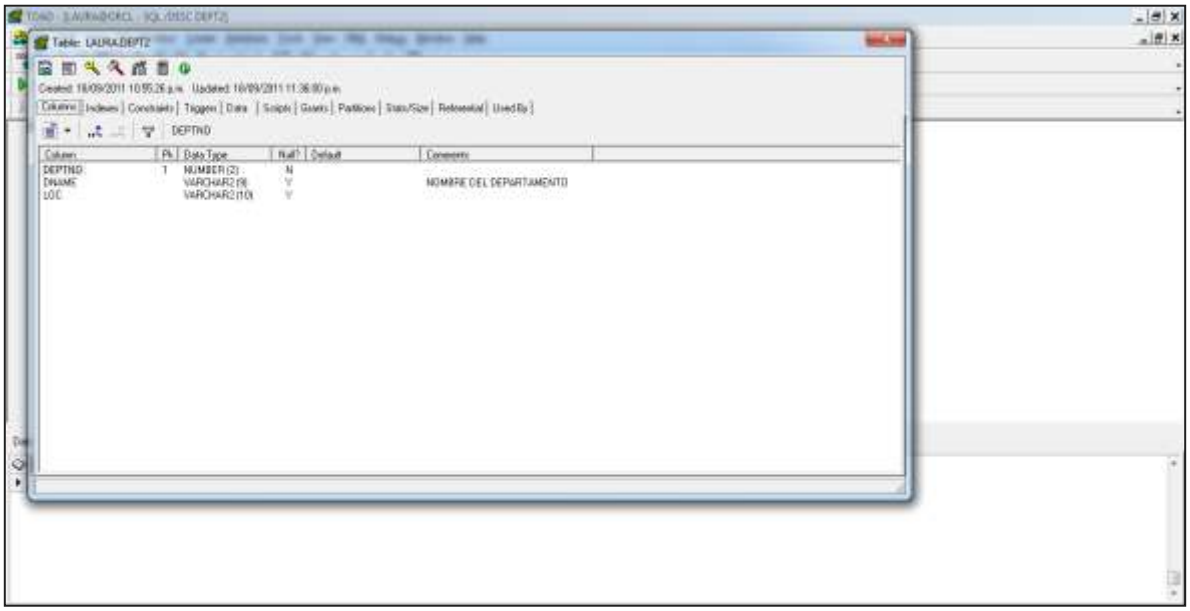


Figura 7. Descripción de tabla DEPT2.

En esta tabla EMP3 se creara con los mismos campos que la tabla EMP de modo que la columna *deptno* tenga una clave externa de nombre *fk_deptno*.

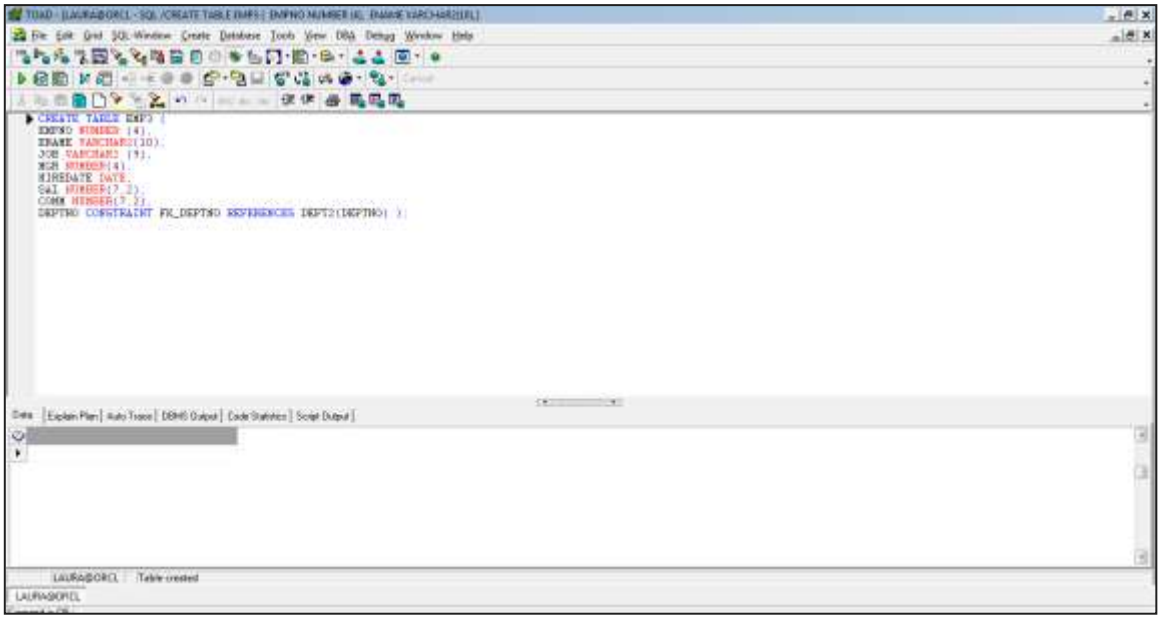


Figura 8. Creación de tabla EMP3.

Confirmación de campos en la tabla EMP3.

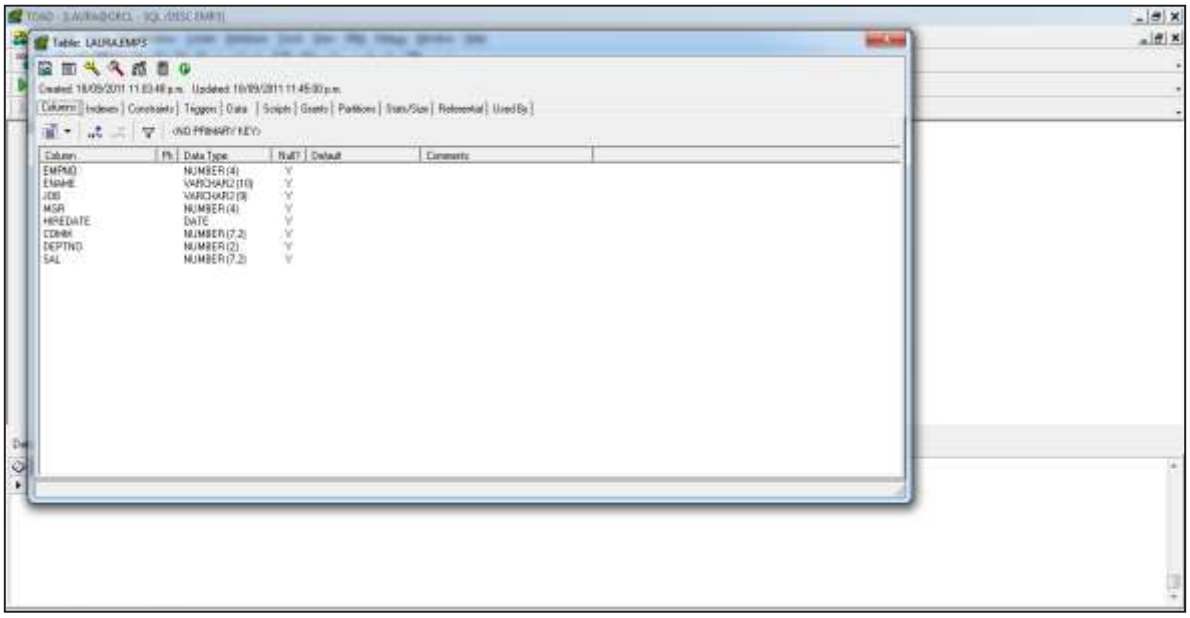


Figura 9. Descripción de tabla EMP3.

En el ejemplo siguiente se crea la tabla PROBLEMA situado como clave única dos columnas simultáneamente, la columna *ciudad* y la columna *fechamuestra*.

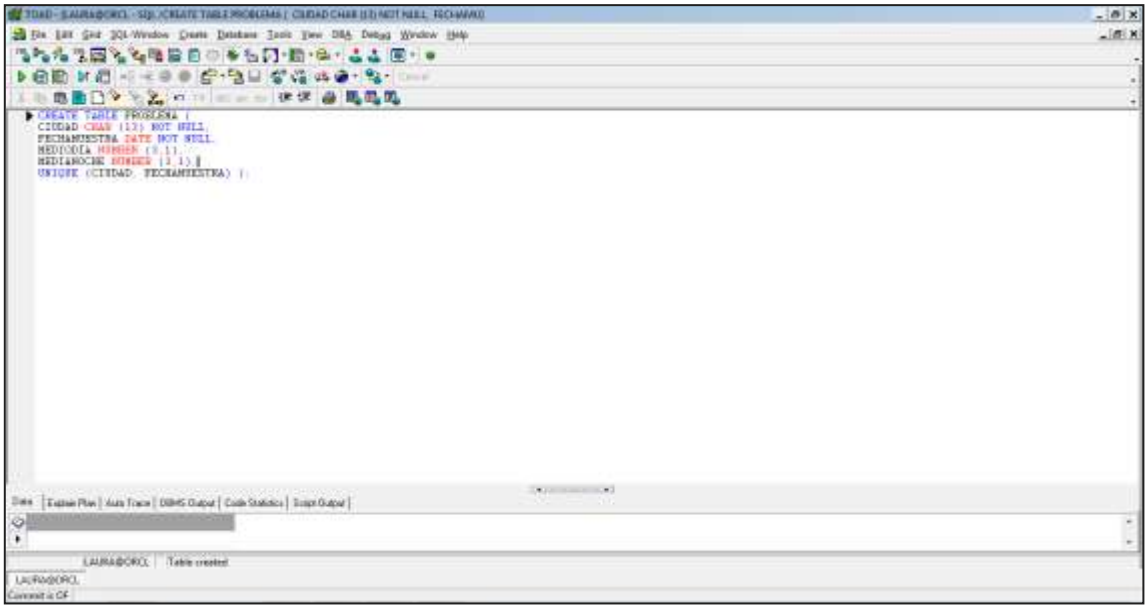


Figura 10. Creación de tabla PROBLEMA.

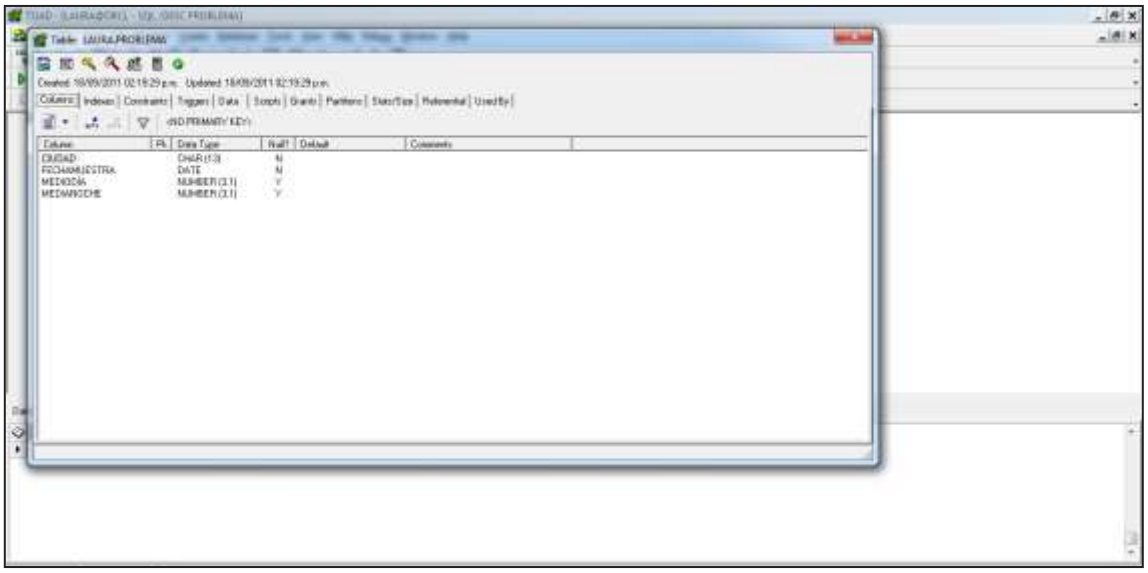


Figura 11. Descripción de tabla PROBLEMA.

También podíamos haber situado las dos columnas anteriores como clave única mediante la consulta que se presenta a continuación.

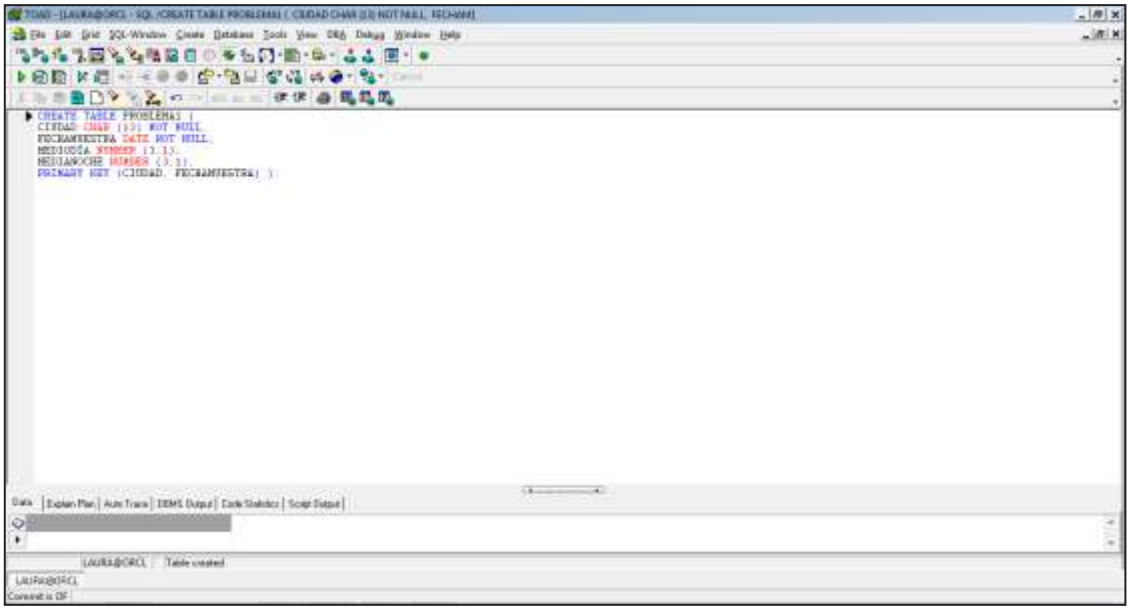


Figura 12. Creación de tabla PROBLEMA1.

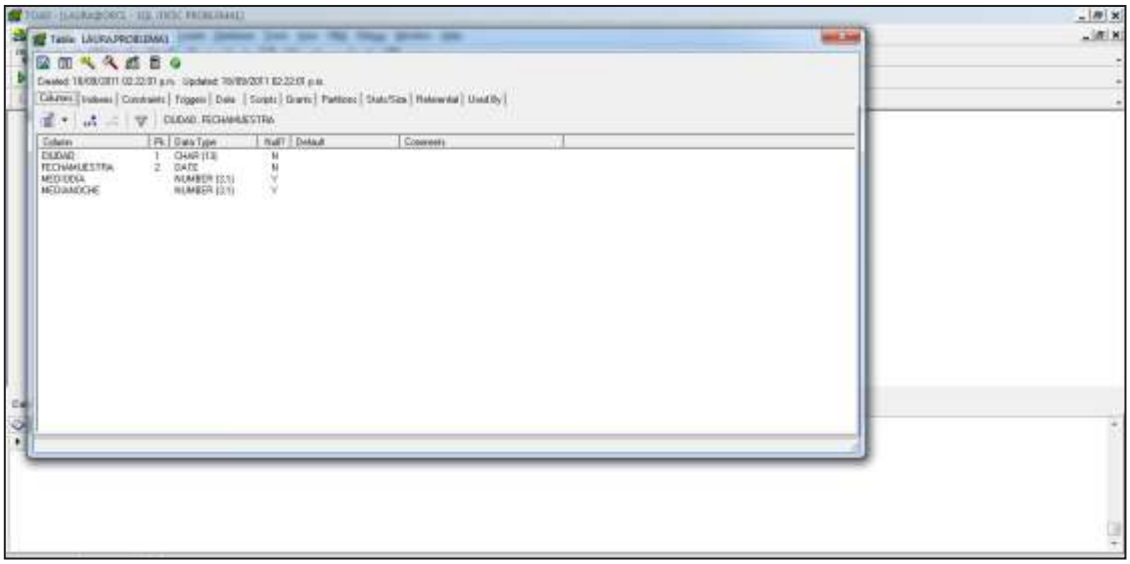


Figura 13. Descripción de tabla Problema1.

Si la clave primaria o la clave única consta de una sola columna, se sitúa las restricción de integridad a continuación de la definición de la columna tal y como se muestra en los ejemplos que se ven a continuación.

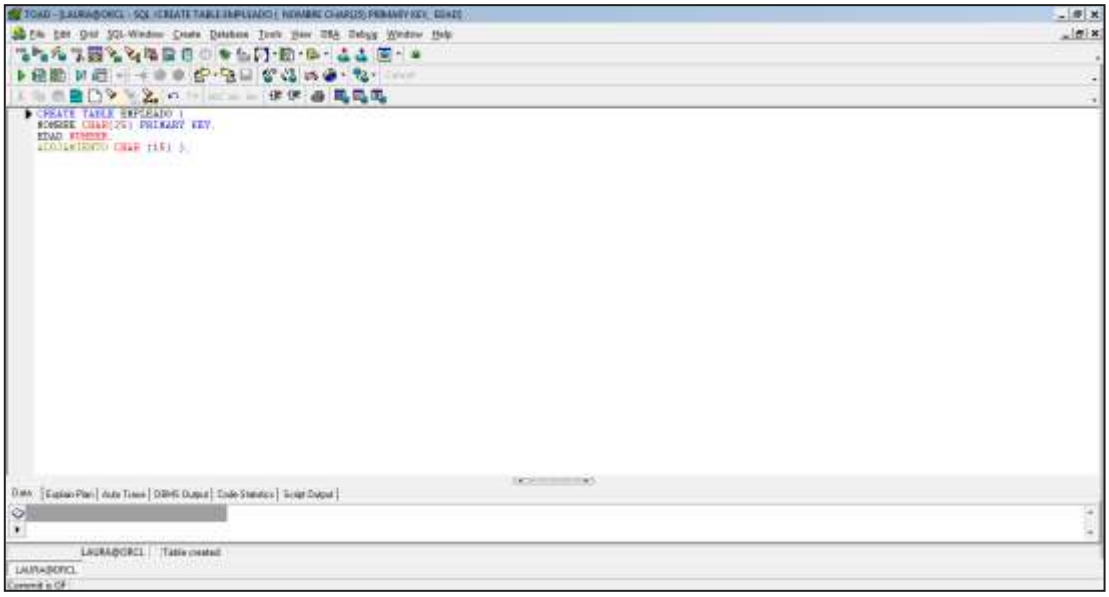


Figura 14. Creación de tabla EMPLEADO.

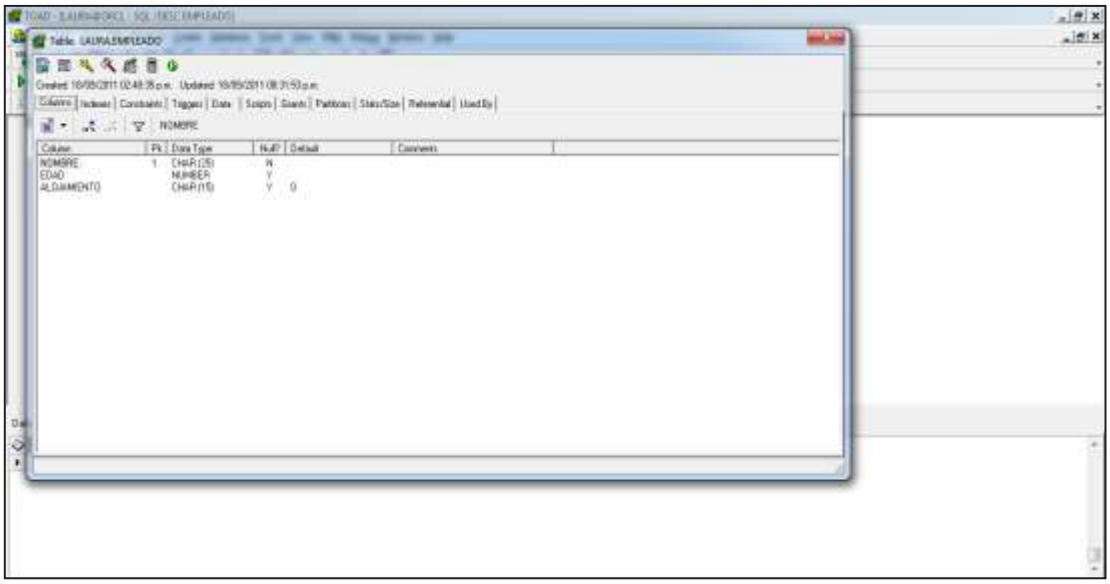


Figura 15. Descripción de tabla EMPLEADO.

La sentencia siguiente crea la tabla *departments* con una restricción NOT NULL y la sitúa en estado de validación ENABLE.

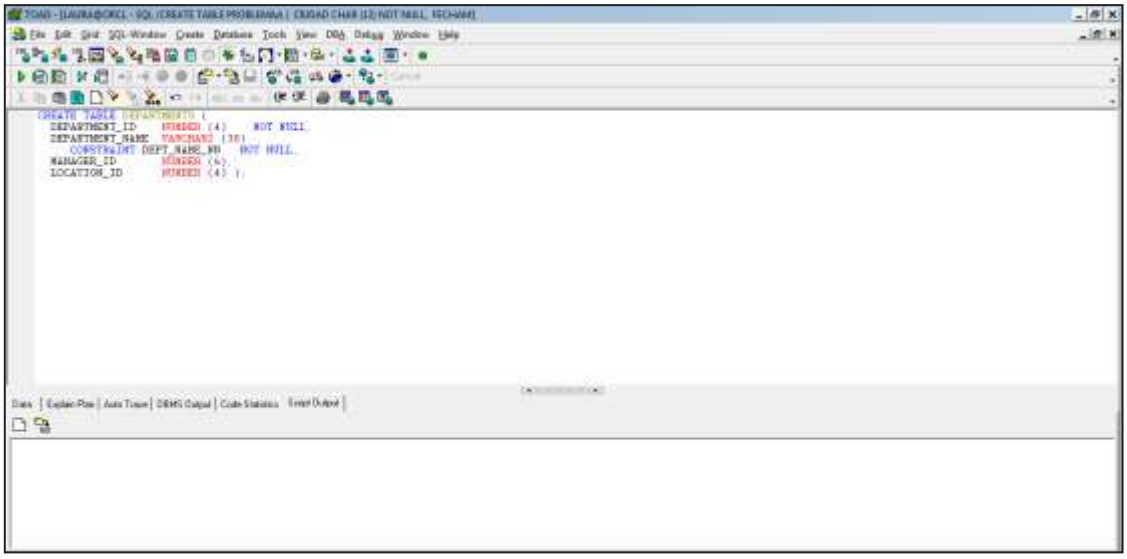


Figura 16. Declarando una restricción a la tabla DEPARTMENS.

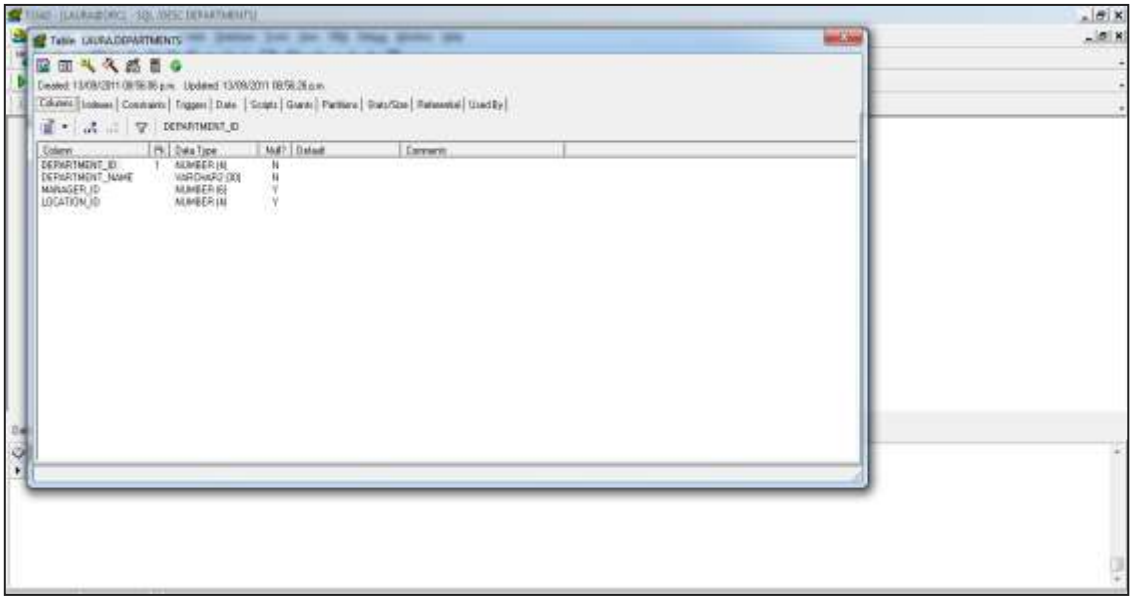


Figura 17. Descripción de tabla DEPARTMENTS.

La secuencia siguiente crea la tabla *departments* con una restricción PRIMARY KEY y la sitúa en estado de validación DISABLE.

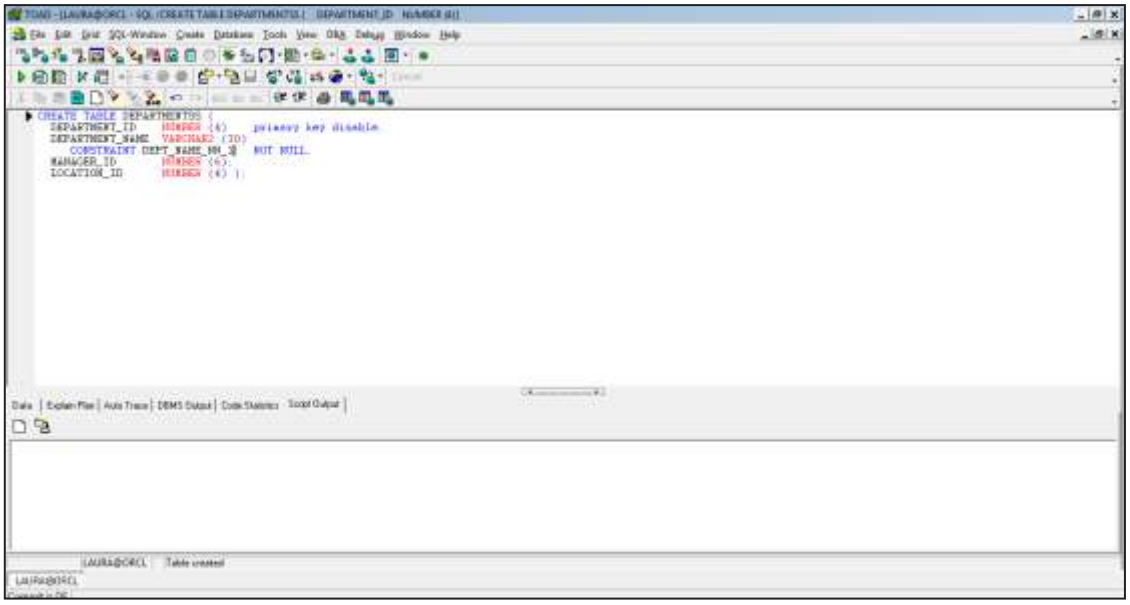


Figura 18. Declarando restricción PRIMARY KEY y un DISABLE a tabla DEPARTMENTS.

La consulta siguiente crea la tabla *dept_80* utilizando un número óptimo de servidores de ejecución en paralelo para verla y obteniendo los datos de la tabla *employees* mediante la cláusula AS y una consulta.

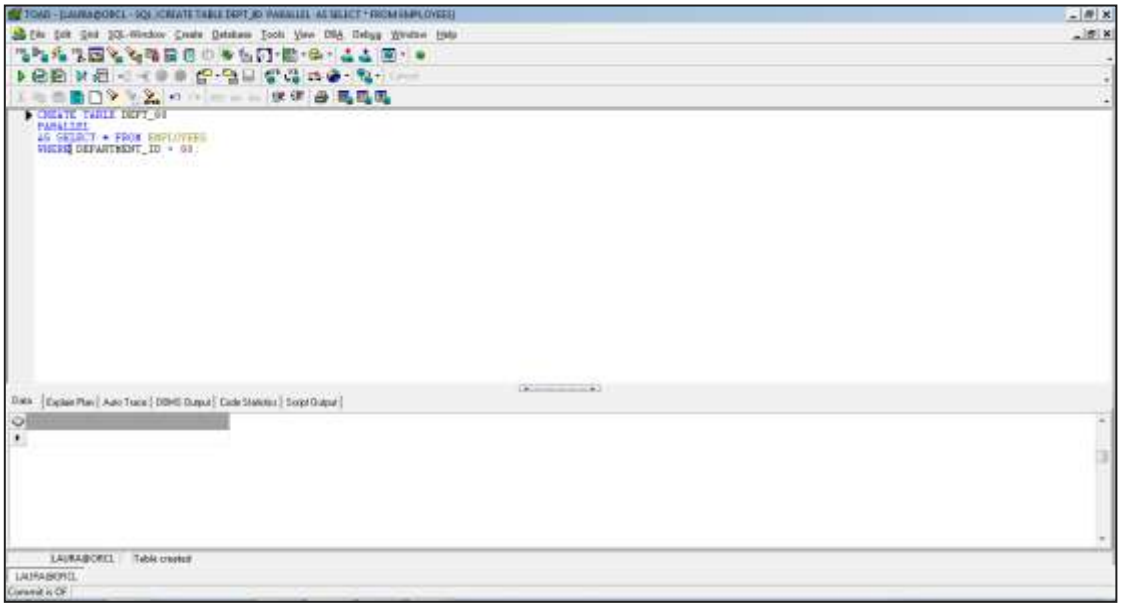


Figura 19. Utilización de ejecución PARALLEL de tabla EMPLOYEES a tabla DEPT_80.

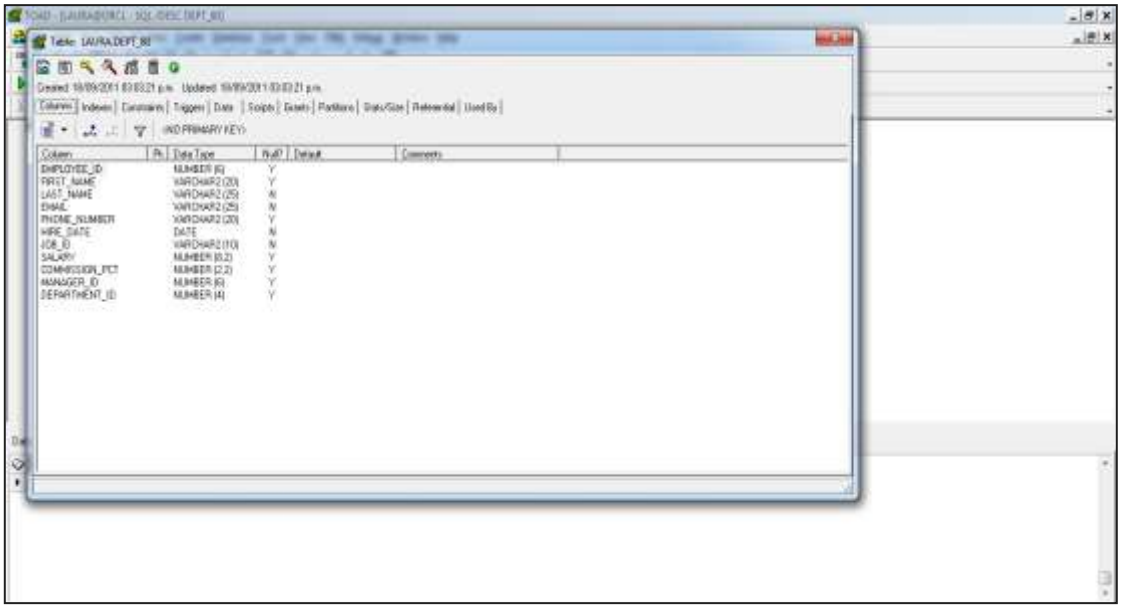


Figura 20. Verificación con una descripción de la tabla EMPLOYEES a la tabla DEPT_80.

La consulta siguiente crea la misma tabla anterior pero sólo para la consulta en serie, sin posibilidad de paralelismo.

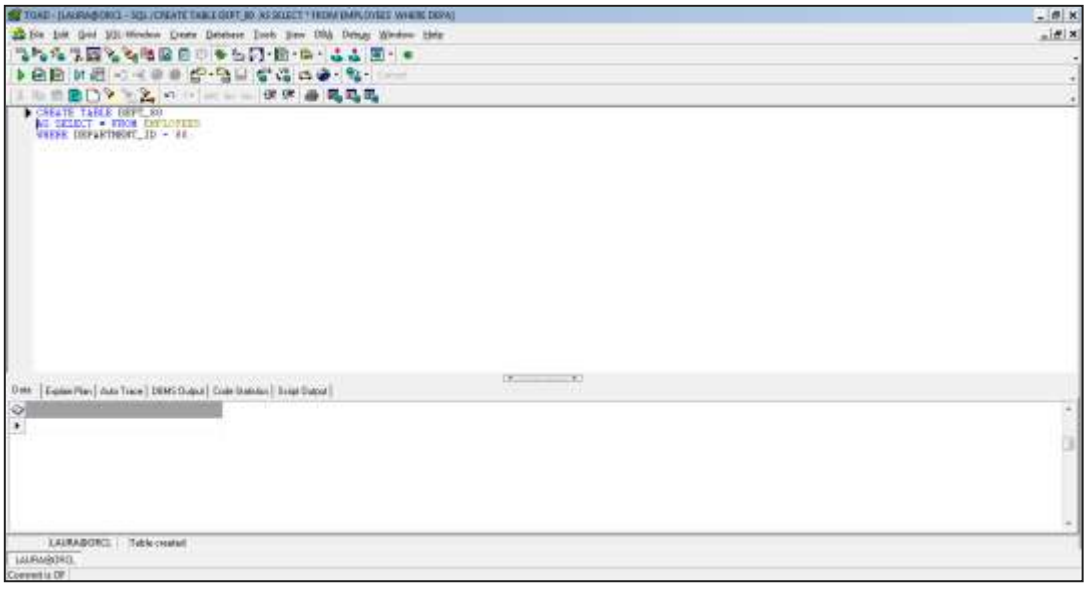


Figura 21. Creación de tabla DEPT_80 de EMPLOYEES.

ANIDAMIENTO:

La consulta siguiente crea la tabla *print_media* con condición de anidamiento mediante la columna *add_textdocs_ntab* y con devolución de copia de la tabla anidada.

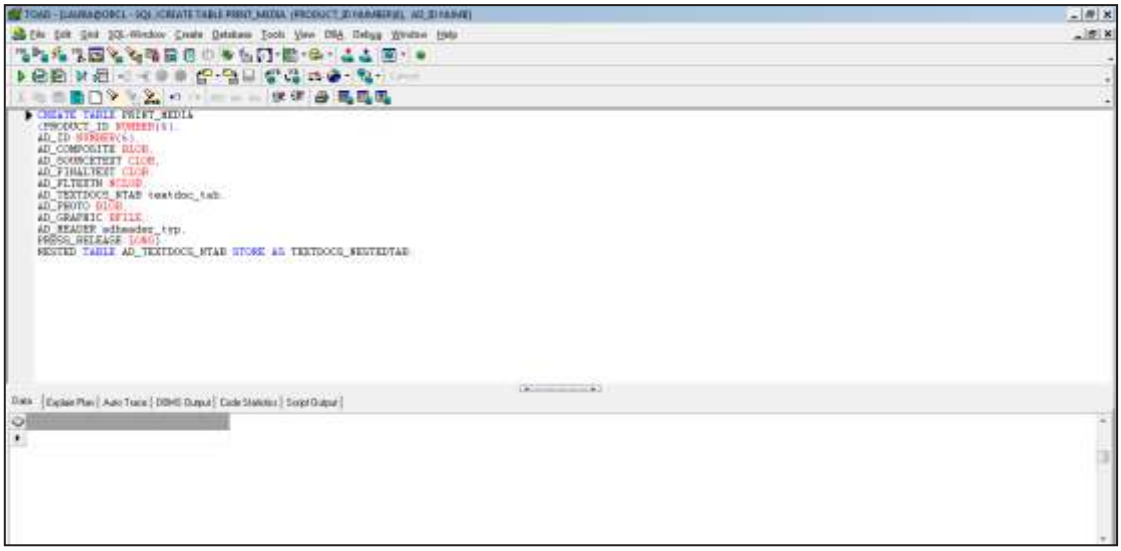


Figura 22. Creación de tabla PRINT_MEDIA mediante una anudamiento a columnas.

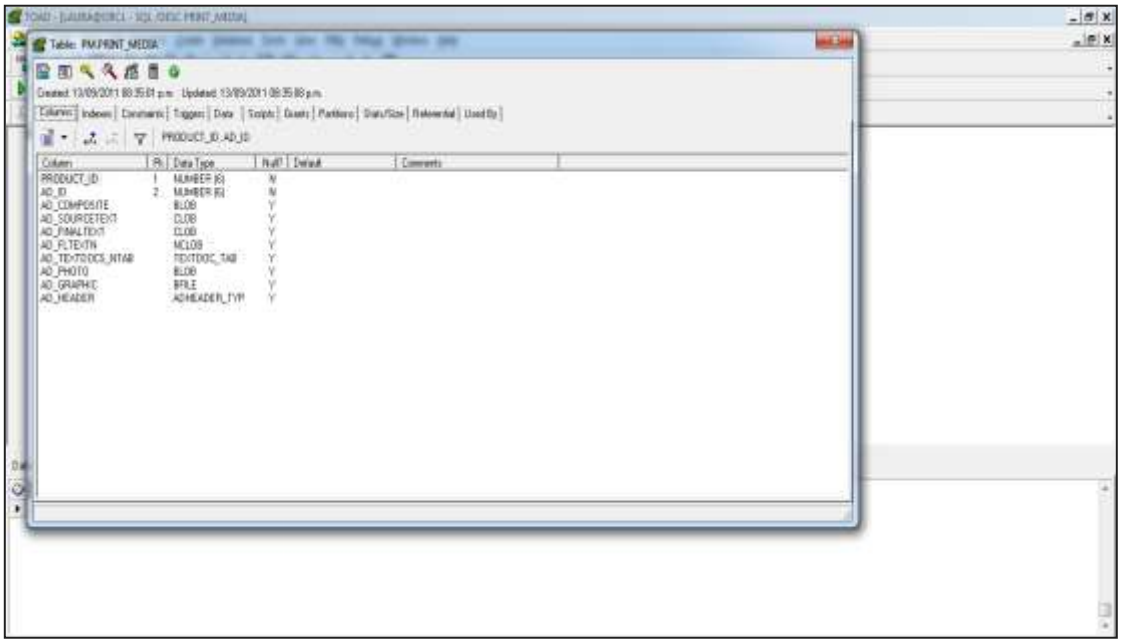


Figura 23. Descripción de tabla PRINT_MEDIA con el anidamiento.

La consulta siguiente crea la tabla anterior añadiendo algunas condiciones LOB de almacenamiento.

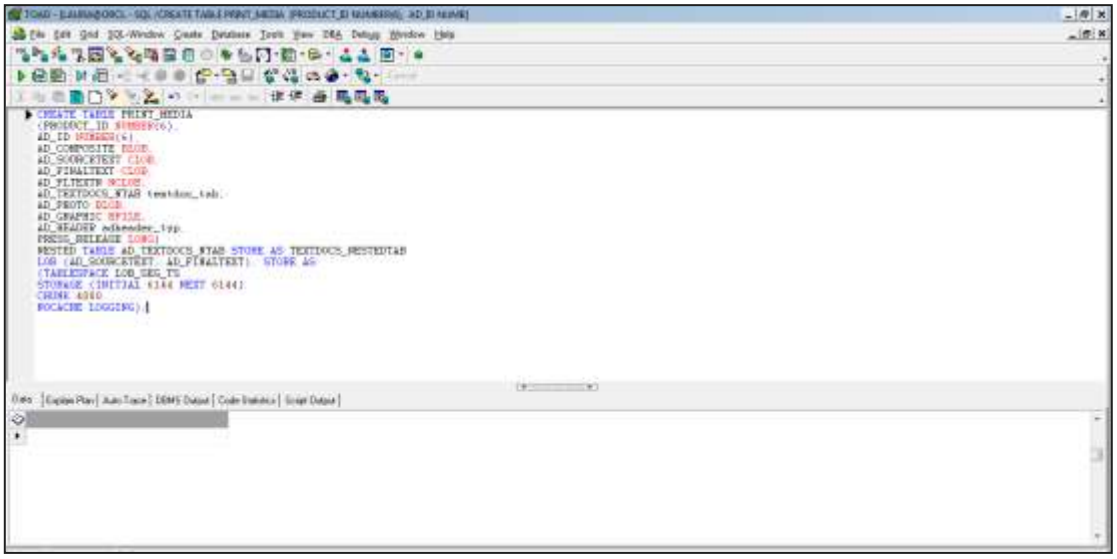


Figura 24. Creación de tabla PRINT_MEDIA con condición de almacenamiento LOB.

PARTICIONAMIENTO:

A continuación, se particiona la tabla *product_information* por el campo *product_id* almacenado en 5 tablespace.

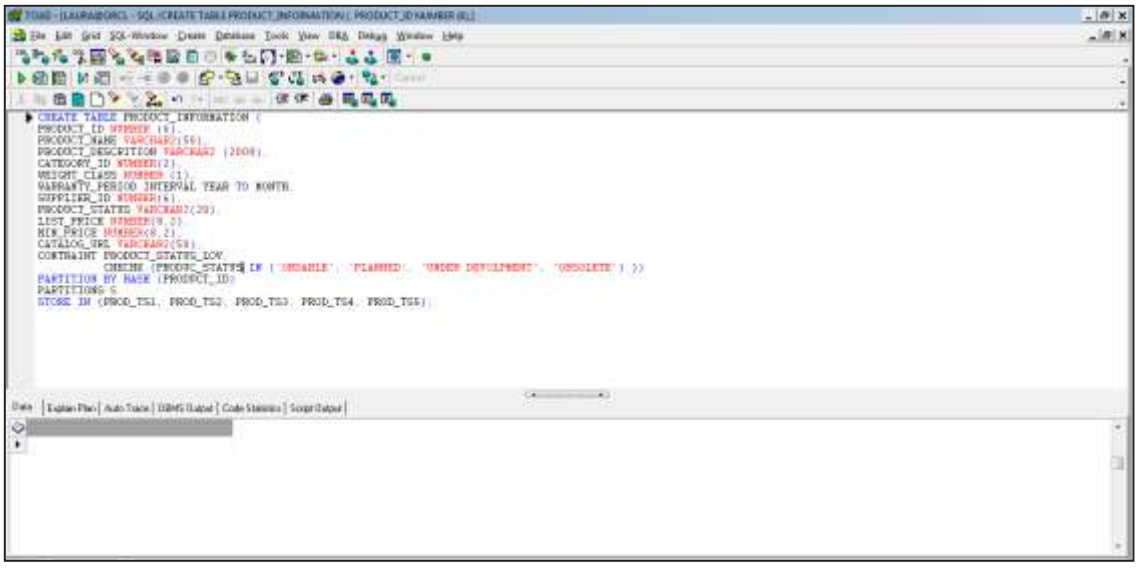


Figura 25. Creación y particionada la tabla PRODUCT_INFORMATION.

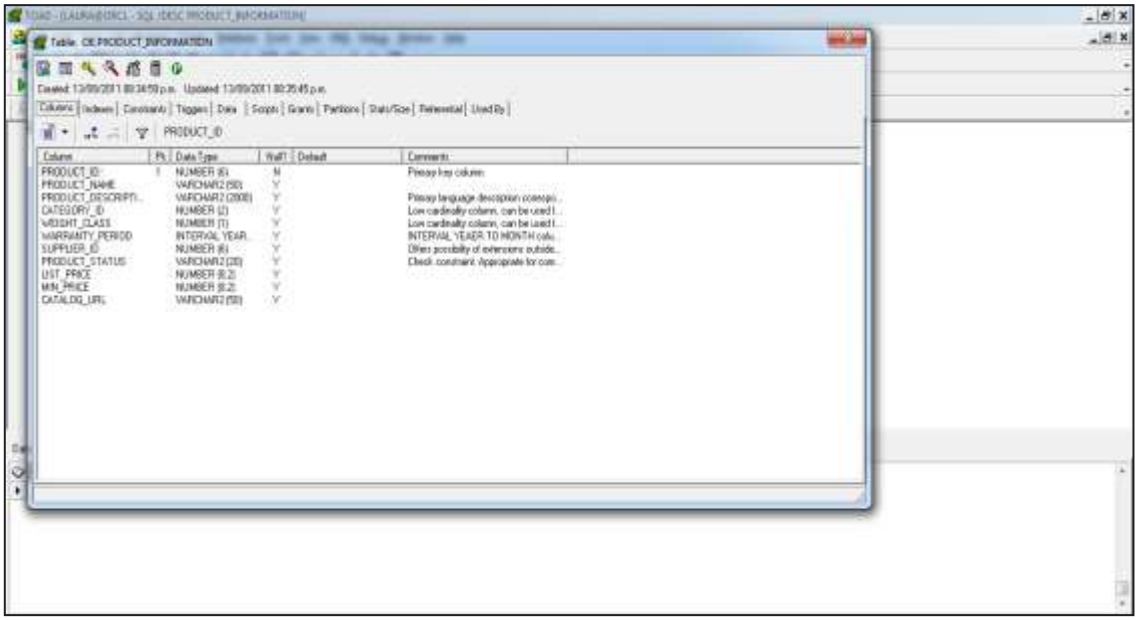


Figura 26. Descripción de tabla particionada PRODUCT_INFORMATION.

A continuación, se presenta un ejemplo de particionamiento por lista.

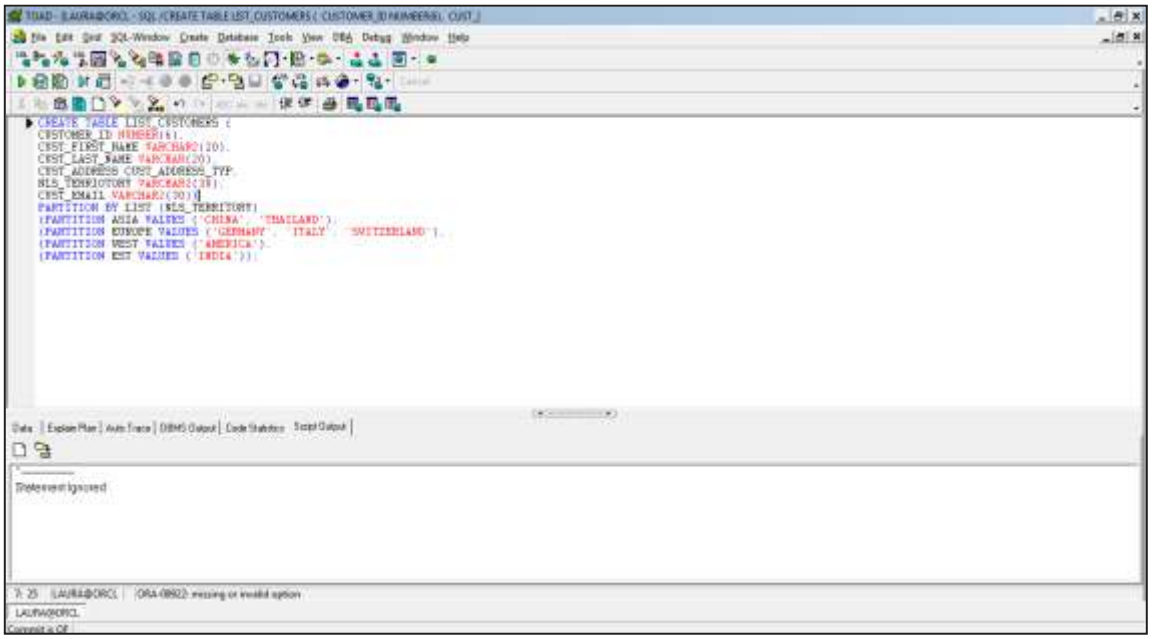


Figura 27. Creación de tabla LIST_CUSTOMERS con particionamiento por lista.

INDICES:

A continuación, se crea una tabla con un índice organizado.

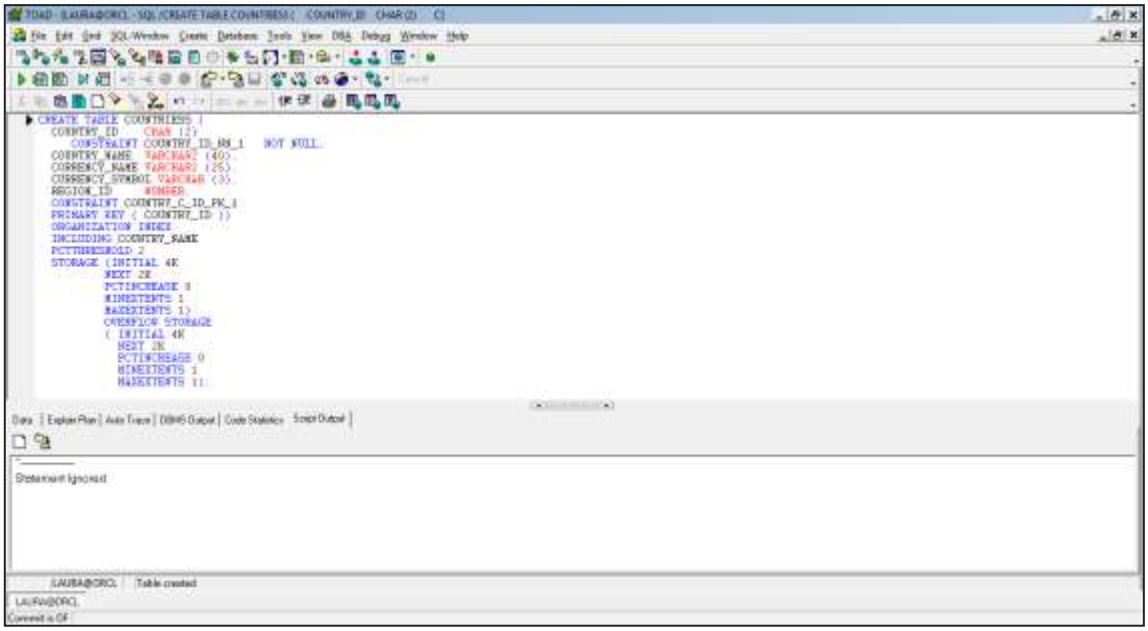


Figura 28. Creación de índice en la tabla COUNTRIES.

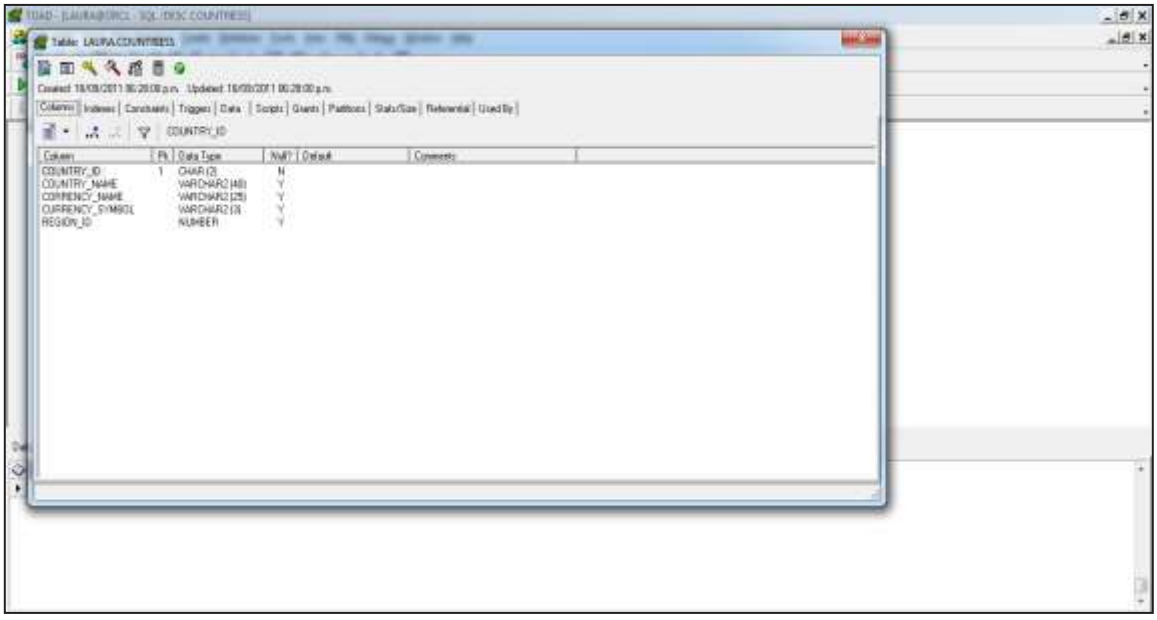


Figura 29. Descripción de tabla COUNTRIES.

En el ejemplo siguiente se crea una tabla externa.

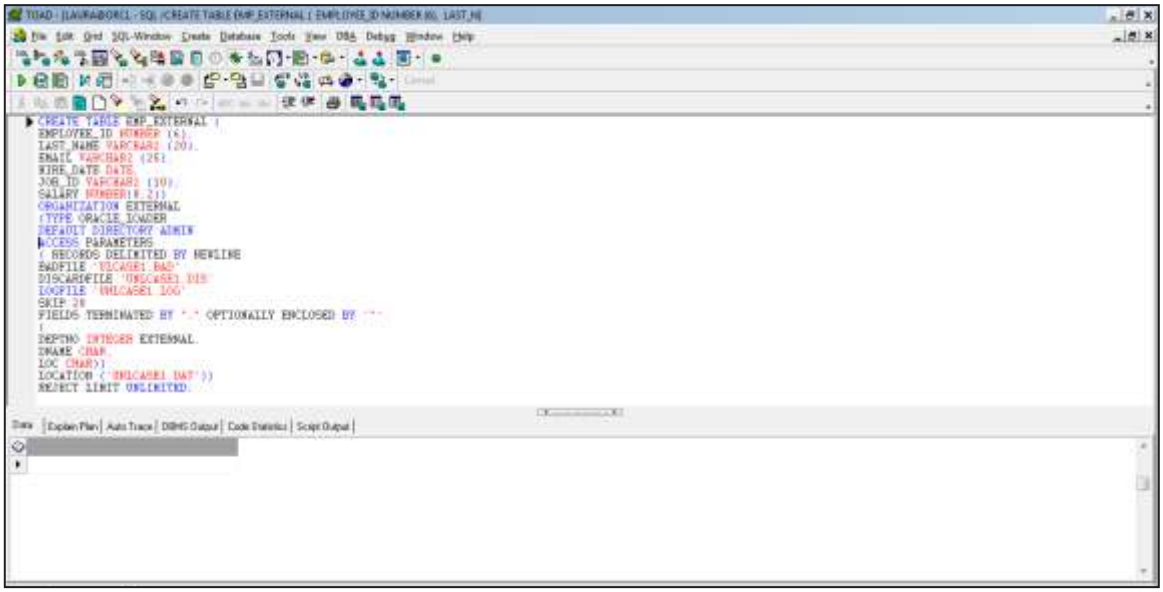


Figura 30. Creación de tabla externa EMP_EXTERNAL.

MODIFICACION DE TABLAS:

En el ejemplo siguiente se cambia los valores a la columna last_name.

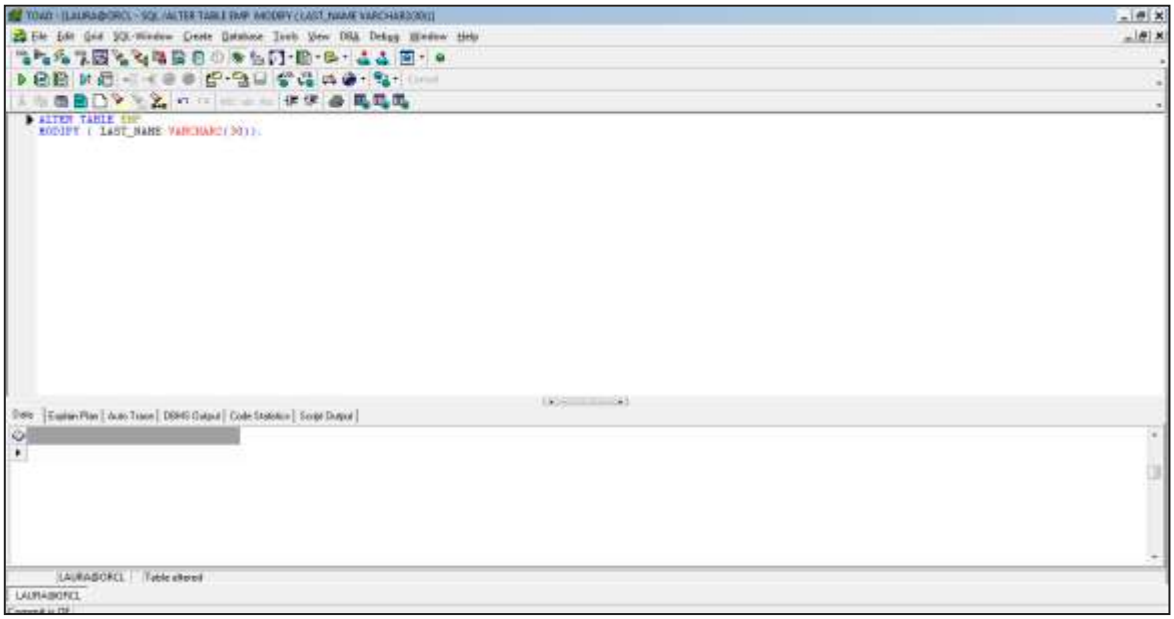


Figura 31. Modificación en la columna *last_name* a la tabla EMP.

En el ejemplo siguiente asignamos un nuevo espacio adicional para la tabla EMP con una extensión de 5 K y lo hacemos disponible para la instancia 4.

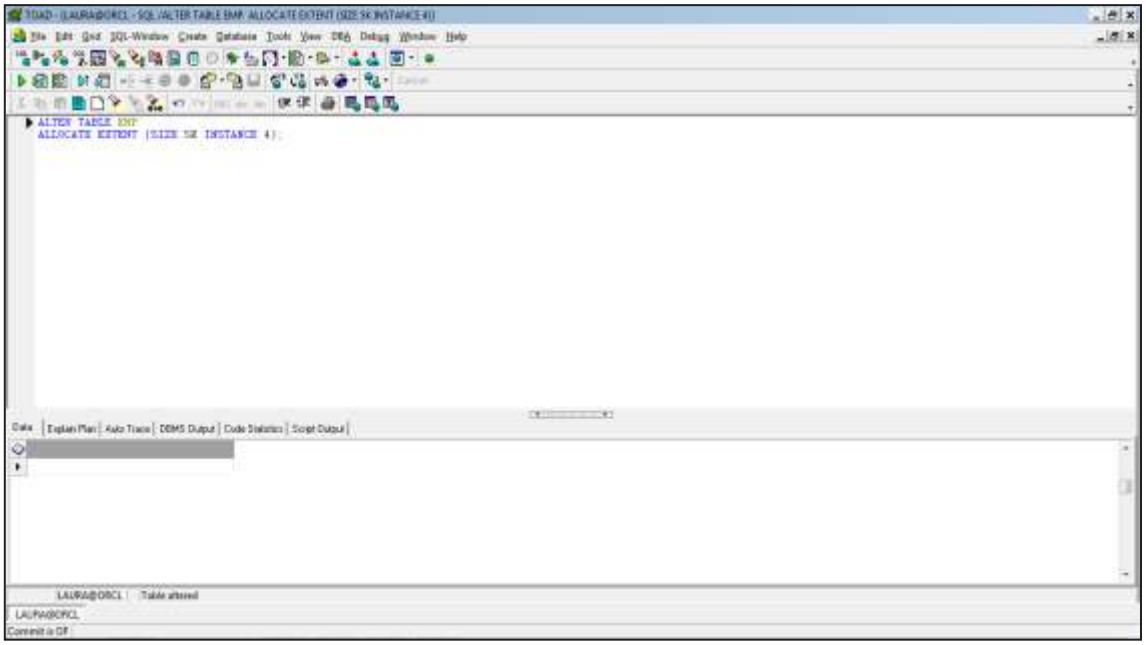


Figura 32. Modificación en tabla EMP asignación de espacio adicional.

La sentencia siguiente desasigna el espacio que no se usa al final de la tabla EMP.

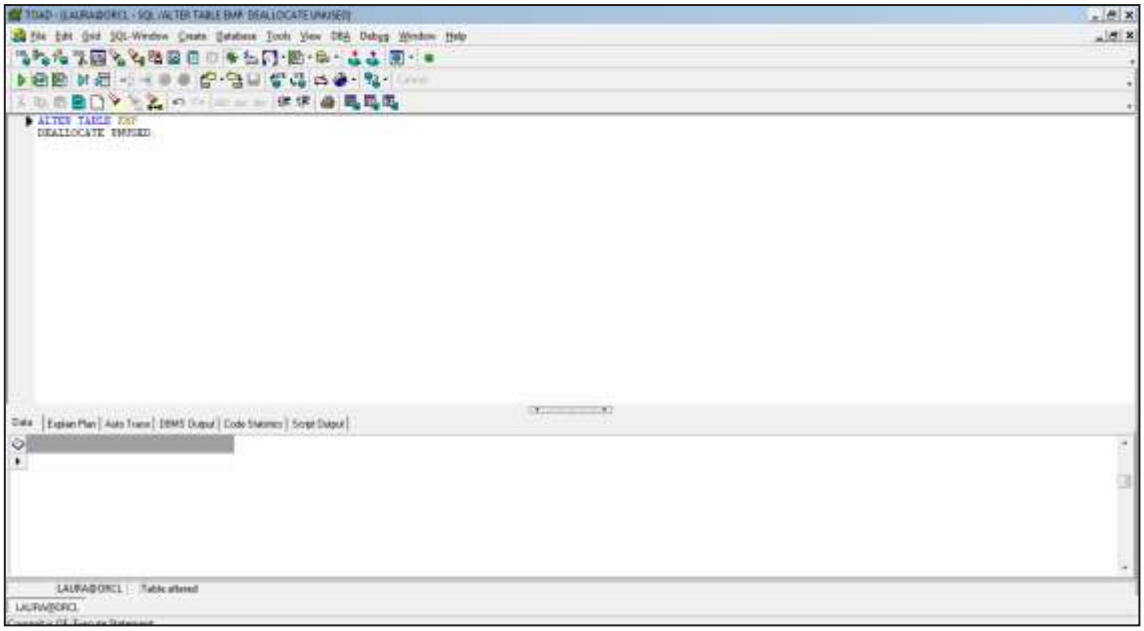


Figura 33. Modificación con una designación al espacio sin utilizar en la tabla EMP.

MODIFICACION COLUMNAS:

La sentencia siguiente borra las columnas que no se han utilizado en la tabla EMP.

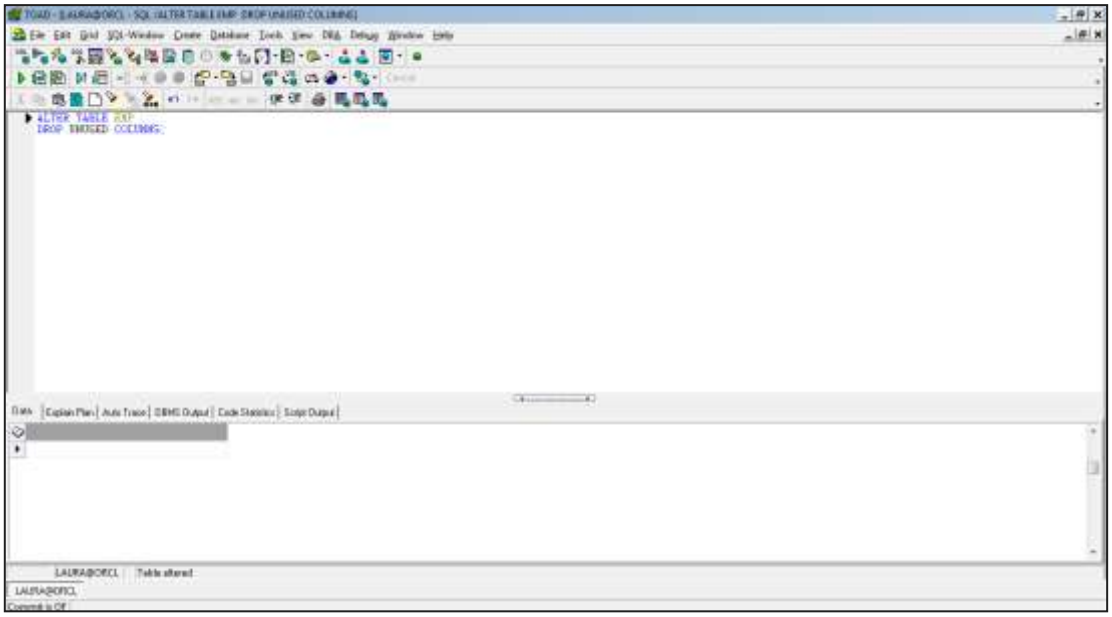


Figura 34. Alteración a la tabla EMP, se borran tablas no utilizadas.

En la consulta siguiente se agrega la columna con valores numéricos de nombre *c_id* a la tabla EMP con la restricción de que no puede tomar valores nulos.

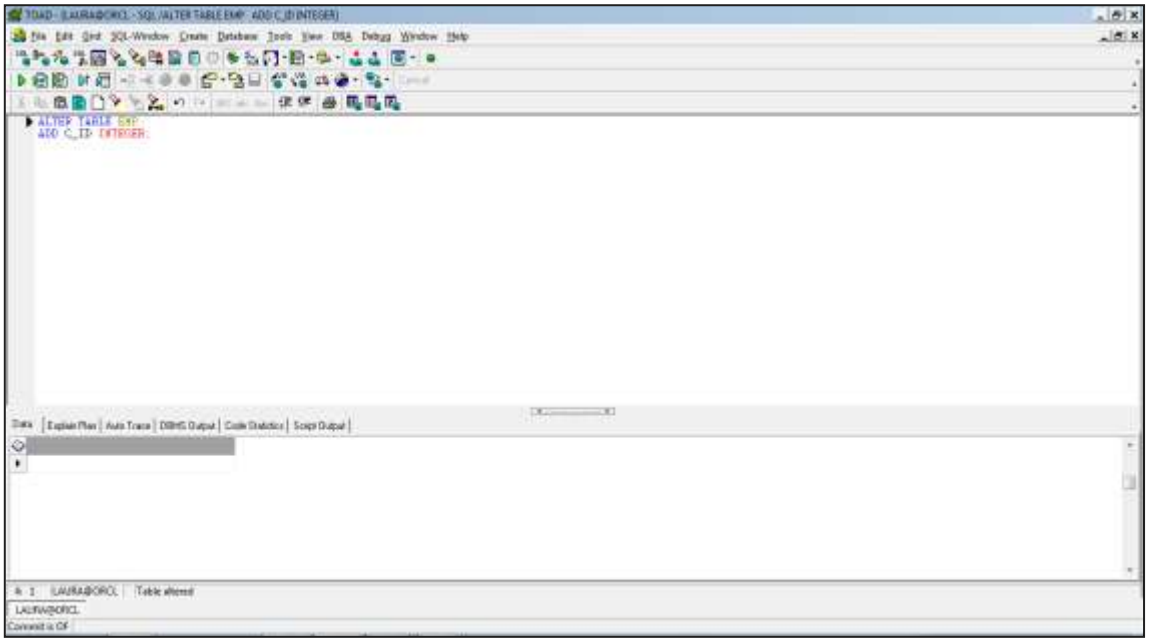


Figura 35. Modificación, se agrega una columna a la tabla EMP.

La siguiente modifica la columna *ID* de la tabla DEPT a tipo numérico sin restricciones de valores nulos.

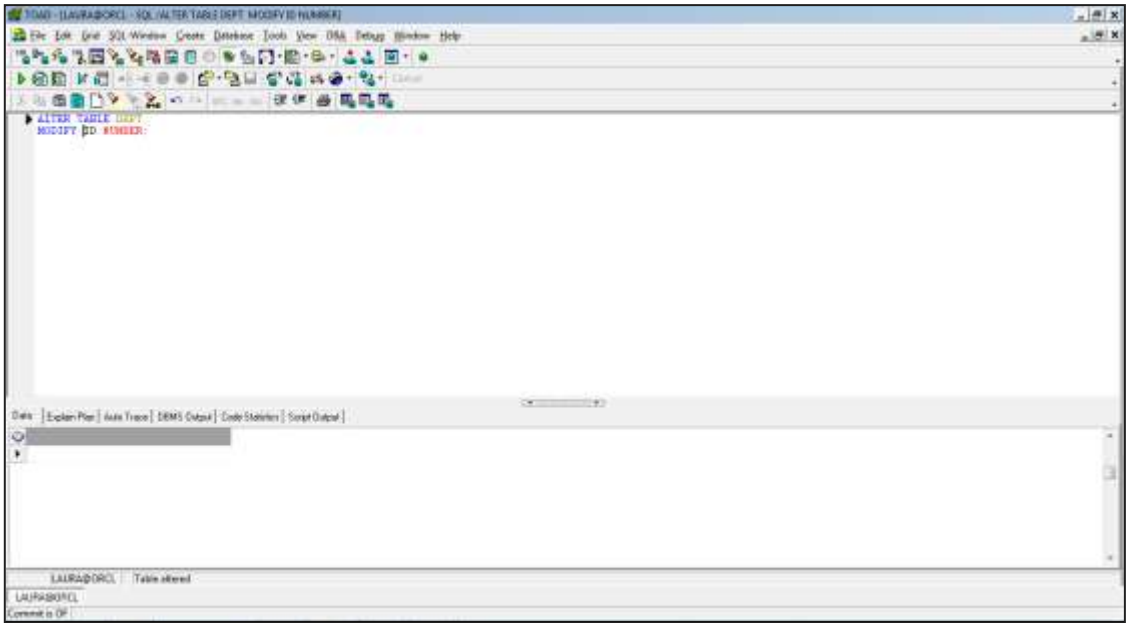


Figura 36. Modificación a la columna *id* con valores numéricos en la columna DEPT.

La consulta siguiente se define y deshabilita restricciones de integridad CHECK en la columnas de la tabla EMP.

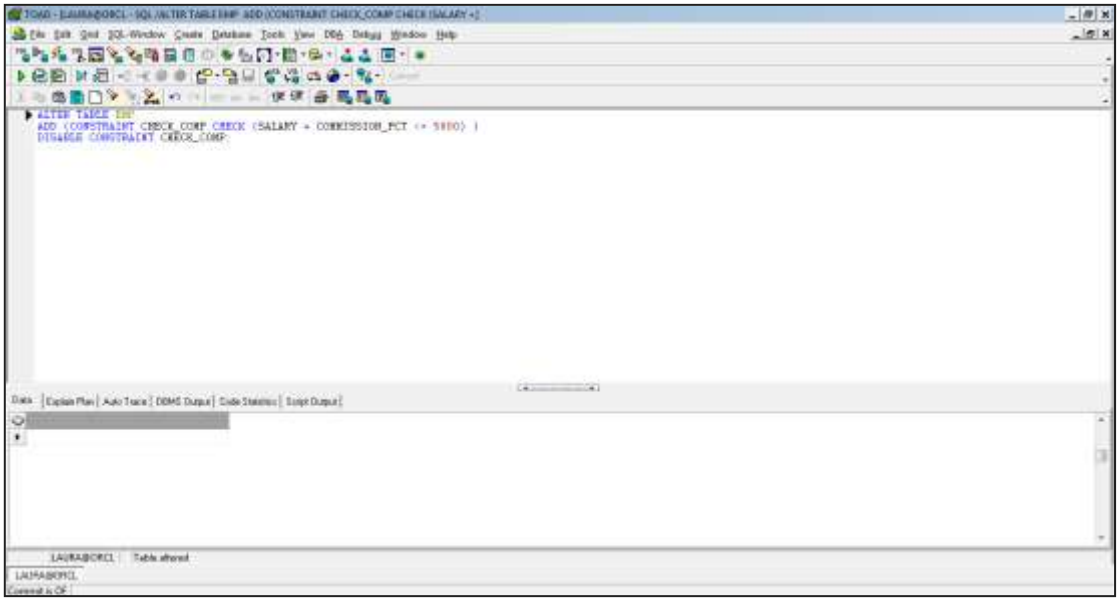


Figura 37. Deshabilitación de restricción CHECK en la columna EMP.

La consulta siguiente añade una restricción de integridad referencial a la tabla CUSTOMERS que asegura que el *s_origen* de cada registro hace referencia a un id de la tabla SALESREP

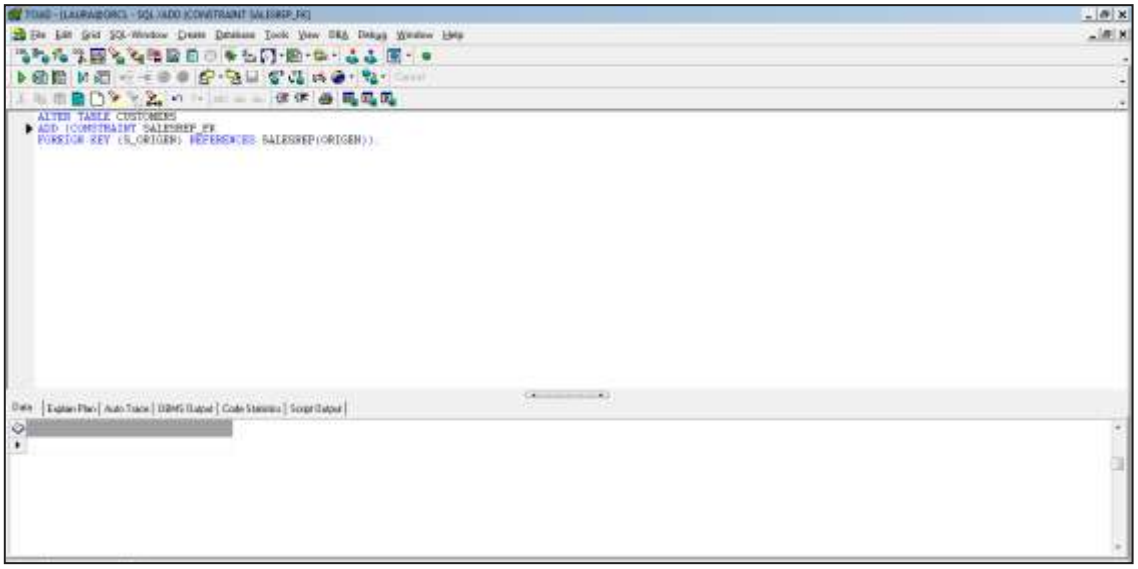


Figura 38. Modificación a la restricción de integración de la tabla CUSTOMERS.

La consulta siguiente agrega la restricción de integridad referencial clave principal a la columna *motivo* de la tabla SALESREP.

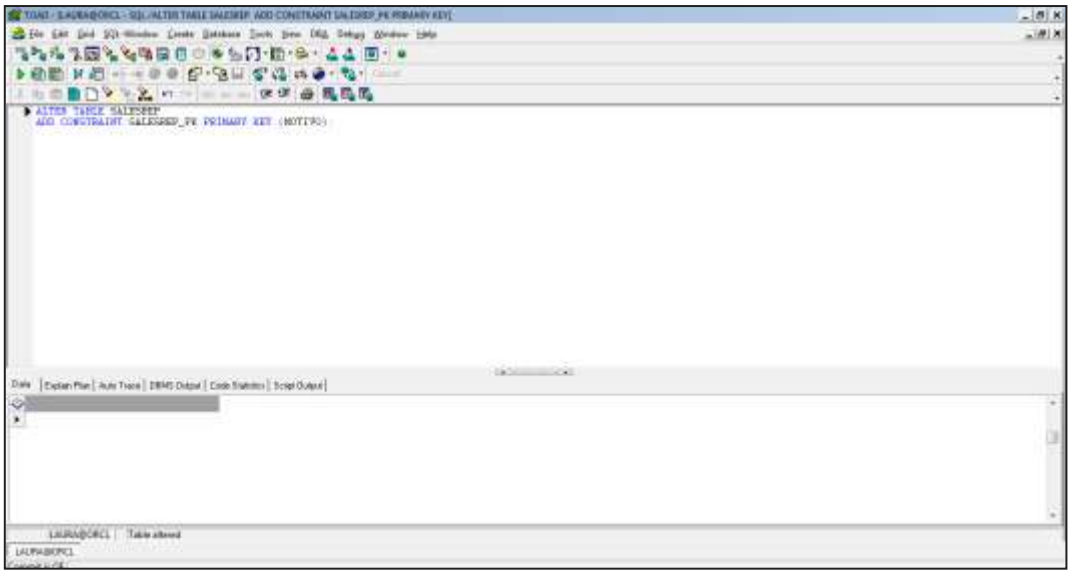


Figura 39. Modificación en la tabla SALESREP con restricción en la columna *motivo*.

En la consulta siguiente se añaden a la tabla EMP las columnas de nombres *thriftplan* (numérica con 7 dígitos y dos decimales) y *loancode* (con 1 carácter).

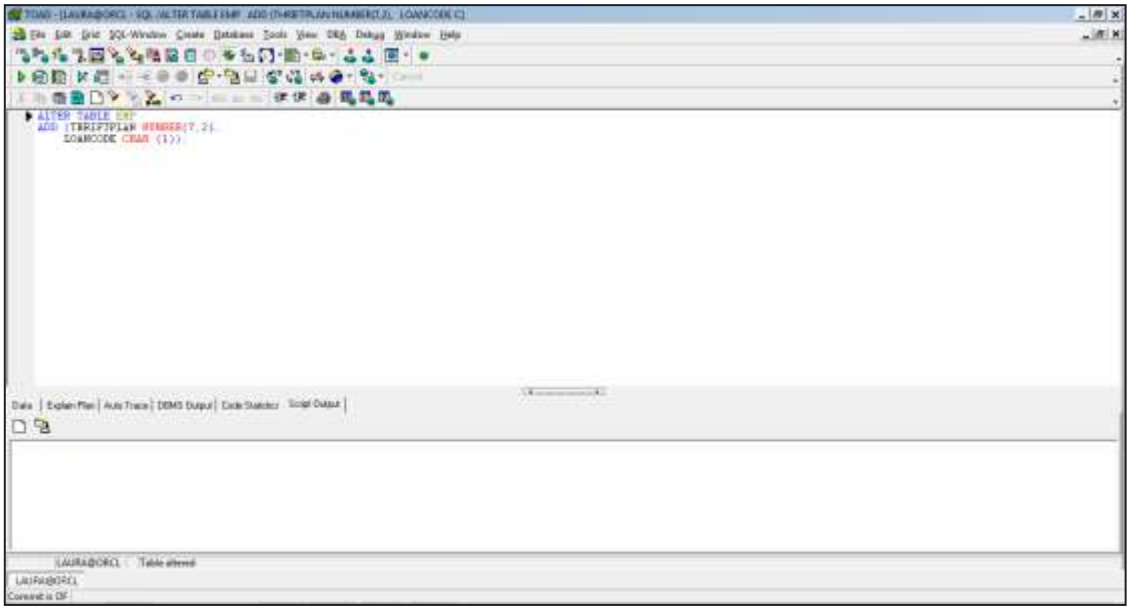
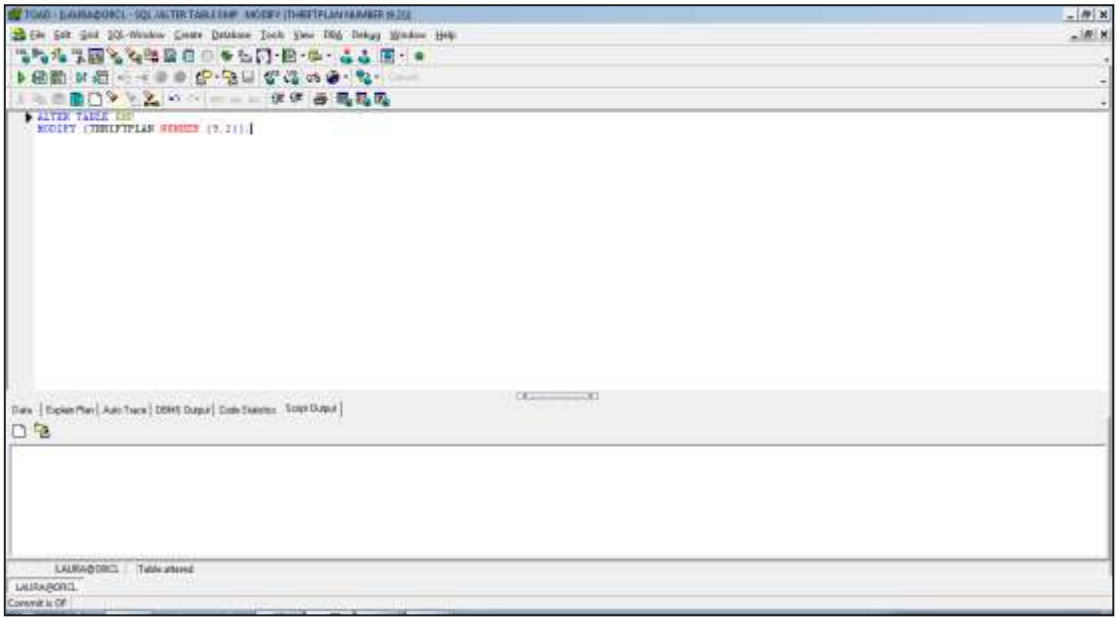
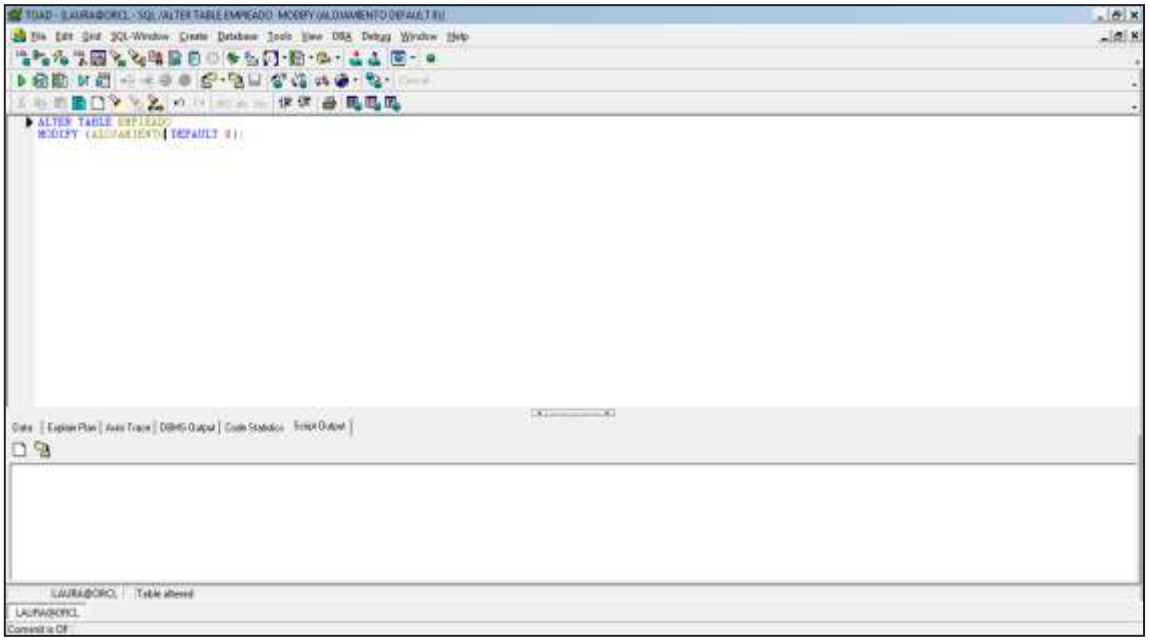


Figura 40. Se añaden dos columnas a la tabla EMP.

En la consulta siguiente se aumenta a 9 dígitos el campo *thriftplan* de la tabla EMP.

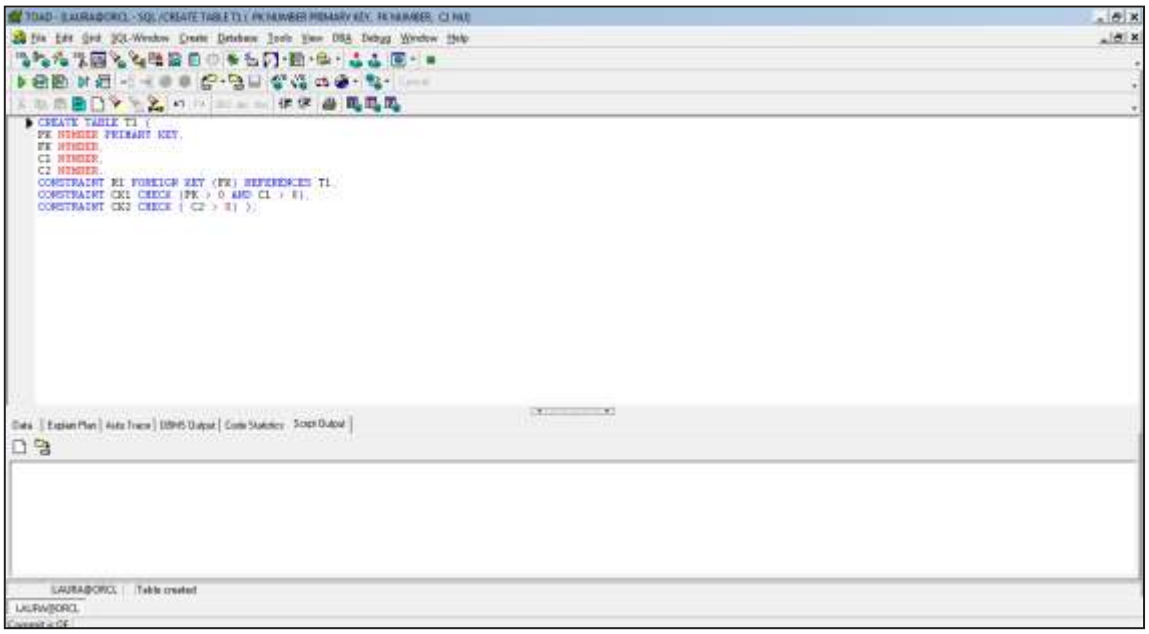


La sentencia siguiente modifica la columna *alojamiento* de la tabla EMPLEADO situando sus valores por defecto a cero.

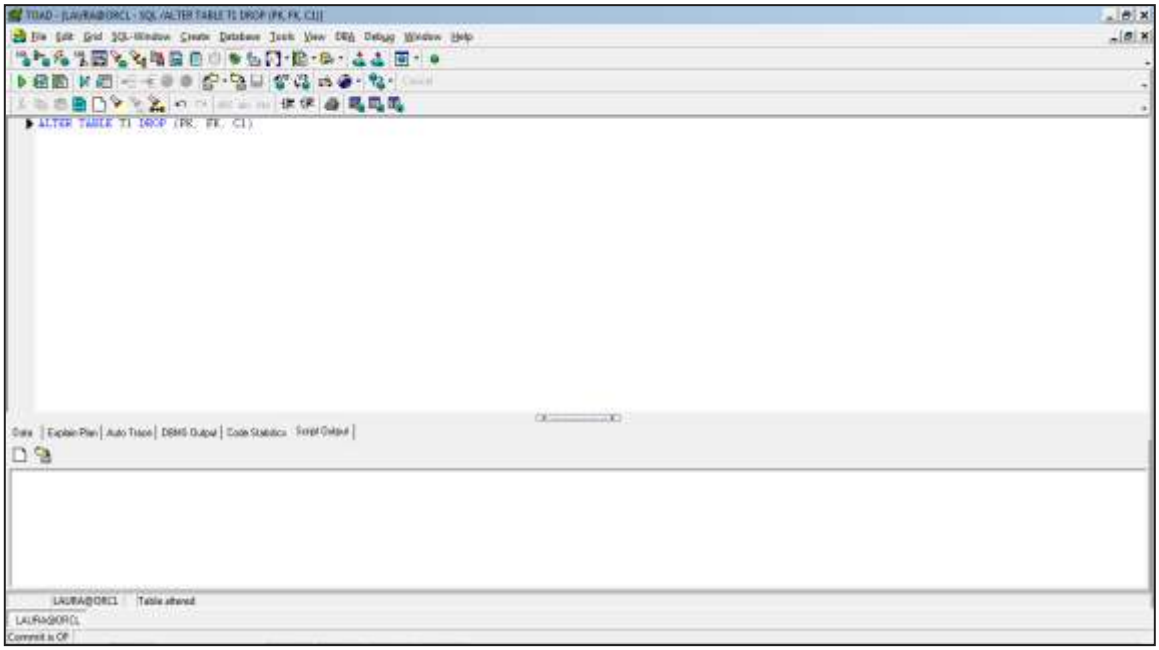


MODIFICACION DE RESTRICCION DE INTEGRIDAD:

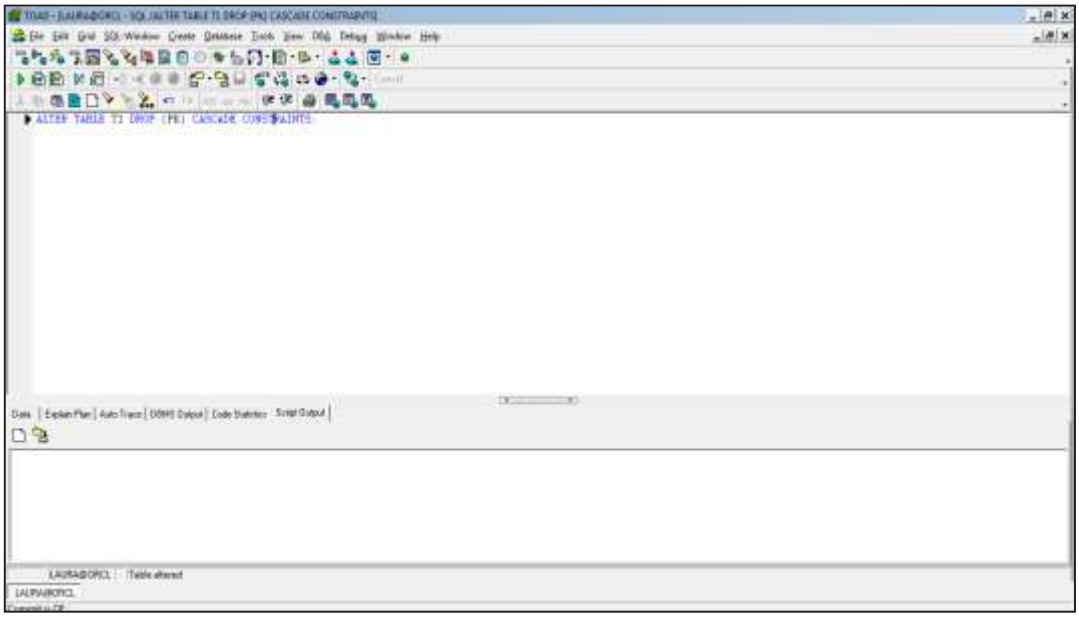
La consulta siguiente crea la tabla TI definiendo sobre sus columnas determinadas restricciones de integridad.



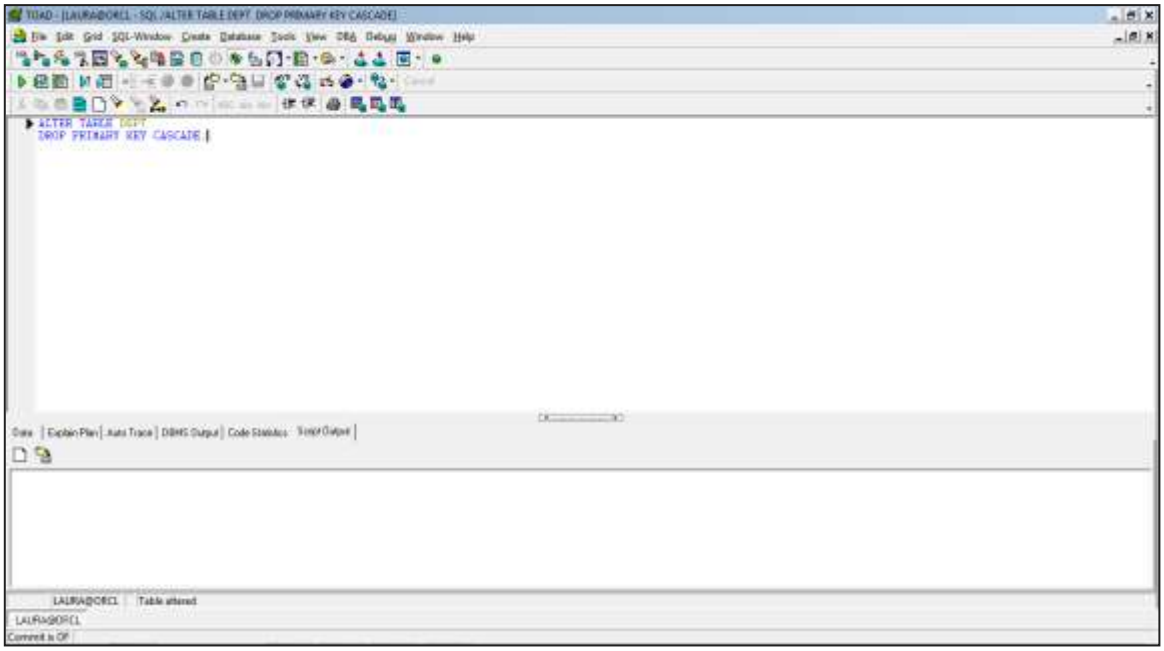
A continuación, borramos la restricción de clave primaria (pk), la restricción de clave foránea (fk) y la restricción CHECK (c1).



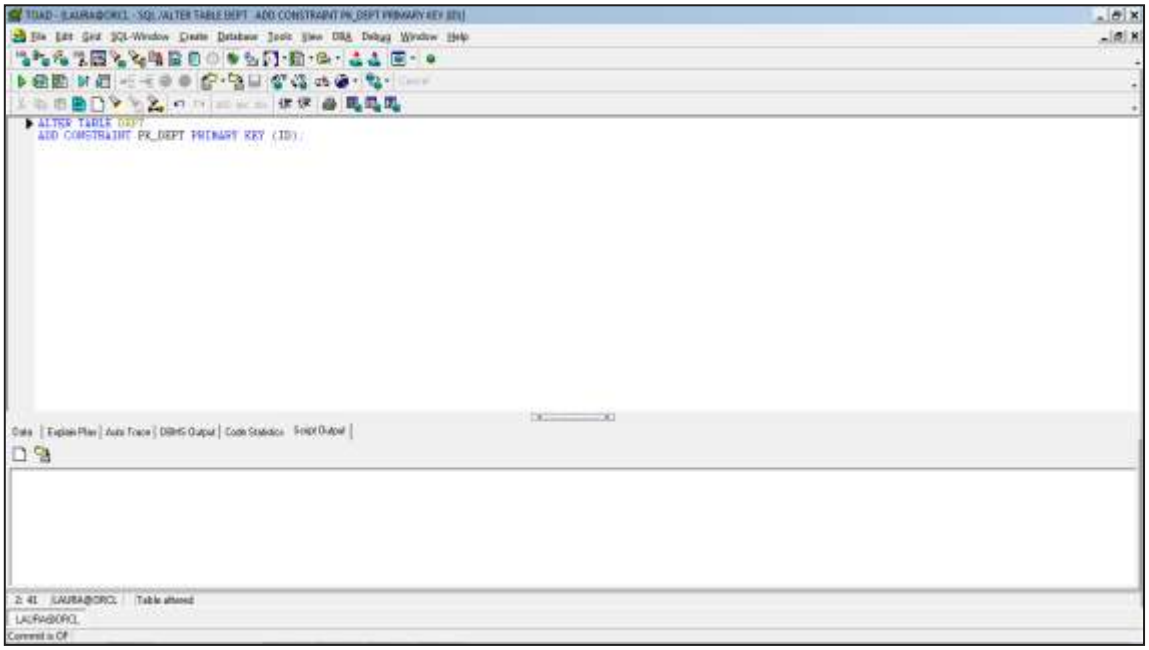
El borrado anterior también podía hacerse realizando con la cláusula CASCADE CONSTRAINTS.



La consulta siguiente bora la clave primaria de la tabla DEPT.

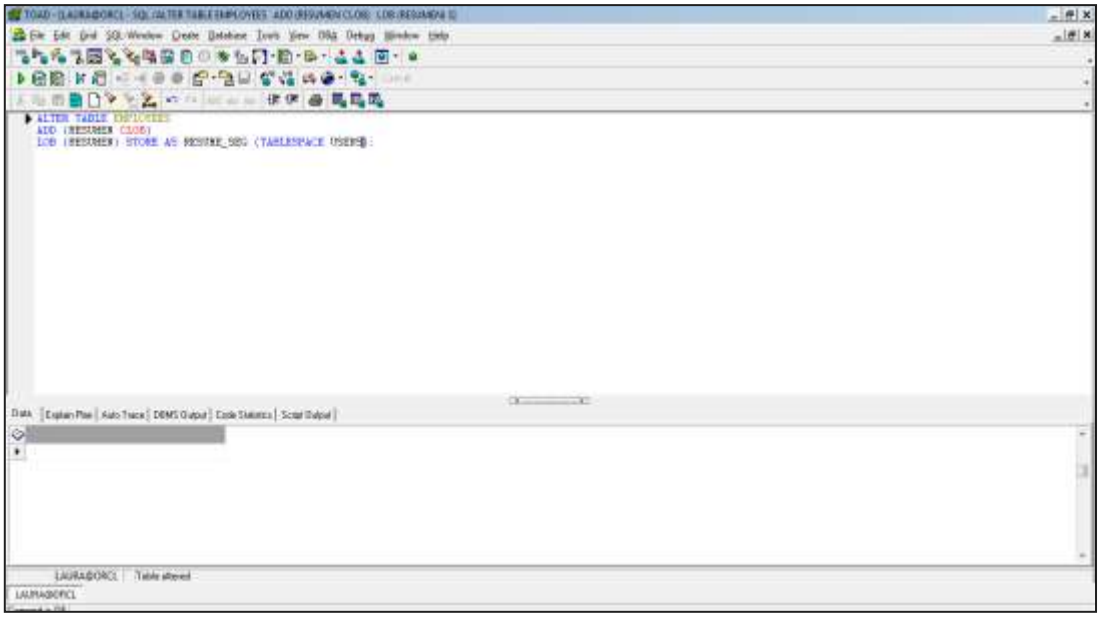


Para no dejar la tabla DEPT sin clave primaria volvemos a añadírsela.

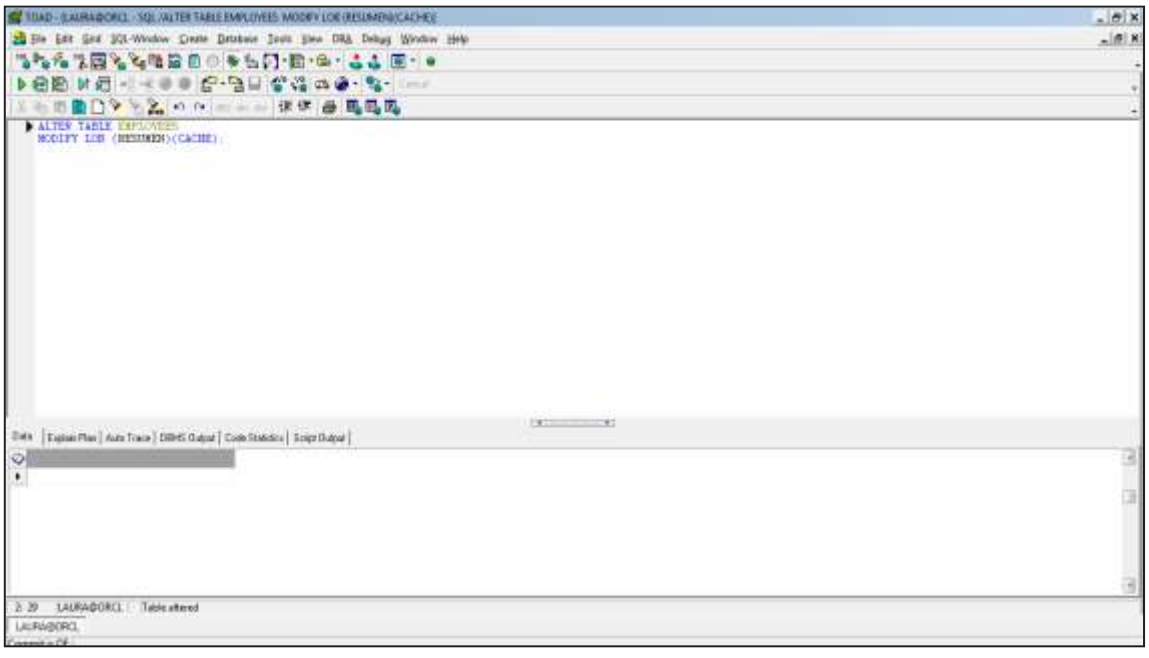


MODIFICACIONDE PROPIEDADES DE COLUMNAS:

La consulta siguiente añade la columna CLOB de nombre *resume* a la tabla EMPLOYEE y específica para ella características de almacenamiento LOB.



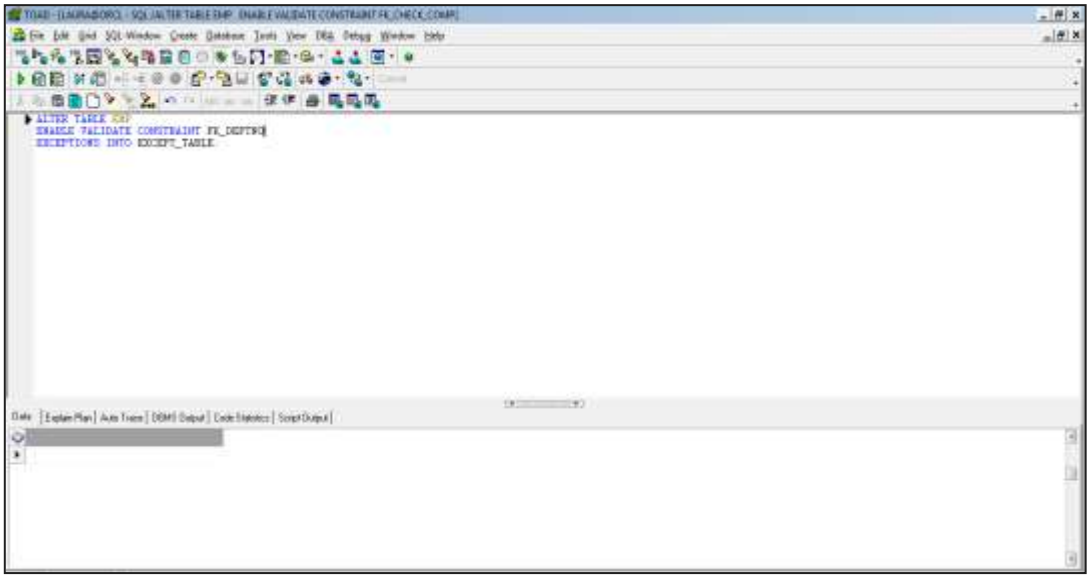
La consulta siguiente modifica la columna LOB de nombre *resume* en la tabla EMPLOYEE.



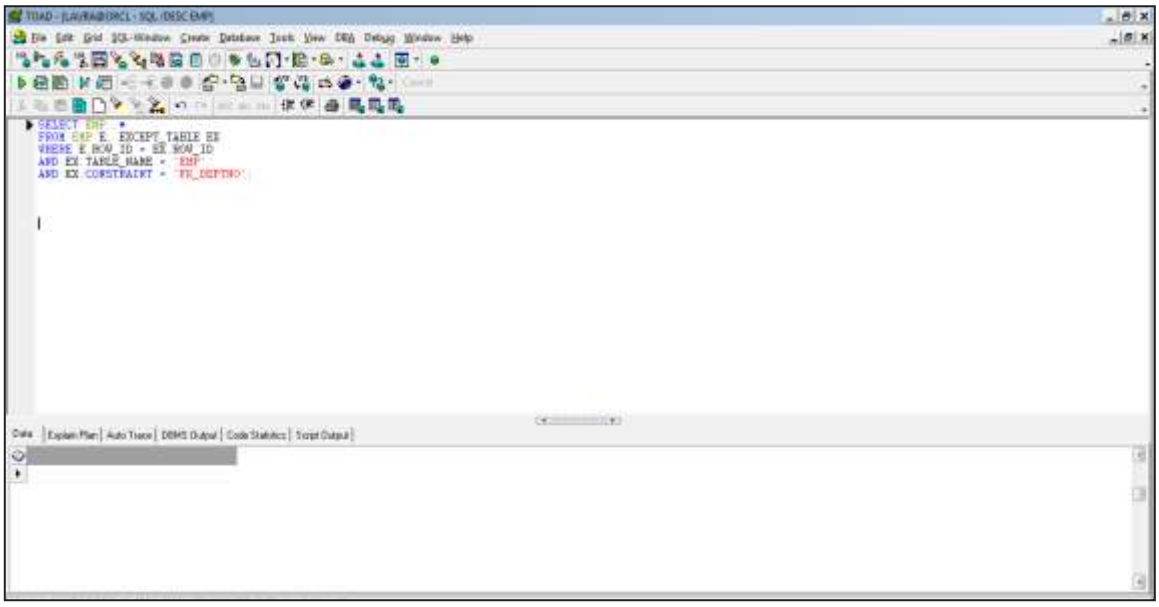
MOVIMIENTO DE TABLAS:

Cláusula ENABLE/DISABLE

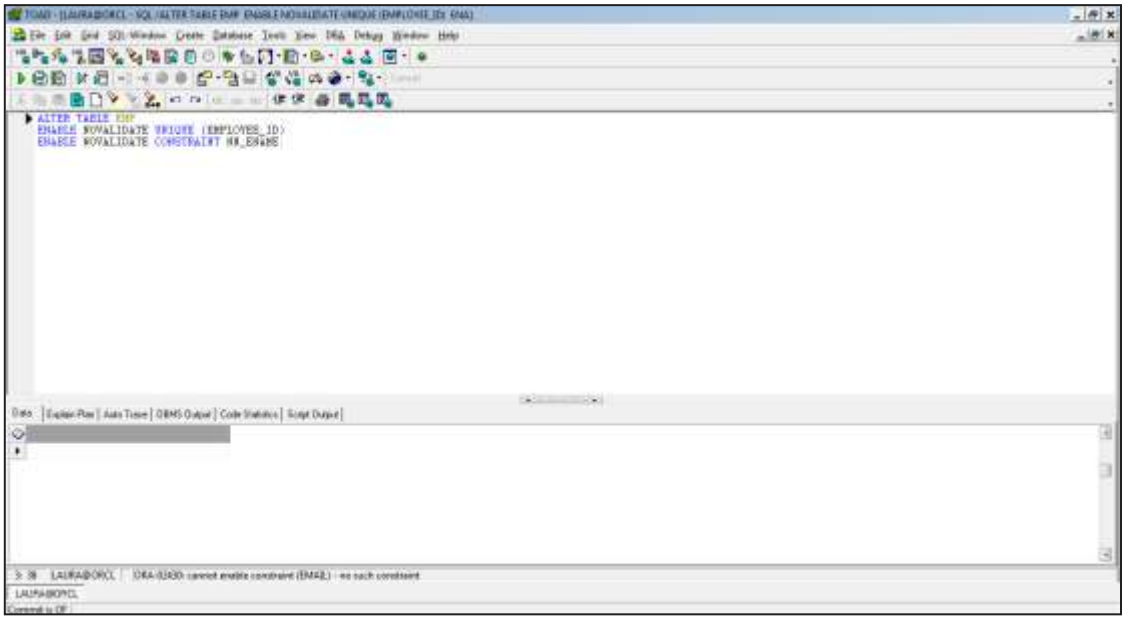
La consulta sitúa el estado ENABLE VALIDATE en la restricción de integridad *fk_deptno* en la tabla EMP.



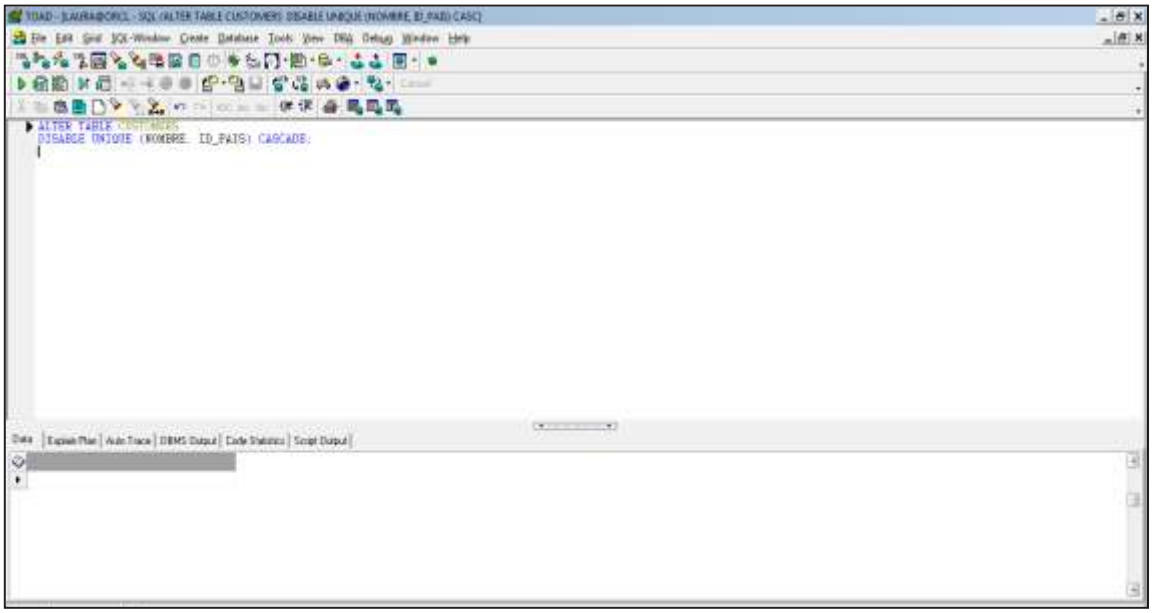
Cada fila de la tabla EMP debe satisfacer la restricción para que ORACLE la habilite. Si una fila viola la restricción, ésta se deshabilita. Oracle lista cada excepción en la tabla DEPARTMENTS. Se puede identificar la excepción en la tabla EMP con la siguiente secuencia.



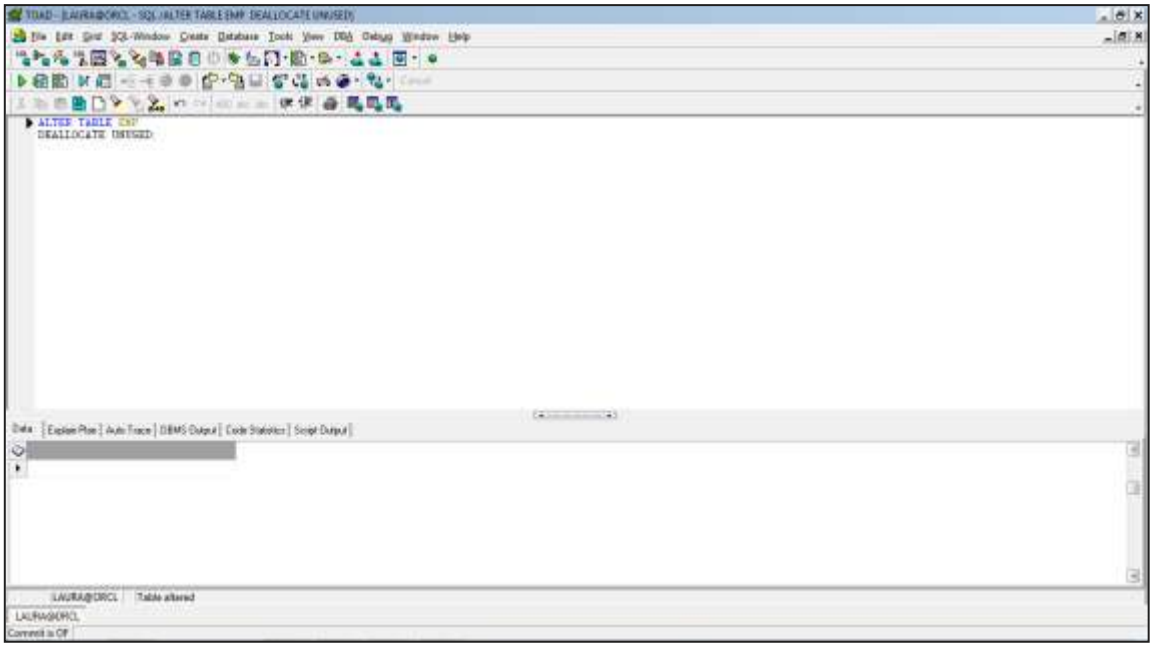
La consulta siguiente intenta situar en estado NO VALIDATE dos restricciones en la tabla EMP.



La consulta siguiente deshabilita la clave única doble formada por las columnas *nombre* y *id_pais* en la tabla de customers.

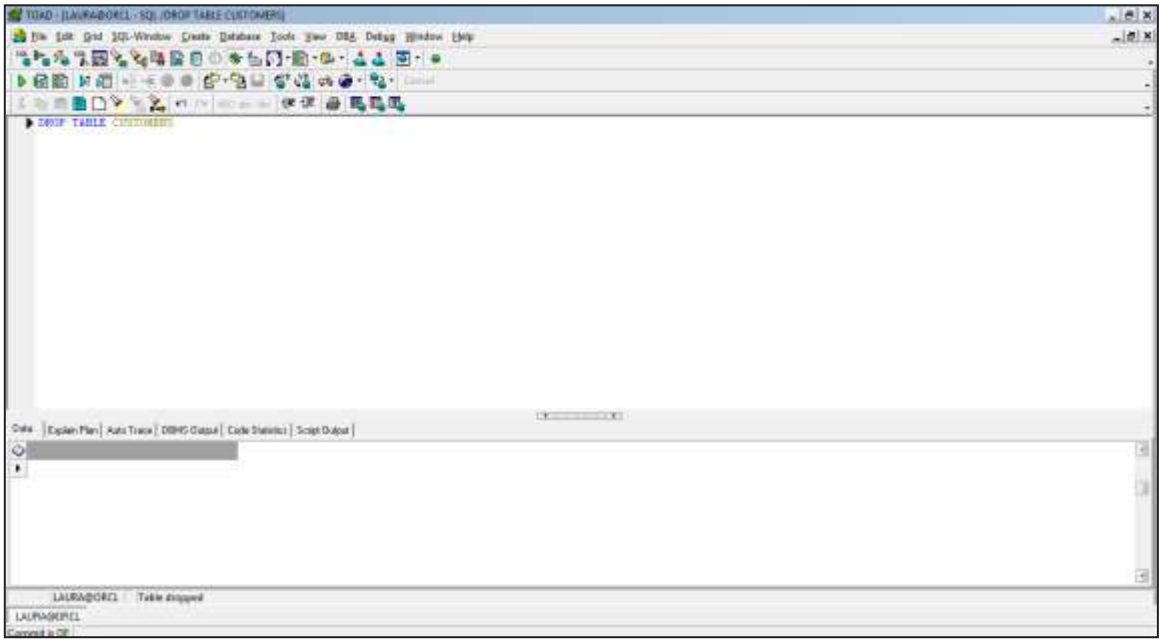


La sentencia habilita los disparadores asociados a la tabla EMP

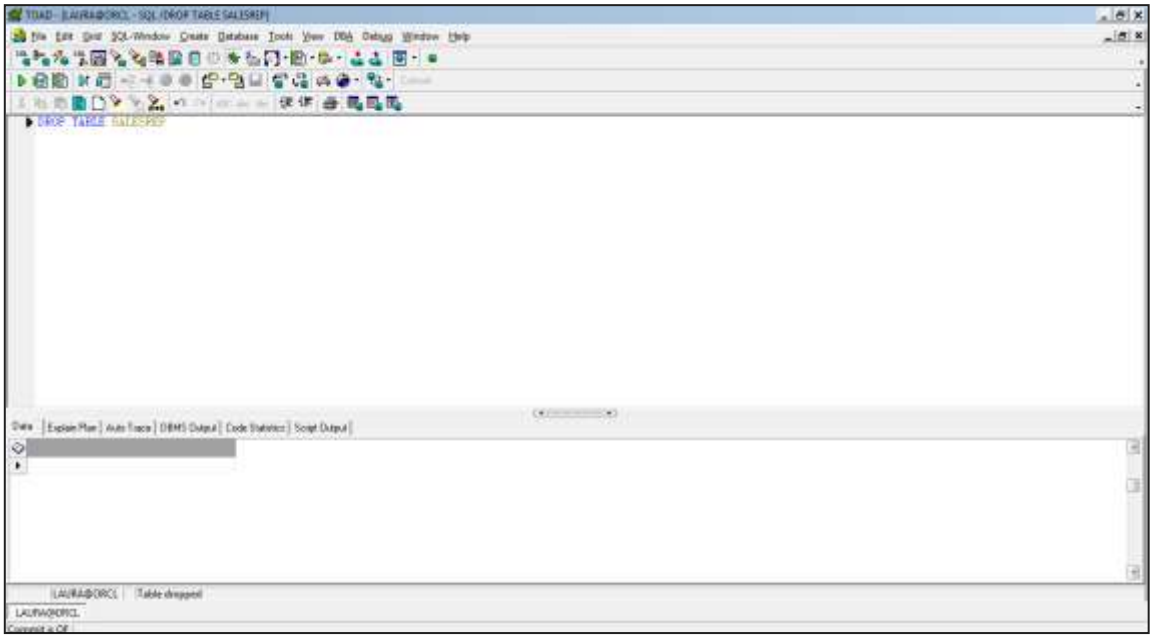


BORRADO DE TABLAS:

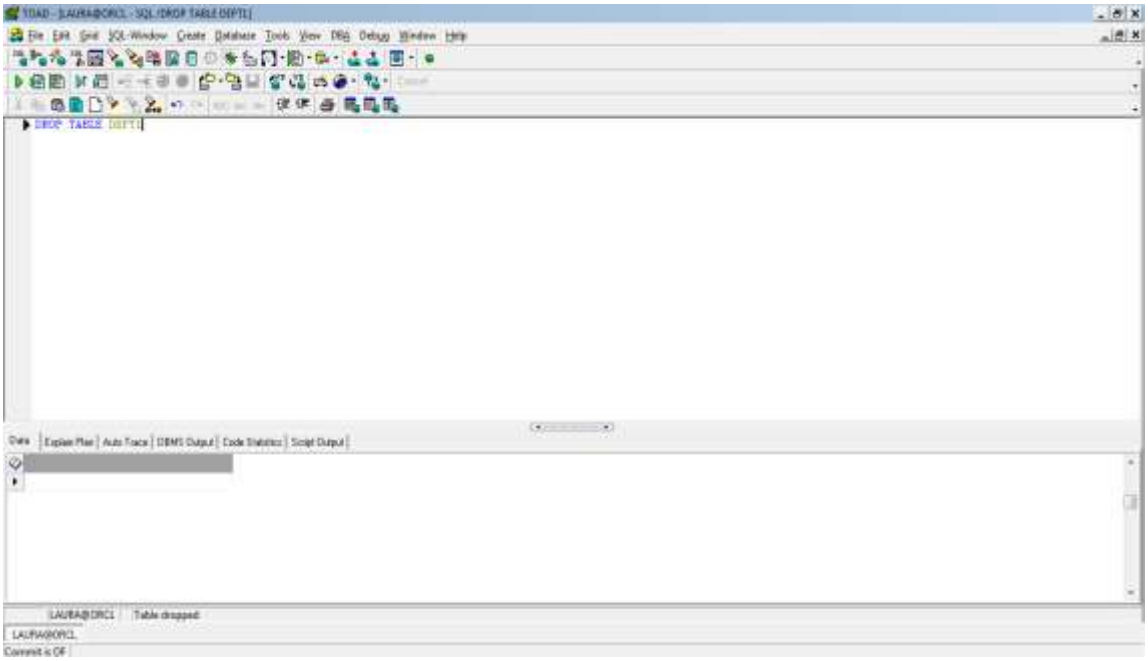
La consulta borra la tabla CUSTOMERS.



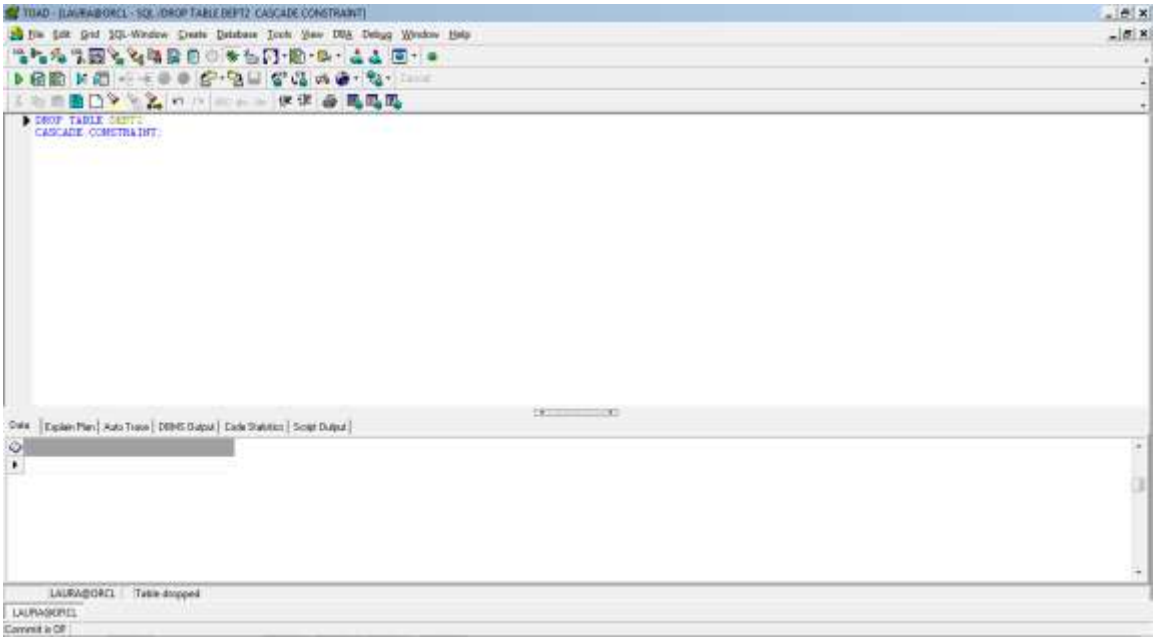
En esta sentencia se borra la tabla SALESREP.



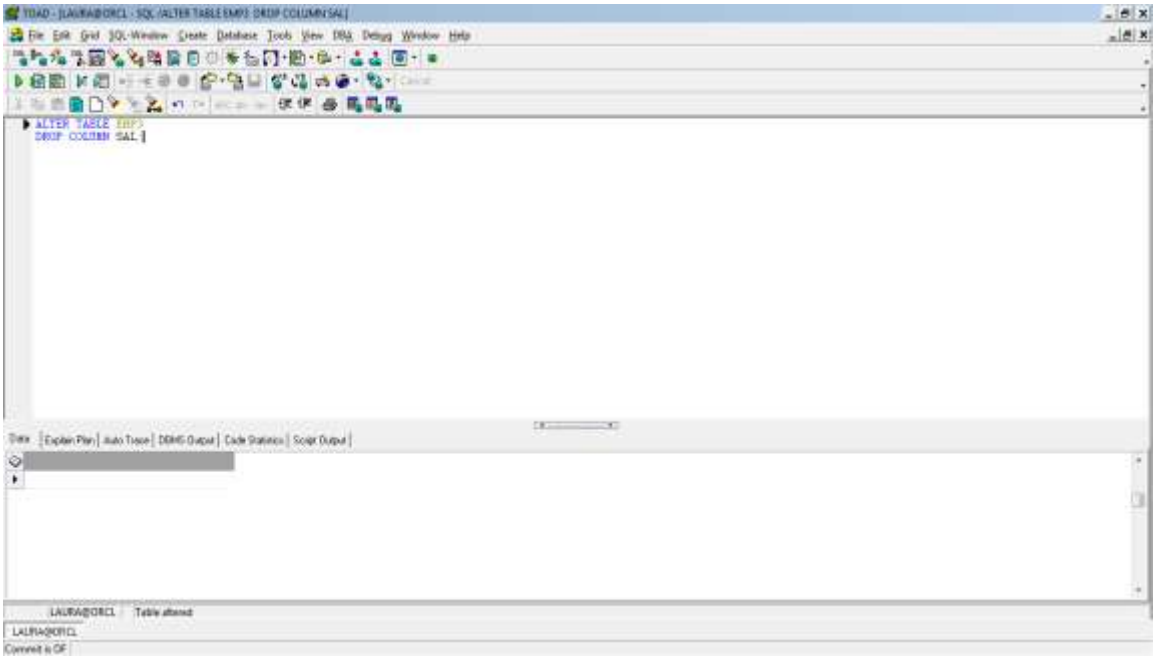
Borrado de tabla DEPT1.



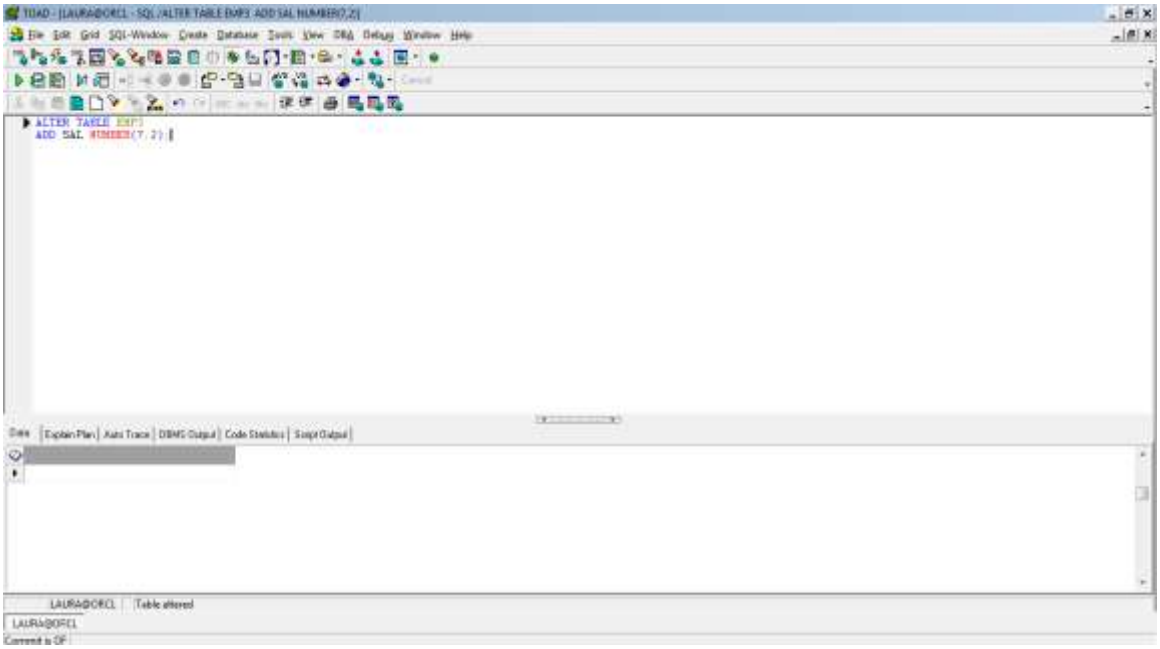
Ejemplo de borrar tabla DEPT2, de modo que las restricciones de integridad referencial para que pueda quedar suprimida.



Se borrara la columna *sal* de la tabla EMP3.

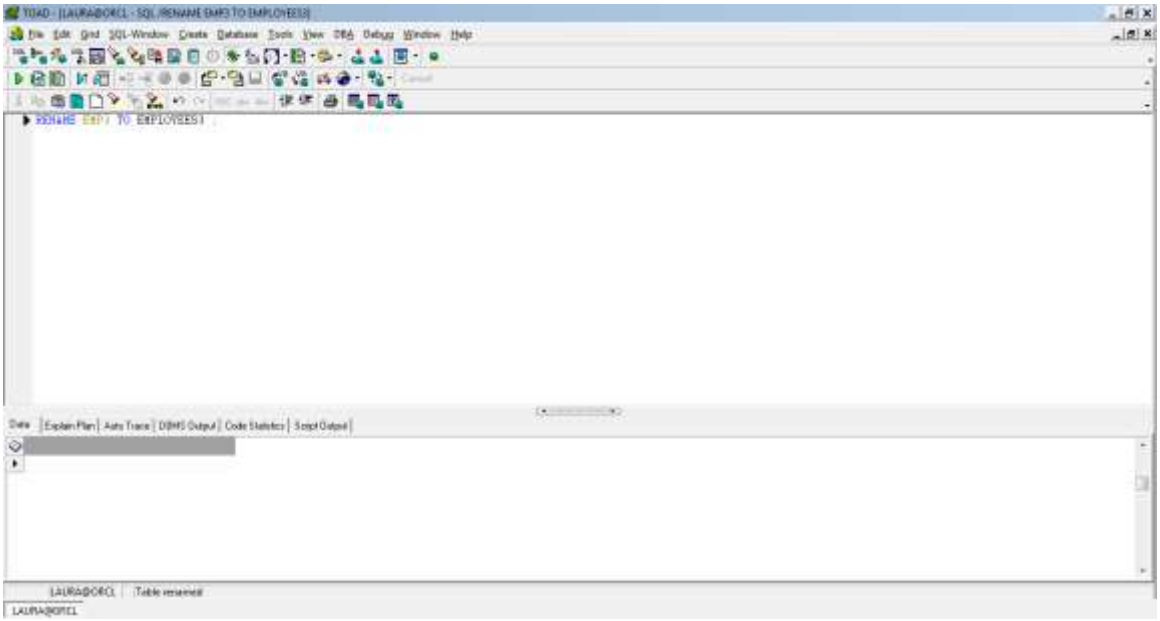


En este ejemplo se va a deshacer el cambio de añadimiento a la tabla la columna recién borrada.



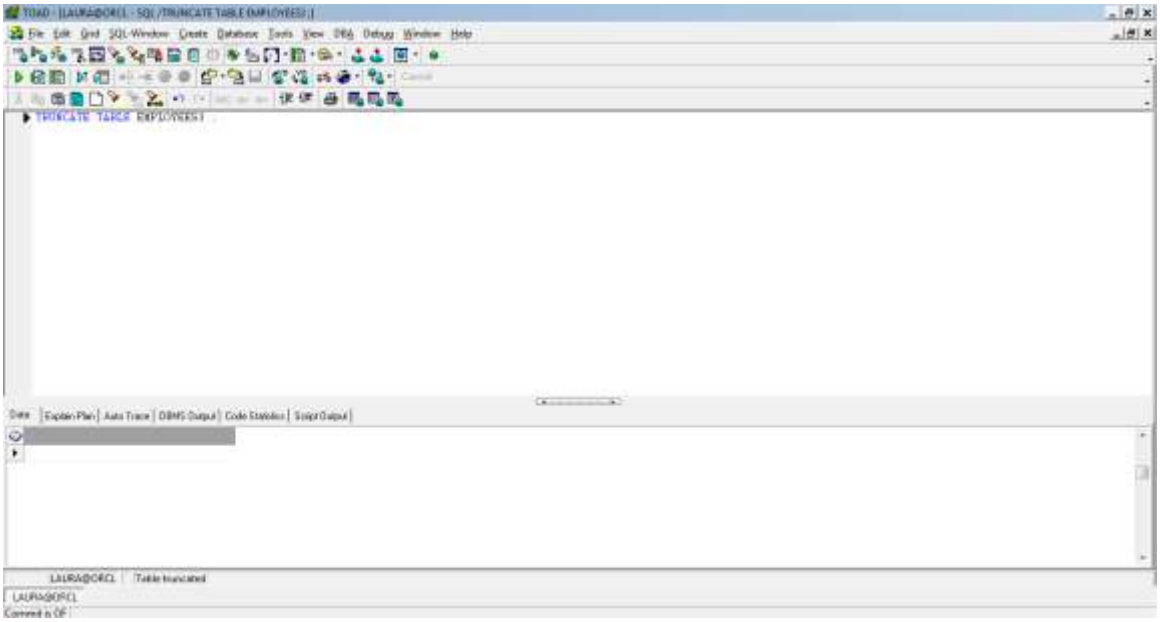
RENOMBRAR:

En el ejemplo se renombrara a la tabla EMP3 a EMPLOYEES3.



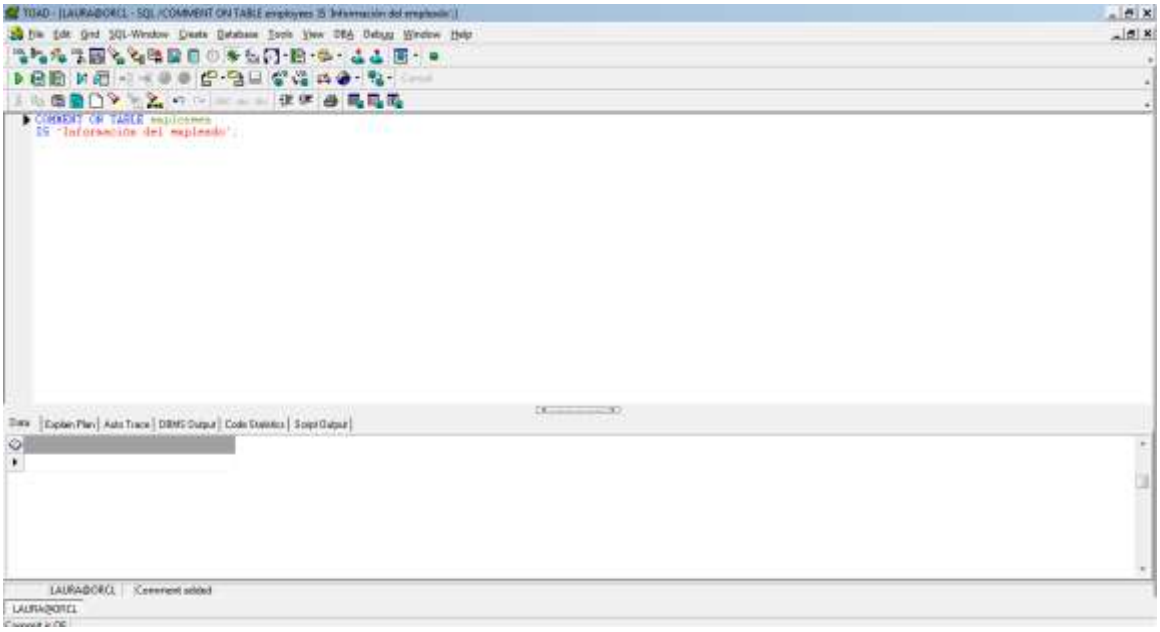
TRUNCAMIENTO:

En el siguiente ejemplo usaremos la sentencia TRUNCATE para la tabla EMPLOYEES3.

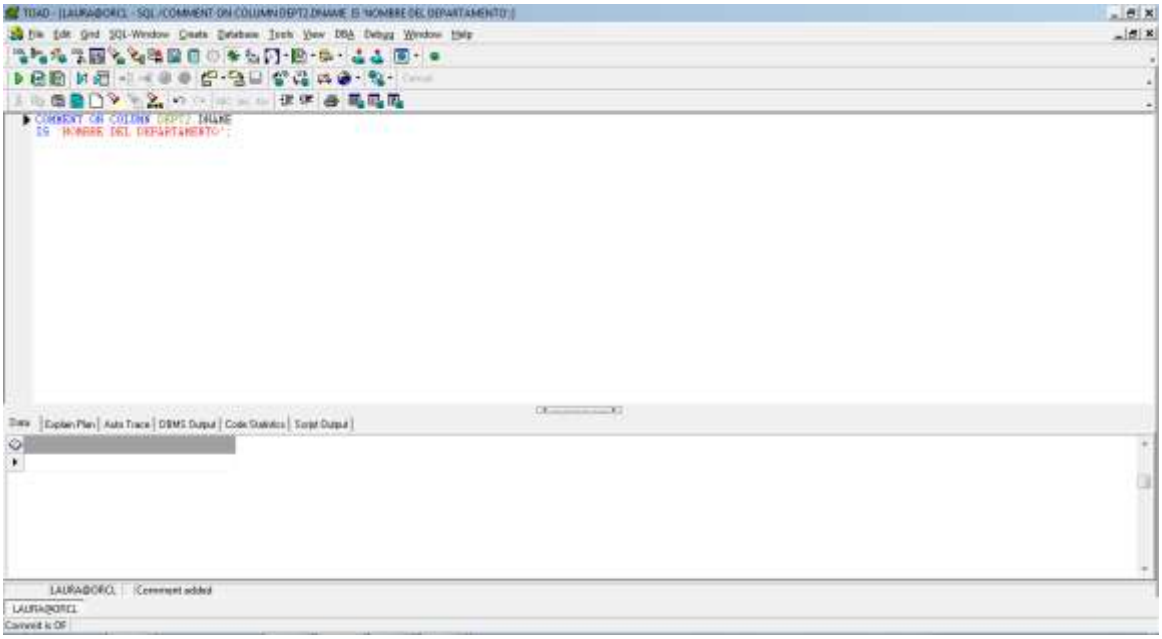


COMENTARIOS:

Para adicionar un comentario a las tablas ya creadas se realiza de la siguiente con el siguiente ejemplo.

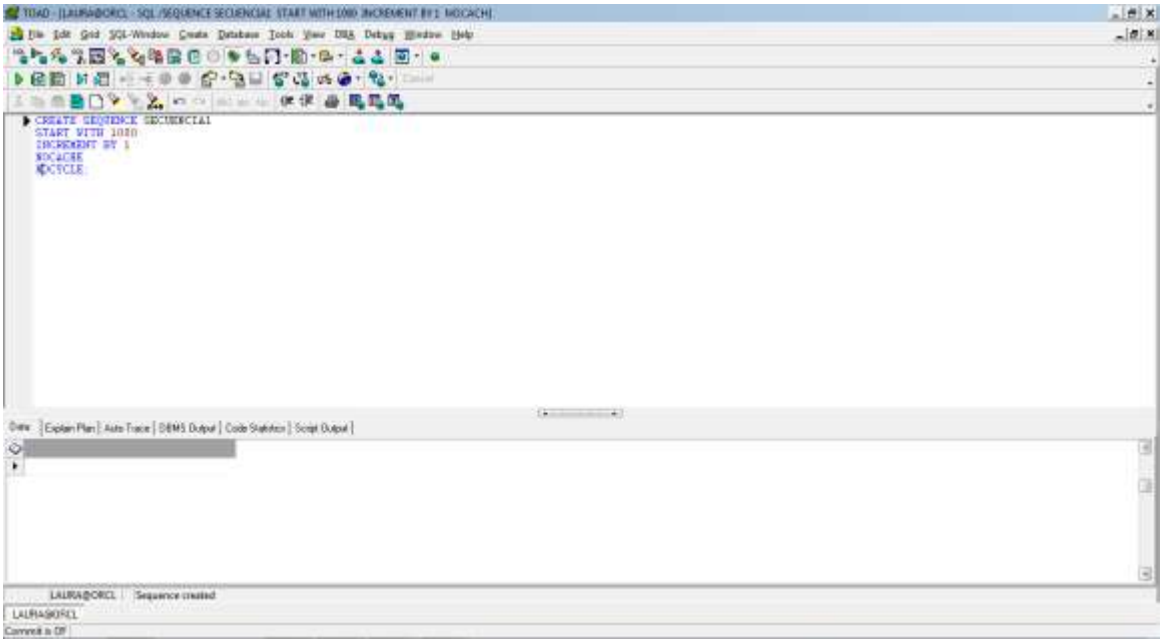


En el ejemplo se muestra que también se le pueden realizar comentarios a las columnas de las tablas.

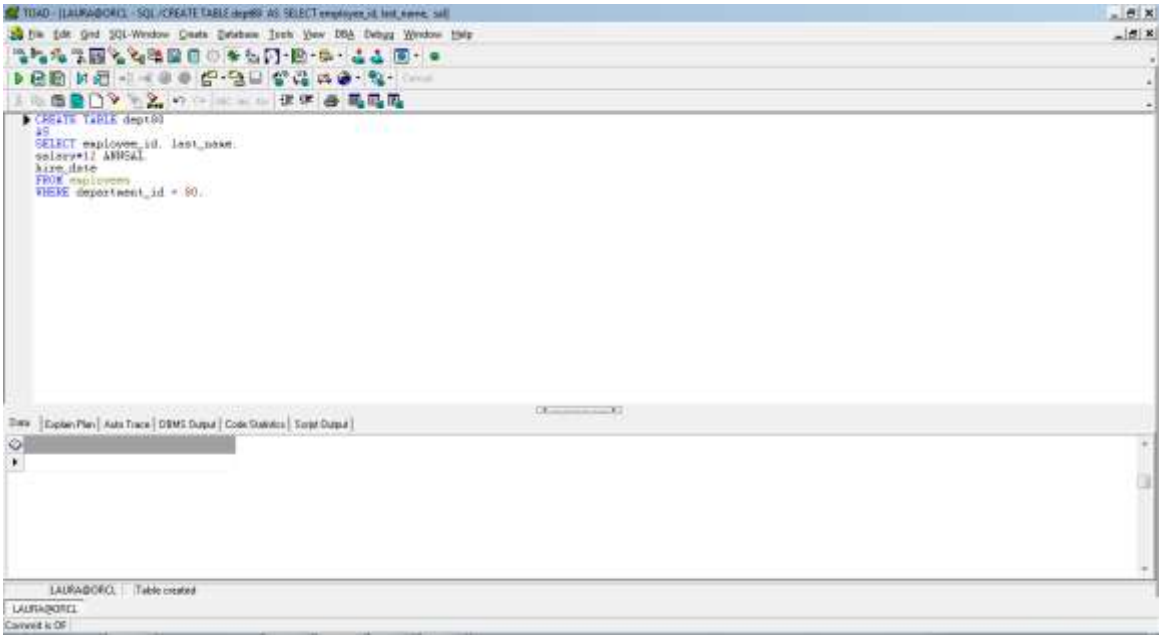


SECUENCIAS:

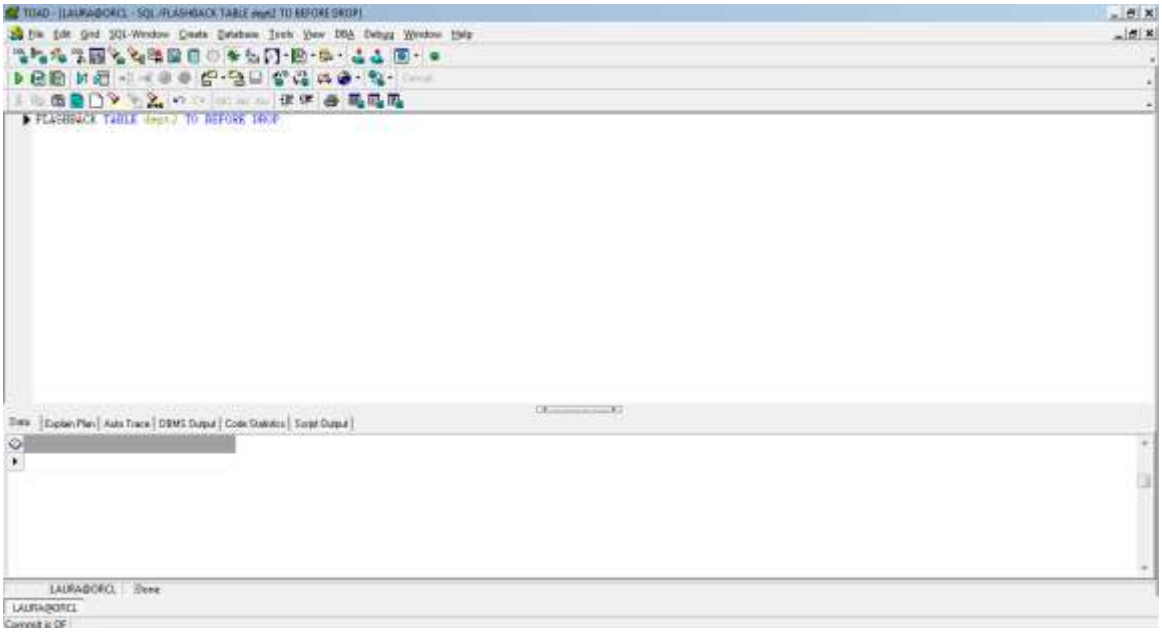
En la secuencia siguiente se crea una secuencia de nombre SECUENCIA1 que comienza en 1,000 y se incrementa unidad a unidad.



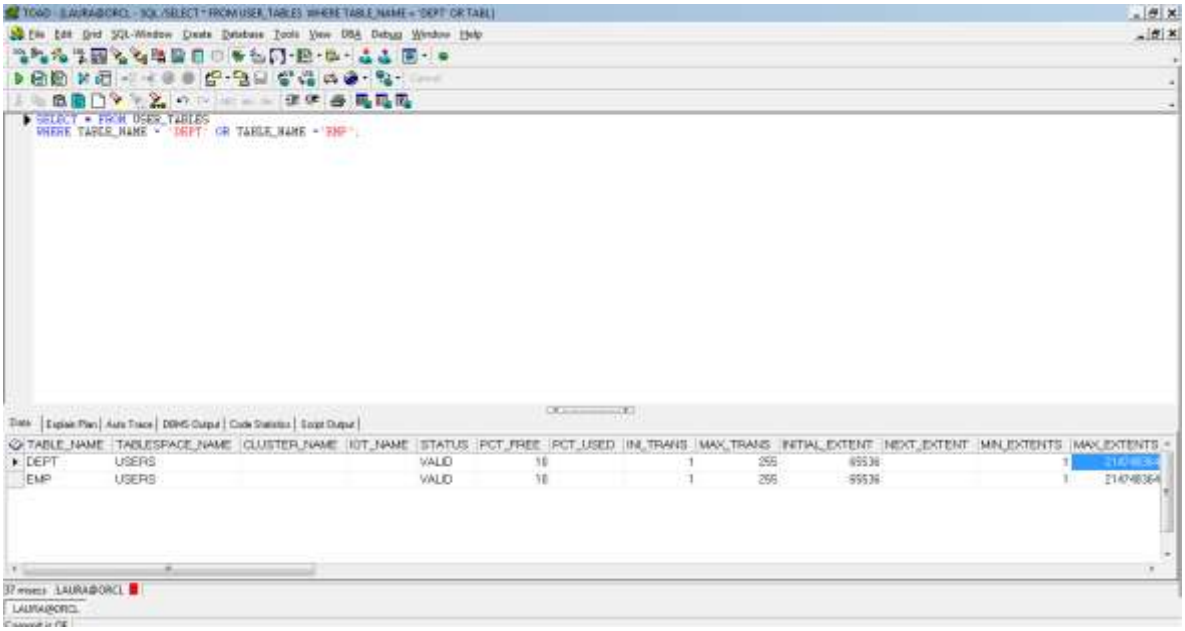
En este ejemplo se puede apreciar la creación de una tabla utilizando una consulta.



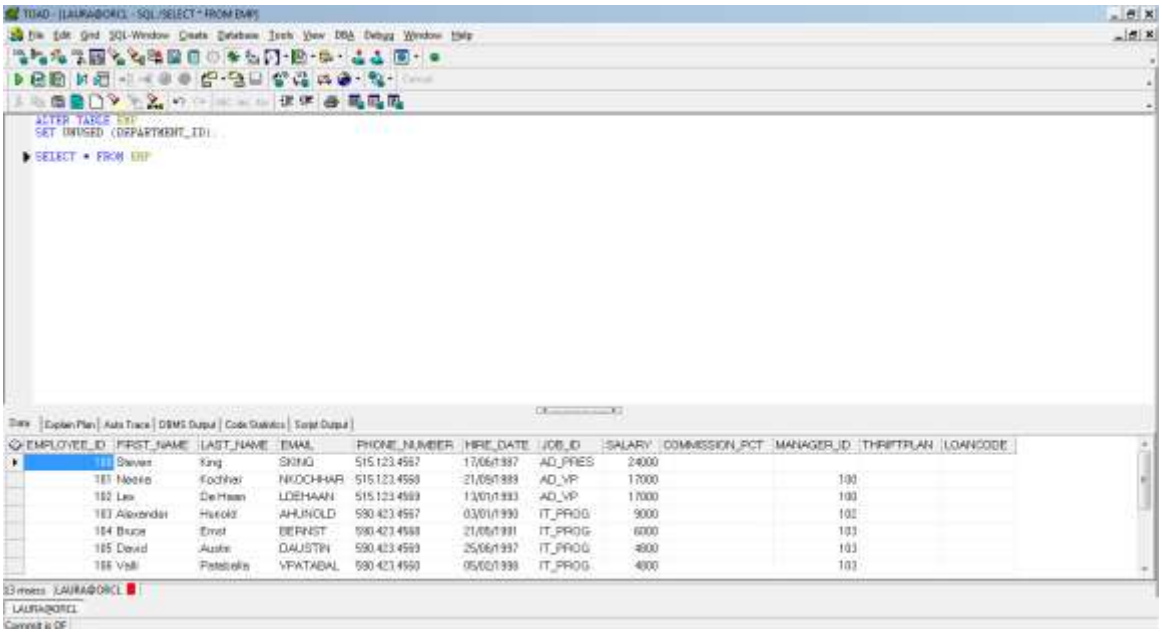
En Oracle se puede recuperar una tabla dependiendo del tamaño de la tabla, es decir, la recupera de un sitio conocido también como papelera de reciclaje. Como se muestra a continuación.



En este ejemplo se puede apreciar como las tablas EMP y DEPT se almacenan en el diccionario de datos mediante la indicación USER_TABES.



En el siguiente ejemplo se marca en la tabla EMP, la columna *department_id como UNUSED* (inutilizable). Se confirma su modificación.



Fuentes

BIBLIOGRAFIAS:

Título: ORACLE 9*i* Guía de aprendizaje

Autor: Michael Abbery, Mike Corey, Ian Abramson

Edición: Oracle Press Oficial, 2002

Editorial: Mc Graw Hill

Título: Oracle 10g Administración y análisis de bases de datos

Autor: César Pérez

Edición: 2° Edición, México, marzo 2008

Editorial: Alfaomega Grupo Editor

http://www.slideshare.net/calderonperaza/utilizando-ddl-sql-oracle-z051-cap-11

http://www.sceu.frba.utn.edu.ar/formacion/science/oracle/Oracle-10g-SQL-Introductorio

http://ora.u440.com/ddl/create%20index.html