[](http://www.google.com/imgres?imgurl=http://www.addenda.com.mx/blog/wp-content/uploads/2011/03/sql-server-logo-411303.jpg&imgrefurl=http://www.addenda.com.mx/blog/?cat=6&usg=__B61t7_HO-JZUMCtbXG3U48e-qso=&h=262&w=319&sz=17&hl=es&start=8&zoom=1&tbnid=ItihwXdKCTv8aM:&tbnh=97&tbnw=118&ei=6bWoTargJ-qY0QHk-8X5CA&prev=/search?q=SQL&hl=es&biw=1051&bih=559&gbv=2&tbm=isch&itbs=1) LENGUAJE DE CONSULTA ESTRUCTURADO (SQL)

**Microsoft SQL Server**

**(Lenguaje de Base de Datos Normalizado)**

[](http://www.google.com/imgres?imgurl=http://www.addenda.com.mx/blog/wp-content/uploads/2011/03/sql-server-logo-411303.jpg&imgrefurl=http://www.addenda.com.mx/blog/?cat=6&usg=__B61t7_HO-JZUMCtbXG3U48e-qso=&h=262&w=319&sz=17&hl=es&start=8&zoom=1&tbnid=ItihwXdKCTv8aM:&tbnh=97&tbnw=118&ei=6bWoTargJ-qY0QHk-8X5CA&prev=/search?q=SQL&hl=es&biw=1051&bih=559&gbv=2&tbm=isch&itbs=1)

**Clase: 19/04/11**

CREACION DE UNA BASE DE DATOS

GO

CREATE DATABASE [relacionesx]

On Primary

(NAME ='relacionesx\_Data1',

FILENAME ='D:\relacionesx\_Data1.Mdf',

SIZE = 10Mb,

MAXSIZE= 20Mb,

FILEGROWTH = 1Mb)

COLLATE Modern\_Spanish\_CI\_AS

CREACION DE TABLAS

go

use relacionesx;

--creando la tabla cliente

create table Cliente

(cod\_cliente int not null,

nom\_cliente varchar(30),

dir\_cliente varchar(40),

cod\_distrito int ,

tel\_cliente varchar(10))

--estableciendo pk

alter table cliente

add constraint pk\_cliente

primary key(cod\_cliente)

--creando tabla distrito

create table distrito

(cod\_distrito int not null,

nom\_distrito varchar(40))

--estableciendo pk

alter table distrito

add constraint pk\_distrito

primary key(cod\_distrito)

--relacionando las tablas

alter table cliente

add constraint fk\_cliente\_distrito

foreign key(cod\_distrito)references distrito

--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MODIFICAR EL NOMBRE DE UNA BASE DE DATOS

GO

USE [relaciones];

GO

ALTER DATABASE relaciones

Modify Name = xx ;

GO

AGREGAR REGISTRO A UNA TABLA

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

USE relacionesx;

insert into distrito values(1,'victoria')

insert into distrito values (2,'peru')

AGREGAR REGISTROS A UNA TABLA MEDIANTE UN ARCHIVO DE TEXTO

TODAS LAS TAREAS

Importar Datos.

CONECTAR PHP CON SQL 2000

NOTA: PRIMERO CONFIGURAR PHP,SYSTEM32

1.      Editar el archivo **c:windowsphp.ini** en la línea 585.

;extension=php\_mhash.dll

;extension=php\_mime\_magic.dll

extension=php\_ming.dll

**;extension=php\_mssql.dll** Quitar el **“;”** para activar las extensiones de Ms SQL.

;extension=php\_msql.dll

;extension=php\_oci8.dll

;extension=php\_openssl.dll

2.      Copiar el archivo **C:AppServphpextphp\_mssql.dll** (en caso sea necesario) de  a la carpeta **c:windowssystem32**

3.      Reiniciar el servidor **Apache.**

<?

//esta parte realiza la conexión al servidor, `se tiene configurado autentificación de Windows, no es necesario ni user ni password.

$conection = mssql\_connect("CARMONAU-1D83A6","","") or die("no se puede conectar a SQL Server");

//aqui selecciono la base Northwind que ya trae MS SQL v.8 como ejemplo

mssql\_select\_db("Northwind",$conection);

//tu consulta la asignas a un string $SQL

$SQL = "SELECT \* FROM Products";

//lo demas es muy sencillo solo estudialo

?>

<html>

<head>

<title>Conexion mssql</title>

</head>

<body>

<table border="1" cellspacing="3" cellpading="4">

<tr>

<td>Product ID</td>

<td>Product Name</td>

<td>Suplier ID</td>

<td>Category ID</td>

<td>Quantity per Unit</td>

<td>Unit Price</td>

</tr>

<?

$result = mssql\_query( $SQL );

while( $row = mssql\_fetch\_array( $result ) ){

?>

<tr>

<td> <? echo $row['ProductID'] ?> </td>

<td> <? echo $row['ProductName'] ?> </td>

<td> <? echo $row['SupplierID'] ?> </td>

<td> <? echo $row['CategoryID'] ?> </td>

<td> <? echo $row['QuantityPerUnit'] ?></td>

<td> <? echo $row['UnitPrice'] ?></td>

</tr>

<?

}

?>

</table>

<?

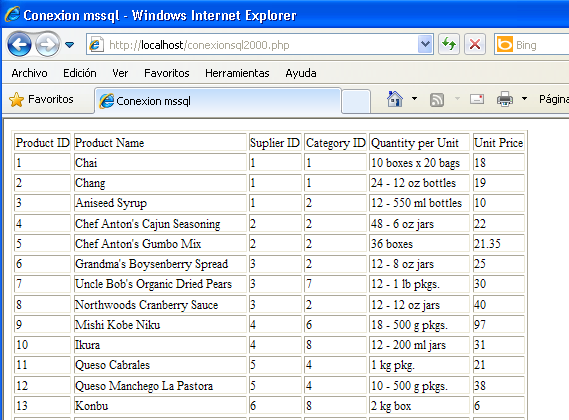
mssql\_close( $conection );

?>

</body>

</html>

Resultado :



**Mssql Funciones**

**Table of Contents**

* [mssql\_bind](http://cl.php.net/manual/es/function.mssql-bind.php) — Adds a parameter to a stored procedure or a remote stored procedure
* [mssql\_close](http://cl.php.net/manual/es/function.mssql-close.php) — Close MS SQL Server connection
* [mssql\_connect](http://cl.php.net/manual/es/function.mssql-connect.php) — Open MS SQL server connection
* [mssql\_data\_seek](http://cl.php.net/manual/es/function.mssql-data-seek.php) — Moves internal row pointer
* [mssql\_execute](http://cl.php.net/manual/es/function.mssql-execute.php) — Executes a stored procedure on a MS SQL server database
* [mssql\_fetch\_array](http://cl.php.net/manual/es/function.mssql-fetch-array.php) — Fetch a result row as an associative array, a numeric array, or both
* [mssql\_fetch\_assoc](http://cl.php.net/manual/es/function.mssql-fetch-assoc.php) — Returns an associative array of the current row in the result
* [mssql\_fetch\_batch](http://cl.php.net/manual/es/function.mssql-fetch-batch.php) — Returns the next batch of records
* [mssql\_fetch\_field](http://cl.php.net/manual/es/function.mssql-fetch-field.php) — Get field information
* [mssql\_fetch\_object](http://cl.php.net/manual/es/function.mssql-fetch-object.php) — Fetch row as object
* [mssql\_fetch\_row](http://cl.php.net/manual/es/function.mssql-fetch-row.php) — Get row as enumerated array
* [mssql\_field\_length](http://cl.php.net/manual/es/function.mssql-field-length.php) — Get the length of a field
* [mssql\_field\_name](http://cl.php.net/manual/es/function.mssql-field-name.php) — Get the name of a field
* [mssql\_field\_seek](http://cl.php.net/manual/es/function.mssql-field-seek.php) — Seeks to the specified field offset
* [mssql\_field\_type](http://cl.php.net/manual/es/function.mssql-field-type.php) — Gets the type of a field
* [mssql\_free\_result](http://cl.php.net/manual/es/function.mssql-free-result.php) — Free result memory
* [mssql\_free\_statement](http://cl.php.net/manual/es/function.mssql-free-statement.php) — Free statement memory
* [mssql\_get\_last\_message](http://cl.php.net/manual/es/function.mssql-get-last-message.php) — Returns the last message from the server
* [mssql\_guid\_string](http://cl.php.net/manual/es/function.mssql-guid-string.php) — Converts a 16 byte binary GUID to a string
* [mssql\_init](http://cl.php.net/manual/es/function.mssql-init.php) — Initializes a stored procedure or a remote stored procedure
* [mssql\_min\_error\_severity](http://cl.php.net/manual/es/function.mssql-min-error-severity.php) — Sets the minimum error severity
* [mssql\_min\_message\_severity](http://cl.php.net/manual/es/function.mssql-min-message-severity.php) — Sets the minimum message severity
* [mssql\_next\_result](http://cl.php.net/manual/es/function.mssql-next-result.php) — Move the internal result pointer to the next result
* [mssql\_num\_fields](http://cl.php.net/manual/es/function.mssql-num-fields.php) — Gets the number of fields in result
* [mssql\_num\_rows](http://cl.php.net/manual/es/function.mssql-num-rows.php) — Gets the number of rows in result
* [mssql\_pconnect](http://cl.php.net/manual/es/function.mssql-pconnect.php) — Open persistent MS SQL connection
* [mssql\_query](http://cl.php.net/manual/es/function.mssql-query.php) — Send MS SQL query
* [mssql\_result](http://cl.php.net/manual/es/function.mssql-result.php) — Get result data
* [mssql\_rows\_affected](http://cl.php.net/manual/es/function.mssql-rows-affected.php) — Returns the number of records affected by the query
* [mssql\_select\_db](http://cl.php.net/manual/es/function.mssql-select-db.php) — Select MS SQL database

[mssql\_bind[>](http://cl.php.net/manual/es/function.mssql-bind.php)](http://cl.php.net/manual/es/function.mssql-bind.php) [[<](http://cl.php.net/manual/es/mssql.constants.php)Constantes predefinidas](http://cl.php.net/manual/es/mssql.constants.php)

**Clase 3: 21/04/11**

PRACTICAS (CLASE 3)

=======================

1. Crear una Base de datos Practica1 con los parámetros .En forma predeterminada y mostrar la información como resultado en cuadricula
2. Crear la base de datos practica2 con un archivo de datos de 15Mb, un tamaño máximo de 30 Mb y un crecimiento de 1Mb, el archivo registros debe asumir los valores por defecto.

Para mostrar un informe de una base de datos

o de todas las bases de datos de un

servidor con SQL Server, ejecute sp\_helpdb.

Para obtener un informe acerca del espacio

utilizado en una base de datos, emplee sp\_spaceused.

Para obtener un informe de los grupos de archivos de

una base de datos, utilice sp\_helpfilegroup,

y utilice sp\_helpfile para obtener el informe de

los archivos de la base de datos.

**Clase 21/04/11 (Profesor :Jorge Carmona E. Telesup)**

COMANDOS DDL

**PRACTICAS:**

**Creando la Base de Datos:**

GO

CREATE DATABASE [labiblioteca]

On Primary

(NAME ='labiblioteca\_Data1',

FILENAME ='D:\labiblioteca\_Data1.Mdf',

SIZE = 10Mb,

MAXSIZE= 20Mb,

FILEGROWTH = 1Mb)

COLLATE Modern\_Spanish\_CI\_AS

go

use labiblioteca;

--creando la tabla alumno

create table Alumno

(id\_alumno char(4) not null,

nom\_alumno varchar(25),

dir\_alumno varchar(40),

dis\_alum varchar(40))

-- estableciendo pk

-- estableciendo el índice idx

alter table alumno

add constraint pk\_alumno

primary key(id\_alumno)

create index idx\_alumno on alumno (nom\_alumno)

--creando la tabla Libro

create table Libro

(id\_libro integer not null,

nomlibro varchar(20),

fechaedicion datetime)

-- estableciendo pk

--estableciendo el índice idx

alter table Libro

add constraint pk\_libro

primary key(id\_libro)

create index idx\_libro on Libro (fechaedicion)

--creando la tabla préstamo

create table Prestamo

(id\_prestamo integer not null,

id\_alumno char(4) not null,

id\_libro integer not null,

fechaprestamo datetime)

-- estableciendo pk

alter table Prestamo

add constraint pk\_prestamo

primary key(id\_prestamo,id\_alumno,id\_libro)

--relacionando las 3 tablas

alter table Prestamo

add constraint fk\_prestamo\_alumno

foreign key(id\_alumno)references Alumno,

constraint fk\_prestamo\_libro

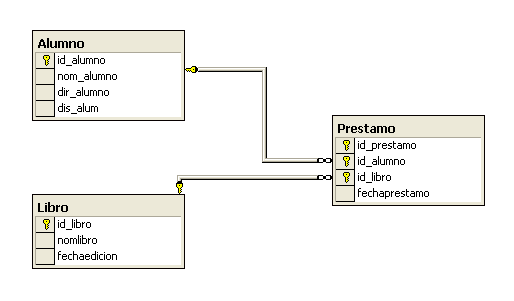
foreign key(id\_libro)references Libro

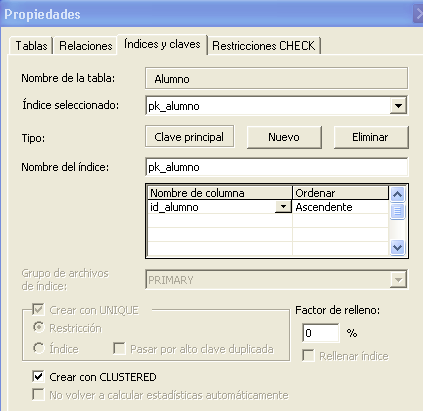
ó (otra forma similar)

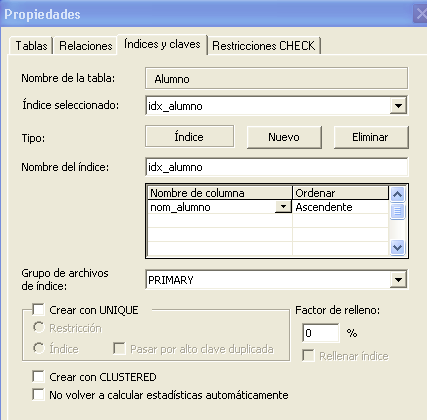
alter table Prestamo

ADD CONSTRAINT fk\_prestamo\_alumno FOREIGN KEY(id\_alumno) REFERENCES Alumno(id\_alumno), CONSTRAINT fk\_prestamo\_libro FOREIGN KEY(id\_libro) REFERENCES Libro(id\_libro)

Resultado :







CREATE TABLE nombre Tabla   
      (nombre\_campo1 Tipo\_de\_variable  [(tamaño)] [Null/Not Null] [Primay Key] [Check (condición)],  
       nombre\_campo2 Tipo\_de\_variable  [(tamaño)] [Null/Not Null] [Primay Key],  
       [FOREIGN KEY(Clave Secundaria) REFERENCES Tabla\_Primaria(Clave\_Primaria\_Tabla\_Primaria1, [Clave\_Primaria\_Tabla\_Primaria2)]  
       [CONSTRAINT pk\_NombreRestriccion PRIMARY KEY (clave1, clave2)]  
       [CONSTRAINT fk\_NombreRestriccion FOREIGN KEY (clave secundaria) REFERENCES TablaPrimaria(ClavePrimaria)]  
      etc... );

**Create Index:** Se utilizan para que las búsquedas por ciertos campos sean más ágiles. No es recomendable cuando los datos que se muestren sean superiores al 5% del total. El índice puede ser **unívoco** (cuando no se admiten campos duplicados en dicho índice) o **no unívoco**.

CREATE [UNIQUE] INDEX Nombre Indice  
     ON Nombre de tabla (Nombre del Campo) [ASC/DESC];

          Las columnas deben de ir por orden de prioridad.  
          UNIQUE: Si se emplea,el índice comprobará, e impedirá, que no se introduzca un valor repetido en la columna con la que está asociado.  
          ASC/DESC: Indica si el índice lleva orden ascendente o descendente.

Ejemplo

CREATE UNIQUE INDEX ui\_Cliente\_Nombre  
     ON Cliente (Cl\_Nombre);

DIFERENCIAS ENTRE UN CLUSTERED Y UN INDICE CLUSTERED

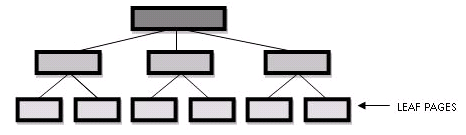
La diferencia entre un índice clustered y un índice non-clustered y en qué caso conviene usar un índice u otro.

Los índices son objetos de la bases de datos, cuya función es optimizar el acceso a datos. A medida que las tablas se van haciendo más grandes y se desea hacer consultar sobre estas tablas, los índices son indispensables.

Internamente un índice normal es una estructura de árbol, que cuenta con una página principal y luego esta con paginas hijas, que a su vez tiene más paginas hijas hasta llegar a la pagina final del índice (leaf level).

La clave del índice está repartida en las páginas del índice, de modo tal que la búsqueda se haga leyendo la menor cantidad posible de datos.

**Estructura interna de un índice:**



Después de esta brevísima introducción, donde está la diferencia entre un índice clustered y uno non-clustered? En la leaf level (la ultima pagina) del índice. **En un índice non-clustered, la clave por la que buscamos tiene un puntero a la página de datos donde se encuentra el registro. Mientras que en índice clustered, la leaf level es la pagina de datos!. Con lo cual, el SQL Server, se ahorra hacer un salto para leer los datos del registro (Bookmark lookup).** La diferencia es importante, ya que el uso de este tipo de índices al evitar tener que hacer lecturas adicionales para traer el registro, son más performantes.

|  |  |
| --- | --- |
| **Búsqueda por clustered index:**  http://grimpidev.files.wordpress.com/2008/03/030808-2111-diferencias216.png | **Búsqueda por non-clustered index:**  http://grimpidev.files.wordpress.com/2008/03/030808-2111-diferencias317.png |
| SQL Server 2005 incorpora una nueva feature interesante en los índices non-clustered. Ahora es posible incluir dentro de la leaf page del índice, campos que en sí, no son parte de la clave. Esto nos permitirá en algunos casos, evitar el salto a la página de datos (Bookmark Lookup) que habíamos hablado anteriormente. Aunque hay que tener cuidado de seleccionar bien que campos se desean incluir al índice, porque de poner demasiados se expandería mucho el índice, haciendo ineficiente. Por ejemplo, si tenemos una tabla Personas cuyo campo DNI es un índice non-clustered y queremos hacer una consulta que solo traiga el Apellido y DNI, entonces si incluimos el campo Apellido , nos ahorraríamos tener que ir a la página de datos para buscar el valor. Es importante recalcar que el campo Apellido no sería parte de la tabla, sino un campo mas en pagina final del índice. | |

Ahora bien, entonces porque no siempre usar índices clustered? Bueno, en primer lugar, lamentablemente solo puede haber 1 solo índice clustered por tabla. La razón es muy sencilla y lógica: Los registros de la tabla físicamente son las paginas leaf-level del índice clustered. Los datos de la tabla esta ordenados según el índice. Y obviamente una tabla no puede simultáneamente estar físicamente ordenada de 2 maneras diferentes.  
Por lo tanto, en tablas grandes y muy consultadas, tenemos que ser cuidadosos y analizar a que campos vamos a seleccionar para ser llaves del índice clustered. Tenemos 1 solo índice de este tipo por tabla, no hay que desperdiciarlo!!!  
Este último punto es importante para saber en qué situaciones y para que campos se debe utilizar un clustered index o un non-clustered.

Guía general de uso de índices:

* Campos autoincrementales (Identitys, newsequentialid, etc), deben convenientemente ser del tipo clustered index. La razón es reducir el page split (fragmentación) de la tabla.
* Los clustered index son convenientes si se va seleccionar un rango de valores, ordenar (ORDER BY) o agrupar (GROUP BY).
* La PK es un buen candidato para un clustered index. Pero no siempre. Por ejemplo, si tenemos una tabla de ventas, cuya PK es un identity en donde se efectúan muchas consultar por rangos de fecha, el campo Fecha seria un mejor candidato para el clustered que la PK.
* Para búsquedas de valores específicos, conviene utilizar un non-clustered index.
* Para índices compuestos, mejor utilizar non-clustered index (generalmente).

**Alter Table**: Esta instrucción DDL se suele utilizar para cambiar características de las tablas, como pueden ser insertar campos, modificar campos, añadir restricciones (CHECK, Claves, etc.). Al ser una función tan diversa,

ADD: Se emplea para añadir un nuevo archivo de control a la base de datos  
DROP: Permite borrar un archivo de control  
RENAME: Permite renombrar un archivo de control

**Añadir Restricción de Clave Primaria**: Si no se añade la clave primaria a la hora de crear la tabla, se puede anexar después.

ALTER TABLE Nombre Tabla  
     ADD CONSTRAINT Nombre\_Restricción   
     PRIMARY KEY (Nombre de Campo1, Nombre de Campo2, ... etc);

Ejemplo

ALTER TABLE Cliente  
      ADD CONSTRAINT pk\_Codigo\_Cliente   
      PRIMARY KEY (Cl\_Cliente);

**Añadir Restricción de Clave Secundaria**:  Se utiliza para definir la relación entre dos tablas. Es necesario que en la tabla referenciada esté definida la Primary Key, porque la relación se crea entre la Primary Key de la tabla referenciada y las columnas que indicamos en la cláusula Foreign Key.  
  
          ALTER TABLE Nombre Tabla  
                  ADD CONSTRAINT Nombre\_Restricción  
                  FOREIGN KEY (Nombre de Campo1, Nombre de Campo 2, .. etc..)  
                  REFERENCES Tabla Referenciada;

Ejemplo

          ALTER TABLE Cliente  
                   ADD CONSTRAINT fk\_Codigo\_Cliente  
                   FOREIGN KEY (Cl\_Cliente)  
                   REFERENCES Nominas;

**Añadir Campo Nuevo a Tabla**:  Sirve para añadir un nuevo campo (una nueva columna) a una tabla ya creada. Debe de tenerse en cuenta que sólo se puede añadir una columna por sentencia.

ALTER TABLE Nombre Tabla  
     ADD (Nombre\_Campo Tipo\_de\_Variable [(tamaño)] [Null/Not Null])  
  
La restricción NOT NULL sólo puede aplicarse si la tabla está vacía, es decir, aún no tiene ninguna fila. Para crear una nueva columna con la restricción NOT NULL. Si la tabla ya tiene filas (datos), primero deberá añadirse la columna sin la restricción, rellenar la columna, y modificarla después para que tenga la restricción NOT NULL.

Ejemplo

ALTER TABLE Clientes  
    ADD (Cl\_Poblacion Varchar2 (30) );

**Modificar Campo de una Tabla**: Podemos cambiar alguna de las características de un campo, también con este comando

ALTER TABLE Nombre Tabla  
      MODIFY Nombre\_Campo Nuevas\_Condiciones

No se pueden borrar ni modificar nombres de columnas ya existentes. Sólo se pueden modificar las variables de tipo carácter a otras variables de tipo carácter si toda la columna está compuesta de NULL, pero sí está permitido aumentar el tamaño de las variables carácter. Si puede cambiarse el tamaño de las variables numéricas. Las modificaciones mediante DEFAULT de una columan afectarán sólo a los datos que se introduzcan después de la modificación, no a laos anteriores.

Ejemplo

ALTER TABLE Clientes  
       MODIFY Nombre Varchar(30) not null;

En este caso, lo que hemos hecho es cambiar de un tamaño 10 que había en el momento de crear el campo a uno de tamaño 30, añadiendole también que no se admiten campos nulos en dicho campo.  
  
Otro caso sería poner un campo por defecto, en este cao, quremos que el número de vacantes por defecto para cualquier curso sea de 20

ALTER TABLE Curso  
     MODIFY (Vacantes default 20);

**Reglas de Validación (Restricción Check):** Las reglas de validación son muy importantes porque permiten establecer una condición a los valores que debe aceptar una columna.

ALTER TABLE Nombre Tabla  
        ADD CONSTRAINT CK\_NombreTabla\_NombreCampo  
        CHECK (Condición);  
  
Ejemplo:  
  
ALTER TABLE Curso  
         ADD CONSTRAINT CK\_Curso\_PreCurso  
         CHECK ( Precurso > 0 );

[Subir Arriba](http://www.alu.ua.es/m/maab5/PL_SQL%20DDL.html#%C3%8Dndice)

**Alter Constraint** : En muchos casos debemos garantizar que los valores de una columna o conjunto de columnas de una tabla acepten sólo valores únicos

ALTER CONSTRAINT Nombre Tabla  
        ADD CONSTRAINT Nombre Restricción  
        UNIQUE (Nombre Campo 1, Nombre Campo 2, ... etc.);  
  
Ejemplo  
  
ALTER TABLE Alumno  
       ADD CONSTRAINT U\_Alumno\_NomAlumno  
       UNIQUE (NomAlumno);

[Subir Arriba](file:///M:\FTP_Manuales\PL_SQL%20DDL.html#%C3%8Dndice)

**Drop:** Este comando se utiliza para eliminar objetos de nuestra base de datos. Se suele utilizar la palabra Drop, a continuación el nombre del objeto que se desea eliminar (por ejemplo, Table) y por último, el nombre de dicho objeto (Clientes). Veamos algunos ejemplos:

DROP Nombre\_Objeto Nombre\_Elemento [CASCADE CONSTRAINTS];  
  
Si se borra una tabla, se borran, a su vez, los índices a ella asociados, vistas, vistas creadas a partir de dichas vistas, etc.  
Si se borra una tabla a la que otras hacen referencia en claves externas habrá que incluir la cláusula CASCADE CONSTRAINT  
  
Ejemplos:  
  
DROP TABLE Clientes  
DROP VIEW Informe\_Clientes\_vw  
DROP INDEX pk\_Codigo\_Cliente

**Otros ejemplos:**

create database renta\_auto

go

use renta\_auto

create table cliente (

idcliente int not null,

nombre varchar (50),

direccion varchar (50),

telefono varchar (50),

notarjeta bigint

)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

go

create table automóvil (

placa varchar (9) not null,

nombre varchar (50),

marca varchar (50),

modelo int,

transmision varchar (50),

color varchar (50)

)

go

create table renta (

idcliente int not null,

placa varchar (9) not null,

fecha smalldatetime not null,

kilometraje int not null,

precio money not null

)

go

ALTER TABLE cliente

add constraint PK\_cliente primary key (idcliente)

go

alter table automovil

add constraint PK\_automovil primary key (placa)

go

alter table renta

add constraint PK\_renta primary key (idcliente, placa, fecha)

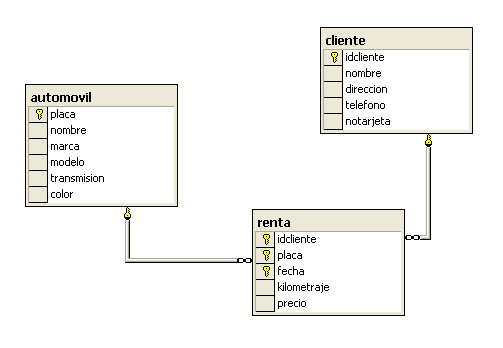
go

ALTER TABLE renta

ADD CONSTRAINT FK\_renta\_cliente FOREIGN KEY (idcliente) REFERENCES cliente (idcliente),

CONSTRAINT FK\_renta\_automovil FOREIGN KEY (placa) REFERENCES automovil (placa)

**Resultado:**



**Otro Ejemplo:**

**CREATE TABLE TIPO\_USUARIO**

**(**

**idTipoUsuario int Identity(1,1),**

**descTipoUsuario varchar(20) NOT NULL,**

**CONSTRAINT PK\_TIPO\_USUARIO PRIMARY KEY(idTipoUsuario)**

**)**

**GO**

**CREATE TABLE USUARIO**

**(**

**idUsuario char(8)NOT NULL,**

**idTipoUsuario int NOT NULL,**

**apelUsuario varchar(35)NOT NULL,**

**nomUsuario varchar(35) NOT NULL,**

**direccion varchar(50)NULL,**

**habilitado bit NOT NULL,**

**fechaExpCarnet smalldatetime NOT NULL,**

**fechaVencCarnet smalldatetime NOT NULL,**

**CONSTRAINT PK\_USUARIO\_idUsuario PRIMARY KEY(idUsuario),**

**CONSTRAINT FK\_USUARIO\_idTipoUsuario FOREIGN KEY (idTipoUsuario) REFERENCES TIPO\_USUARIO(idTipoUsuario)**

**)**

**GO**

**CREATE TABLE MAT\_BIBLIO**

**(**

**idMatBiblio varchar(20)NOT NULL,**

**tituloMatBiblio varchar(150)NOT NULL,**

**editorial varchar(50)NULL,**

**pais varchar(20)NULL,**

**año smalldatetime NULL,**

**nPag int NULL,**

**existencia int NOT NULL,**

**CONSTRAINT PK\_MAT\_BIBLIO\_idMatBiblio PRIMARY KEY(idMatBiblio)**

**)**

**CREATE TABLE PRESTAMO**

**(**

**codOper char(7)NOT NULL,**

**idMatBiblio varchar(20)NOT NULL,**

**idUsuario char(8)NOT NULL,**

**fechaP smalldatetime NOT NULL,**

**fechaD smalldatetime NOT NULL,**

**ndias int NOT NULL,**

**CONSTRAINT PK\_PRESTAMO\_codOper PRIMARY KEY(codOper),**

**CONSTRAINT FK\_PRESTAMO\_idMatBiblio FOREIGN KEY(idMatBiblio) REFERENCES MAT\_BIBLIO(idMatBiblio)**

**)**

**GO**

**CREATE TABLE AUTOR**

**(**

**idAutor char(4)NOT NULL,**

**nomAutor varchar(50)NOT NULL,**

**CONSTRAINT PK\_AUTOR\_idAutor PRIMARY KEY(idAutor)**

**)**

**GO**

**CREATE TABLE MB\_AUTOR**

**(**

**idMatBiblio varchar(20)NOT NULL,**

**idAutor char(4)NOT NULL,**

**CONSTRAINT PK\_MB\_AUTOR\_idMatBiblio\_idAutor PRIMARY KEY(idMatBiblio,idAutor),**

**CONSTRAINT FK\_MB\_AUTOR\_idMatBiblio FOREIGN KEY(idMatBiblio) REFERENCES MAT\_BIBLIO(idMatBiblio),**

**CONSTRAINT FK\_MB\_AUTOR\_idAutor FOREIGN KEY(idAutor) REFERENCES AUTOR(idAutor)**

**)**

**GO**

Resultado:

