



BASE DE DATOS

DDL

Lenguaje de Definición de Datos

Lenguaje de Descripción de Datos

ÍNDICE

- INTRODUCCIÓN
- CREATE DATABASE
- CREATE TABLE
 - Estructura
 - TIPS
 - Estructura con Restricciones
 - NOT NULL
 - CONSTRAINTS (Primary Key, Unique, Foreign Key,...)
- TIPOS DE DATOS
- ALTER TABLE
 - Operaciones
 - Estructura
- DROP TABLE
- INTEGRIDAD REFERENCIAL



INTRODUCCIÓN

- LENGUAJE QUE DEFINE LA **ESTRUCTURA**.
- Su especificación puede cambiar de un gestor de base de datos a otro.
- Define como el **sistema organiza internamente los datos**.
- Se encarga de la **creación, modificación y eliminación de los objetos** de la base de datos (es decir de los **metadatos**).

INTRODUCCIÓN

- Una **base de datos** posee un **esquema**. El esquema suele tener el mismo nombre que el usuario y **sirve para almacenar los objetos de esquema**, es decir los objetos que posee el usuario.

INTRODUCCIÓN

- CREATE objeto → Crear
- ALTER objeto → Modificar
- DROP objeto → Eliminar
- RENAME objeto → Renombrar
- TRUNCATE objeto → Eliminar

CREATE DATABASE

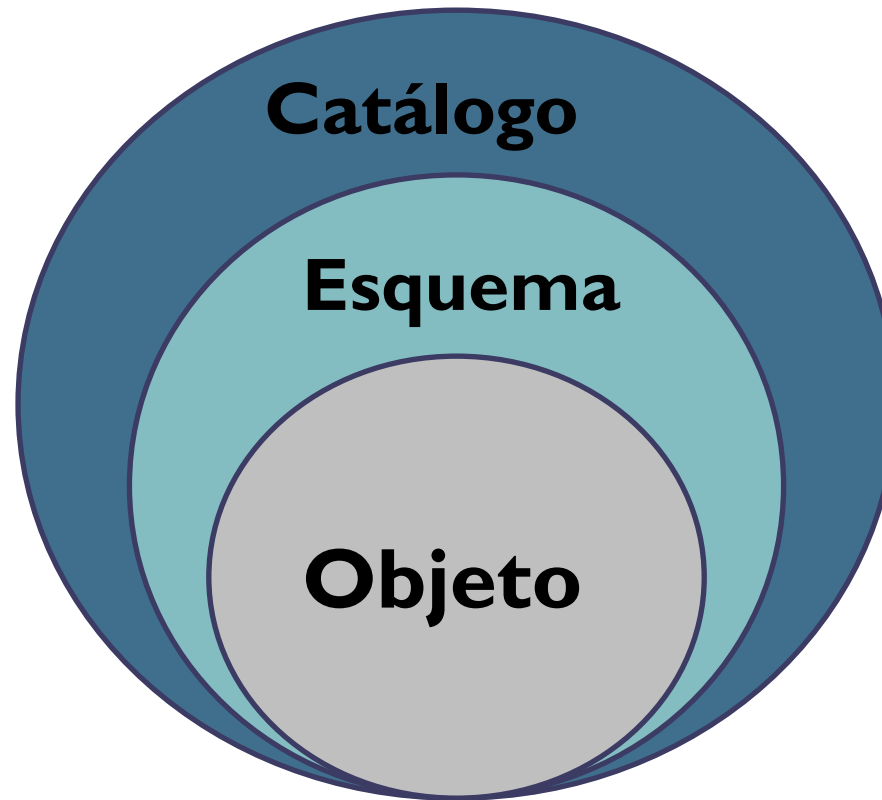
- Se requiere especificar los archivos y ubicaciones que se utilizarán para la misma, además de otras indicaciones técnicas y administrativas que no se comentarán en este tema.

CREATE DATABASE prueba;

CREATE DATABASE

- Una base de datos es un **conjunto de objetos pensados para gestionar datos**. En particular existe la siguiente organización:

Catálogo.Esquema.Objeto



CREATE TABLE

- Permite crear una tabla
- Permite **definir** **las**
columnas **y** **las**
restricciones en estas.

CREATE TABLE - ESTRUCTURA

```
CREATE TABLE nombreTabla (  
    nombreColumna        tipoDatos,  
    nombreColumna        tipoDatos,  
    nombreColumna        tipoDatos  
);
```

Tipos datos, varían dependiendo del motor de datos:

Para SQL Server:

<http://msdn.microsoft.com/es-es/library/ms187752.aspx>

CREATE TABLE - ESTRUCTURA

```
CREATE TABLE usuario (  
    codigo          int,  
    nombre          varchar(60),  
    clave            varchar(15)  
);
```

CREATE TABLE - TIPS

- Deben comenzar con una **letra**
- **No** deben tener más de **30 caracteres**
- Sólo se permiten utilizar letras del alfabeto (inglés), números o el signo de subrayado (también el signo \$ y #, pero esos se utilizan de manera especial por lo que no son recomendados)
- **No** puede haber **dos tablas con el mismo nombre para el mismo esquema** (pueden coincidir los nombres si están en distintas bases de datos o esquemas)

CREATE TABLE – ESTRUCTURA CON RESTRICCIONES

```
CREATE TABLE nombreTabla (  
    nombreColumna          tipoDatos  
    RESTRICCION,  
    nombreColumna          tipoDatos  
    RESTRICCION,  
    nombreColumna          tipoDatos  
    RESTRICCION  
);
```

CREATE TABLE – ESTRUCTURA CON RESTRICCIONES

- Una **restricción** consiste en la definición de una **característica adicional** que tiene una columna o una combinación de columnas

RESTRICCIONES

Restricciones (CONSTRAINT)
NOT NULL
UNIQUE
PRIMARY KEY
FOREING KEY
CHECK

RESTRICCIONES

- **NOT NULL** indica que la columna no podrá contener un valor nulo
- **CONSTRAINT** sirve para definir una restricción que se podrá eliminar cuando queramos sin tener que borrar la columna. A cada restricción se le asigna un nombre que se utiliza para identificarla y para poder eliminarla cuando se quiera.

Como constraint definimos la de clave primaria (clave principal), la de índice único (sin duplicados), la de valor no nulo y la de clave foránea .

CONSTRAINT – PRIMARY KEY

- Define la columna como **clave principal de la tabla**.
- **Las columnas no puede contener valores nulos.**
- **No pueden haber valores duplicados** en esa columna, es decir que dos filas no pueden tener el mismo valor en esa columna.
- **Sólo hay una clave principal por tabla.**

CONSTRAINT - UNIQUE

- Define un **índice único** sobre la columna. Un índice único es un índice que **no permite valores duplicados**, es decir que si una columna tiene definida una restricción de **UNIQUE** no podrán haber dos filas con el mismo valor en esa columna.
- Se suele emplear para que el sistema compruebe el mismo que no se añaden valores que ya existen.

CONSTRAINT – FOREIGN KEY

- **Es una columna o conjunto de columnas que contiene un valor que hace referencia a una fila de otra tabla. (Clave ajena o foránea)**

CONSTRAINT - CHECK

- CHECK Restricción que **debe cumplir el campo** sobre el cual se define el constraint a **través de una condición de tipo lógica** (condición que determina si algo se cumple o no – **falso ó verdadero**).
- En la condición pueden emplearse;
 - Operadores lógicos relacionales (menor, mayor, etc...)
 - Operadores lógicos booleanos (AND, OR, NOT, etc...)

CREATE TABLE – ESTRUCTURA CON RESTRICCIONES

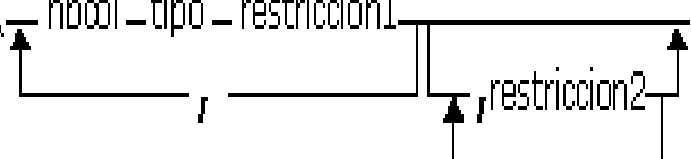
➤ En general:

```
CREATE TABLE usuario (  
    codigo int CONSTRAINT pkNombre PRIMARY KEY,  
    nombre    VARCHAR(25) NOT NULL,  
    identificacion CHAR(10) CONSTRAINT uknombre  
UNIQUE,  
    fec_ingreso datetime,  
    cod_empresa      int CONSTRAINT fkNombre  
REFERENCES empresa(codigo)  
);
```

Este es el concepto básico de cómo se utilizan sin embargo puede variar de una base de datos a otra.

➤ ESTRUCTURA

```
CREATE TABLE _nbtabla ( _nbcol _tipo _restriccion1  
                        , _restriccion2 )
```

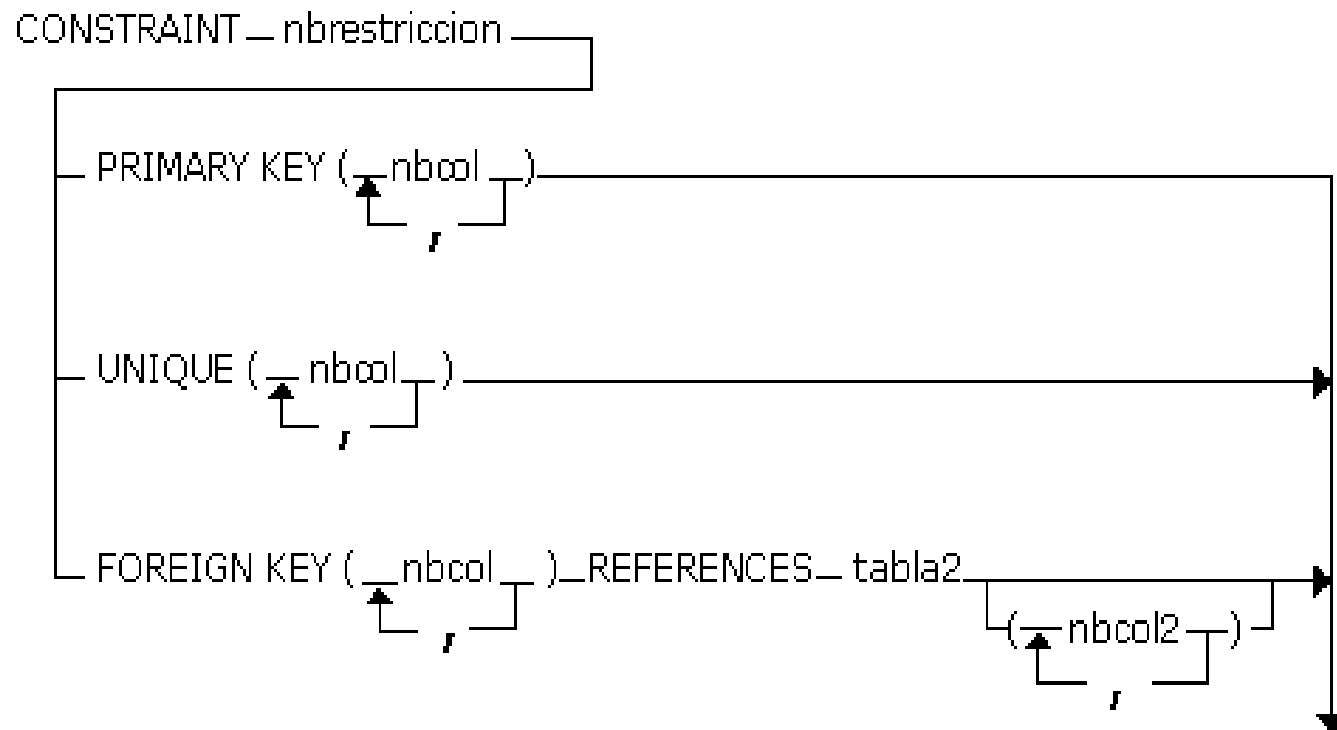


nbtabla: nombre de la **tabla** que estamos definiendo

nbcol: nombre de la **columna** que estamos definiendo

tipo: **tipo de dato** de la columna, todos los datos almacenados en la columna deberán ser de ese tipo.

➤ ESTRUCTURA



OTRA FORMA DE ESPECIFICAR

```
CREATE TABLE tab1 (col1 INTEGER,  
    col2    CHAR(25) NOT NULL,  
    col3    CHAR(10),  
    col4    INTEGER,  
    col5    INT,  
    CONSTRAINT pk PRIMARY KEY (col1),  
    CONSTRAINT uni1 UNIQUE (col3),  
    CONSTRAINT fk5 FOREIGN KEY (col5)  
    REFERENCES tab2 (camporeferencia) );
```

TIPOS DE DATOS

○ Numéricos

➤ *Exactos*

- ❖ Enteros, según el RDBMS: SMALLINT, INTEGER, CURRENCY, MONEY, ...
- ❖ Con precisión y escala, con signo y punto decimal: NUMBER(X,Y)

nº total de dígitos a almacenar

dígitos a la derecha del punto decimal

➤ *Aproximados*

- ❖ De punto flotante, según el RDBMS: REAL/DOUBLE

○ Booleanos

- No todos los RDBMS lo soportan

TIPOS DE DATOS

○ Fecha y hora

- **DATE:** año, mes y día
- **TIME:** hora, minuto y segundo
- **DATETIME:** DATE + TIME
- El formato dependerá de la configuración del “NationalLanguage”

○ Texto

- **De longitud fija**
 - ❖ CHAR(N), los caracteres no usados a la derecha se rellenan con espacios en blanco
- **De longitud variable**
 - ❖ VARCHAR(N), se mantiene la longitud en un entero
- Los caracteres deben ser parte del juego de caracteres de la base (UNICODE es un juego de caracteres que soporta internacionalización)

TIPOS DE DATOS

- Objetos grandes (LOBs: Long OBjects)

- ***CLOB: CharacterLong Object***

- ❖ Permite almacenar textos muy grandes, que superen los límites de VARCHAR
- ❖ Los caracteres deben ser parte del juego de caracteres de la base

- ***BLOB: BinaryLong Object***

- ❖ Permite almacenar archivos binarios, como imágenes o documentos

TIPOS DE DATOS

- Todos los tipos de datos **aceptan** el valor **NULL**.
- Los límites dependen de la implementación, por lo que varían de un RDBMS a otro
- El tipo de datos de una columna debe permitir almacenar todos los valores previstos, pero usar un tipo demasiado grande puede desperdiciar espacio y limitar el desempeño (no usar DATETIME si alcanza con DATE, no usar CLOB si alcanza con VARCHAR, etc.)

ALTER TABLE

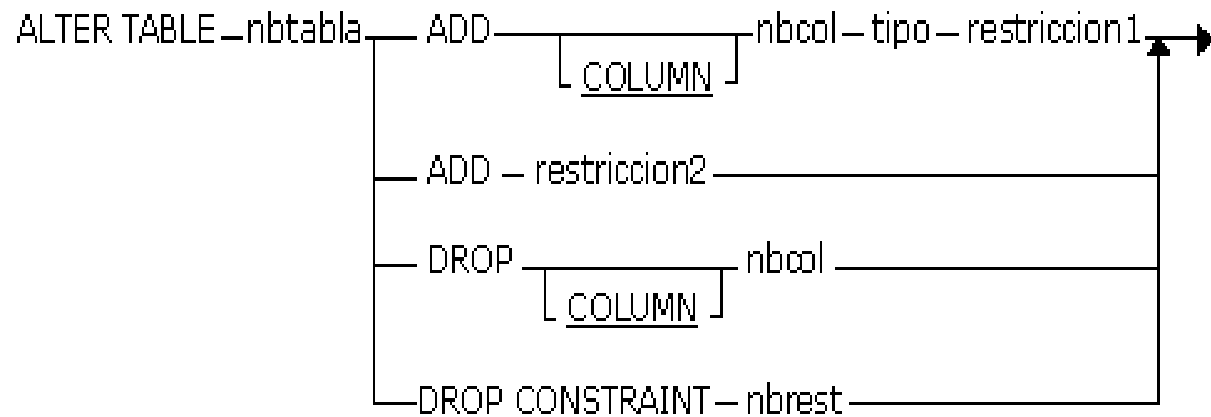
- La sentencia **ALTER TABLE** sirve para **modificar la estructura de una tabla** que ya existe:
 - Añadir columnas.
 - Eliminar columnas.
 - Modificar la definición de la columna.
 - Adicionar restricciones a las columnas.
 - Eliminar las restricciones de las columnas.

ALTER TABLE - OPERACIONES

- ADD (añade),
- ALTER (modifica),
- DROP (elimina),
- COLUMN (columna),
- CONSTRAINT (restricción).

ALTER TABLE - ESTRUCTURA

➤ Estructura



DROP TABLE

- La sentencia **DROP TABLE** sirve para **eliminar una tabla** que ya existe.
- **NOTA:** Una tabla **no se puede eliminar** si esta **abierta**, o **infringe** las reglas de **integridad referencial**.

DROP TABLE NOMBRE_TABLA;

INTEGRIDAD REFERENCIAL

- Sistema de **reglas** que utilizan las bases de datos relacionales para **asegurarse que los registros de tablas relacionadas son válidos**, que no se borren o cambien datos relacionados de forma accidental produciendo errores de integridad.

INTEGRIDAD REFERENCIAL

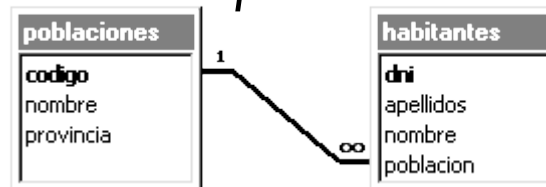
– TIPOS DE RELACIONES

○ Relación Uno a Uno

- **Profesores y Departamentos** → qué profesor es jefe de qué departamento, relación uno a uno entre las dos tablas ya que un departamento tiene un solo jefe y un profesor puede ser jefe de un solo departamento

Relación Uno a Varios

- **Poblaciones y Habitantes** → una población puede tener más de un habitante, pero un habitante pertenecerá (estará empadronado) en una única población.



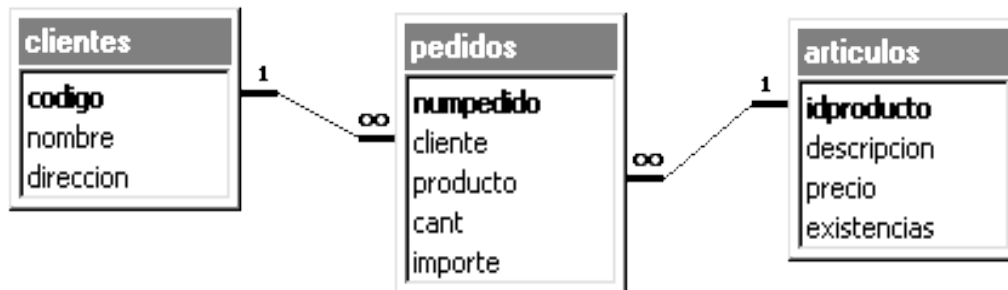
- Tabla principal 'padre' y tabla secundaria 'hijo' → 'un padre puede tener varios hijos pero un hijo solo tiene un padre.

INTEGRIDAD REFERENCIAL – TIPOS DE RELACIONES

○ Relación Varios a Varios

- **Cientes y Artículos** → *un cliente podrá realizar un pedido con varios artículos, y un artículo podrá ser vendido a más de un cliente.*

No se puede definir entre clientes y artículos, hace falta otra tabla (tabla de pedidos) relacionada con clientes y con artículos. La tabla pedidos estará relacionada con cliente por una relación uno a muchos y también estará relacionada con artículos por un relación uno a muchos.



INTEGRIDAD REFERENCIAL – ¿CUÁNDO SE PUEDEN PRODUCIR ERRORES EN LOS DATOS?

- Cuando insertamos una nueva fila en la tabla secundaria y el valor de la clave foránea no existe en la tabla principal.
- Cuando modificamos el valor de la clave principal de un registro que tiene 'hijos'
- Cuando modificamos el valor de la clave foránea, el nuevo valor debe existir en la tabla principal.
- Cuando queremos borrar una fila de la tabla principal y ese registro tiene 'hijos'

INTEGRIDAD REFERENCIAL – ACTUALIZACIÓN Y BORRADO EN CASCADA

○ Actualizar registros en cascada

- Si cambiamos en la tabla de poblaciones (tabla principal) el valor 1 por el valor 10 en el campo codigo (la clave principal), automáticamente se actualizan todos los habitantes (en la tabla secundaria) que tienen el valor 1 en el campo poblacion (en la clave ajena) dejando 10 en vez de 1.

○ Eliminar registros en cascada

- Si borramos la población Cuenca en la tabla de poblaciones, automáticamente todos los habitantes de Cuenca se borrarán de la tabla de habitantes.

DDL (LENGUAJE DE DEFINICIÓN DE DATOS)

DROP
TABLE

ALTER
TABLE

CREATE
TABLE