

---

# SQL

## (Structured Query Language)

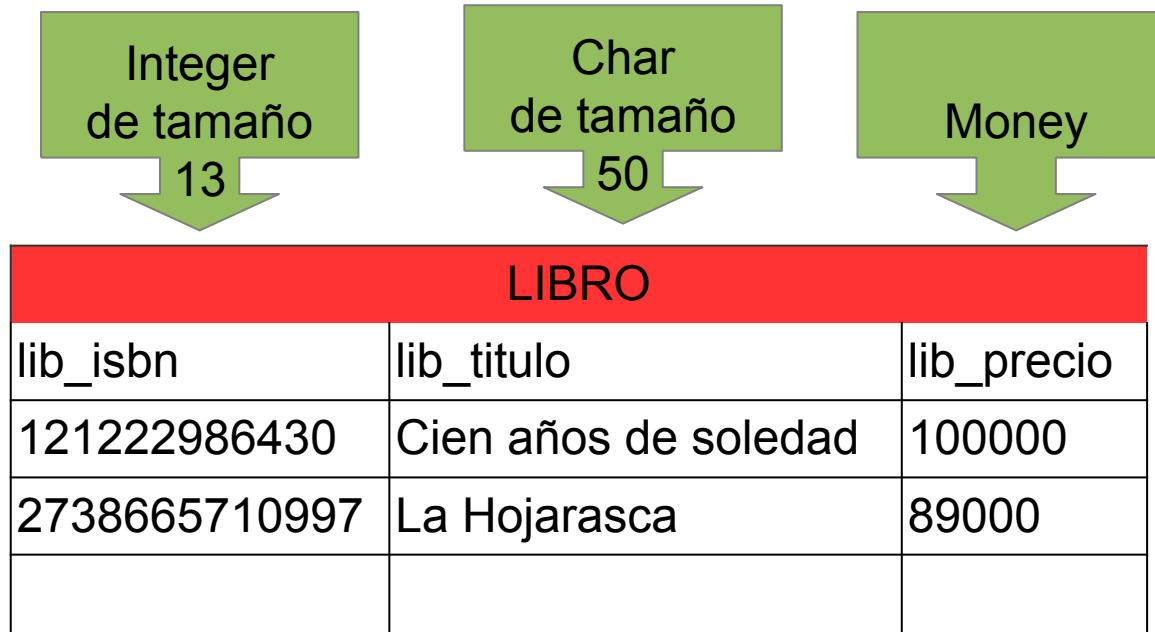
### Bases de Datos Relacionales

Por  
**Elizabeth León Guzmán, Ph.D.**  
Profesora  
Ingeniería de Sistemas

LIBRO			

- Una Tabla es un objeto de base de datos que almacena datos en filas y columnas -> **Representa una relación del modelo relacional**
- Antes de crear una tabla, se debe diseñar su estructura:
  1. nombre de la tabla y de cada columna (modelo físico de la BD)
  2. Selección de un tipo de dato para cada columna (modelo físico de la BD)
  3. Determinar las propiedades de cada columna

- Un tipo de datos (datatype) es asignado a un **atributo** (columna) que especifica el tipo de información que puede almacenarse en esa columna



The diagram illustrates the mapping of data types to attributes in a table. Three green boxes at the top define the data types: 'Integer de tamaño 13' pointing to the 'lib\_isbn' attribute, 'Char de tamaño 50' pointing to the 'lib\_titulo' attribute, and 'Money' pointing to the 'lib\_precio' attribute. Below this, a red header row labeled 'LIBRO' contains three columns: 'lib\_isbn', 'lib\_titulo', and 'lib\_precio'. The data rows show two entries: one for 'Cien años de soledad' with ISBN 121222986430 and price 100000, and another for 'La Hojarasca' with ISBN 2738665710997 and price 89000.

LIBRO		
lib_isbn	lib_titulo	lib_precio
121222986430	Cien años de soledad	100000
2738665710997	La Hojarasca	89000

- Su definición depende del Gestor de Bases de Datos

- Para seleccionar el tipo de datos de una columna se debe:
  1. Analizar y determinar el conjunto posible de valores de la columna: **dominio**
  2. Determinar la precisión requerida
  3. Encontrar el tipo de datos que:
    - Pueda guardar todos los posibles valores
    - Proporcione la exactitud y precisión requeridas
    - Use eficientemente el almacenamiento
    - Facilite el futuro crecimiento

# Ejemplos de Datatypes por Categorías

- **Números exactos**  
integer, numeric (p,s), decimal (p,s)
- **Números aproximados**  
float (p), real, double precision (dependientes de máquina)
- **Money**  
money
- **Date y time**  
datetime
- **Character**  
char (n), varchar (n), text
- **Binary**  
bit, binary, varbinary, image

# Ejemplos de Datatypes por Categorías

## • Numéricos

- INT (Integer)
- SMALLINT
- TINYINT
- FLOAT( $n$ ) donde  $n$  es la precisión
- DOUBLE(REAL)
- DOUBLE PRECISION
- DECIMAL ( $i,j$ ) o NUMERIC ( $i,j$ ) donde  $i$  es la precisión y  $j$  la cantidad de dígitos decimales

# Ejemplos de Datatypes por Categorías

---

- **Caracteres o cadenas de caracteres**

- CHAR( $n$ ) donde  $n$  es la longitud de la cadena de caracteres
- VARCHAR( $n$ ) es una cadena de tamaño variable cuya longitud máxima es  $n$ .
- VARBINARY( $n$ ) es una cadena que almacena cadenas de binarias de bits

# Ejemplos de Datatypes por Categorías

- **Fecha**

- DATE  $\Rightarrow$  YYYY-MM-DD
- TIME  $\Rightarrow$  HH:MM:SS
- TIMESTAMP  $\Rightarrow$  Incluye fecha y hora más un mínimo de 6 fracciones decimales para los segundos y con TIME ZONE opcional (depende del gestor de BD)

# Propiedades de las Columnas

Una columna puede tener una de las siguientes **propiedades**:

- NULL
- NOT NULL
- IDENTITY o AUTOINCREMENTAL
- DEFAULT
- PRIMARY KEY
- FOREIGN KEY

Si no se especifica NOT NULL, generalmente se asume NULL

- Un NULL representa un **valor desconocido** o que no aplica
  - Para valores numéricos, NULL **no** es igual a 0
  - Para caracteres, NULL **no** es igual a " " ( en blanco)
- NULL no se considera menor que, mayor que, o igual a cualquier otro valor
- Dos NULL no se consideran iguales

# Propiedad IDENTITY o AUTOINCREMENTAL

- La propiedad IDENTITY (AUTOINCREMENTAL) permite que el DBMS asigne automáticamente valores únicos a cada fila de manera incremental



empleados		
1		
2		
3		

- el servidor automáticamente asigna un número secuencial a la columna que tiene la propiedad de IDENTITY o AUTOINCREMENTAL en el momento de insertar datos en la tabla

# Reglas para columnas con IDENTITY

- Sólo puede existir una por tabla
- Debe de ser de tipo numérico sin posiciones decimales  
Ejemplo: *numeric(5,0)*
- No se puede actualizar
- No acepta valores tipo NULL
- Inicia por default en 1. Se puede asignar un valor de arranque

# Valores por defecto DEFAULT

Los atributos pueden tener **valores por defecto**. Es decir, en el momento de la inserción si no hay valor para ese atributo se insertará el valor por defecto.

## Listado de Valores permitidos ENUM

Los atributos pueden tener una lista de **valores permitidos** (DOMINIO). Es decir, en el momento de la inserción solo podrá tener un valor de esa lista.

# SQL (Structured Query Language)

---

SQL lenguaje usado para definir, manipular, y controlar bases de datos relacionales

Definido por ANSI (American National Standards Institute)

Comandos SQL se pueden dividir en tres categorías:

**DDL** (Data definition language)

**create, alter, drop**

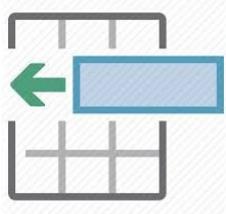
**DML** (Data manipulation language)

**select, insert, update, delete**

**DCL** (Data control language)

**grant, revoke**

# SQL (Structured Query Language)



1. Creación de tablas
2. Inserción de datos
3. Borrado de datos
4. Consulta de datos
5. Actualización de datos



# Crear Tablas

---

Sintaxis SQL simplificada para **Crear una Tabla o relación :**

```
CREATE TABLE  table_name (
    column_name  datatype [NULL | NOT NULL] ,
    ...
    column_name  datatype [NULL | NOT NULL ]
) ;
```

# Crear Tabla en MySQL

---

Ejemplo:

```
CREATE TABLE empleado (
    emp_id          int (10)      AUTO_INCREMENT ,
    emp_apellido   varchar(30)   NOT NULL ,
    emp_nombre     varchar(30)   NOT NULL ,
    emp_e_mail     char(6)       NULL ,
    emp_departamento  varchar(30) NULL
);
```

# Crear Tabla en MySQL

Llave primaria

---

```
CREATE TABLE empleado (
    emp_id      int (10)  AUTO_INCREMENT PRIMARY KEY,
    emp_apellido varchar(30)  NOT NULL,
    emp_nombre   varchar(30)  NOT NULL,
    emp_e_mail   char(6)      NULL,
    emp_departamento  varchar(30) NULL
);
```

# Crear Tabla en MySQL

## Llave primaria compuestas

```
CREATE TABLE venta (
    vta_id_cliente      integer NOT NULL,
    vta_id_producto     integer NOT NULL,
    vta_cantidad        integer NULL,
    vta_fecha           date    NOT NULL,
    vta_precio          money   NULL,
    vta_total           money   NULL,
    PRIMARY KEY (vta_id_cliente,vta_id_producto,vta_fecha)
);
```

# Crear Tabla en MySQL con llaves foráneas

- Tablas tipo INNODB
- Usar la sintaxis FOREIGN KEY(campo\_fk) REFERENCES nombre\_tabla (nombre\_campo)

```
CREATE TABLE venta (
    vta_id_factura INT NOT NULL,
    vta_id_cliente INT NOT NULL,
    vta_cantidad INT,
    PRIMARY KEY(vta_id_factura),
    FOREIGN KEY (vta_id_cliente) REFERENCES cliente(id_cliente)
);
```

Nombre de la  
tabla a la que  
referencia

Atributo de de la  
tabla a la que  
referencia

# Crear Tabla en MySQL

## Valores por defecto y listas de valores

```
CREATE TABLE pais(
    pais_codigo CHAR(3) PRIMARY KEY
    pais_nombre VARCHAR(30) NOT NULL DEFAULT "",  

    pais_contienente enum('Asia','Europe','North America',
    'Africa','Oceania','Antarctica','South America') NOT
    NULL DEFAULT 'Asia'
);
```

# Crear Tabla en MySQL

## Valores por defecto y listas de valores

```
CREATE TABLE pais(
    pais_codigo CHAR(3) PRIMARY KEY
    pais_nombre VARCHAR(30) NOT NULL DEFAULT "",
    pais_contienente enum('Asia','Europe','North America',
    'Africa','Oceania','Antarctica','South America') NOT
    NULL DEFAULT 'Asia'
);
```

- Comando DROP. Borra el objeto tabla con sus datos.
- Sintaxis Simplificada para **drop**:

**DROP TABLE *table\_name***

- Ejemplo:
  - borra la tabla empleado
  - DROP TABLE empleado

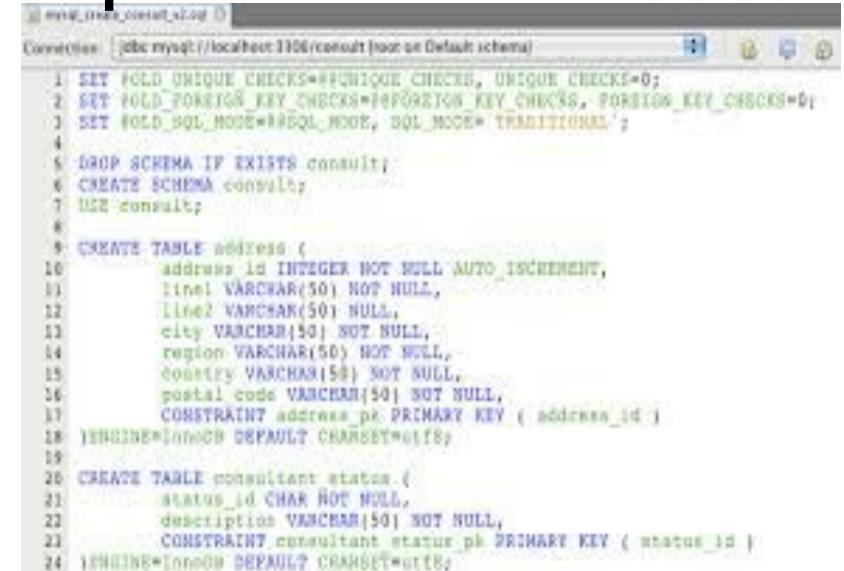
# Scripts

## Recomendaciones para desarrollo

- Crear todas las tablas a través de un **script**

- Facilita la recreación de los objetos
  - Sirve como material permanente
  - de referencia

- Especificar una propiedad para cada columna



```
1 SET FOREIGN_KEY_CHECKS=0;
2 SET UNIQUE_CHECKS=0;
3 SET CHECK_CONSTRAINTS=0;
4 SET SQL_MODE='TRADITIONAL';
5
6 DROP SCHEMA IF EXISTS consult;
7 CREATE SCHEMA consult;
8 USE consult;
9
10 CREATE TABLE address (
11     address_id INTEGER NOT NULL AUTO_INCREMENT,
12     line1 VARCHAR(50) NOT NULL,
13     line2 VARCHAR(50) NULL,
14     city VARCHAR(50) NOT NULL,
15     region VARCHAR(50) NOT NULL,
16     country VARCHAR(50) NOT NULL,
17     postal_code VARCHAR(50) NOT NULL,
18     CONSTRAINT address_pk PRIMARY KEY (address_id)
19 )ENGINE=InnoDB DEFAULT CHARSET=utf8;
20
21 CREATE TABLE consultant_status (
22     status_id CHAR NOT NULL,
23     description VARCHAR(50) NOT NULL,
24     CONSTRAINT consultant_status_pk PRIMARY KEY (status_id)
25 )ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

- Usar **tipos de datos de usuario** para columnas que almacenen el mismo conjunto de valores (no disponible en todos los DBMS)

# Ejercicio

- Abrir MySQL Workbench o una terminal de MySQL y crear un schema llamado museo y luego crear un script llamado creacionMuseo.sql
- Crear la tabla exposición y la tabla obra

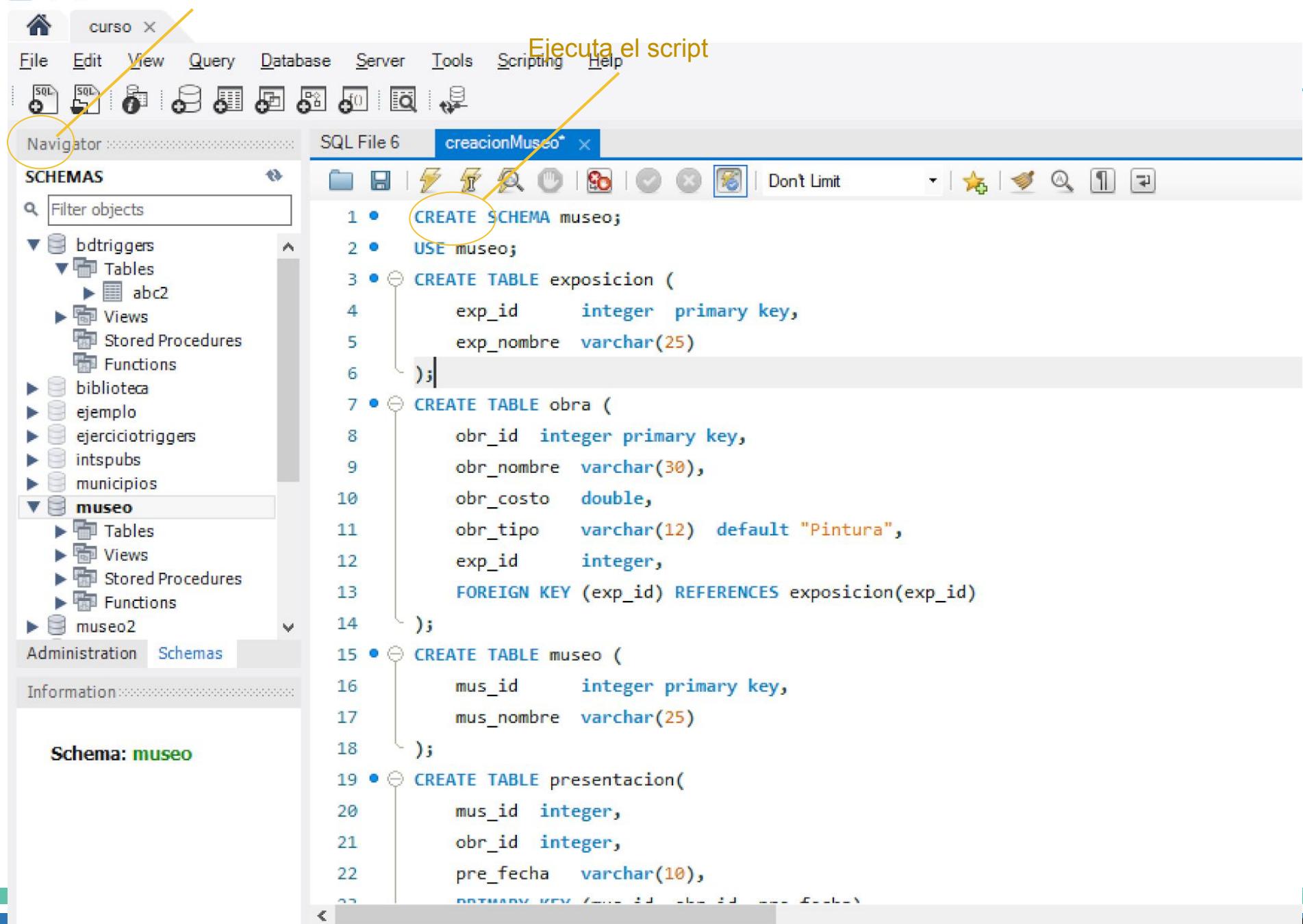
```
CREATE TABLE exposicion (
    exp_id integer PRIMARY KEY,
    exp_nombre varchar(30) not null
);
```

```
CREATE TABLE obra (
    obr_id integer PRIMARY KEY,
    obr_nombre varchar(30) not null,
    obr_tipo varchar(15) not null,
    obr_costo double not null,
    exp_id integer not null,
    FOREIGN KEY (exp_id) REFERENCES exposicion(exp_id)
);
```

## Fiercicio

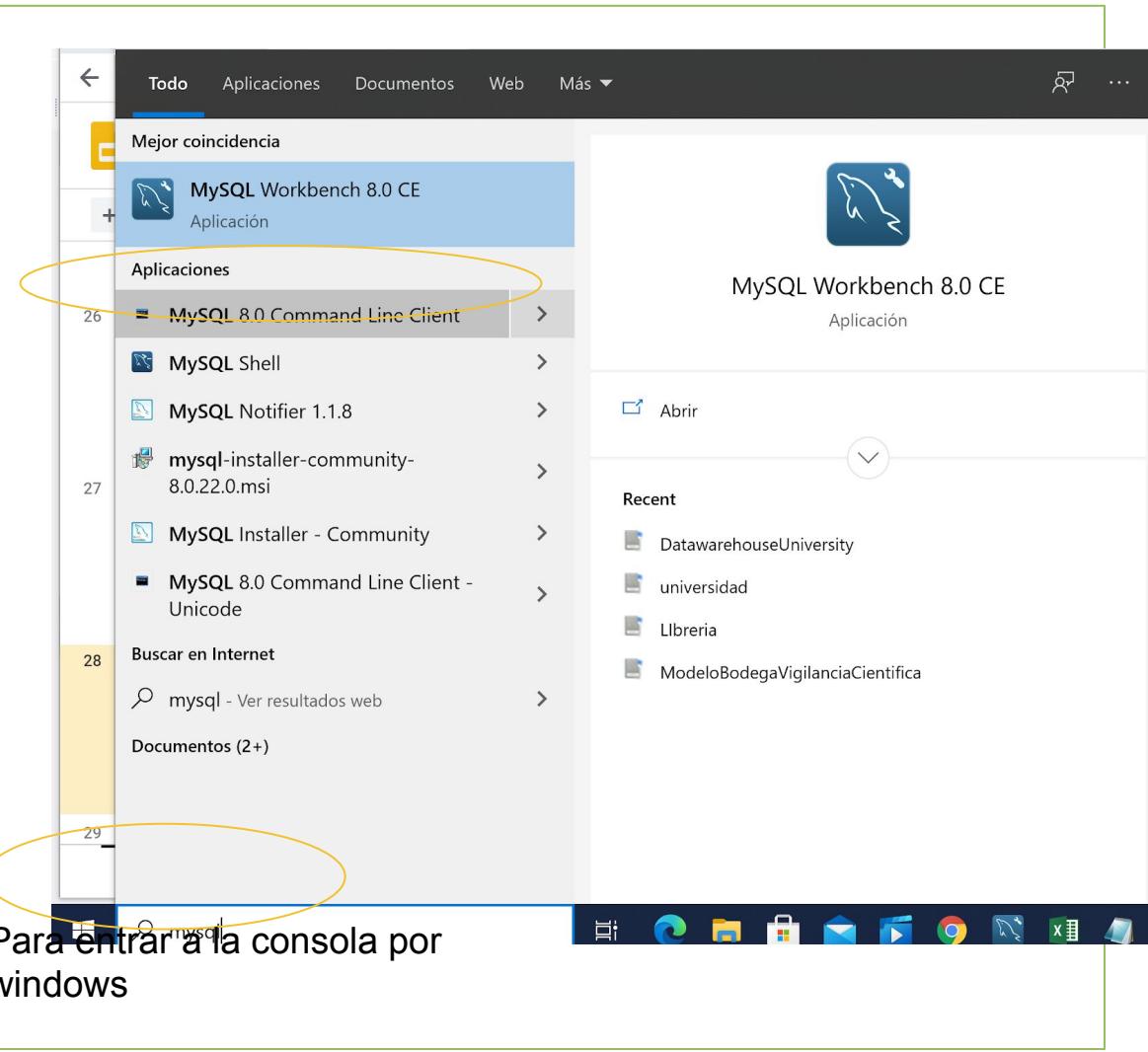
Crea un script

Ejecuta el script

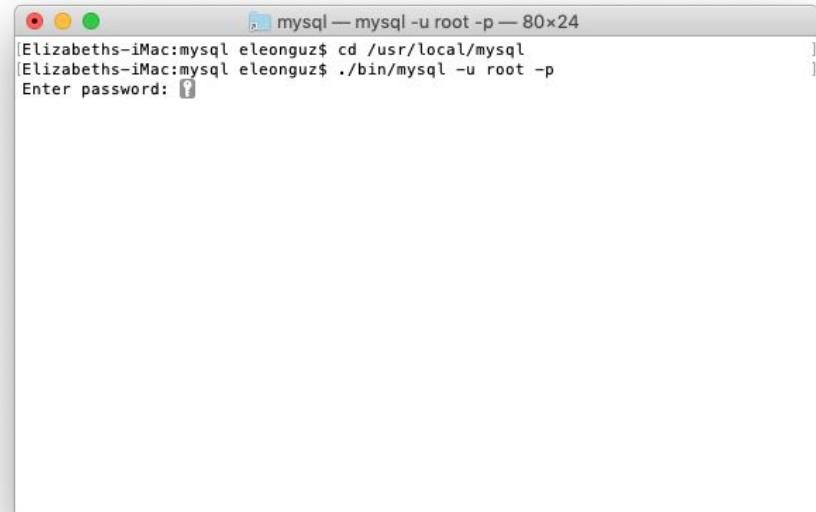


```
1 • CREATE SCHEMA museo;
2 • USE museo;
3 • ◇ CREATE TABLE exposicion (
4     exp_id      integer primary key,
5     exp_nombre  varchar(25)
6 );
7 • ◇ CREATE TABLE obra (
8     obr_id      integer primary key,
9     obr_nombre  varchar(30),
10    obr_costo   double,
11    obr_tipo    varchar(12) default "Pintura",
12    exp_id      integer,
13    FOREIGN KEY (exp_id) REFERENCES exposicion(exp_id)
14 );
15 • ◇ CREATE TABLE museo (
16     mus_id      integer primary key,
17     mus_nombre  varchar(25)
18 );
19 • ◇ CREATE TABLE presentacion(
20     mus_id      integer,
21     obr_id      integer,
22     pre_fecha   varchar(10),
23     PRIMARY KEY (mus_id, obr_id)
```

# Si se usa consola...



Abrir una consola del SO e ir a la carpeta donde esta MySQL  
`/usr/local/mysql`  
ejecutar:  
`./bin/mysql -u root -p`



```
[Elizabeths-iMac:mysql eleonguz$ cd /usr/local/mysql
[Elizabeths-iMac:mysql eleonguz$ ./bin/mysql -u root -p
Enter password:
```

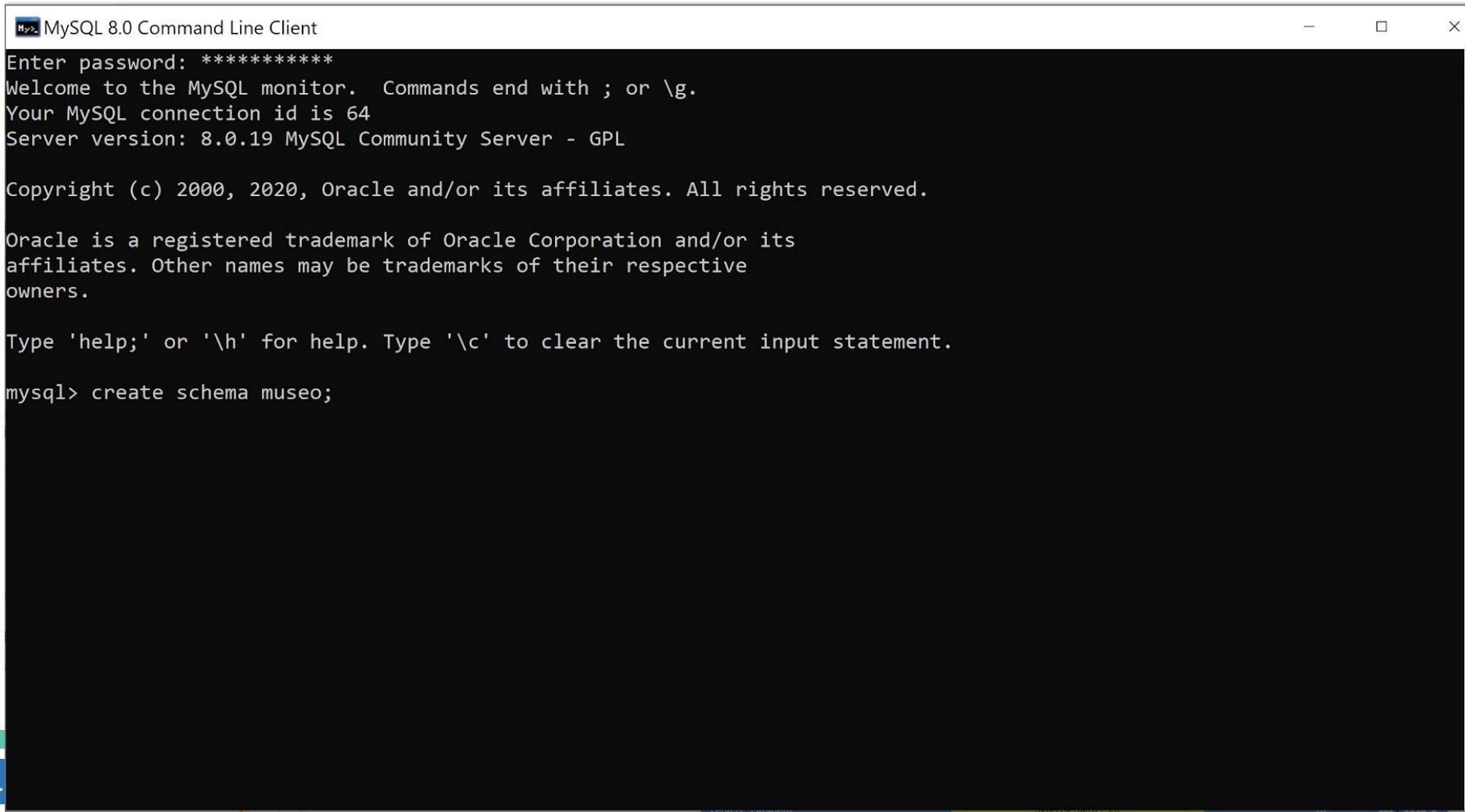
Para entrar por consola en Linux o Mac

# Ejercicio

---

Digitar el password y luego en el prompt:

```
create schema museo;  
use museo;
```



The screenshot shows a terminal window titled "MySQL 8.0 Command Line Client". It displays the MySQL monitor welcome message, server version information, copyright notice, and a trademark disclaimer. At the bottom, the command "mysql> create schema museo;" is entered.

```
MySQL 8.0 Command Line Client  
Enter password: *****  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 64  
Server version: 8.0.19 MySQL Community Server - GPL  
  
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> create schema museo;
```

### MySQL 8.0 Command Line Client

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
mysql> create schema museos;
Query OK, 1 row affected (0.01 sec)
```

```
mysql> use museos;
Database changed
mysql> CREATE TABLE exposicion (exp_id integer primary key, exp_nombre varchar(25));
Query OK, 0 rows affected (0.09 sec)
```

```
mysql> CREATE TABLE obra (obr_id integer primary key, obr_nombre varchar(30), obr_tipo varchar(12) default "Pintura", obr_costo int, exp_id integer, FOREIGN KEY (exp_id) REFERENCES exposicion(exp_id));
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> CREATE TABLE museo (mus_id integer primary key, mus_nombre varchar(25));
Query OK, 0 rows affected (0.05 sec)
```

```
mysql> CREATE TABLE presentacion( pre_fecha varchar(15), obr_id integer, mus_id integer, PRIMARY KEY (mus_id, obr_id, pre_fecha), FOREIGN KEY (mus_id) REFERENCES museo(mus_id), FOREIGN KEY (obr_id) REFERENCES obra(obr_id) );
Query OK, 0 rows affected (0.05 sec)
```

```
mysql>
```

# Ejercicio

---

- Modificar el script para crear las demás tablas: museo y presentación. Tener en cuenta que presentación debe tener llaves foráneas a museo y obra (por lo que la tabla museo debe crearse primero que presentación).

- Después de creada una tabla, se pueden añadir columnas
- Algunos DBMS no permiten borrar directamente una columna de una tabla
  - Para borrar una columna, se debe borrar y volver a crear la tabla

## ALTER TABLE

- Sintaxis para añadir una columna a una tabla:

```
ALTER TABLE table_name
  add column_name datatype NULL
    [, column_name datatype NULL ...]
```

- Todas las columnas añadidas deben tener la propiedad NULL.

# Adicionar Columnas

---

```
ALTER TABLE editor
add dirección varchar(40) NULL,
    país         varchar(20) NULL;
```

# Colocar valores por defecto

---

```
ALTER TABLE editor
ALTER país SET DEFAULT
'Colombia';
```

# Ejercicio

---

- Adicionar una nueva columna a presentación con la Hora de la presentación. Permitir valores NULLs

```
ALTER TABLE presentacion
    add pre_hora varchar(6) NULL;
```

- Modificar la columna de costo de obra dejando por defecto el valor 500

```
ALTER TABLE obra
ALTER obr_costo SET DEFAULT 500;
```

# Inserción de datos

---

## INSERT

Sintaxis INSERT Simplificada:

```
INSERT [into] table_name [(column_list) ]  
{ VALUES (value_list) | select_statement }
```

# Sintaxis para INSERT

editor			
edi_codigo	edi_nombre	edi_ciudad	edi_departamento

Ejemplos:

```
INSERT INTO editor VALUES ("736", "Nuevas BD",
"Pasto", "Nariño");
```

```
INSERT INTO editor (edi_nombre, edi_codigo)
VALUES ("Prensa Actual", "2003");
```

# Sintaxis para INSERT

editor			
edi_codigo	edi_nombre	edi_ciudad	edi_departamento
736	Nuevas BD	Pasto	Nariño
2003	Prensa Actual	NULL	NULL

Ejemplos:

```
INSERT INTO editor VALUES ("736", "Nuevas BD",
"Pasto", "Nariño");
```

```
INSERT INTO editor (edi_nombre, edi_codigo)
VALUES ("Prensa Actual", "2003");
```

# Scripts

## Ejercicio

- Insertar las exposiciones

```
insert into exposicion values (1003, 'Da Vinci');  
insert into exposicion values (1004, 'Renacimiento');  
insert into exposicion values (1005, 'Cubismo');  
insert into exposicion values (1006, 'Impresionismo');
```

- Insertar todas las obras

```
insert into obra values (111, 'Mona lisa', 'pintura', 1000, 1003);  
insert into obra values (112, 'Ultima cena', 'pintura', 800,  
1003);  
insert into obra values (113, 'Hombre vitruvio', 'boceto', 400,  
1003);  
insert into obra values (114, 'Planos', 'planos', 200, 1003);  
insert into obra values (200, 'Fornarina', 'pintura', 400, 1004);  
.  
.  
.
```

# Scripts

## Ejercicio

- Verificar las inserciones

```
SELECT * FROM obra;
```

```
SELECT * FROM exposicion;
```

- Insertar una nueva obra de Da Vinci sin costo (verificar que tenga el valor por defecto después de la inserción)

```
insert into obra (obr_id, obr_nombre, obr_tipo, exp_id) values  
(115, 'Salvator Mundi', 'pintura', 1003);
```

```
SELECT * FROM obra;
```

- Modificar el script de la creación de las tablas de museo, para insertar los datos de todas las tablas (museo y presentación).

# Borrado de datos

## delete

### Comando delete

- Borra toda la tabla

```
DELETE FROM empleado
```

- Borra los registros que cumplen la condición

```
DELETE FROM empleado WHERE nacionalidad =  
'colombiana'
```

# Ejercicio

- Borrar la obra que tiene obr\_id = 115

```
DELETE FROM obra WHERE obr_id=115;
```

Consultar las obras para verificar que el borrado funcionó.

```
SELECT * FROM obra;
```

# Actualización de datos

---

## UPDATE

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Ejemplo:

```
UPDATE editor
SET edi_ciudad = 'Tunja', edi_departamento='Cundinamarca'
WHERE ediCodigo = 736 OR 738;
```

```
UPDATE editor
SET edi_ciudad = 'Tunja', edi_departamento='Cundinamarca'
WHERE ediCodigo IN (736,738);
```

# Ejercicio

- Cambiar el precio de la obra “Mona Lisa” por \$1300

```
SELECT * FROM obra;
```

```
UPDATE obra SET obr_costo=1300 WHERE obr_nombre = "Mona Lisa";
```

```
SELECT * FROM obra;
```

- Incrementar el precio de las esculturas en 10%

```
SELECT * FROM obra;
```

```
UPDATE obra SET obr_costo=obr_costo*1.1 WHERE obr_tipo =  
"escultura";
```

```
SELECT * FROM obra;
```

# Consultas

**SELECT A1, A2, A3 FROM R WHERE condición;**

Proyección

Selección

 $\pi_{A_1, A_2, \dots, A_n}(R)$  $\sigma_{condición}(R)$ 

SELECT  $A_1, A_2, \dots, A_n$  FROM R

SELECT \* FROM R WHERE condición

# Para recordar....

 $R_1 \times R_2$ 


```
SELECT * FROM R1, R2
```

 $\sigma_{R_1.k=R_2.k}(R_1 \times R_2)$ 


```
SELECT * FROM R1, R2 WHERE R1.k = R2.k
```

 $R_1 \bowtie_k R_2$ 


```
SELECT * FROM R1 JOIN (R2) USING (k)
```

 $\delta(R)$ 


```
SELECT DISTINCT * FROM R
```

 $R_1 \bowtie R_2$ 


```
SELECT * FROM R1 NATURAL JOIN (R2)
```

# Ejercicio

- Consultar todos los atributos de todas las obras

```
SELECT * FROM obra;
```

- Consultar los nombres de las obras con su correspondiente tipo

```
SELECT obr_nombre, obr_tipo FROM obra;
```

- Consultar los diferentes tipos de obras sin repetidos y renombrando el atributo por TIPO

```
SELECT DISTINCT obr_tipo AS TIPO FROM obra;
```

# Consultas

El operador más usado en MySQL para buscar strings es **LIKE**. Con este comando se pueden realizar consultas como:

- ◆ Consulta Exacta  
    **LIKE 'Mona Lisa'**
- ◆ Consulta omitiendo una parte  
    **LIKE 'Mona%**'
- ◆ Consulta omitiendo un carácter  
    **LIKE 'Mo\_a\_Lis\_'**

Se pueden combinar usando AND, OR y NOT

Cuál es el resultado de:

```
SELECT obr_nombre FROM obra WHERE obr_nombre LIKE  
'%ma%' AND obr_nombre NOT LIKE '%cena%';
```

# Consultas

---

Otras funciones que se pueden aplicar sobre los strings son:

- ◆ LENGTH('Pintura') -> 7
- ◆ CONCAT('Mona', 'Lisa') -> MonaLisa
- ◆ LOWER('DaVinci') -> davinci
- ◆ UPPER('Ultima Cena') -> ULTIMA CENA
- ◆ RTRIM('Planos ') -> Planos

# Ejercicio

- Listar todas los nombres de las obras en mayúscula

```
SELECT UPPER(obr_nombre) FROM obra;
```

- Consultar el código de la exposición “da vinci” (en minúscula)

```
SELECT exp_id FROM exposicion WHERE LOWER(exp_nombre)=  
“da vinci”;
```

# Consultas

Operadores numéricos : <,>,!=,<= y >=

```
SELECT obr_nombre from obra WHERE obr_costo > 400
```

Operadores de agregación. Usan los valores de una columna completa. Los más usados son:

- ◆ AVG: promedio
- ◆ MIN y MAX : mínimo y máximo
- ◆ SUM: suma
- ◆ Count: contar

```
SELECT MAX(obr_costo) FROM obra;
```

```
SELECT AVG(obr_costo) FROM obra;
```

```
SELECT obr_costo FROM obra WHERE obr_costo >(SELECT  
AVG(obr_costo) FROM obra);
```

# Consultas

Se pueden ordenar los resultados por una de las columnas usando **ORDER BY**. Se puede usar **LIMIT** para restringir el máximo de elementos a retornar.

**Ejemplo:** Nombre de las 5 obras más costosas

```
SELECT obr_nombre FROM obra ORDER BY obr_costo  
DESC LIMIT 5
```

## Ejercicios

- Nombre de las obras que tienen un costo menor a 1/3 del promedio.
- Listar los nombres de las obras ordenadas alfabéticamente

# Consultas

Se pueden agrupar los resultados por valores de una columna y calcular funciones de agregación por cada una de las agrupaciones, usando **GROUP BY**.

**Ejemplo:** Contar el número de obras por cada uno de los tipos de obras

```
SELECT obr_tipo, count(obr_id) FROM obra GROUP  
BY obr_tipo
```

## Ejercicios:

- Sumar el costo de las obras por tipo de obra
- Contar las obras que tiene cada una de las exposiciones

# Consultas de más de una tabla (JOINS)

- Listar los nombres de las obras y exposiciones de las obras tipo pintura

```
SELECT obr_nombre, exp_nombre FROM obra JOIN exposicion  
USING (exp_id) WHERE obr_tipo = "pintura";
```

- Listar los nombres, tipo y costo de las obras de la exposición “Da Vinci”
- Ejecutar las consultas en SQL realizadas en la clase de ayer con AR

## Referencias

---

- [1] Gillenson, M. *Administración de Bases de Datos*. LIMUSA WILEY
- [2] Coronel, Morris, Rob. *Bases de Datos: Diseño, Implementación y Administración*. CENGAGE Learning
- [3] Elmasri, R.; Navathe, S.B. *Fundamentos de Sistemas de Bases de Datos*. 3<sup>a</sup> ed. Addison-Wesley
- [4] Silberschatz, A; Korth, H; Sudarshan, S. *Fundamentos de Bases de Datos*. 3<sup>a</sup> edición. Madrid: McGraw-Hill.
- [5] León, E. Notas curso Bases de Datos. Universidad Nacional de Colombia