Universidad Nacional Mar del Plata
Facultad de Ingeniería
Departamento de Informática
Curso de Base de Datos SQLite y MySQL

Clase 2

Bases de Datos

Lic. Fernando Genin
geninfernando@hotmail.com

Curso de Base de Datos SQLite y MySQL

Clase 2

SQL

DDL - Lenguaje de definición de datos

DML - Lenguaje de manipulación de datos

Presentación MySQL

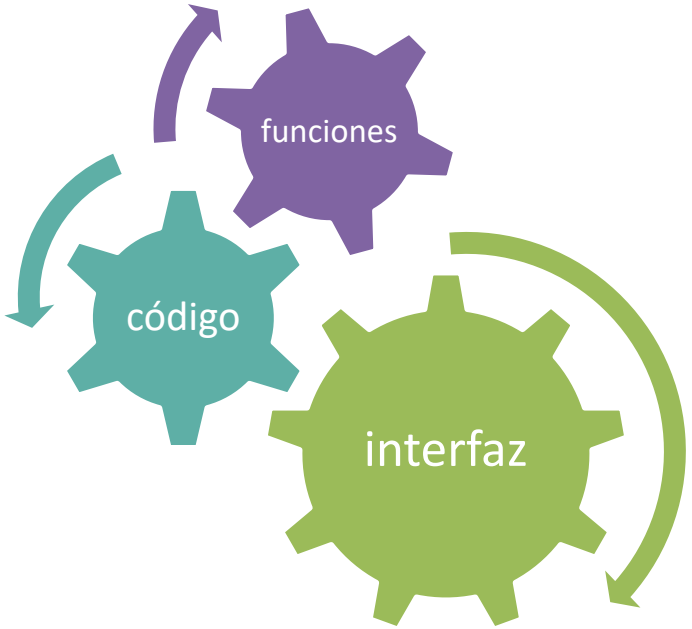
Práctica en MySQL

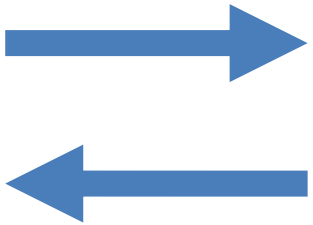
SQL

Lenguaje estructurado de consultas

Lenguaje universal

SQL





LENGUAJE DE PROGRAMACION

BASE DE DATOS

DDL - Lenguaje de definición de datos

Se encarga de la modificación de la estructura de los objetos de la BD

- **CREATE**
- **ALTER**
- **DROP**
- **TRUNCATE**

DML - Lenguaje de manipulación de datos

Permite realizar la consulta o manipulación de los datos

- **INSERT**
- **UPDATE**
- **DELETE**
- **SELECT**

Sentencia CREATE

La sentencia CREATE TABLE define una tabla. El usuario proporciona el nombre de la tabla, los atributos y sus tipos de datos.

Crear una tabla llamada ciudades con dos columnas.

- **idciudad** clave primaria y autoincremental
- **ciudad**, donde se guardará el nombre de la misma

```
CREATE TABLE `ciudades` (
  `idciudad` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Clave primaria',
  `ciudad` varchar(50) NOT NULL ,
   PRIMARY KEY (`idciudad`)  )
```

Sentencia CREATE

Otra opción:

```
CREATE TABLE `ciudades` (  `idciudad` int(11) NOT NULL,  `ciudad` text NOT NULL)
```

```
ALTER TABLE `ciudades`  ADD PRIMARY KEY (`idciudad`);
```

```
ALTER TABLE `ciudades`  MODIFY `idciudad` int(11) NOT NULL AUTO_INCREMENT;
```

Sentencia ALTER

La sentencia ALTER TABLE sirve para modificar una tabla.

Modificar el nombre de la tabla ciudad por ciudades_copia
ALTER TABLE **ciudades** RENAME TO ciudades_copia

Agregar una columna en la tabla ciudades, con el nombre cod_postal de tipo varchar
ALTER TABLE **ciudades** ADD cod_postal varchar(15)

Eliminar la columna cod_postal
ALTER TABLE **ciudades** drop column cod_postal

Sentencia DROP

La sentencia DROP TABLE borra una tabla de la base de datos.

Eliminar la tabla ciudades.
DROP TABLE ciudades

Sentencia INSERT

La sentencia INSERT se utiliza para agregar registros en una tabla.

Agregar 3 registros en la tabla ciudades
INSERT INTO ciudades (ciudad) VALUES ('Mar del Plata')
INSERT INTO ciudades (ciudad) VALUES ('Santa Clara')
INSERT INTO ciudades (ciudad) VALUES ('Necochea')

Sentencia SELECT

La sentencia SELECT se utiliza para recuperar los datos de una o varias tablas.

Mostrar los registros cargados en la tabla ciudades
SELECT * FROM ciudades

Sentencia UPDATE

La sentencia UPDATE se utiliza para modificar uno o varios campos de uno o varios registros.

Modificar el nombre de la ciudad “Santa Clara” por el nombre “Santa Clara 2”. Posteriormente verificar que el cambio se realizó correctamente.

```
UPDATE ciudades
SET ciudad = 'Santa Clara 2'
WHERE idciudad = 2;
```

```
SELECT * FROM ciudades
```

Sentencia UPDATE

IMPORTANTE: al momento de hacer una modificación, debemos tener cuidado en la clausula WHERE, ya que en la misma se establece el criterio para elegir los registros que serán afectados por la modificación.

Sentencia UPDATE

Analizar el impacto de la ejecución de las siguientes consultas:

```
UPDATE ciudades
SET ciudad = 'Santa Clara 2'
```

```
UPDATE ciudades
SET ciudad = 'Santa Clara 2'
WHERE idciudad >1;
```

Sentencia DELETE

La sentencia DELETE se utiliza para eliminar uno o varios registros.

Eliminar la ciudad de Necochea. Posteriormente verificar que el cambio se realizó correctamente.
DELETE FROM ciudades WHERE ciudad='Necochea'
SELECT * FROM ciudades

Sentencia DELETE

IMPORTANTE: al momento de hacer una eliminación, debemos tener cuidado en la cláusula WHERE, ya que en la misma se establece el criterio para elegir los registros que serán afectados por la eliminación.

Ejecutar las siguientes sentencias y analizar que sucede con la correlación del campo idciudad
INSERT INTO ciudades (ciudad) VALUES ('Necochea')
SELECT * FROM ciudades

Práctica en MySQL

Crear la tabla personas

```
CREATE TABLE personas (
        idpersona INTEGER PRIMARY KEY AUTO_INCREMENT,
        nombre TEXT,
        apellido TEXT,
        fecha_nac DATE,
        dni INTEGER,
        idciudad INTEGER,
        sexo TEXT,
        foreign key(idciudad) references ciudades(idciudad)
)
```

Diferencia en la creación de tablas SQLite vs MySQL


```
CREATE TABLE "personas" (
"idpersona"          INTEGER PRIMARY KEY
AUTOINCREMENT,
"nombre"             TEXT,
"apellido"           TEXT,
"fecha_nac"          DATE,
"dni"      INTEGER,
"idciudad"           INTEGER,
"sexo"     TEXT,
foreign key(idciudad) references ciudades(idciudad)
)
```

```
CREATE TABLE personas (
idpersona INTEGER PRIMARY KEY AUTO_INCREMENT,
nombre TEXT,
apellido TEXT,
fecha_nac DATE,
dni INTEGER,
idciudad INTEGER,
sexo TEXT,
foreign key(idciudad) references ciudades(idciudad)
)
```

Práctica en MySQL

Insertar 10 registros en la tabla personas

```
INSERT INTO `personas` (`idpersona`, `nombre`, `apellido`, `fecha_nac`, `dni`, `idciudad`,
`sexo`) VALUES(
1, 'Analia', 'Garcia', '1982-04-15', 27272727, 6, 'F'),(
2, 'Tomas', 'Rodriguez', '1979-12-15', 27272727, 4, 'M'),(
3, 'Fernando', 'Acosta', '1980-11-05', 23232323, 6, 'M'),(
4, 'Ezequiel', 'Genin', '1979-12-15', 27272727, 4, 'M'),(
5, 'Fernando', 'Lara', '1980-11-05', 23232323, 4, 'M'),(
6, 'Debora', 'Garcia', '1969-11-05', 34343434, 5, 'F'),(
7, 'Jonas', 'Llorenzo', '1995-12-15', 45454545, 6, 'M'),(
8, 'Juan', 'Messi', '1980-11-15', 56565656, 4, 'M');
```

Sentencia SELECT

Estructura genérica de la sentencia SELECT

SELECT columna1, columna2
FROM tabla1, tabla2
WHERE condiciones
ORDER BY columna1

Existen otros parámetros que veremos mas adelante

Sentencia SELECT

Mostrar todas las columnas de la tabla personas

```
SELECT * FROM personas
```

Mostrar apellido y nombre de las personas

```
SELECT apellido,nombre FROM personas
```

Mostrar nombre y apellido de las personas

```
SELECT nombre,apellido FROM personas
```

Sentencia SELECT AS

AS permite renombrar columnas si lo utilizamos en la cláusula SELECT, o renombrar tablas si lo utilizamos en la cláusula FROM. Es opcional. Con ello podremos crear diversos alias de columnas y tablas.

Mostrar nombre y apellido de las personas y renombrar las columnas de nombre y apellido por n y a respectivamente
SELECT nombre as n, apellido as a FROM personas

El renombramiento no se realiza en el campo de la tabla, se realiza en el resultado de la consulta.

Sentencia SELECT WHERE

Especifica la condición de filtro de las filas devueltas. Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas condiciones.

Mostrar nombre y apellido de las personas de sexo femenino
SELECT nombre, apellido FROM personas WHERE sexo='f'

Sentencia SELECT CONDICIONES

Son expresiones lógicas a comprobar para la condición de filtro, que tras su resolución devuelven para cada fila TRUE o FALSE, en función de que se cumplan o no. Se puede utilizar cualquier expresión lógica y en ella utilizar diversos operadores como:

`>` `>=` `<` `<=` `=` `<>` `IS NULL` `IS NOT NULL` `LIKE` `BETWEEN` `IN`

Por supuesto es posible combinar varias condiciones simples de los operadores anteriores utilizando los operadores lógicos:

`OR` `AND` `NOT`

Sentencia SELECT

CONDICIONES

IS NULL

Una columna de una fila es NULL si está completamente vacía. Hay que tener en cuenta que si se ha introducido cualquier dato, incluso en un campo alfanumérico si se introduce una cadena en blanco o un cero en un campo numérico, deja de ser NULL

```
SELECT * FROM personas WHERE dni IS NULL
```

Sentencia SELECT

CONDICIONES

LIKE

Se utiliza para la comparación de un modelo. Para ello utiliza los caracteres comodín especiales: “%” y “_”. Con el primero indicamos que en su lugar puede ir cualquier cadena de caracteres, y con el segundo que puede ir cualquier carácter individual (un solo carácter). Con la combinación de estos caracteres podremos obtener múltiples patrones de búsqueda.

Sentencia SELECT

CONDICIONES

LIKE

El nombre empieza con A: nombre LIKE 'A%'

El nombre termina con A: nombre LIKE '%A'

El nombre contiene la letra A: nombre LIKE '%A%'

El nombre empieza por A y después contiene un solo carácter cualquiera: nombre LIKE 'A_'

El nombre empieza una A, después cualquier carácter, luego una E y al final cualquier cadena de caracteres: nombre LIKE 'A_E%'

Sentencia SELECT

CONDICIONES

BETWEEN

Se utiliza para un intervalo de valores.

Personas con dni entre 30 millones y 100 millones:

```
SELECT * FROM personas WHERE dni BETWEEN 10000000 AND 30000000
```

Sentencia SELECT

CONDICIONES

IN

Se utiliza para especificar una relación de valores concretos

Mostrar las personas que viven en la ciudad 1 y 2

```
SELECT * FROM personas WHERE idciudad=1 OR idciudad =2
```

```
SELECT * FROM personas WHERE idciudad IN(1,2)
```

Sentencia SELECT

ORDER BY

Define el orden de las filas del conjunto de resultados. Se especifica el campo o campos (separados por comas) por los cuales queremos ordenar los resultados.

DESC

ASC

Mostrar las personas que viven en ciudad 1 y 2 ordenas por apellido y nombre

```
SELECT * FROM personas WHERE idciudad=1 OR idciudad=2
ORDER BY apellido,nombre
```

ACLARACION: ASC es el valor predeterminado, especifica que la columna indicada en la cláusula ORDER BY se ordenará de forma ascendente, o sea de menor a mayor

Repaso

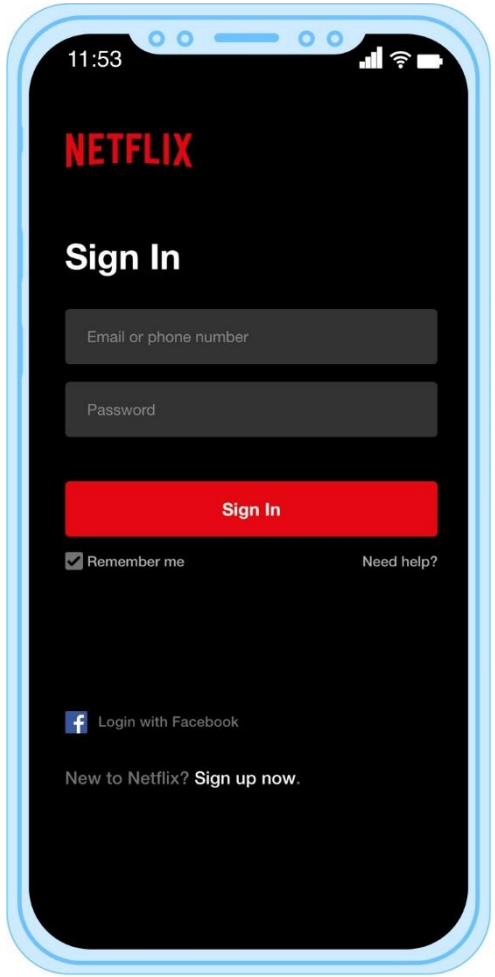
DDL

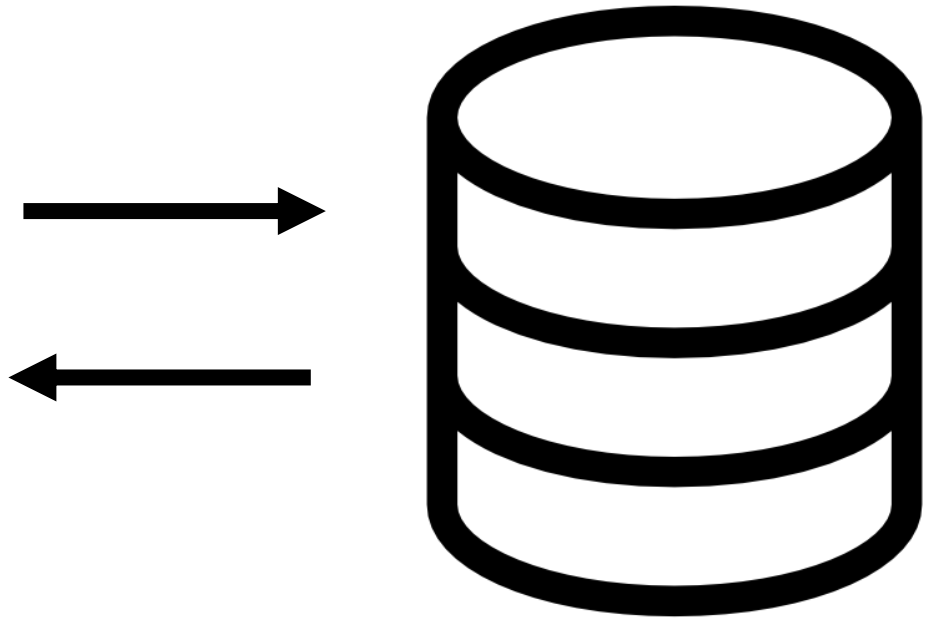
- Crear tabla, BD, etc. Sentencia CREATE
- Modificar tabla, BD, etc. Sentencia ALTER
- Eliminar tabla, BD, etc. Sentencia DROP
- Borrar tabla Sentencia TRUNCATE

DML

- Mostrar los datos Sentencia SELECT
- Insertar un registro Sentencia INSERT
- Modificar un registro Sentencia UPDATE
- Eliminar un registro Sentencia DELETE

Ejemplo





1. Conexión BD
2. Se realiza SELECT con usuario y contraseña

SELECT * **FROM** usuarios **WHERE** usuario='valor ingresado' and clave='valor ingresado'

3. Se envía respuesta del SELECT
4. La aplicación te deja ingresar o te muestra mensaje de error

Ejemplo

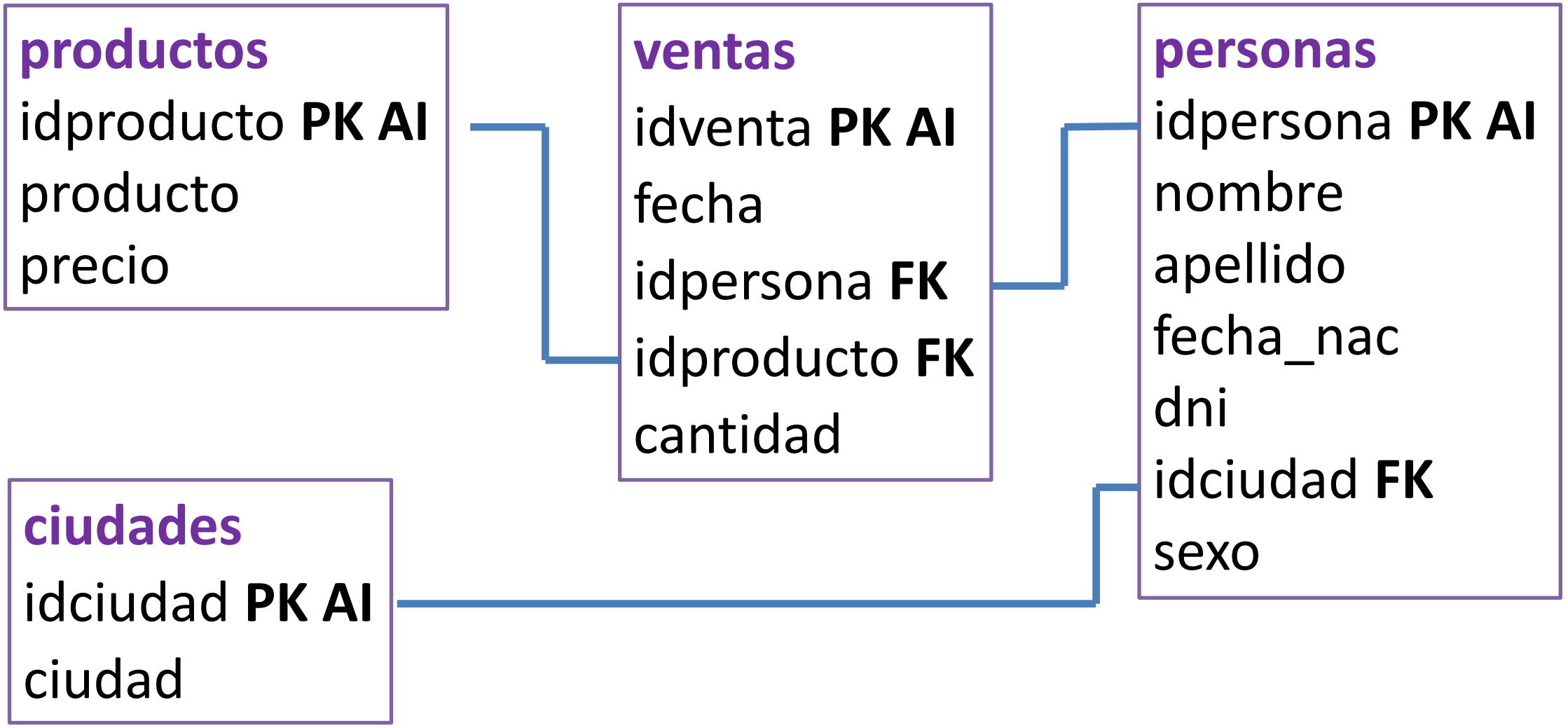
Buscar películas de suspenso

```
SELECT titulo
FROM peliculas
WHERE genero='suspenso'
```

Buscar películas que empiecen con "La casa de p"

```
SELECT titulo
FROM peliculas
WHERE titulo like 'La casa de p%'
```

Práctica en MySQL



Bibliografía

- Elmasri R. Navathe S. (2016). ***Sistemas de Bases de Datos***. 4°ed. España: Addison Wesley Iberoamericana.
- Marqués M. (2011). ***Bases de Datos***. España: Universidad Jaumet I.
- Silberschatz A. Korth H. Sudarshan S. (2002) ***Fundamentos de Bases de Datos***. 4° ed. Buenos Aires: Mc Graw Hill.