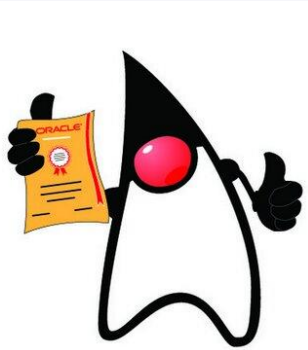
Maurisandev @MauriDeveloper mau.sanchez@bue.edu.ar

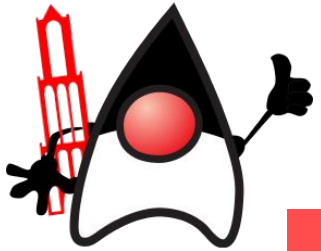
Curso Java FullStack



Codo a Codo 4.0

Clase-24 –.

ACCESO Y CONTROL DE DATOS



Introducción a bases de datos

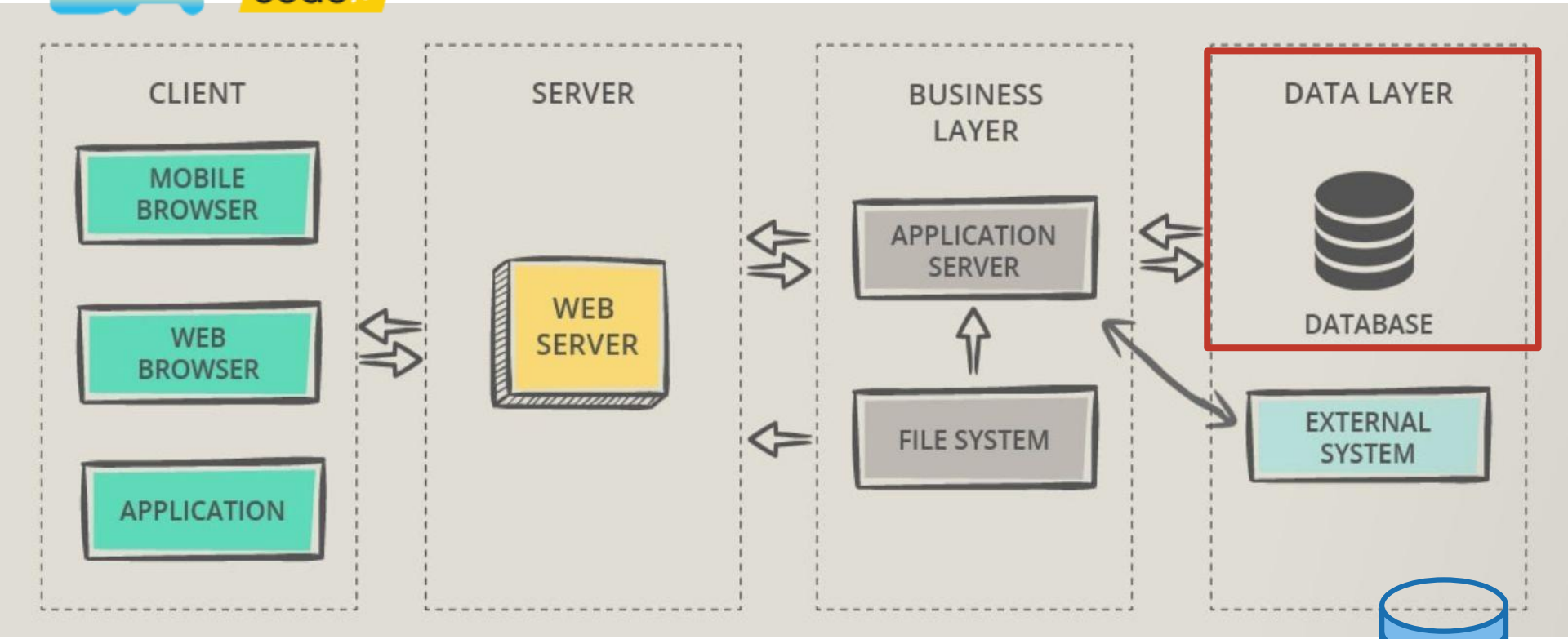
¿Por qué necesitamos bases de datos?

Introducción a bases de datos

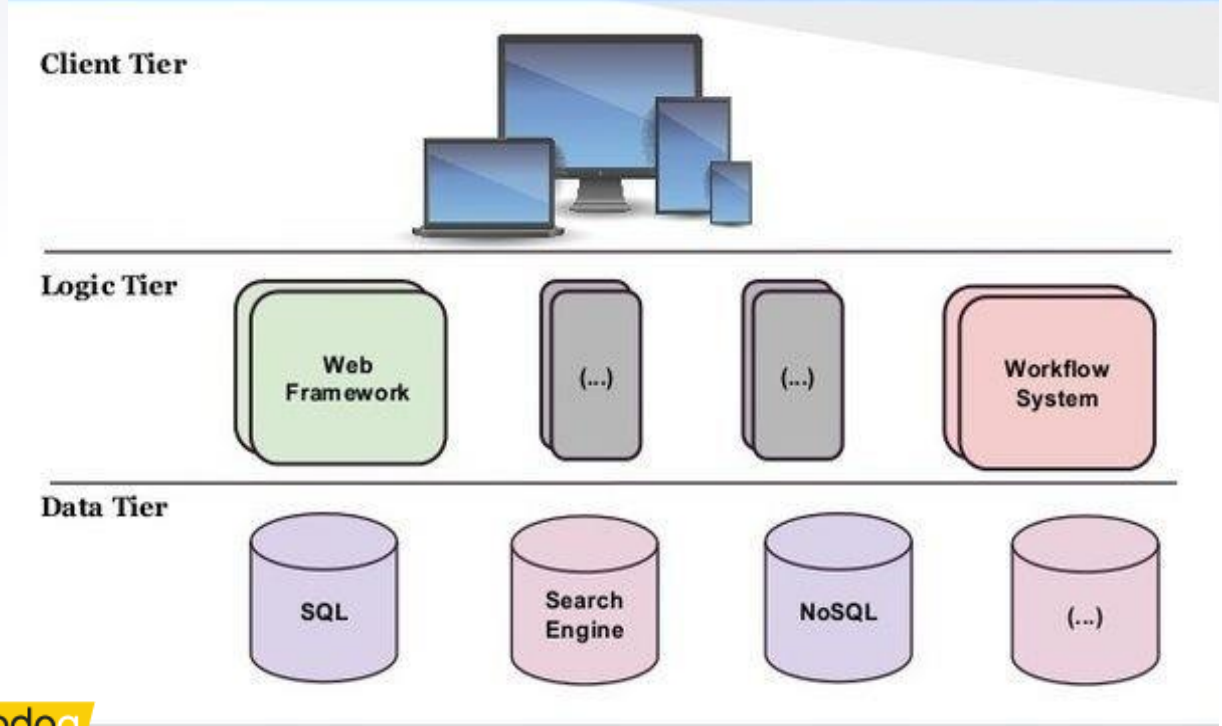
Una **base de datos** se entiende como un conjunto de datos estructurado y almacenado de forma sistemática con objeto de facilitar su posterior utilización.

Introducción a bases de datos

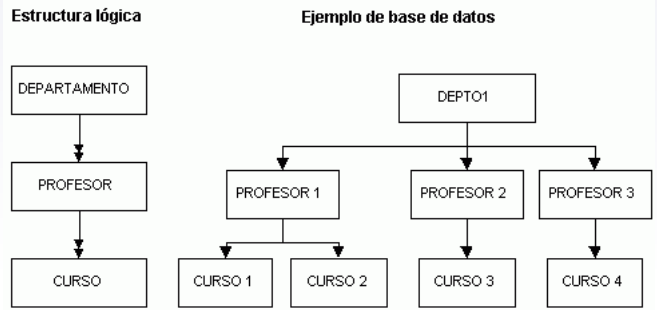
Diremos que un **dato** es un hecho conocido que podemos registrar y que tiene un significado implícito, es decir, según el contexto sobre el cual estemos trabajando.


CLIENT
MOBILE BROWSER
WEB BROWSER
APPLICATION
SERVER
WEB SERVER
BUSINESS LAYER
APPLICATION SERVER
FILE SYSTEM
DATA LAYER
DATABASE
EXTERNAL SYSTEM

3-Tier Architecture



Introducción a bases de datos - Modelos

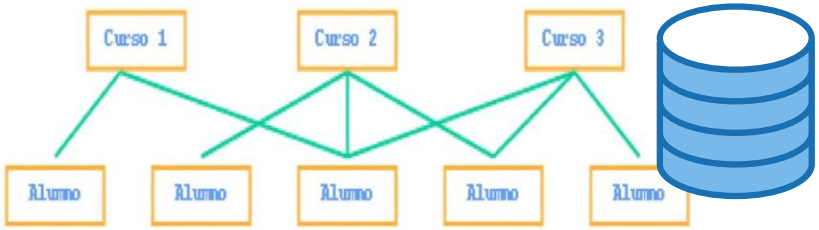


Jerárquica: Nodos interconectados.
Cada nodo tiene un único padre y cero, uno o varios hijos.

En red: Aparición de ciclos en la base de datos.

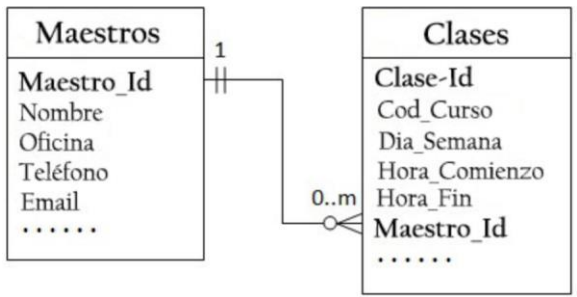
RED

Permite que un mismo nodo tenga varios padres (posibilidad no permitida en el modelo jerárquico).



Introducción a bases de datos - Modelos

Relacional: Esquema basado en tablas.
Las tablas contienen registros (filas) y campos (columnas).



Orientada a Objetos:
Derivado del OOP.
Extiende las capacidades de las bases de datos relacionales.

```
[
 {
  "student_id":1,
  "age":12,
  "score":77
 },
 {
  "student_id":2,
  "age":12,
  "score":68
 },
 {
  "student_id":3,
  "age":11,
  "score":75
 }
]
```

Introducción a bases de datos - Historia

FICHEROS FÍSICOS

BASES DE DATOS RELACIONALES

BASES DE DATOS NO RELACIONALES

Introducción a bases de datos - Beneficios

- Acceso rápido y preciso a lo buscado
- Usuarios múltiples y desde cualquier lugar
- Reutilización de la información
- Gestión de cambios realizados
- Respaldo y restauración de datos

Introducción a bases de datos – Modelo relacional

MOTOR ELEGIDO

- Una **base de datos relacional** es un conjunto de una o más tablas estructuradas en registros (líneas) y campos (columnas), que se vinculan entre sí por un campo en común, en ambos casos posee las mismas características como por ejemplo el nombre de campo, tipo y longitud; a este campo generalmente se le denomina ID, identificador o clave. A esta manera de construir bases de datos se le denomina modelo relacional.

Introducción a bases de datos – Características

- Una **base de datos** se compone de varias **tablas**, denominadas relaciones.
- **No** pueden existir dos tablas con el mismo nombre.
- Cada tabla es un conjunto de **campos** y **registros**.

DBA (Relacionales y No Relacionales)

DBA (Relacionales/No Relacionales) Diferencias

Una *base de datos relacional* es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas.

Una base de datos no relacional Están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Son ampliamente reconocidas porque son fáciles de desarrollar, tanto en funcionalidad como en rendimiento a escala.

<codoa codo/>

Instalación MySQL

https://dev.mysql.com/downloads/mysql/

Instalación Cliente MySQL

- MySQL Workbench
- Heidi SQL

Instalación Apache Friends (Otra forma)

- https://www.apachefriends.org/es/download.html

Instalación DBeaver (Otra forma)

- https://dbeaver.io/

- Tutos: https://www.adictosaltrabajo.com/2015/07/22/dbeaver/
- https://blog.desdelinux.net/dbeaver-una-excelente-herramienta-para-la-gestion-de-diferentes-db/

Instalación MongoDB (NO RELACIONAL)

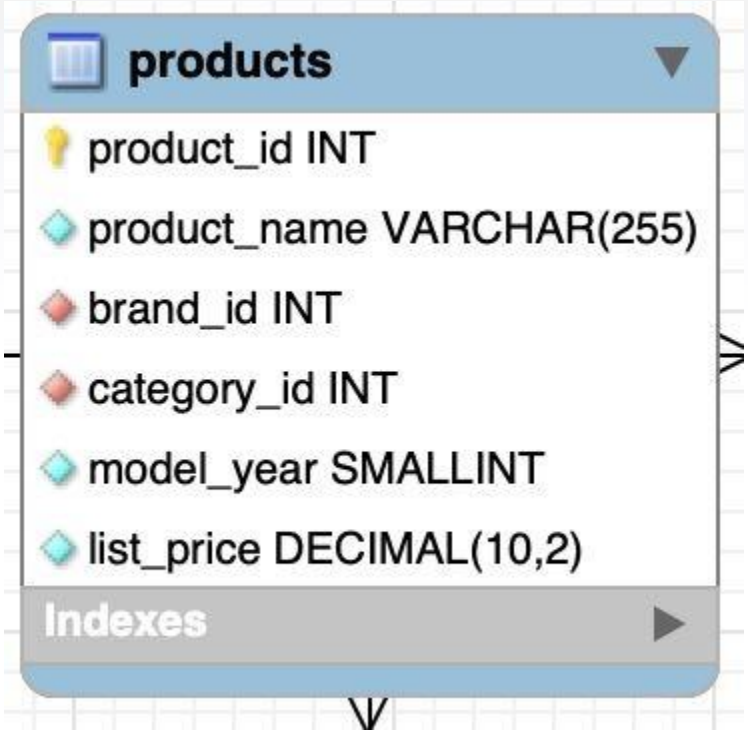
- https://www.mongodb.com/
- Tutorial:
- https://www.codewall.co.uk/mongodb-beginner-tutorial-with-compass-gui-the-mongo-shell-cli/

ESTRUCTURA TABLAS

TABLA

	A	B	C	D	E	F	G	H
1	Purchase ID	Last name	First name	Birthday	Country	Date of purchase	Amount of purchase	
2	1	Davidson	Michael	04/03/1986	United States	10/12/2016	37	
3	2	Vito	Jim	09/01/1994	United Kingdom	02/02/2016	85	
4	3	Johnson	Tom	23/08/1972	France	02/11/2016	83	
5	4	Lewis	Peter	18/10/1979	Germany	22/11/2016	27	
6	5	Koenig	Edward	13/05/1983	Argentina	26/03/2015	43	
7	6	Preston	Jack	16/06/1991	United States	06/11/2016	77	
8	7	Smith	David	11/03/1965	Canada	15/11/2016	23	
9	8	Brown	Luis	03/09/1997	Australia	03/07/2015	74	
10	9	Miller	Thomas	07/01/1980	Germany	07/11/2016	13	
11	10	Williams	Bill	26/07/1960	United States	20/11/2015	80	
12	11	Gemini	Alexia	12/09/1995	Canada	11/03/2017	35	
13	12	Bond	James	25/02/1975	United Kingdom	12/08/2017	40	
14	13	Burgle	Patricia	01/12/1990	United States	18/01/2015	55	
15	14	Reding	Michelle	07/04/1985	Canada	23/02/2017	28	
16	15	Harvey	Billy	14/07/1971	United Kingdom	12/01/2016	41	
17								
18								

De la misma forma que en Microsoft Excel, los motores de base de datos SQL utilizan tablas para almacenar lainformación.


products
product_id INT
product_name VARCHAR(255)
brand_id INT
category_id INT
model_year SMALLINT
list_price DECIMAL(10,2)
Indexes

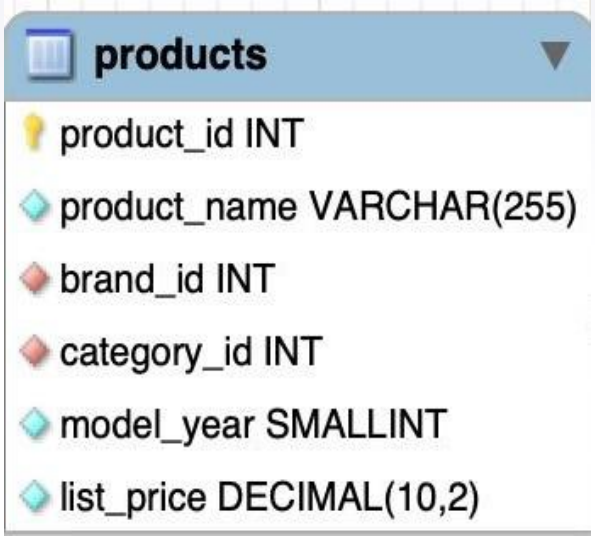
COLUMNAS

TIPOS DE DATOS

DATE TYPE	SPEC
CHAR	String (0 - 255)
VARCHAR	String (0 - 255)
TINYTEXT	String (0 - 255)
TEXT	String (0 - 65535)
BLOB	String (0 - 65535)
MEDIUMTEXT	String (0 - 16777215)
MEDIUMBLOB	String (0 - 16777215)
LONGTEXT	String (0 - 4294967295)
LONGBLOB	String (0 - 4294967295)
TINYINT	Integer (-128 to 127)
SMALLINT	Integer (-32768 to 32767)
MEDIUMINT	Integer (-8388608 to 8388607)

DATA TYPE	SPEC
INT	Integer (-2147483648 to 214748-3647)
BIGINT	Integer (-9223372036854775808 to 9223372036854775807)
FLOAT	Decimal (precise to 23 digits)
DOUBLE	Decimal (24 to 53 digits)
DECIMAL	"DOUBLE" stored as string
DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD HH:MM:SS
TIMESTAMP	YYYYMMDDHHMMSS
TIME	HH:MM:SS
ENUM	One of preset options
SET	Selection of preset options
BOOLEAN	TINYINT(1)

EJEMPLO 1



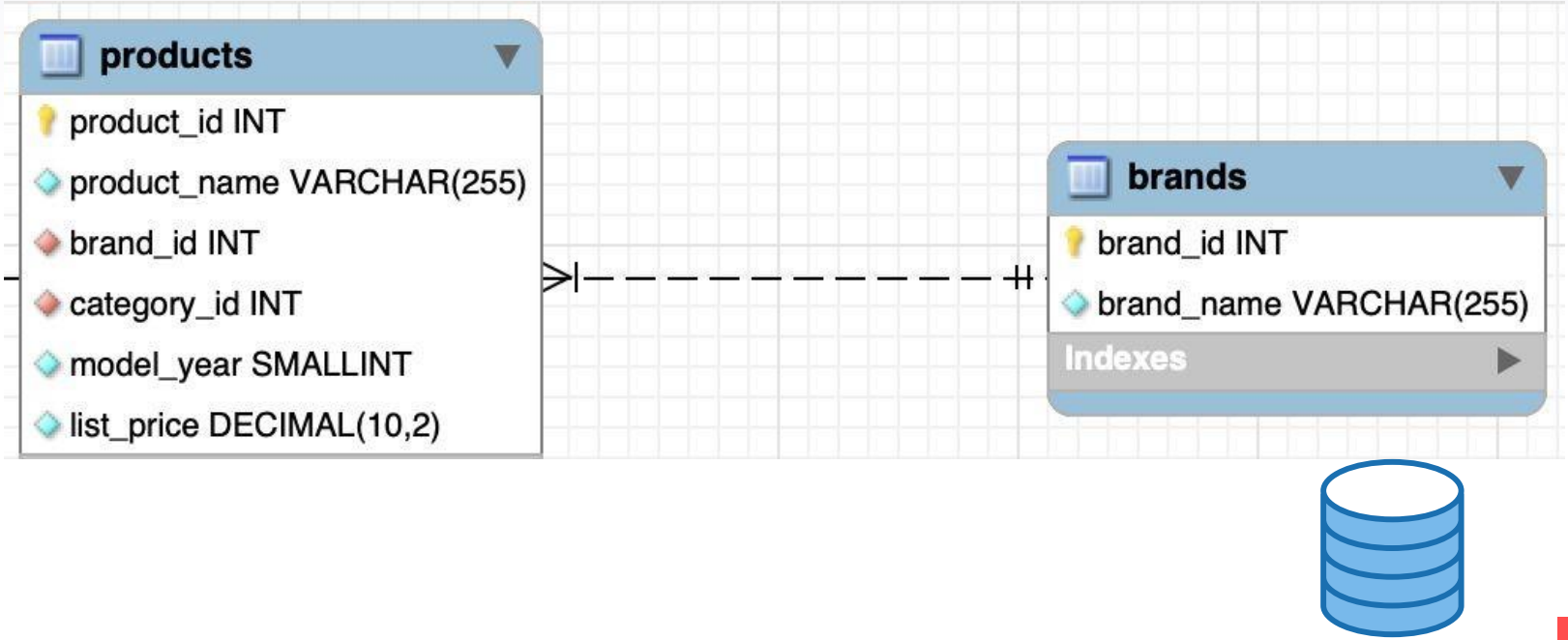
product_id	product_name	brand_id	category_id	model_year	list_price
1	Trek 820 - 2016	9	6	2016	379.99
2	Ritchey Timberwolf Frameset - 2016	5	6	2016	749.99
3	Surly Wednesday Frameset - 2016	8	6	2016	999.99
4	Trek Fuel EX 8 29 - 2016	9	6	2016	2899.99
5	Heller Shagamaw Frame - 2016	3	6	2016	1320.99
6	Surly Ice Cream Truck Frameset - 2016	8	6	2016	469.99
7	Trek Slash 8 27.5 - 2016	9	6	2016	3999.99
8	Trek Remedy 29 Carbon Frameset - 2016	9	6	2016	1799.99
9	Trek Conduit+ - 2016	9	5	2016	2999.99
10	Surly Straggler - 2016	8	4	2016	1549.00
11	Surly Straggler 650b - 2016	8	4	2016	1680.99
12	Electra Townie Original 21D - 2016	1	3	2016	549.99

EJEMPLO 2

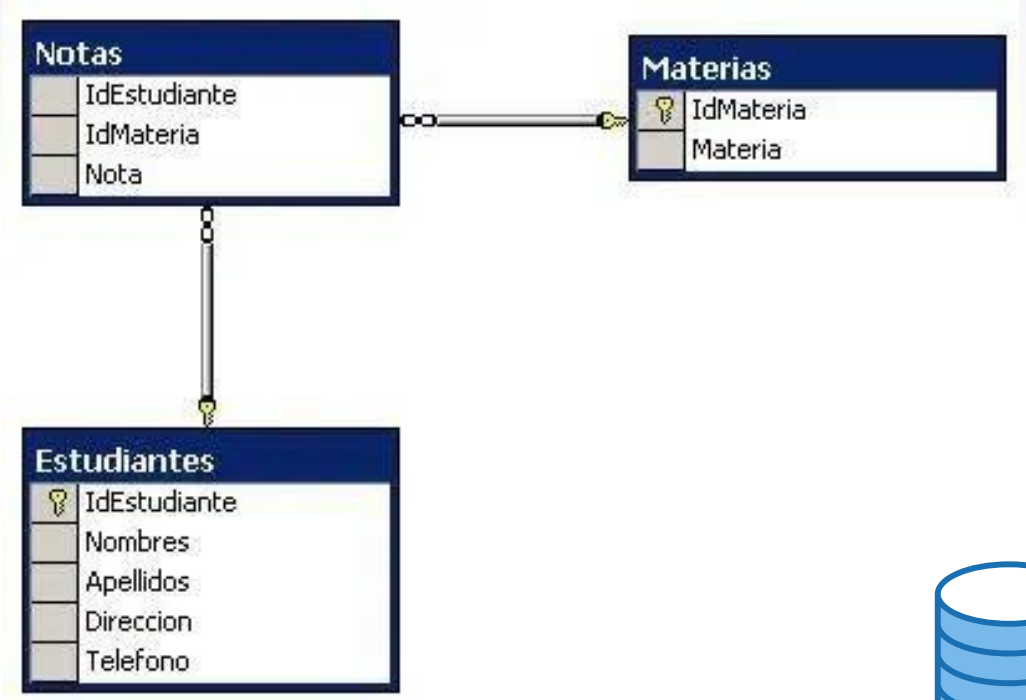
brand_id	brand_name
3	Heller
4	Pure Cycles
5	Ritchey
6	Strider
7	Sun Bicycles
8	Surly
9	Trek

TABLAS CRUZADAS

RELACIONAR MÚLTIPLES TABLAS



RELACIONAR MÚLTIPLES TABLAS



EJERCICIO TABLAS

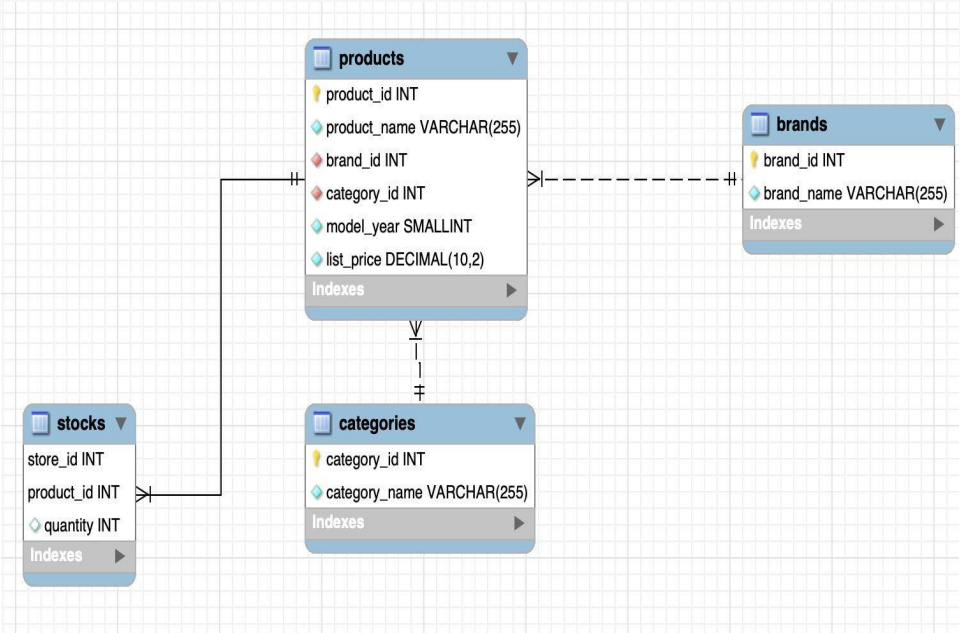
EJERCICIO CONJUNTO: DER

Un cliente concesionario de autos nos pide que diagramemos el esquema de base de datos cumpliendo las siguientes reglas:

- Tiene muchas marcas de autos
- Cada marca de autos tiene muchos modelos
- Los modelos tienen color, número de chasis, descripción del modelo y precio
- Existen vendedores. Cada vendedor tiene asignada solo 1marca de autos para vender.

ESQUEMA DE BASE DE DATOS

DB SCHEMA



Un SCHEMA de BD es un conjunto de tablas.
Para acceder a una de las tablas se utiliza esquema.tabla

Por ejemplo, si mi esquema se llama concesionario la tabla de autos se accede como:

concesionario.auto

INTRO SQL

PARA QUÉ SIRVE SQL

```sql
SELECT * FROM production.brands;
```

SQL (Structured Query Language) es un lenguaje que nos permite interactuar con una base de datos.

Con SQL podemos:

- Buscar, listar y manipular datos en una tabla
- Insertar nuevos datos y modificar datos existentes
- Borrar datos de tablas
- Crear, modificar y eliminar nuevas tablas

SQL posee 4 acciones principales que deberemos aprender:

SELECT | **WHERE** | **UPDATE** | **DELETE** | **INSERT**

CREAR TABLAS

CREAR TABLA

```
CREATE TABLE nombre_tabla (
    columna1 tipo_de_dato,
    columna2 tipo_de_dato,
    columna3 tipo_de_dato,
   ....
);
```

La instrucción **CREATE TABLE** nos permite crear una tabla en la base de datos.

EJEMPLO 1

```sql
CREATE TABLE production.brands (
    brand_id INT,
    brand_name VARCHAR (255) NOT NULL
);
```

Creo la tabla utilizada en el ejemplo anterior.

EJEMPLO 2

```sql
CREATE TABLE production.products (
    product_id INT,
    product_name VARCHAR (255) NOT NULL,
    brand_id INT NOT NULL,
    category_id INT NOT NULL,
    model_year SMALLINT NOT NULL,
    list_price DECIMAL (10, 2) NOT NULL
);
```

SELECT * FROM TABLA

SENTENCIA SELECT

La instrucción **SELECT** nos permite traer datos de una tabla.

```
select * from products;
```

product_id	product_name	brand_id	category_id	model_year	list_price
1	Trek 820 - 2016	9	6	2016	379.99
2	Ritchey Timberwolf Frameset - 2016	5	6	2016	749.99
3	Surly Wednesday Frameset - 2016	8	6	2016	999.99
4	Trek Fuel EX 8 29 - 2016	9	6	2016	2899.99
5	Heller Shagamaw Frame - 2016	3	6	2016	1320.99

SELECT WHERE

SENTENCIA SELECT WHERE

La instrucción **WHERE** nos permite traer datos que cumplan una condición.

```
SELECT * FROM products WHERE model_year = '2017';
```

product_id	product_name	brand_id	category_id	model_year	list_price
27	Surly Big Dummy Frameset - 2017	8	6	2017	999.99
28	Surly Karate Monkey 27.5+ Frameset - 2017	8	6	2017	2499.99
29	Trek X-Caliber 8 - 2017	9	6	2017	999.99
30	Surly Ice Cream Truck Frameset - 2017	8	6	2017	999.99

SHOW TABLES

- Mediante el uso de la sentencia SHOW TABLES podremos listar las tablas de una base de datos.
- Con el agregado de LIKE ‘patrón’, podremos filtrar por alguna tabla en particular

```
1 SHOW TABLES FROM production
```

Consultas favoritas | Histor

Tables_in_production
brands
categories
products
stocks

```
1 SHOW TABLES FROM production LIKE '%nds'
```

Consultas favoritas | Historial de consu

Tables_in_production (%nds)
brands

INSERT INTO

- La sentencia “INSERT INTO” es utilizada para **crear** nuevos registros en una tabla.
- Existen dos formas de crear registros dentro de una tabla
- En la siguiente forma, se especifica las columnas sobre las cuales queremos agregar datos:

```
INSERT INTO brands (brand_name) VALUES ('Audi')
```

- Otra forma de crear registros es indicando valores para todas las columnas de la tabla:

```
INSERT INTO brands VALUES (11, 'BMW');
```

DELETE

DELETE

- La sentencia “DELETE” es utilizada para **eliminar** registros existentes en una tabla.

- Al momento de ejecutar dicha sentencia, debemos prestar mucha atención con los registros que queremos eliminar, ya que si no utilizamos ningún filtro, se eliminarán todos los registros de la tabla. Es decir:

  - DELETE FROM brands, eliminará todos los registros de dicha tabla
  - DELETE FROM brands WHERE brand_id = 1, eliminará el registro cuyo id es igual 1.
  - *Como eliminar el registro con brand Trek?*

EJERCICIO

EJERCICIO FULL

- CREAR UNA BASE DE DATOS DE TEMA LIBRE
  - MÍNIMO 3 TABLAS RELACIONALES
  - CREAR REGISTROS SOBRE LAS MISMAS (UTILIZANDO INSERT)
    - Almacenar las sentencias porque serán evaluadas
  - PREPARAR ALGUNAS CONSULTAS PARA VISUALIZAR DATOS
    - Almacenar las sentencias porque serán evaluadas
  - ELIMINAR ALGUNOS REGISTROS
    - Almacenar las sentencias porque serán evaluadas

UPDATE

UPDATE

- La sentencia “UPDATE” es utilizada para **actualizar** registros existentes en una tabla.

  - Al igual que sucedía con la sentencia DELETE, debemos prestar mucha atención con los registros que queremos actualizar, ya que si no utilizamos ningún filtro, se actualizarán todos los registros de la tabla.

  - Sintaxis

UPDATE [nombre_tabla]

SET nombre_columna = valor WHERE

UPDATE

- Actualizando un campo, a partir de su campo identidad

```
UPDATE usuario
SET estado = 2  WHERE usuario_id = 1;
```

UPDATE

- Actualizando múltiples campos, a partir de su campo identidad.

```
UPDATE usuario
SET estado = 1, usuario = 'juanca' WHERE
usuario_id = 1;
```

PRIMARY KEYS

CLAVE PRIMARIA COMO ID

La instrucción PRIMARY KEY define la clave primaria de la tabla.

```sql
CREATE TABLE Personas (
    id int PRIMARY KEY,
    apellido varchar(255) NOT NULL,
    nombre varchar(255),
    edad int
);
```

Una clave primaria no puede ser NULL.

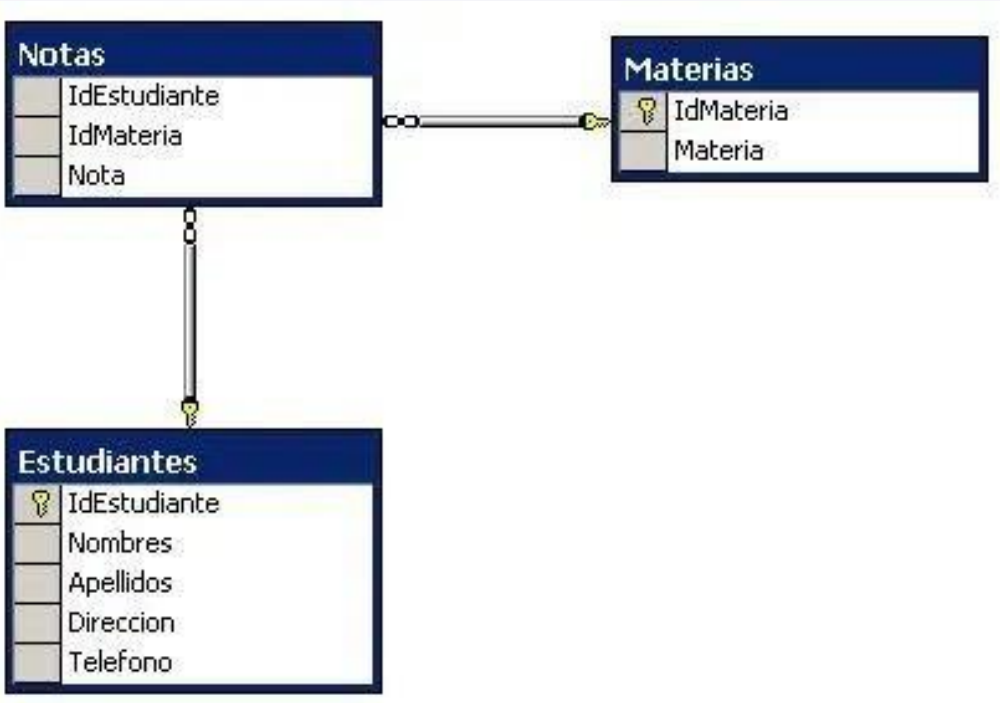
AUTOINCREMENTAL

La instrucción **AUTO_INCREMENT** produce que el identificador se incremente cada vez que se inserte un nuevo registro.

```
CREATE TABLE Personas (
    id int PRIMARY KEY AUTO_INCREMENT,
    apellido varchar(255) NOT NULL,
    nombre varchar(255),
    edad int
);
```

FOREIGN KEYS

RELACIONAR MÚLTIPLES TABLAS



FOREIGN KEY

```sql
CREATE TABLE Estudiantes (
    idEstudiante int PRIMARY KEY AUTO_INCREMENT,
    apellido varchar(255) NOT NULL,
    nombre varchar(255) NOT NULL,
    direccion varchar(255),
    telefono varchar(255)
);
```

```sql
CREATE TABLE Materias (
    idMateria int PRIMARY KEY AUTO_INCREMENT,
    materia varchar(255)
);
```

FOREIGN KEY

La instrucción **FOREIGN KEY** la utilizamos para unir lógicamente dos tablas.

```
CREATE TABLE Notas (
    idEstudiante int PRIMARY KEY,
    idMateria int,
    nota int,
    FOREIGN KEY (idEstudiante) REFERENCES Estudiantes(idEstudiante),
    FOREIGN KEY (idMateria) REFERENCES Materias(idMateria)
);
```

JOINS

JOIN

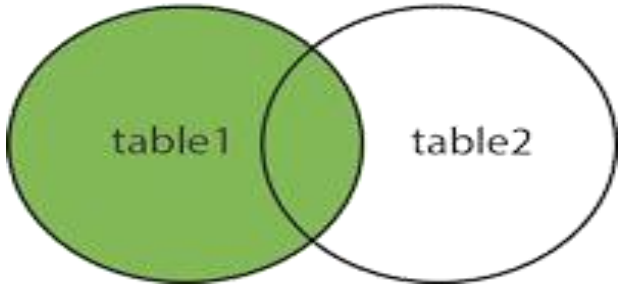
Con un **JOIN** puedo traer todas las Notas de un estudiante y también su información.

```
SELECT * FROM Notas
JOIN Estudiantes ON Estudiantes.idEstudiante = Notas.idEstudiante
WHERE isEstudiante = 1;
```

JOIN

Con un **LEFT JOIN** puedo traer todas las Notas de un estudiante y también su información incluso aunque el estudiante no tenga ninguna nota registrada.

```
SELECT * FROM Estudiantes
LEFT JOIN Notas ON Estudiantes.idEstudiante = Notas.idEstudiante
WHERE isEstudiante = 1;
```



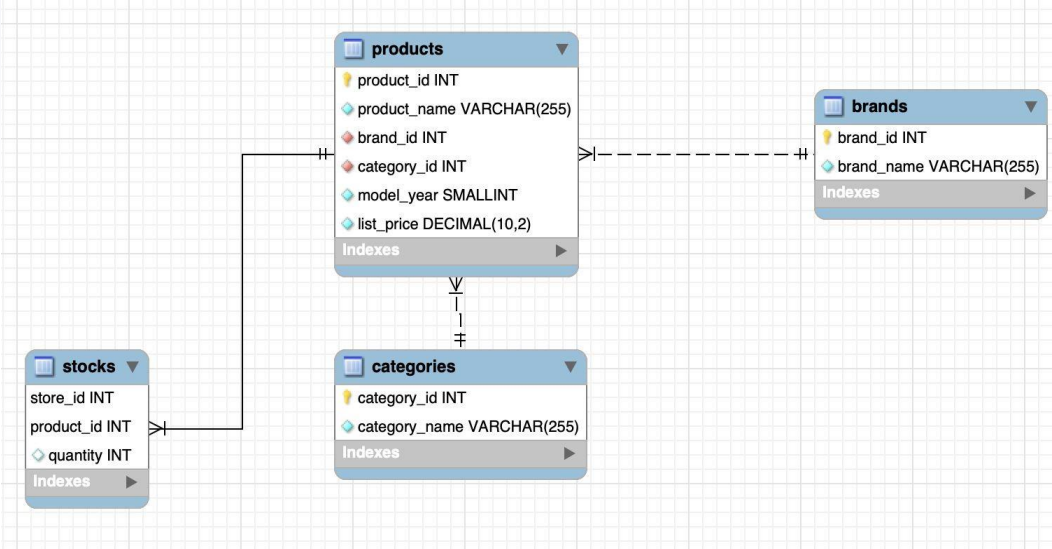
JOINS & GROUP BY

JOINS & GROUP BY

La instrucción **JOIN** en conjunto con **GROUP BY** nos permite agrupar registros de múltiples tablas.

```
SELECT store_name, count(*)
FROM sales.staffs st
JOIN sales.stores s ON s.store_id = st.store_id
GROUP BY s.store_name;
```

DB SCHEMA

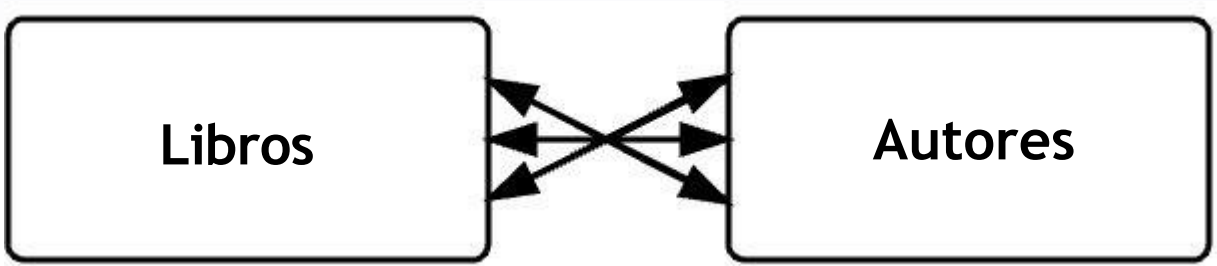


Un SCHEMA de BD es un conjunto de tablas. Para acceder a una de las tablas se utiliza esquema.tabla

Por ejemplo, si mi esquema se llama concesionario la tabla de autos se accede como:

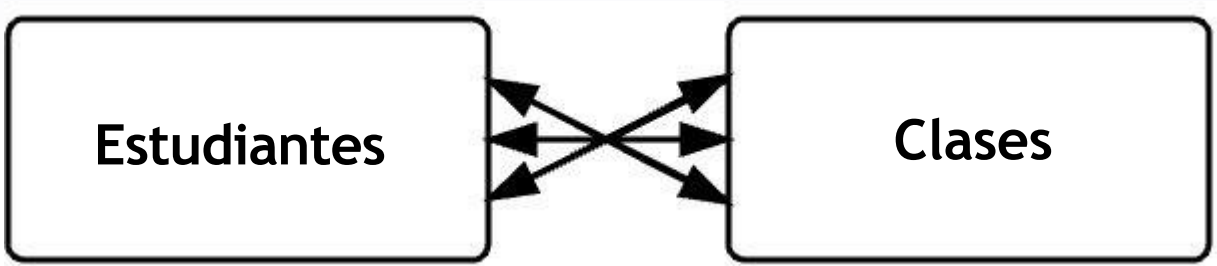
concesionario.auto

Libros <-> Autores



Relación muchos a muchos

Relación muchos a muchos



Relación muchos a muchos

TABLA INTERMEDIA

La tabla intermedia almacena la relación entre los estudiantes y las clases.

```sql
CREATE TABLE estudiante_clase (
    id_estudiante INT,
    id_clase INT,
    FOREIGN KEY (id_estudiante) REFERENCES estudiante(id_estudiante),
    FOREIGN KEY (id_clase) REFERENCES clase(id_clase)
);
```

TABLA INTERMEDIA

La tabla intermedia puede contener información extra sobre la relación.

```sql
CREATE TABLE estudiante_clase (
    id_estudiante INT,
    id_clase INT,
    nota INT,
    FOREIGN KEY (id_estudiante) REFERENCES estudiante(id_estudiante),
    FOREIGN KEY (id_clase) REFERENCES clase(id_clase)
);
```

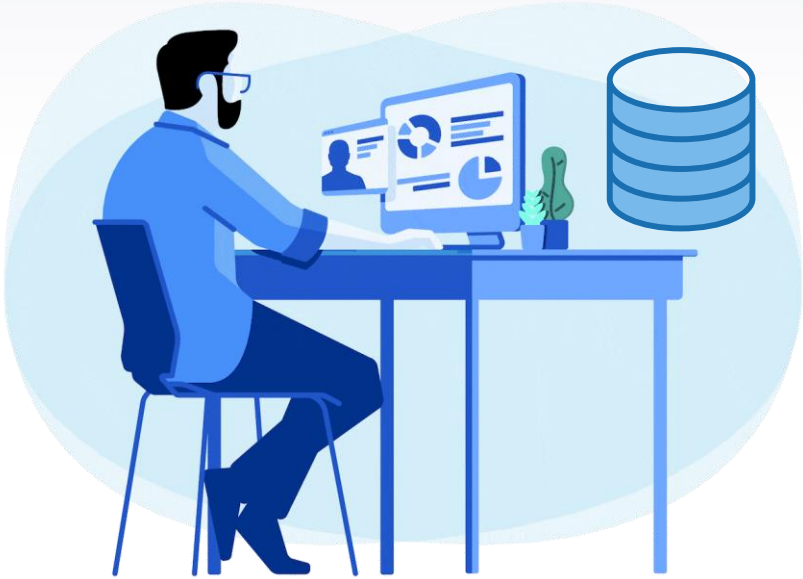
FUNCIONES QUE AGREGAN DATOS

Las **funciones** que agregan datos se utilizan en conjunto con los **GROUP BY**

AVG	Utilizada para calcular el promedio de los valores de un campo determinado
COUNT	Utilizada para devolver el número de registros de la selección
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado
MAX	Utilizada para devolver el valor más alto de un campo especificado
MIN	Utilizada para devolver el valor más bajo de un campo especificado

Maurisandev @MauriDeveloper maurisan4011@gmail.com

Muchas Gracias!



****

NOS VEMOS EL Martes 19:00!!