PROYECTO ABP
GESTIÓN DE GUARDERÍA

ANDRES FELIPE MOLINARES BOLAÑOS
ESTUDIANTE

JOHN ARRIETA ARRIETA
DOCENTE

V SEMESTRE
ESCUELA DE INGENIERÍA DE SISTEMAS

UNIVERSIDAD DEL SINÚ
CARTAGENA DE INDIAS D, T Y C
30 DE ABRIL DE 2021

INTRODUCCIÓN

En el siguiente informe se explicará de manera detallada el desarrollo del caso problema asignado para su respectivo análisis e implementación en 4 motores de bases de datos los cuales son: MySQL, PostgreSQL, SQL Server y Oracle DB.

Se evidenciará con una breve descripción cada una de las sentencias DDL aplicadas al caso problema, tales como, CREATE, ALTER, DROP, entre otras. Así como también, se tendrá en cuenta las sentencias DML (INSERT, UPDATE, DELETE, SELECT) con ejemplos aplicados al ejercicio.

OBJETIVO GENERAL

Analizar el caso de estudio planteado para sistematizar el proceso de gestión de datos, modelar y desarrollar la base de datos, y proporcionar las soluciones para el cliente.

OBJETIVOS ESPECÍFICOS

- ➢ Investigar y analizar el caso problema para diseñar los diagramas entidad relación y modelo relacional de base de datos.
- ➢ Identificar las sentencias DDL y DML que se utilizarán para el desarrollo de las bases de datos
- ➢ Implementar aquellas consultas que se consideran necesarias para las pruebas de funcionamiento.

CASO PROBLEMA:

Una guardería desea controlar los gastos que cada uno de los niños realiza a través de su asistencia y de las comidas que consume. De cada niño se desea conocer los datos propios de su matrícula en el centro educativo, es decir, el número de matrícula, el nombre, la fecha de nacimiento y la fecha de ingreso en la guardería. Para aquellos niños que se hayan dado de baja, también se desea conocer la fecha de la baja.

Los niños sólo pueden ser recogidos en la guardería por un conjunto de personas que suelen ser un familiar del niño o un conocido de sus familiares De éstos se desea conocer el DNI, el nombre, la dirección y al menos un número de teléfono de contacto. Además, debe de quedar constancia de cuál es la relación entre la persona autorizada y el niño. El coste mensual del niño en la guardería es abonado por una persona, de la que se desea conocer el DNI, el nombre, la dirección, el teléfono, y el número de la cuenta corriente en la que se realizará el cargo. Estas personas también pueden estar autorizadas para recoger al niño. En la guardería aparece un conjunto de menús, compuesto por una serie de platos concretos, cada uno de los cuales presentan unos ingredientes determinados. Cada menú se identifica por un número, mientras que los platos y los ingredientes se caracterizan por su nombre. Un niño puede ser alérgico a diferentes ingredientes, y por tanto no puede consumir los platos en los que aparece este ingrediente. Estas alergias deben de ser controladas para evitar posibles intoxicaciones en los niños.

El cargo mensual de un niño se calcula como la suma de un coste fijo mensual y el coste de las comidas realizadas. Este último se obtiene a partir del número de días que el niño ha comido en la guardería, por lo que resulta necesario controlar dicho número. Además, se desea saber el menú que ha consumido cada niño cada día.

CREACIÓN DE LOS DIAGRAMAS DE ENTIDAD-RELACIÓN MODELO RELACIONAL.

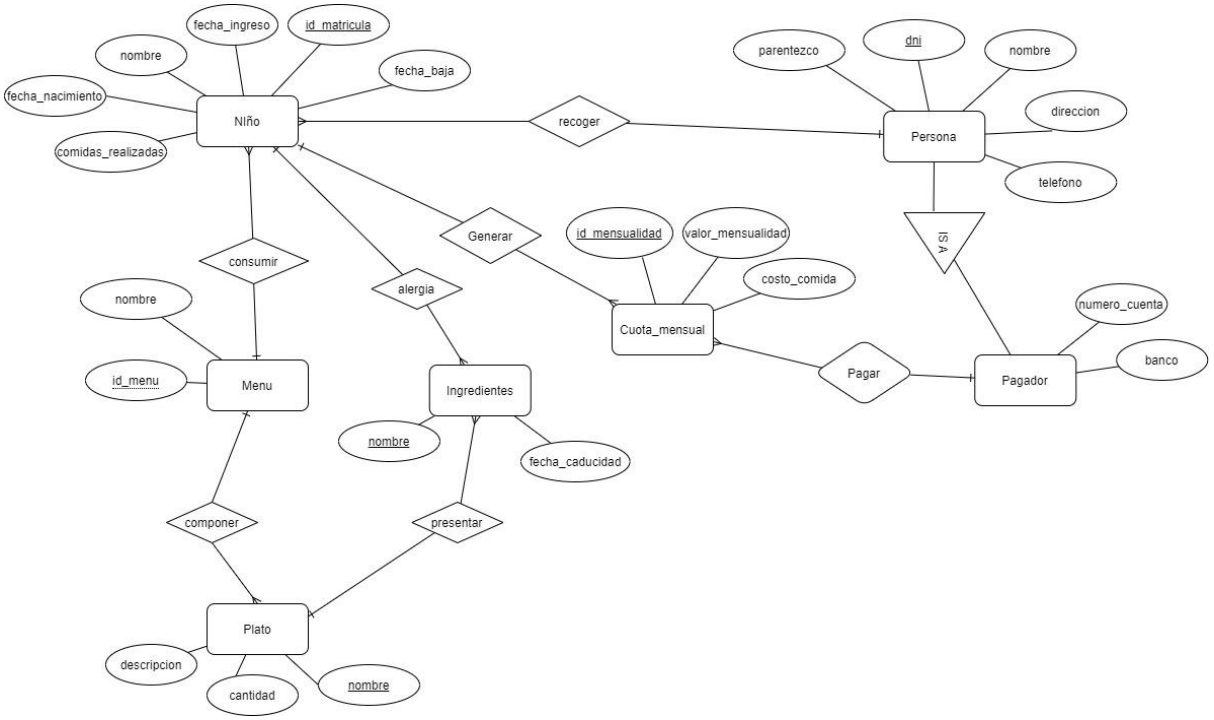
El modelo entidad relación es una herramienta que permite representar de manera simplificada los componentes que participan en un proceso de negocio y el modo en el que estos se relacionan entre sí. En él las entidades se representan utilizando rectángulos, los atributos por medio de círculos o elipses y las relaciones como líneas que conectan las entidades que tienen algún tipo de vínculo.

El modelo relacional se basa en el concepto matemático de relación, que gráficamente se representa mediante una tabla. Es decir, una relación es una tabla, con columnas y filas. Un SGBD (Sistema Gestor de Bases de Datos) sólo necesita que el usuario pueda percibir la base de datos como un conjunto de tablas.

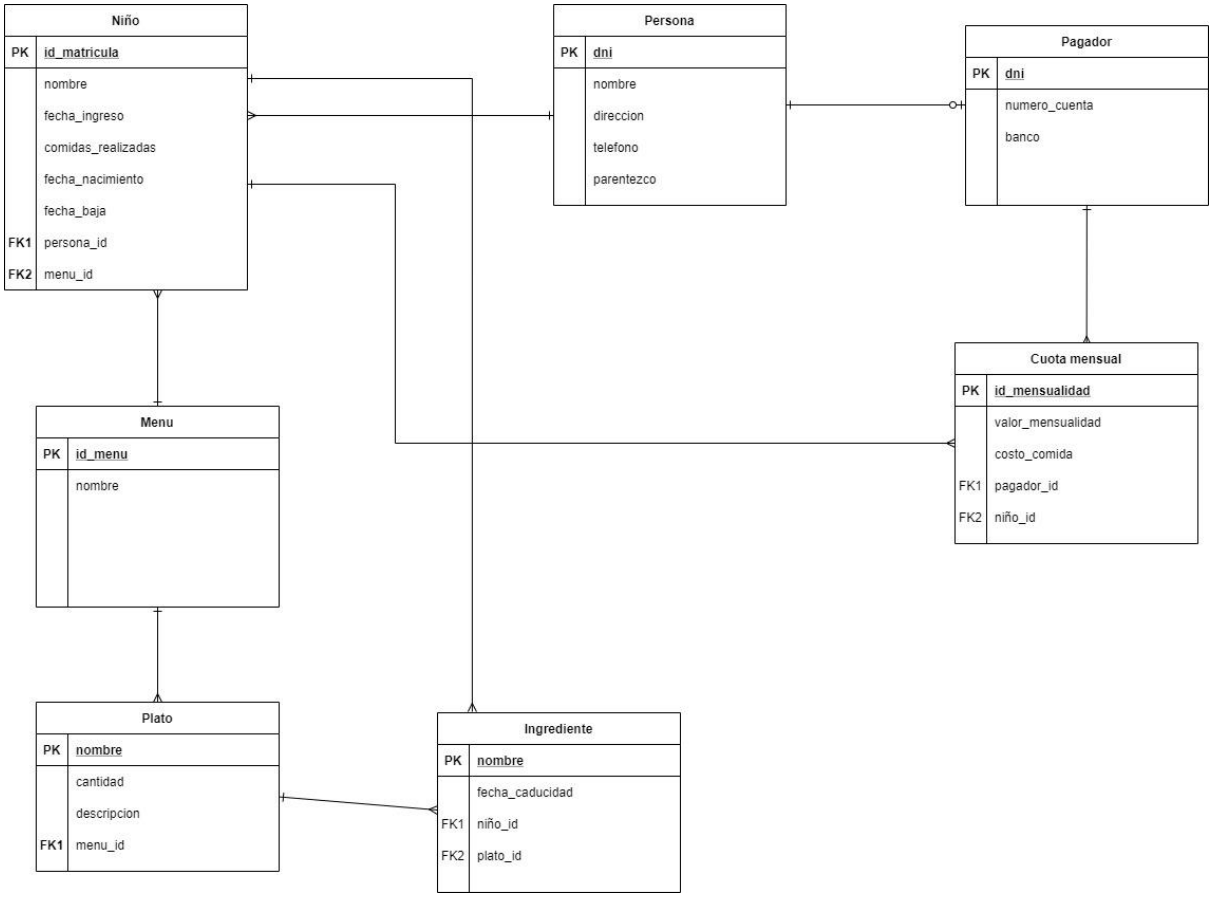
Es muy importante iniciar desarrollando los diagramas mencionados anteriormente, ya que así tenemos una percepción más detallada del funcionamiento de la base de datos.

Para este caso, los diagramas que cree fueron los siguientes:

Diagrama entidad-relación.



Modelo relacional.



DESARROLLO DE LA BASE DE DATOS

- **SENTENCIAS DDL**

Sentencias DDL son aquellas utilizadas para la creación de una base de datos y todos sus componentes: tablas, índices, relaciones, disparadores (triggers), procedimientos almacenados, etc.

Creación de base de datos.

- **MYSQL**

```
CREATE DATABASE guarderia;
USE guarderia;
```

- **POSTGRESQL**

```
CREATE DATABASE guarderia;
\c guarderia;
```

- **ORACLE**

```
CREATE DATABASE guarderia;
```

- **SQL SERVER**

```
CREATE DATABASE guarderia;
go
```

Creación de las tablas en la base de datos.

- **MYSQL**

```
CREATE TABLE `niño` (
  `id_matricula` int PRIMARY KEY NOT NULL,
  `nombre` varchar(50) NOT NULL,
  `fecha_ingreso` date DEFAULT NULL,
  `comidas_realizadas` int DEFAULT NULL,
  `fecha_baja` date DEFAULT NULL,
  `persona_id` varchar(11) NOT NULL,
  `menu_id` int NOT NULL,
  PRIMARY KEY (`id_matricula`)
) ENGINE = INNODB;

CREATE TABLE `cuota_mensual` (
  `id_mensualidad` varchar(10) PRIMARY KEY NOT NULL,
  `valor_mensualidad` float NOT NULL,
  `costo_comida` float DEFAULT NULL,
  `pagador_id` varchar(11) NOT NULL,
  `niño_id` int NOT NULL,
  PRIMARY KEY (`id_mensualidad`)
```

```
) ENGINE = INNODB;
CREATE TABLE `ingredientes` (
  `nombre` varchar(50) PRIMARY KEY NOT NULL,
  `fecha_caducidad` date DEFAULT NULL,
  `niño_id` int NOT NULL,
  `plato_id` varchar(50) NOT NULL,
  PRIMARY KEY (`nombre`)
) ENGINE = INNODB;

CREATE TABLE `menu` (
  `id_menu` int PRIMARY KEY NOT NULL AUTO_INCREMENT,
  `nombre` varchar(100) NOT NULL
  PRIMARY KEY (`id_menu`)
) ENGINE=InnoDB;

CREATE TABLE `personas` (
  `dni` varchar(11) PRIMARY KEY NOT NULL,
  `nombre` varchar(50) NOT NULL,
  `direccion` text,
  `telefono` varchar(15) NOT NULL,
  `parentezco` varchar(20) NOT NULL,
  PRIMARY KEY (`dni`)
) ENGINE=InnoDB;

CREATE TABLE `pagador` (
  `dni` varchar(11) PRIMARY KEY NOT NULL,
  `numero_cuenta` varchar(20) NOT NULL,
  `banco` varchar(100) NOT NULL,
   PRIMARY KEY (`dni`)
) ENGINE=InnoDB;

CREATE TABLE `plato` (
  `nombre` varchar(50) PRIMARY KEY NOT NULL,
  `cantidad` float NOT NULL,
  `descripcion` text,
  `menu_id` int NOT NULL,
  PRIMARY KEY (`nombre`)
) ENGINE=InnoDB;
```

- **POSTGRESQL**

```
CREATE TABLE public."nino" (
    id_matricula integer PRIMARY KEY NOT NULL,
    fecha_ingreso date,
    comidas_realizadas integer,
    fecha_nacimiento date NOT NULL,
    fecha_baja date,
    persona_id character varying(15) NOT NULL,
    menu_id integer NOT NULL
);

CREATE TABLE public.cuota_mensual (
    id_mensualidad character varying(20) PRIMARY KEY NOT NULL,
    valor_mensualidad double precision NOT NULL,
    costo_comida double precision,
    pagador_id character varying(15) NOT NULL,
    "nino_id" integer NOT NULL
);

CREATE TABLE public.menu (
    id_menu integer PRIMARY KEY NOT NULL,
    nombre varying(100) NOT NULL,
);

CREATE TABLE public.pagador (
    dni character varying(15)  PRIMARY KEY NOT NULL,
    numero_cuenta character varying(20) NOT NULL,
    banco character varying(50) NOT NULL
);

CREATE TABLE public.persona (
    dni character varying(15) PRIMARY KEY NOT NULL,
    nombre character varying(50) NOT NULL,
    direccion text,
    telefono character varying(10),
    parentezco character varying(50) NOT NULL
);
```

```
CREATE TABLE public.plato (
    nombre character varying(50) PRIMARY KEY NOT NULL,
    cantidad double precision NOT NULL,
    descripcion text,
    menu_id integer NOT NULL
);
```

- **ORACLE**

```
 CREATE TABLE CUOTA_MENSUAL
   (  "ID_MENSUALIDAD" VARCHAR2(10 BYTE) PRIMARY KEY,
      "VALOR_MENSUALIDAD" FLOAT(126),
      "COSTO_COMIDA" FLOAT(126),
      "PAGADOR_ID" VARCHAR2(15 BYTE),
      "NINO_ID" VARCHAR2(15 BYTE)
   );

  CREATE TABLE INGREDIENTE
   (  "NOMBRE" VARCHAR2(50 BYTE) PRIMARY KEY,
      "FECHA_CADUCIDAD" DATE,
      "NINO_ID" VARCHAR2(15 BYTE),
      "PLATO_ID" VARCHAR2(50 BYTE)
   );

  CREATE TABLE MENU
   (  "ID_MENU" NUMBER(*,0) PRIMARY KEY,
      "NOMBRE" VARCHAR2(100 BYTE)
   );

  CREATE TABLE NINO
   (  "ID_MATRICULA" VARCHAR2(15 BYTE) PRIMARY KEY,
      "NOMBRE" VARCHAR2(50 BYTE),
      "FECHA_INGRESO" DATE,
      "COMIDAS_REALIZADAS" NUMBER(*,0),
      "FECHA_NACIMIENTO" DATE,
      "FECHA_BAJA" DATE,
      "PERSONA_ID" VARCHAR2(15 BYTE),
      "MENU_ID" NUMBER(*,0)
   );
```

```
  CREATE TABLE PAGADOR
   (  "DNI" VARCHAR2(15 BYTE) PRIMARY KEY,
      "NUMERO_CUENTA" VARCHAR2(20 BYTE),
      "BANCO" VARCHAR2(50 BYTE)
   ) ;

  CREATE TABLE PERSONA
   (  "DNI" VARCHAR2(15 BYTE),
      "NOMBRE" VARCHAR2(50 BYTE),
      "DIRECCION" VARCHAR2(100 BYTE),
      "TELEFONO" VARCHAR2(10 BYTE),
      "PARENTEZCO" VARCHAR2(50 BYTE)
   );

  CREATE TABLE PLATO
   (  "NOMBRE" VARCHAR2(50 BYTE) PRIMARY KEY,
      "CANTIDAD" FLOAT(126),
      "DESCRIPCION" VARCHAR2(100 BYTE),
      "MENU_ID" NUMBER(*,0)
   );
```

- **SQL SERVER**

```
CREATE TABLE cuota_mensual(
      id_mensualidad varchar(10) PRIMARY KEY NOT NULL,
      valor_mensualidad float NOT NULL,
      costo_comida float NOT NULL,
      pagador_id varchar(15) NOT NULL,
      niño_id varchar(5) NOT NULL);
go

CREATE TABLE ingrediente(
      nombre varchar(50) PRIMARY KEY NOT NULL,
      fecha_caducidad date NULL,
      niño_id varchar(5) NOT NULL,
      plato_id varchar(50) NOT NULL);
go

CREATE TABLE Menu(
      id_menu int NOT NULL,
      nombre varchar (100) NOT NULL);
```

```
go

CREATE TABLE nino(
      id_matricula varchar(5) PRIMARY KEY NOT NULL,
      nombre varchar(50) NOT NULL,
      fecha_ingreso date NOT NULL,
      comidas_realizadas int NULL,
      fecha_nacimiento date NOT NULL,
      fecha_baja date NULL,
      persona_id varchar(15) NOT NULL,
      menu_id int NOT NULL);
go

CREATE TABLE pagador(
      dni varchar(15)  PRIMARY KEY NOT NULL,
      numero_cuenta varchar(20) NOT NULL,
      banco varchar(50) NOT NULL);
go

CREATE TABLE Persona(
      dni varchar(15) PRIMARY KEY NOT NULL,
      nombre varchar(50) NOT NULL,
      direccion text NULL,
      telefono varchar(10) NOT NULL,
      parentezco varchar(50) NOT NULL);
go

CREATE TABLE Plato(
      nombre varchar(50) PRIMARY KEY NOT NULL,
      cantidad float NULL,
      descripcion text NULL,
      menu_id int NOT NULL);
go
```

Relacionando las tablas (Índices y Foreign Keys).

- **MYSQL**

```
CREATE INDEX `idx_cuota_pagador` ON cuota_mensual(`pagador_id`);
CREATE INDEX `idx_cuota_niño` ON cuota_mensual(`niño_id`);

ALTER TABLE cuota_mensual
ADD CONSTRAINT `fk_cuota_niño`
FOREIGN KEY (`niño_id`)
REFERENCES `niño` (`id_matricula`);

ALTER TABLE cuota_mensual
ADD CONSTRAINT `fk_cuota_pagador`
FOREIGN KEY (`pagador_id`)
REFERENCES `pagador` (`dni`);

CREATE INDEX  `idx_ingrediente_niño` ON ingrediente(`niño_id`);
CREATE INDEX `idx_ingredientes_plato` ON ingrediente(`plato_id`);

ALTER TABLE ingredientes
ADD CONSTRAINT `fk_ingredientes_niño`
FOREIGN KEY (`niño_id`)
REFERENCES `niño` (`id_matricula`);

ALTER TABLE ingredientes
ADD CONSTRAINT `fk_ingredientes_plato`
FOREIGN KEY (`plato_id`)
REFERENCES `plato` (`nombre`);

CREATE INDEX `idx_niño_persona` ON niño(`persona_id`),
CREATE INDEX `idx_niño_menu` ON niño(`menu_id`);

ALTER TABLE niño
ADD CONSTRAINT `fk_niño_menu`
FOREIGN KEY (`menu_id`)
REFERENCES `menu` (`id_menu`);

ALTER TABLE niño
ADD CONSTRAINT `fk_niño_persona`
FOREIGN KEY (`persona_id`)
REFERENCES `personas` (`dni`);
```

```
CREATE INDEX `idx_persona_pagador` ON pagador(`dni`);

ALTER TABLE pagador
ADD CONSTRAINT `fk_persona_pagador`
FOREIGN KEY (`dni`)
REFERENCES `personas` (`dni`);

CREATE INDEX `idx_plato_menu` ON plato(`menu_id`),
ADD CONSTRAINT `fk_plato_menu`
FOREIGN KEY (`menu_id`)
REFERENCES `menu` (`id_menu`);
```

- **POSTGRESQL**

```
CREATE INDEX "idx_cuota_nino" ON public.cuota_mensual ("nino_id");
ALTER TABLE public.cuota_mensual
ADD CONSTRAINT "fk_cuota_nino" FOREIGN KEY ("nino_id") REFERENCES
public.nino(id_matricula);

CREATE INDEX idx_cuota_pagador ON public.cuota_mensual
(pagador_id);
ALTER TABLE public.cuota_mensual
ADD CONSTRAINT fk_cuota_pagador FOREIGN KEY (pagador_id)
REFERENCES public.pagador(dni);

CREATE INDEX "idx_ingrediente_nino" ON public.ingrediente
("nino_id");
ALTER TABLE public.ingrediente
ADD CONSTRAINT "fk_ingrediente_nino" FOREIGN KEY ("nino_id")
REFERENCES public."nino"(id_matricula);

CREATE INDEX idx_ingrediente_plato ON
public.ingrediente(plato_id);
ALTER TABLE public.ingrediente
ADD CONSTRAINT fk_ingrediente_plato FOREIGN KEY (plato_id)
REFERENCES public.plato(nombre);

CREATE INDEX "idx_nino_menu" ON public.nino(menu_id);
ALTER TABLE public.nino
ADD CONSTRAINT "fk_nino_menu" FOREIGN KEY (menu_id)
REFERENCES public.menu(id_menu);
```

```
CREATE INDEX "idx_nino_persona" ON public.nino(persona_id);
ALTER TABLE public.nino
ADD CONSTRAINT "fk_nino_persona" FOREIGN KEY (persona_id)
REFERENCES public.persona(dni);
CREATE INDEX idx_pagador_persona ON public.pagador(dni);
ALTER TABLE public.pagador
ADD CONSTRAINT fk_pagador_persona FOREIGN KEY (dni)
REFERENCES public.persona(dni);

CREATE INDEX idx_plato_menu ON public.plato(menu_id);
ALTER TABLE public.plato
ADD CONSTRAINT fk_plato_menu FOREIGN KEY (menu_id)
REFERENCES public.menu(id_menu);
```

- **ORACLE SQL**

```
CREATE INDEX "IDX_NINO_MENU" ON NINO(MENU_ID);
ALTER TABLE NINO
ADD CONSTRAINT "FK_NINO_MENU"
FOREIGN KEY (MENU_ID)
REFERENCES MENU (ID_MENU);

CREATE INDEX "IDX_NINO_PERSONA" ON NINO(PERSONA_ID);
ALTER TABLE NINO
ADD CONSTRAINT "FK_NINO_PERSONA"
FOREIGN KEY (PERSONA_ID)
REFERENCES PERSONA(DNI);

CREATE INDEX "IDX_INGREDIENTE_NINO" ON INGREDIENTE(NINO_ID) ;
ALTER TABLE INGREDIENTE
ADD CONSTRAINT "FK_INGREDIENTE_NINO"
FOREIGN KEY (NINO_ID)
REFERENCES NINO (ID_MATRICULA);

CREATE INDEX "IDX_CUOTA_NINO" ON CUOTA_MENSUAL(NINO_ID);
ALTER TABLE CUOTA_MENSUAL
ADD CONSTRAINT "FK_CUOTA_NINO"
FOREIGN KEY (NINO_ID)
REFERENCES NINO(ID_MATRICULA);
CREATE INDEX "IDX_NINO_PERSONA" ON NINO(PERSONA_ID) ;
```

```
CREATE INDEX "IDX_PLATO_MENU" ON PLATO(MENU_ID);
ALTER TABLE INGREDIENTE
ADD CONSTRAINT "FK_INGREDIENTE_PLATO"
FOREIGN KEY (PLATO_ID)
REFERENCES PLATO (NOMBRE);

CREATE INDEX "IDX_CUOTA_PAGADOR" ON CUOTA_MENSUAL (PAGADOR_ID);
ALTER TABLE CUOTA_MENSUAL
ADD CONSTRAINT "FK_CUOTA_PAGADOR"
FOREIGN KEY (PAGADOR_ID)
REFERENCES PAGADOR(DNI);

ALTER TABLE PAGADOR
ADD CONSTRAINT "FK_PAGADOR_PERSONA"
FOREIGN KEY (DNI)
REFERENCES PERSONA (DNI);
```

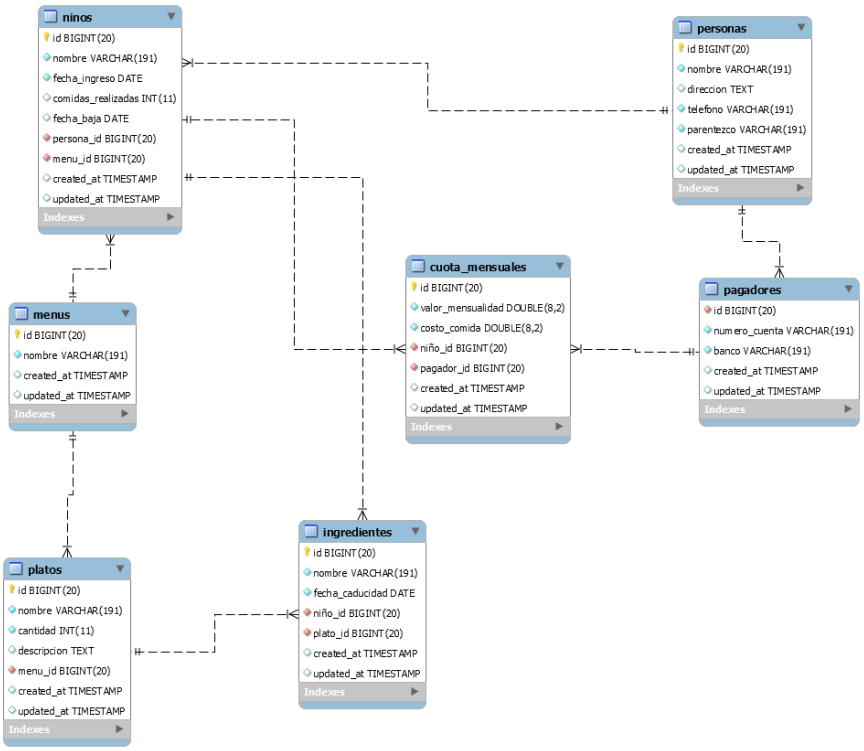
- **SQL SERVER**

```
CREATE INDEX "IDX_NINO_MENU" ON NINO(MENU_ID);
go
ALTER TABLE NINO
ADD CONSTRAINT "FK_NINO_MENU"
FOREIGN KEY (MENU_ID)
REFERENCES MENU (ID_MENU);
go

CREATE INDEX "IDX_NINO_PERSONA" ON NINO(PERSONA_ID);
go
ALTER TABLE NINO
ADD CONSTRAINT "FK_NINO_PERSONA"
FOREIGN KEY (PERSONA_ID)
REFERENCES PERSONA(DNI);
go

CREATE INDEX "IDX_INGREDIENTE_NINO" ON INGREDIENTE(NINO_ID) ;
go
ALTER TABLE INGREDIENTE
ADD CONSTRAINT "FK_INGREDIENTE_NINO"
FOREIGN KEY (NINO_ID)
REFERENCES NINO (ID_MATRICULA);
```

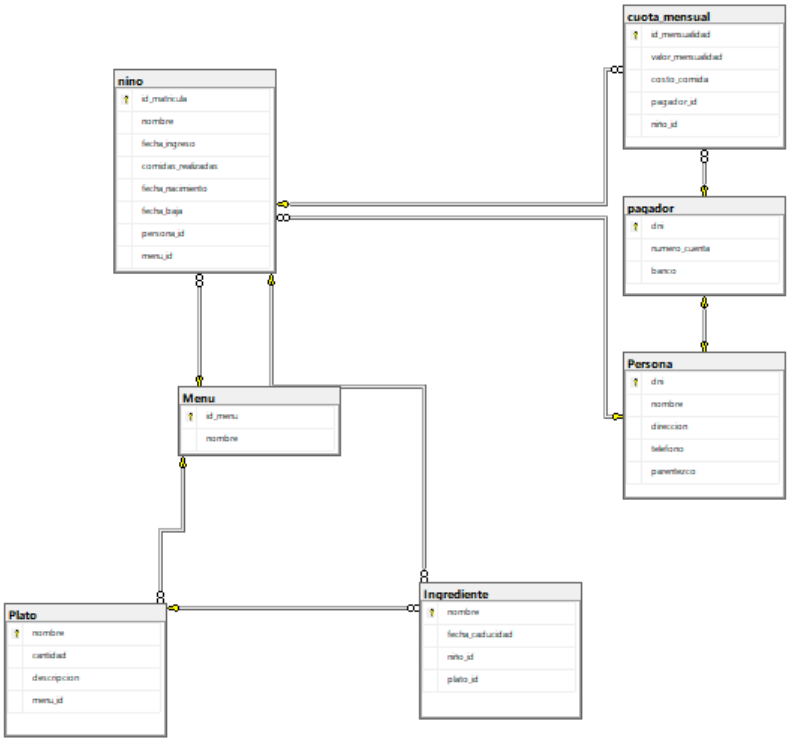
```
go

CREATE INDEX "IDX_CUOTA_NINO" ON CUOTA_MENSUAL(NINO_ID);
go
ALTER TABLE CUOTA_MENSUAL
ADD CONSTRAINT "FK_CUOTA_NINO"
FOREIGN KEY (NINO_ID)
REFERENCES NINO(ID_MATRICULA);
CREATE INDEX "IDX_NINO_PERSONA" ON NINO(PERSONA_ID) ;
go

CREATE INDEX "IDX_PLATO_MENU" ON PLATO(MENU_ID);
go
ALTER TABLE INGREDIENTE
ADD CONSTRAINT "FK_INGREDIENTE_PLATO"
FOREIGN KEY (PLATO_ID)
REFERENCES PLATO (NOMBRE);
go

CREATE INDEX "IDX_CUOTA_PAGADOR" ON CUOTA_MENSUAL (PAGADOR_ID);
go
ALTER TABLE CUOTA_MENSUAL
ADD CONSTRAINT "FK_CUOTA_PAGADOR"
FOREIGN KEY (PAGADOR_ID)
REFERENCES PAGADOR(DNI);
go

ALTER TABLE PAGADOR
ADD CONSTRAINT "FK_PAGADOR_PERSONA"
FOREIGN KEY (DNI)
REFERENCES PERSONA (DNI);
go
```

Luego de crear todos los comandos y ejecutar todas las instrucciones, en los respectivos clientes para administración de base de datos se ven de la siguiente manera:

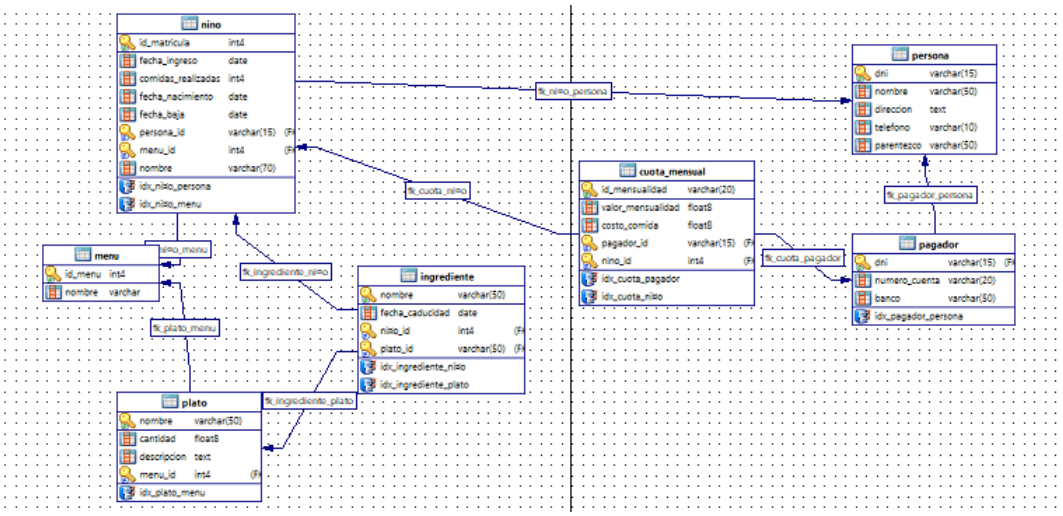
MySQL Workbench



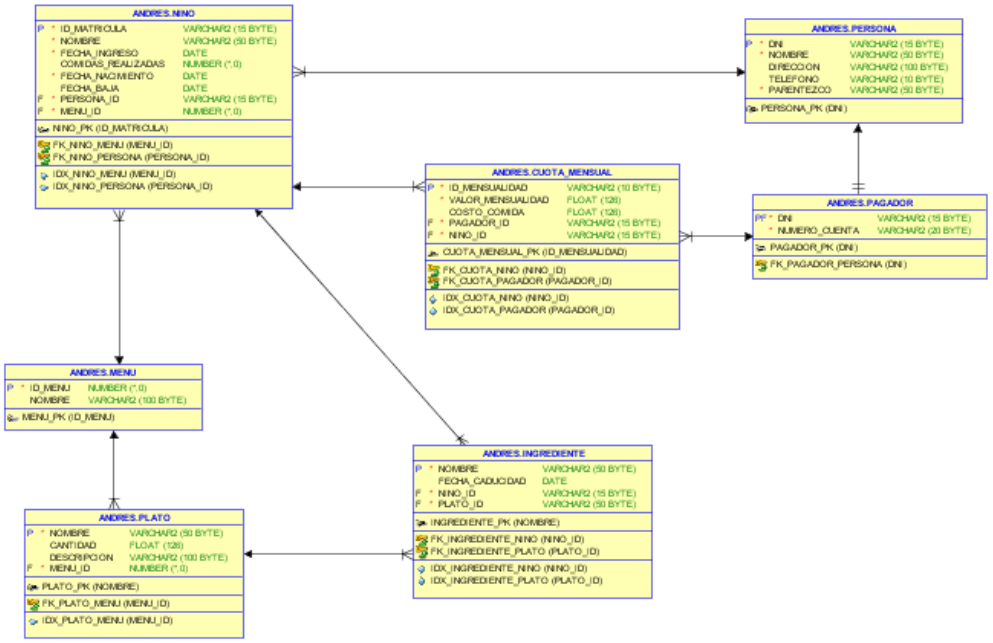
- **Microsoft SQL Management Studio**



- **MicroOLAP Database Designer for POSTGRESQL**



- **Sqldeveloper**



- **SENTENCIAS DML**

Las sentencias DML son aquellas utilizadas para insertar, borrar, modificar y consultar los datos de una base de datos. Las que mencionaré serán ***INSERT, UPDATE, DELETE y SELECT***

- **INSERT**

La instrucción INSERT de SQL permite añadir registros a una tabla. Con ella podemos ir añadiendo registros uno a uno, o añadir registros masivamente.

❖ **INSERT TOTAL**

Permite insertar registros sin dejar ninguno vacío, es decir, si o si hay que ingresar todos los registros correspondiente a cada columna.

Ejemplo:

- **MYSQL**

```
-- INSERT TOTAL
-- INSERTAR NIÑOS
INSERT INTO nino VALUES
(223, 'Marcos', '2020-01-27', 3, NULL, '1007975579',1),
(243, 'Ana', '2020-10-03', 2, '2021-03-23', '111',2);

-- INSERTAR PLATO
INSERT INTO plato
VALUES ('Nuggets', 5, 'asdasdas', 1);
```

- **POSTGRESQL**

```
-- INSERT TOTAL
-- INSERTAR NIÑOS
INSERT INTO nino VALUES
(223,'2020-01-27', 3, '2015-05-21', NULL, '1007975579',1,
'Marcos'),
(243,'2020-10-03', 2, '2012-01-15', '2021-03-23', '111',2, 'Ana');

-- INSERTAR PLATO
INSERT INTO plato
VALUES ('Nuggets', 5, 'asdasdas', 1);
```

- **SQL SERVER**

```
-- INSERT TOTAL
-- INSERTAR NIÑOS
INSERT INTO nino VALUES
('223', 'Marcos', '2020-01-27', 3, '2015-05-21', NULL,
'1007975579',1),
('243','Ana','2020-10-03', 2, '2012-01-15', '2021-03-23',
'111',2);
go
```

- **ORACLE SQL**

```
-- INSERT TOTAL
INSERT INTO persona
VALUES ('223344','Raul',NULL,'4655','fasgf');
```

❖ **INSERT PARCIAL**

Permite ingresar registros a la tabla teniendo en cuenta las columnas que especifica el usuario.

Ejemplos:

- **MYSQL**

```
-- INSERT PARCIAL
-- INSERTAR PLATOS
INSERT INTO plato (nombre, cantidad, menu_id)
VALUES ('paella', 1, 1),
('bandeja paisa', 1, 2),
('Ceviche', 1, 1),
('sopa', 1, 2);
```

- **POSTGRESQL**

```
-- INSERT PARCIAL
-- INSERTAR PLATOS
INSERT INTO plato (nombre, cantidad, menu_id)
VALUES ('paella', 1, 1),
('bandeja paisa', 1, 2),
('Ceviche', 1, 1),
('sopa', 1, 2);
```

- **ORACLE SQL**

```
-- INSERT PARCIAL
INSERT INTO persona(dni, nombre, telefono, parentezco)
VALUES('4','jorge','5553','padre');

INSERT INTO persona(dni, nombre, telefono, parentezco)
VALUES('55','antonio','5552','padre');
```

- **SQL SERVER**

```
-- INSERT PARCIAL
-- INSERTAR PLATOS
INSERT INTO plato (nombre, cantidad, menu_id)
VALUES ('paella', 1, 1),
('bandeja paisa', 1, 2),
('Ceviche', 1, 1),
('sopa', 1, 2);
go
-- INSERTAR PAGADORES
INSERT INTO pagador (dni, numero_cuenta, banco) VALUES ('111',
'111', 'Bancolombia'),
('12321', '00002222', 'Bancolombia');
go
```

- **INSERT MASIVO**

Permite insertar muchos registros a una tabla en una sola instrucción.

Ejemplos:

- **MYSQL**

```
-- INSERT MASIVO
INSERT INTO `personas` VALUES
('1007975579','Andres Molinares',NULL,'3015109237','Hermano'),
('111','John',NULL,'6558741','asd'),
('12321','Marco','Cartagena','22556644','tio'),
('12337','felipe','bocagrande','555','padre'),
('4546','Antonio','sanfernando','444','tio'),
('555','salma','getsemani','999','madre'),
('56564','alberto','libano','777','hermano'),
('744','gabriela','alameda','888','prima'),
('877977','juan','pozon','666','abuelo');
```

- **POSTGRESQL**

```
-- INSERT MASIVO
INSERT INTO persona VALUES
('1007975579','Andres Molinares',NULL,'3015109237','Hermano'),
('111','John',NULL,'6558741','asd'),
('12321','Marco','Cartagena','22556644','tio'),
('12337','felipe','bocagrande','555','padre'),
('4546','Antonio','sanfernando','444','tio'),
('555','salma','getsemani','999','madre'),
('56564','alberto','libano','777','hermano'),
('744','gabriela','alameda','888','prima'),
('877977','juan','pozon','666','abuelo');
```

```
INSERT INTO menu VALUES (2, 'menu2'), (1, 'menu1');
```

- **ORACLE SQL**

```
INSERT ALL
INTO pagador (dni, numero_cuenta, banco) VALUES ('55', '259844',
'BBVA')
INTO pagador (dni, numero_cuenta, banco) VALUES ('4', '4234',
'BBVA')
SELECT * FROM pagador;
```

- **UPDATE**

Update es la instrucción del lenguaje SQL que nos sirve para modificar los registros de una tabla

Ejemplos:

- **MySQL**

```
-- UPDATE
-- ACTUALIZA AQUELLAS PERSONAS QUE TENGAN NULL EN LA DIRECCIÓN
-- Y LA CAMBIA POR 'CARTAGENA'
UPDATE persona
SET direccion = 'Cartagena'
WHERE direccion IS NULL;

-- Actualiza la tabla personas para colocar sus nombres en
mayúscula
UPDATE persona
```

```
SET nombre = upper(nombre);
```

- **POSTGRESQL**

```
-- UPDATE
UPDATE persona
SET direccion = 'Cartagena'
WHERE direccion IS NULL;

-- Actualiza la tabla personas para colocar sus nombres en
mayúscula
UPDATE persona
SET nombre = upper(nombre);
```

- **ORACLE**

```
-- UPDATE
UPDATE persona
SET direccion = 'Cartagena'
WHERE direccion IS NULL;

-- Actualiza la tabla personas para colocar sus nombres en
mayuscula
UPDATE persona
SET nombre = upper(nombre);

-- ACTUALIZAR MENU PARA AGREGAR NOMBRE
UPDATE menu SET nombre = 'menu1' WHERE id_menu=1;
```

- **SQL SERVER**

```
UPDATE persona
SET direccion = 'Cartagena'
WHERE direccion IS NULL;
go

-- Actualiza la tabla personas para colocar sus nombres en
mayuscula (HACER)
UPDATE persona
SET nombre = upper(nombre);
go
```

- **DELETE**

La instrucción DELETE permite eliminar uno o múltiples registros. Incluso todos los registros de una tabla, dejándola vacía.

Ejemplos:

Como tal, el proceso es el mismo para todos los motores de bases de datos, aquí un ejemplo con cada uno de ellos.

- **MySQL**

```
-- DELETE
-- Elimina todos los registros de la tabla niño
DELETE FROM niño WHERE id_matricula IS NOT NULL;

-- Elimina aquellos niños que se retiraron de la guarderia
DELETE FROM niño
WHERE fecha_baja IS NOT NULL;
```

- **POSTGRESQL**

```
-- DELETE
-- Elimina todos los registros de la tabla niño
DELETE FROM nino WHERE id_matricula IS NOT NULL;

-- Elimina aquellos niños que se retiraron de la guarderia
DELETE FROM nino
WHERE fecha_baja IS NOT NULL;
```

- **ORACLE SQL**

```
-- DELETE
-- Elimina todos los registros de la tabla niño
DELETE FROM niño WHERE id_matricula IS NOT NULL;

-- Elimina aquellos niños que se retiraron de la guarderia
DELETE FROM niño
WHERE fecha_baja IS NOT NULL;
```

- **SQL SERVER**

```
-- DELETE
-- Elimina todos los registros de la tabla niño
DELETE FROM niño WHERE id_matricula IS NOT NULL;
go
-- Elimina aquellos niños que se retiraron de la guarderia
```

```
DELETE FROM niño
WHERE fecha_baja IS NOT NULL;
go
```

- **SELECT**

Una sentencia SELECT devuelve una tabla al cliente que coincide con la consulta. Se trata de una tabla en el sentido de que los resultados tienen la forma de filas y columnas. La sentencia SELECT es la base para realizar consultas en cualquier base de datos de lenguaje de consulta estructurado (SQL).

Ejemplo:
Se desea saber cuáles niños se retiraron de la guardería.

- **MySQL**

```
-- NIÑOS QUE SALIERON DE LA GUARDERIA
SELECT * FROM ninos
WHERE fecha_baja IS NOT NULL;
```

Resultado:

id	nombre	fecha_ingreso	comidas_realizadas	fecha_baja	persona_id	menu_id
2	ANDRES FELIPE M	2021-04-23	6	2021-04-23	11	1
4	Myke	2021-04-21	2	2021-04-22	1007975579	2
5	juan	2021-04-24	4	2021-04-20	12	2

- **POSTGRE**

```
-- NIÑOS QUE SALIERON DE LA GUARDERIA
SELECT * FROM nino
WHERE fecha_baja IS NOT NULL;
```

Resultado:

	id_matricula [PK] integer	fecha_ingreso date	comidas_realizadas integer	fecha_nacimiento date	fecha_baja date	persona_id character varying (15)	menu_id integer	nombre character varying (70)
1	243	2020-10-03	2	2012-01-15	2021-03-23	111	2	Ana

- **ORACLE**

```
-- NIÑOS QUE SE RETIRARON DE LA GUARDERIA
SELECT * FROM nino
WHERE fecha_baja IS NOT NULL;
```

Resultado:

	ID_MATRICULA	NOMBRE	FECHA_INGRESO	COMIDAS_REALIZADAS	FECHA_NACIMIENTO	FECHA_BAJA	PERSONA_ID	MENU_ID
1	2704	Andres	08/06/15	2	10/01/13	15/04/20	12337	2
2	243	Ana	03/10/20	2	03/11/19	03/11/20	111	1

- **SQL SERVER**

```
-- NIÑOS QUE SALIERON DE LA GUARDERIA
SELECT * FROM nino
WHERE fecha_baja IS NOT NULL;
go
```

Resultado:

	id_matricula	nombre	fecha_ingreso	comidas_realizadas	fecha_nacimiento	fecha_baja	persona_id	menu_id
1	243	Ana	2020-10-03	2	2012-01-15	2021-03-23	111	2

Ejemplo2:
Se desea saber cuáles niños son alérgicos a algún ingrediente

- **MySQL**

```
-- NIÑOS QUE SON ALERGICOS A ALGUN INGREDIENTE
SELECT ingredientes.nombre AS Nombre_ingrediente, ninos.nombre AS
NOMBRE_NIÑO
FROM ingredientes, ninos WHERE ninos.id = ingredientes.niño_id;
```

Resultado:

	Nombre_ingrediente	NOMBRE_NIÑO
▶	Ajo	Ana
	Cebolla morada	ANDRES FELIPE M
	mostaza	Myke

- **PostgreSQL**

```
-- NIÑOS QUE SON ALERGICOS A ALGUN INGREDIENTE
SELECT ingrediente.nombre AS Nombre_ingrediente, nino.nombre AS
NOMBRE_NIÑO
FROM ingrediente, nino WHERE nino.id_matricula =
ingrediente.nino_id;
```

Resultado:

	nombre_ingrediente character varying (50)	nombre_niÑo character varying (70)
1	Ajo	Marcos
2	mostaza	Ana
3	pollo	Andres

- **ORACLE SQL**

```
-- NIÑOS QUE SON ALERGICOS A ALGUN INGREDIENTE
SELECT ingrediente.nombre AS Nombre_ingrediente, nino.nombre AS
NOMBRE_NIÑO
FROM ingrediente, nino WHERE nino.id_matricula =
ingrediente.nino_id;
```

Resultados:

	NOMBRE_INGREDIENTE	NOMBRE_NIÑO
1	Ajo	Marcos
2	mostaza	Ana
3	pollo	Andres

- **SQL SERVER**

```
-- NIÑOS QUE SON ALERGICOS A ALGUN INGREDIENTE
SELECT ingrediente.nombre AS Nombre_ingrediente, nino.nombre AS
NOMBRE_NIÑO
FROM ingrediente, nino WHERE nino.id_matricula =
ingrediente.niño_id;
go
```

Resultado:

	Nombre_ingrediente	NOMBRE_NIÑO
1	Ajo	Marcos
2	mostaza	Ana
3	pollo	Andres

Ejemplo3:
Se desea saber el total de la mensualidad a pagar por cada niño, el cual se calcula costo por comidas por las comidas realizadas de cada niño más el valor de la mensualidad.

-MySQL

```
-- Mostrar MENSUALIDAD TOTAL

SELECT ninos.nombre AS Nombre_niño, ninos.comidas_realizadas,
cuota_mensuales.valor_mensualidad AS cargo_mensual,
cuota_mensuales.costo_comida, (cuota_mensuales.costo_comida *
ninos.comidas_realizadas) as total_comidas ,
(cuota_mensuales.costo_comida * ninos.comidas_realizadas) +
cuota_mensuales.valor_mensualidad AS total_mensualidad FROM ninos
JOIN cuota_mensuales ON ninos.id = cuota_mensuales.niño_id;
```

Resultado:

	Nombre_niño	comidas_realizadas	cargo_mensual	costo_comida	total_comidas	total_mensualidad
▶	Ana	4	30000.00	1000.00	4000.00	34000.00
	ANDRES FELIPE M	6	100000.00	20000.00	120000.00	220000.00

- **POSTGRESQL**

```
-- MOSTRAR TOTAL MENSUALIDADES
SELECT nino.nombre AS Nombre_niño, nino.comidas_realizadas,
cuota_mensual.valor_mensualidad AS cargo_mensual,
cuota_mensual.costo_comida,
(cuota_mensual.costo_comida * nino.comidas_realizadas) as
total_comidas,
(cuota_mensual.costo_comida * nino.comidas_realizadas) +
cuota_mensual.valor_mensualidad AS total_mensualidad
FROM nino JOIN cuota_mensual ON nino.id_matricula =
cuota_mensual.nino_id;
```

Resultado:

	nombre_niño character varying (70)	comidas_realizadas integer	cargo_mensual double precision	costo_comida double precision	total_comidas double precision	total_mensualidad double precision
1	Marcos	3	300000	1000	3000	303000
2	Ana	2	100000	20000	40000	140000

- **ORACLE SQL**

```
-- MOSTRAR TOTAL MENSUALIDADES
SELECT nino.nombre AS Nombre_niño, nino.comidas_realizadas,
cuota_mensual.valor_mensualidad AS cargo_mensual,
cuota_mensual.costo_comida,
(cuota_mensual.costo_comida * nino.comidas_realizadas) as
total_comidas,
(cuota_mensual.costo_comida * nino.comidas_realizadas) +
cuota_mensual.valor_mensualidad AS total_mensualidad
FROM nino JOIN cuota_mensual ON nino.id_matricula =
cuota_mensual.nino_id;
```

Resultado:

	NOMBRE_...	COMIDAS_REALIZADAS	CARGO_MENSUAL	COSTO_COMIDA	TOTAL_COMIDAS	TOTAL_MENSUALIDAD
1	Ana	2	100000	20000	40000	140000
2	Marcos	3	300000	1000	3000	303000

- **SQL SERVER**

```
-- MOSTRAR TOTAL MENSUALIDADES
SELECT nino.nombre AS Nombre_niño, nino.comidas_realizadas,
cuota_mensual.valor_mensualidad AS cargo_mensual,
cuota_mensual.costo_comida,
(cuota_mensual.costo_comida * nino.comidas_realizadas) as
total_comidas,
(cuota_mensual.costo_comida * nino.comidas_realizadas) +
cuota_mensual.valor_mensualidad AS total_mensualidad
FROM nino JOIN cuota_mensual ON nino.id_matricula =
cuota_mensual.niño_id;
go
```

Resultado:

	Nombre_niño	comidas_realizadas	cargo_mensual	costo_comida	total_comidas	total_mensualidad
1	Ana	2	100000	20000	40000	140000
2	Marcos	3	300000	1000	3000	303000

Ejemplo4:
Se desea saber el número de platos que tiene cada menú.

- **MySQL**

```
-- Numero de platos que tiene cada menú
SELECT menus.nombre, COUNT(platos.menu_id) AS total_platos FROM
platos
JOIN menus ON platos.menu_id = menus.id
group by platos.menu_id;
```

Resultado:

	nombre	total_platos
▶	Menu 1	4
	menu 2	3

- **PostgreSQL**

```
-- Numero de platos que tiene cada menú
SELECT menu.nombre, COUNT(plato.menu_id) AS total_platos FROM
plato
JOIN menu ON plato.menu_id = menu.id_menu
GROUP BY  menu.nombre,plato.menu_id;
```

Resultado:

	nombre character varying	total_platos bigint
1	menu1	3
2	menu2	2

- **ORACLE SQL**

```
-- Numero de platos que tiene cada menú
SELECT menu.nombre, COUNT(plato.menu_id) AS total_platos FROM
plato
JOIN menu ON plato.menu_id = menu.id_menu
GROUP BY  menu.nombre,plato.menu_id;
```

Resultado:

	NOMBRE	TOTAL_PLATOS
1	menu1	3
2	menu2	2

- **SQL SERVER**

```
-- Numero de platos que tiene cada menú
SELECT menu.nombre, COUNT(plato.menu_id) AS total_platos FROM
plato
JOIN menu ON plato.menu_id = menu.id_menu
GROUP BY  menu.nombre,plato.menu_id;
go
```

Resultado:

	nombre	total_platos
1	menu1	2
2	menu2	2

Ejemplo5:
Se desea saber cuántos niños prefieren y consumen cada menú.

- **MySQL**

```
-- Menus favoritos
SELECT COUNT(ninos.id) AS cantidad_niños, menus.nombre AS
nombre_menu FROM ninos
JOIN menus ON ninos.menu_id = menus.id
GROUP BY ninos.menu_id;
```

Resultado:

	cantidad_niños	nombre_menu
▶	3	Menu 1
	2	menu 2

- **POSTGRESQL**

```
-- Menus favoritos
SELECT COUNT(nino.id_matricula) AS cantidad_niños, menu.nombre AS
nombre_menu FROM nino
JOIN menu ON nino.menu_id = menu.id_menu
GROUP BY nino.menu_id, menu.nombre;
```

Resultado:

	cantidad_niños bigint	nombre_menu character varying
1	3	menu2
2	1	menu1

- **ORACLE SQL**

```
-- Menus favoritos
SELECT COUNT(nino.id_matricula) AS cantidad_niños, menu.nombre AS
nombre_menu FROM nino
JOIN menu ON nino.menu_id = menu.id_menu
GROUP BY nino.menu_id, menu.nombre;
```

Resultado:

	CANTIDAD_NIÑOS	NOMBRE_MENU
1	2	menu1
2	2	menu2

- **SQL SERVER**

```
-- Menus favoritos
SELECT COUNT(nino.id_matricula) AS cantidad_niños, menu.nombre AS
nombre_menu FROM nino
JOIN menu ON nino.menu_id = menu.id_menu
GROUP BY nino.menu_id, menu.nombre;
go
```

Resultado:

	cantidad_niños	nombre_menu
1	1	menu1
2	3	menu2

BIBLIOGRAFÍAS

- https://www.esic.edu/rethink/tecnologia/modelo-entidad-relacion-descripcion-aplicaciones
- https://www.danysoft.com/estaticos/free/dei02.pdf
- www.campusmvp.es/recursos/post/Fundamentos-de-SQL-Insercion-de-datos-INSERT.aspx
- https://desarrolloweb.com/articulos/266.php