PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA INFORMÁTICA

Bases de datos

GONZALO TELLO VALENZUELA

Docente Base de Datos: Delia Ibacache
Docente Taller de Base de datos: Sebastian Berrios

2do semestre, 2017
1er Semestre, 2020

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Base de Datos

1.1. Datos v/s Información

- **Datos:** Par ordenado <nombre, valor>. Representa una característica sobre algún concepto o suceso.
- **Información:** Resultado de combinar, comparar y realizar cálculos (procesamientos) sobre los datos. Son datos que han sido organizados o preparados en una forma adecuada para apoyar la toma de decisiones, por lo tanto son valorados por la organización y tienen significado.

La unión de datos bajo una determinada selección también le denominamos objetos. Si nos vemos a nosotros como objetos será:

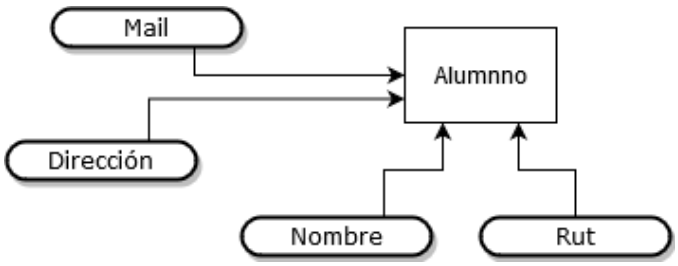


Figura 1.1: Ejemplo de un dato

Estableciendo entre ramos como objetos, tenemos

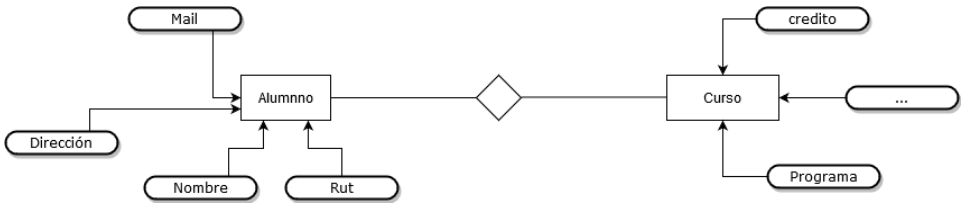


Figura 1.2: Ejemplo asociación entre datos

Dato histórico(no operacional)

- Datos que están almacenados y que son inmutables, es decir, son consultables pero no son modificables.

  BASES DE DATOS $\Longrightarrow$ INFORMACIÓN

1.1.1. Enfoque tradiciona de Bases de datos

- La necesidad de las grandes empresas de almacenar cantidades crecientes de información de una forma rápida, sencilla, fiable, a la que se pueda acceder en cualquier momento, trajo consigo la creación de mecanismos para guardar datos.
- El primer enfoque utilizado para almacenar datos, se conoce como enfoque tradicional de procesamiento de datos.
- Los sistemas de archivos surgieron a raíz de la necesidad de almacenamiento de la información para su correspondiente reutilización (persistencia)
- En el enfoque tradicional del procesamiento de datos,**una colección de programas de aplicación realiza diversos servicios para los usuarios finales, como por ejemplo la producción de informes. Cada programa define y gestiona sus propios datos.**

1.1.1.1. Desventajas del enfoque tradicional de procesamiento de datos

- **Inflexibilidad**
  - Dificultad del enfoque para enfrentar cambios, debido a que no es sencillo adaptarse a nuevos requerimientos que no hayan sido considerados en el diseño inicial.

- Se genera una evolución lenta de información. Los usuarios que existen tienen datos, sin embargo, no pueden obtener la información que necesitan.

- **Redundancia no aplicada**
  Cuando cada aplicación tiene sus propios archivos se produce una redundancia que genera:

  - Perdida de tiempo
  - Tener que repetir las acciones, por ejemplo, al actualizar el mismo dato en cada archivo.
  - Inconsistencia si las acciones de actualización no son coordinadas para cada archivo donde se encuentren los mismos datos.

- **Inconsistencia de datos**

  - Es la fuente más común de errores en las aplicaciones, generando documentos y reportes poco precisos y contradictorios, lo que disminuye la confianza del usuario en la integración del sistema de información.
  - Se produce debido a la descoordinación con la que realizan operaciones de ingreso, actualización o eliminación en archivos que presentan información redundante.

- **Escasa posibilidad de compartir datos**
  Al tener cada aplicativo sus propios archivos, existe poca oportunidad para los usuarios de compartir datos. Esto atrae las siguientes consecuencias:

  - Un mismo dato debe ser ingresado varias veces para actualizar los archivos.
  - Al realizar nuevos aplicativos puede que no sea posible explotar los datos, teniendo que crearse nuevos archivos y aumentando la redundancia.

- **Baja productividad del programador**

  - El programador debe diseñar cada archivo usado y luego codificar las definiciones en el aplicativo.
  - Se deben escribir las instrucciones de I/O requeridas.

- **Excesiva mantención**

- Cualquier modificación de un archivo requiere de la identificación de los programas donde será usado.
- El 80 % del esfuerzo del programador es ocupado en esta tarea

- **Estandarización deficiente**
  Las características de los datos pueden ser distintas en cada aplicación, por ejemplo, es posible encontrar las siguientes inconsistencias:

  - **Sinónimos**: Uno de nombres diferentes para un mismo item. *'ID', 'código', 'N_ requisito'*, son usados para el campo de un artículo.
  - **Homónicos**: El mismo nombre para items de datos distintos, por ejemplo, nombre se usa para clientes, proveedor y producto.

Todas estas limitaciones de los sistemas basados en archivos pueden atribuirse a dos factores.

- La definición de los datos está incluida en los programas de aplicación, en lugar de almacenarse de forma separada e independiente.
- No existe ningún control sobre el acceso y manipulación de datos, más allá que imponen los propios programas de aplicación.

1.1.2. Sistemas de bases de datos

- **¿Por qué surgieron los sistemas de bases de datos?**
  Necesidad de solucionar las debilidades de los sistemas de archivos.
- **Capacidades**

  - Manejo de persistencia
  - Soporte por lo menos de un modelo de datos
  - Soporte de un lenguaje de alto nivel que permita manipular y definir la estructura de la información.
  - Control de acceso
  - Evitar inconsistencia al compartir la información.

1.1.3. Sistema de gestión de base de datos

- Un sistema de gestión de base de datos (SGBD o Database management system o DBMS) es un software que gestiona y controla bases de datos.

- sus principales funciones son facilitar la utilización de la base de datos a muchos usuarios simultáneos y de tipos diferentes, independizar al usuario del mundo físico y mantener la integridad de los datos

1.1.4. Bases de datos

Una base ded datos es un conjunto estructurado y compartir de datos lógicamente relacionados y almacenados en un sistema computacional, diseñados para satisfacer las necesidades de información de una organización.

- Provee facilidades para recuperar, insertar, modificar y eliminar los datos cuando se requiera.
- Provee facilidades para transformar los datos recuperados en información útil.

1.1.4.1. Actores en la base de datos

- Administradores de la base da datos
  - Responsable de la administración de los recursos
  - Autoriza el acceso a la base de datos
  - Coordina y vigila la utilización de la base de datos
  - Adquiere los recursos de software y hardware que sean necesarios.
  - Es el responsable cuando surgen problemas como violaciones de la seguridad o una respuesta lenta del sistema.
- Usuarios
  - Usuario:
    Requieren acceder a la Base de datos para consultarla, actualizarla y generar informes
  - Usuarios finales ocasionales:
    Acceden en forma esporádica. Sus requerimientos de información pueden variar en cada ocasión.
- Desarrolladores
  - Analista de sistema:
    Determinan los requerimientos de los usuarios finales. Desarrollan especificaciones para transacciones programadas que satisfagan dichos requerimientos.

- Programadores de sistema:
  Implementan las especificaciones en forma de programas. Prueban, depuran, documentan y mantienen las transacciones programadas.
- Diseñador de base de datos:
  Identifican los datos que se almacenan en la base de datos, y eligen las estructuras apropiadas para representar y almacenar dichos datos. Tienen la responsabilidad de comunicarse con todos los futuros usuarios de la base de datos con el fín de comprender sus necesidades, y de representar un diseño que satisfaga esos requerimientos.

1.1.5. Construcción y modelado

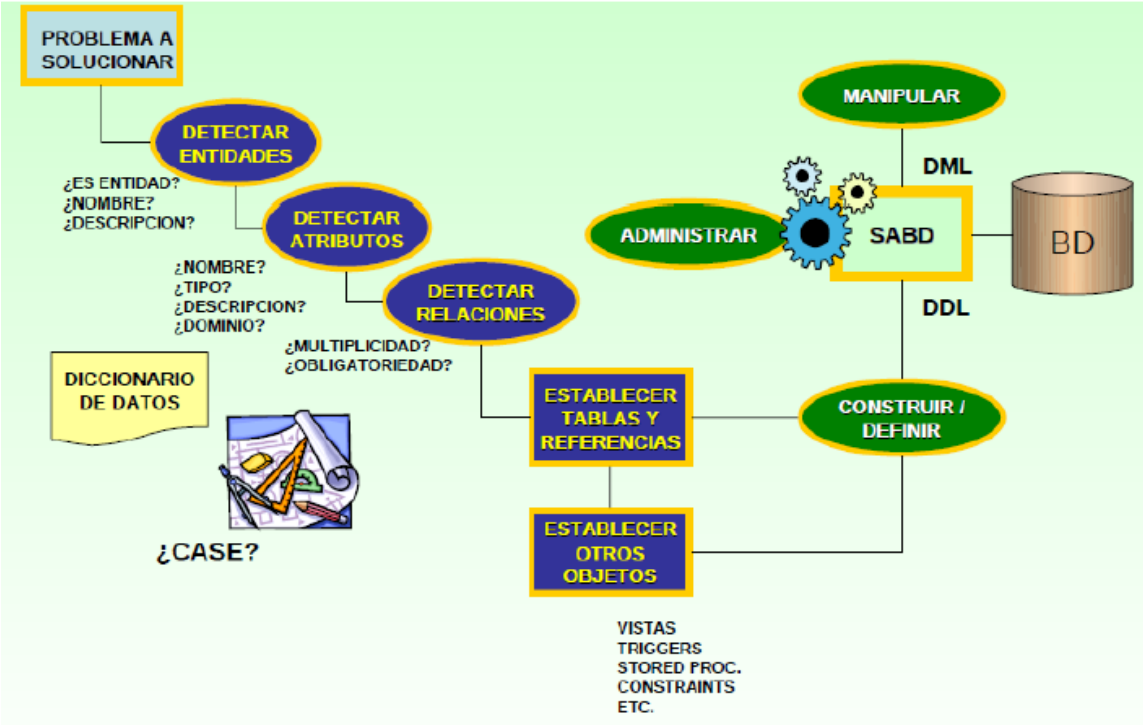


Figura 1.3: Construcción de bases de datos

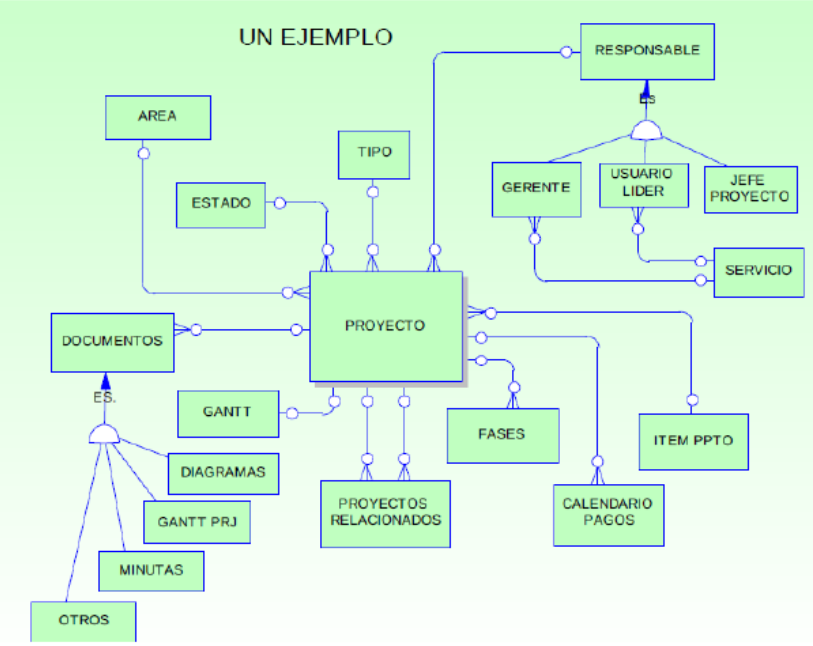


Figura 1.4: Modelar los datos conceptualmente (MER)

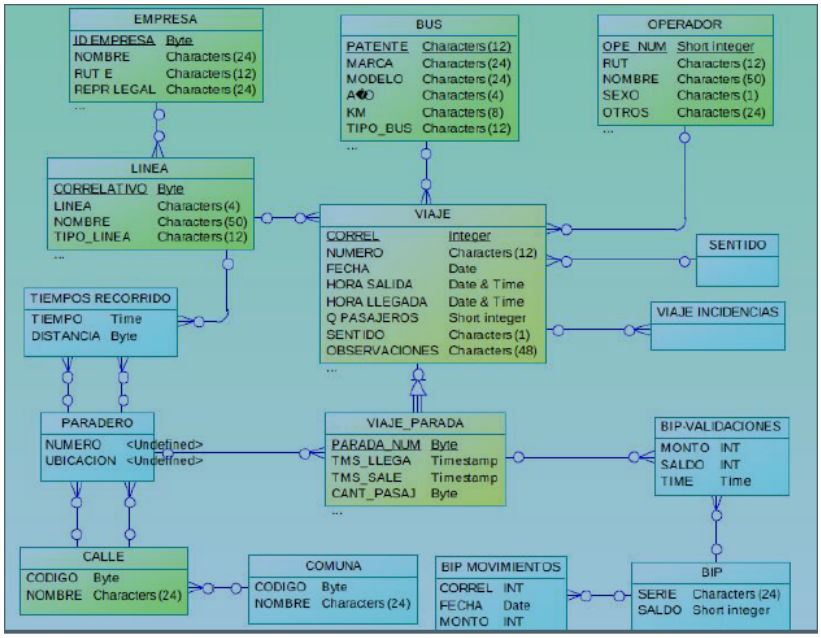


Figura 1.5: Modelar los datos lógicamente (Relacional)

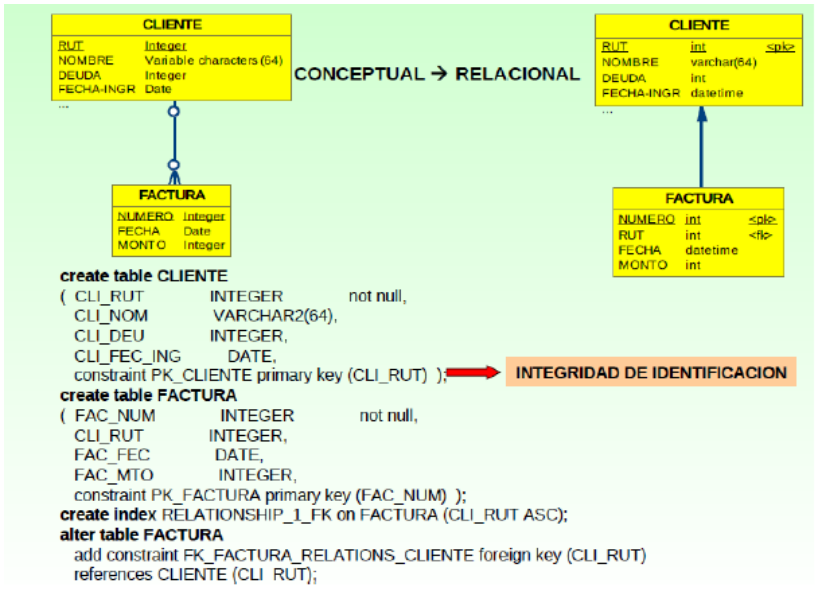


Figura 1.6: Implementación base de datos

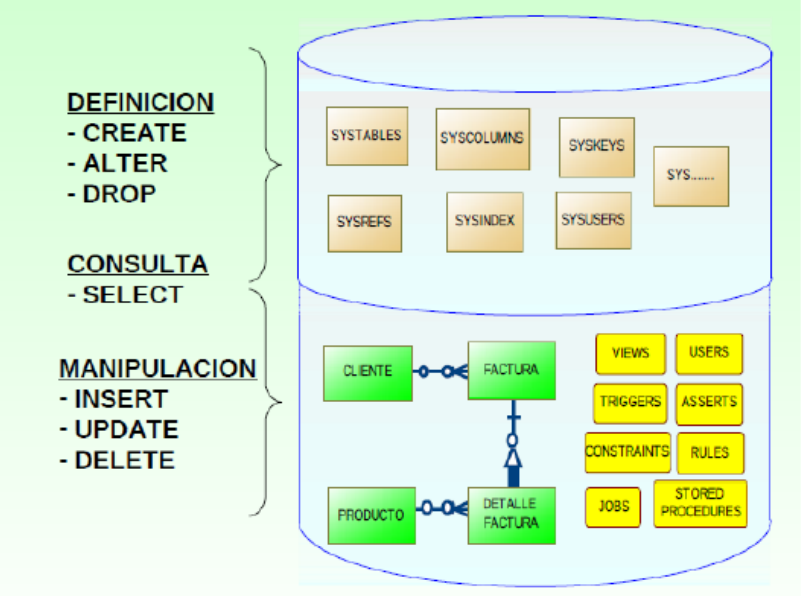


Figura 1.7: Operación de la base de datos

1.1.6. Ventajas y Desventajas de un Sistema de Gestión de Base de datos

1.1.6.1. Ventajas de un SGBD

- **Integridad**
  Validez y coherencia de los datos almacenados. La integridad se suele expresar en términos de **restricciones**, que son reglas de coherencia que no se permite que la base de datos viole.

- **Seguridad**
  Protección de los datos frente a su uso por personas no autorizadas.

- **Rapidez de desarrollo**
  El SGDB proporciona muchas de las funciones estándar que el programador tendría normalmente que incluir dentro de su aplicación basada en archivos.

- **Mantenimiento y reingeniería**
  Cambios en la estructura de datos sin cambiar los programas que los usan.

1.1.6.2. Desventajas de un SGBD

- **Tamaño**
  La complejidad y la cantidad de funcionalidades que entrega el SGBD hacen que el sistema ocupe gran espacio.
- **Susceptibilidad a fallas**
  El fallo de ciertos componentes puede provocar que se detengan varias operaciones
- **Complejidad**
  Si no se conoce adecuadamente el sistema pueden tomarse decisiones erróneas.

1.1.7. Arquitectura de los SGBD

1.1.7.1. Arquitectura ANSI/SPARC

Los SGBD necesitan que les demos una descripción o definición de la BD. Esta descripción reibe el nombre de esquema de la BD, y los SGBD (DBMS) la tendrán continuamente a su alcance en tres niveles:

- La forma en que los usuarios perciben los datos se denomina nivel externo.
- La forma en que el SGBD y el sistema operativo perciben los datos es el nivel interno.
- El nivel conceptual proporciona tanto la correspondencia como la necesaria independencia entre los niveles externo e interno.

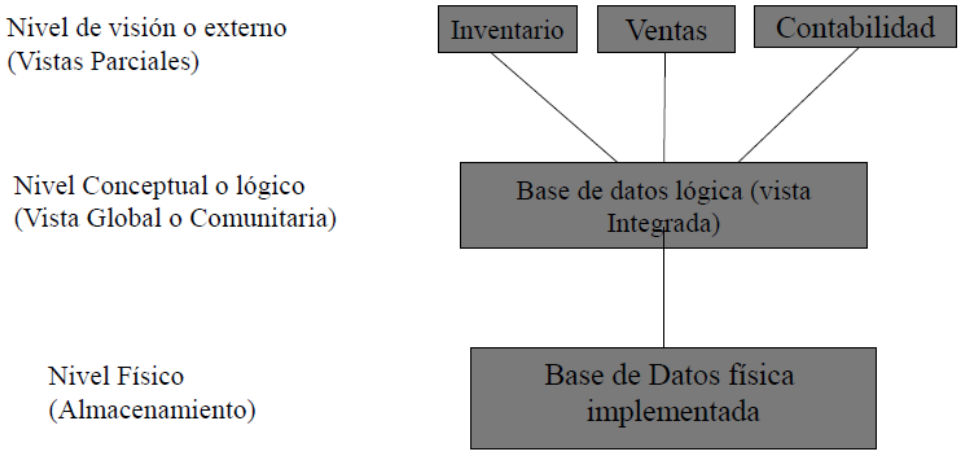


Figura 1.8: Ejemplo Arquitectura ANSI

1.1.7.2. Niveles de abstracción e independencia de los datos

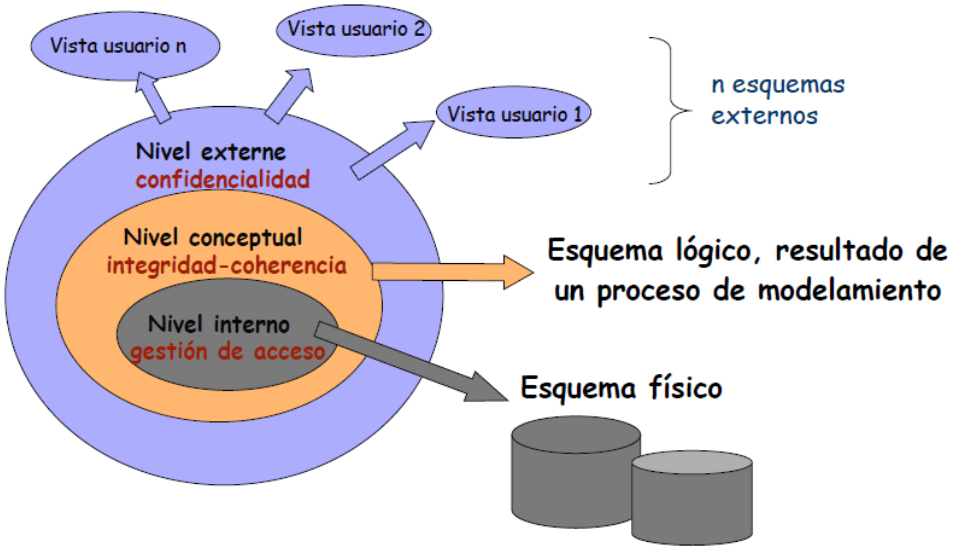


Figura 1.9: Nivel de abstracción de los datos

1.1.8. SQL: Structured Query Languaje

- Permite la definición y manipulación de los datos de una base de datos relacional.

- Implementado por 1° vez en un prototipo de IBM llamado System R.
- Se ha convertido en el lenguaje estándar de los sistema de gestión de base de datos
- Es un lenguaje declarativo.
- Se puede utilizar de manera interactiva en asociación con interfaces gráficas o lenguajes de programación (C#, Java, PHP).
- Permite iterrogar una Base de datos relacional sin preocuparse por la representación física de los datos.
- Conjunto de instrucciones que permiten (Para Data Definition Language)
    - Definir, modificar y eliminar esquemas de relaciones
    - Crear índices
    - Definir vistas
    - Especificar restricciones de integridad
    - Ejemplo:

      CREATE; ALTER; DROP
- Conjunto de instrucciones que permiten (Para Data Manipulation Language)
    - Consultar, actualizar y eliminar los elementos creados con el DDL, como tablas, índices, etc.
    - Está basado en álgebra y cálculo relacional
    - Ejemplo:

      SELECT; INSERT; UPDATE; DELETE
- Data Control Languaje
    - Conjunto de instrucciones que permiten definir permisos de acceso a la base de datos.
    - Ejemplo

      GRANT; REVOKE

1.1.8.1. Lenguaje de definición de datos (DDL)

Un schema SQL es identificado por un nombre de schema, e incluye un identificador de autorización que indica el usuario dueño del schema. La instrucción CREATE SCHEMA es usada para crear un nuevo schema. Por ejemplo, la siguiente instrucción crea un schema llamado 'compañía', cuyo dueño es JPerez:

```
Create Schema compania Authorization jperez;
```

La instrucción CREATE TABLE es usada para especificar una nueva relación, dándole un nombre y especificando atributos y restricciones. A cada atributo se le da un nombre, un tipo de datos (para especificar su dominio de valores) y opcionalmente algunas restricciones.

SQL: Creación de tablas

Npil	ApPa	ApMa	RUT	FNac	Dir	Sexo	Sueld	RutS	NDep
Juan	Perez	Garcia	1234567-8	09/01/55	Toesca 968	M	560	333	5
Alicia	Zelaya	Roa	9998877-7	09/07/58	Blanco 2120	F	630	888	2
Juana	Baeza	Araya	9876543-2	20/06/88	Casa 2540	F	240	886	2
Fran	Cea	Daza	11375103-6	05/12/72	Condell 221	M	350	332	5
Jaime	Ramo	Salas	17896852-9	10/11/90	Ovalle 1015	M	760	987	4

Tabla 1.1: Tabla Empleados

```
Create Table empleado(
    Npil Varchar(15) Not Null,
    ApPa Varchar(15) Not Null,
    ApMa Varchar(15) Not Null,
    RUT Varchar(10) Not Null,
    Fnac Date,
    Dir Varchar(30) Not Null,
    Sexo Char,
    Sueld DecimalL(5,2),
    RutS Varchar(10),
    NDep Int Not Null,
    Primary Key(RUT),
    Foreign Key(RUTS) REFERENCES EMPLEADO(RUT),
    Foreign Key(NDep) REFERENCES DEPARTAMENTO(DNUM));

Create Table departamento(
    DNOMBRE Varchar(15) Not Null,
```

```
    DNUM Int Not Null,
    RUT_GER Varchar(10) Not Null,
    GERFECHAINI Date,
    Primary Key(DNUM),
    Unique(DNOMBRE),
    Foreign Key(RUT_GER) REFERENCES EMPLEADO(RUT));

Create Table ubicaciones_depto(
    DNUM Int Not Null,
    DUBICA Varchar(15) Not Null,
    Primary Key(DNUM, DUBICACION),
    Foreign Key(DNUM) REFERENCES DEPARTAMENTO(DNUM));

Create Table proyecto(
    PNOMBRE Varchar(15) Not Null,
    PNUMERO Int Not Null,
    PUBICA Varchar(15),
    DNUM  Int Not Null,
    Primary Key(PNUMERO),
    Unique(PNOMBRE),
    Foreign Key(DNUM) REFERENCES DEPARTAMENTO(DNUM));
```

La restricción UNIQUE identifica de manera única a cada fila de una tabla. También se puede agregar explícitamente el nombre del schema a cada tabla separado por un punto, por ejemplo:

```
Create table COMPANIA.EMPLEADO;
```

- COMPAÑIA es el nombre de la base de datos
- EMPLEADO es el nombre de la tabla

Los tipos de dato disponibles para los atributos: numérico, tira de cadenas, carácter, fecha y hora. Los tipos numéricos pueden incluir números enteros de varios tamaños (INT y SMALLINT), números reales de varias precisiones (FLOAT, REAL, DOUBLEPRECISION).

Se pueden declarar números con formato, usando DECIMAL(i,j).

- Los caracteres pueden ser de largo fijo (CHAR(n)) o de largo variable (VARCHAR(n), donde n es el máximo de números de caracteres
- La fecha tiene 10 posiciones, típicamente AAAA-MM-DD

- La hora tiene al menos 8 posiciones, típicamente HH:MM:SS.
- Solamente fechas y horas válidas son permitidas en las implementaciones de SQL.

Declarar Dominios.

Esto facilita hacer cambios en los tipos de datos (cambiando sólo el dominio y no cada dato declarado). Por ejemplo, podemos crear el dominio TIPO_RUT con la siguiente instrucción.

```
Create domain tipo_rut as varchar(10);
```

A partir de ahora, podemos usar TIPO_RUT en lugar de VARCHAR(10), por ejemplo en los atributos RUT, RUTS, RUTGER y ERUT.

Debido a que SQL permite el 'NULL' como valor de sus atributos, es necesario especificar las restricción 'NOT NULL' para los atributos que no permiten este valor (por violaciones de integridad). Esta restricción siempre debe ser especificada para los atributos que son llaves primarias en cada relación. Por ejemplo:

```
Create table asistente(
Rut Varchar(13) Not Null,
Cod_empleado Varchar(25) Not Null,
...
);
```

Es posible definir un valor por defecto para un atributo agregando a la clausula DEFAULT 'valor' en la definición del atributo. Por ejemplo:

```
Create table Producto(
    Precio Decimal(1,2) Default '0',
    ...
);
```

Las siguientes clausulas especifican:

- **PRIMARY KEY** especifica uno o más atributos que forman la llave primaria de la relación.
- **UNIQUE** es una claúsula que especifica llaves alternas.
- **FOREIGN KEY** especifica integridad de referencia.

Las restricciones de integridad referencial pueden ser violadas cuando:

- Las tuplas son insertadas o borradas.
- Se modifica una clave foránea.

CONSTRAINT coloca restricciones para limitar el tipo de dato que puede ingresar en una tabla.

Para borrar un schema completo, se usa la instrucción **DROP SCHEMA**, con dos opciones: **CASCADE** o **RESTRICT**. Por ejemplo, para borrar el schema de base de datos 'compañia' y todas sus tablas, dominios y otros elementos, se usa la opción **CASCADE**.

```
Drop Schema compania Cascade;
```

Una relación o tabla puede ser borrada del schema de la Base de datos usando la instrucción **Drop Tables**. Por ejemplo, si la relación 'carga' no va a ser utilizada más en la base de datos 'compañia', se puede borrar de la siguiente manera:

```
Drop Table carga Cascade;
```

Si la opción **RESTRICT** es usada en lugar de **CASCADE**, la tabla es borrada solamente si esta no es referenciada en ninguna restricción (por ejemplo, como clave foránea en otra tabla).

Modificar una tabla

La definición de una tabla puede ser modificada usando la instrucción **Alter Table**. Con esta instrucción es posible agregar o borrar atributos (columnas), cambiar la definición de una columna, y agregar o borrar restricciones. Por ejemplo, para agregar un atributo con el puesto de los empleados de la tabla empleado, se usa:

```
Alter Table compania.empleado Add puesto Varchar(12);
```

Para borrar una columna se puede usar **Cascade** o **Restrict**. Con CASCADE todas las restricciones son borrados automáticamente junto con la columna. Por ejemplo:

```
Alter Table compania.empleado Alter ndpto Drop Default;
Alter Table compania.empleado Alter ndpto Set Default '5';
```

Finalmente, se pueden borrar o agregar restricciones en una tabla. Para borrar una restricción esta debe tener un nombre (dado con **CONNSTRAINT**). Por ejemplo para borrar la restricción 'ndeptolf' de la tabla 'empleado':

```
Alter Table compania.empleado Drop Constraint ndptolf Cascade;
```

1.1.8.2. Lenguaje de manipulación de datos (DML)

Consultas en SQL.

SQL tiene una instrucción principal para recuperar información de una base de datos: el comando **SELECT**. Esta instrucción tiene muchas opciones. La forma básica de la instrucción **SELECT** es la siguiente:

```
Select <lista de atributos>
From <lista de tablas>
Where <condicion>;
```

- Extracción de datos: SELECT
  - **Select**
    - Corresponde a la operación de proyección.
    - Permite listar los atributos que se desean en el resultado de la consulta.
  - **From**
    - Especifica los nombres de la(s) tabla(s) de donde los datos serán almacenados.
    - Cuando son varias tablas las que aparecen en **From**, el sistema ejecuta una operación de producto cartesiado o de **Join**.
  - **Where**
    - Corresponde a la operación de la selección.
    - La condición se hace sobre los de las tablas del **From**.
  - La lista de atributos puede ser sustituida por un * para denotar todos los atributos.

- Extracción de datos: Operadores.

  Los operadores de comparación permitidos son:

  - = (igual)
  - ! = o <>(distinto)
  - > (mayor que)
  - < (menor que)

  También se pueden comparar cadenas de caracteres con operadores como >= y <=.

- Extracció de datos: conectores.

  Se permiten consultas tan complejas como sea necesario, usando conectores **AND, OR, NOT**.

```
Select *
From empleados
Where edad<28 And depto=1;
```

  Se pueden usar un operador especial denominado **BETWEEN**, para especificar un rango de valores sobre el cual puede variar el dominio de un atributo, ejemplo:

```
Select *
From empleados
Where edad Between 28 And 30;
```

  Se puede expresar una lista de valores de dominio específicos con el operador **IN**:

```
Select *
From empleados
Where nombre In ('Jorge Campos','Esteban Paz');
```

  El **WHERE** anterior equivale a:

```
nombre = 'Jorge Campos' Or nombre = 'Esteban Paz'
```

  IN y **BETWEEN** se pueden negar con **NOT**. **BETWEEN** puede lograrse por medio de <= **AND** >=

  En las condiciones se puede usar para realizar comparaciones especiales de cadenas de caracteres aparte de la igualdad (=) y el diferente (<>), así:

- El caracter % reemplaza cualquier cadena.
- El caracter _ reemplaza un caracter.

Ejemplo:

- Atributo **Like** 'ING %': Todo lo que comience por Ing.
- Atributo **Like** '%enieria': Todo lo que contenga por Ing.
- Atributo **Like** '_ _ _ %': Todo lo que tenga al menos 3 caracteres.

- Distinct vs All

  Se pueden eliminar tuplas duplicadas en una consulta colocando explicitamente **DISTINCT** después de **SELECT**.

  ```
  Select Distinct nombre_sucursal From prestamo;
  ```

  SQL no es cerrado relacionalmente, debido a que puede producir tuplas repetidas. Por lo tanto, se puede reescribir la consulta.

  Si explícitamente se quieren ver los duplicados, se coloca **All** después de **SELECT** (ALL es la opción por defecto).

- Join

  Una forma es comparando los atributos que realizan el **JOIN** en la claúsula **WHERE**.

  ```
  Select *
  From empleado, departamento
  Where empleado.depto = Departamento.depto;
  ```

  Se emplea la clausula **AS** para generar alias en las diferentes tablas

  ```
  Select *
  From empleado as e, departamento as d
  Where e.depto = d.depto;
  ```

- Anidamiento de consultas.

  Con **IN** y con otros operadores como **EXISTS**, se pueden anidar consultas. Por ejemplo, presentar el título de las películas que nunca se han prestado:

```
Select Distinct pel.titulo
From pelicula.pel
Where pel.titulo Not In(
    Select pel.titulo
    From pelicula As pel, prestamo As pr, copia As co
    Where pr.codcopia = co.codcopia And co.codpeli = pel.codpeli;
);
```

  Otra manera de realizar la anterior consulta, que evita la realización de uno de los **JOIN** es:

```
Select Distinct pel.titulo
From pelicula As pel
Where pel.codpeli Not In(
    Select co.codpeli
    From prestamo As pr, copia As co
    Where pr.codcopia = co.codcopia;
);
```

- Funciones de agregación:

  - **SUM(atributo)**: Sumatoria del atributo
  - **MAX(atributo)**: Valor máximo del atributo
  - **MIN(atributo)**: Valor mínimo del atributo
  - **AVG(atributo)**: Valor promedio del atributo
  - **COUNT(atributo | *)**: Conteo de tuplas

  Se puede usar **GROUP BY** con estas funciones para consolidar por grupos comunes. Se puede usar **HAVING** para establecer condiciones para los grupos (**HAVING** es lo que el **WHERE** es para las tuplas).

- Inserción de datos: INSERT

```
Insert Into departamento
Values (1, 'ADMINISTRACION');
```

- Borrado de datos: DELETE

```
Delete
From empleado
Where codigo=10;
```

- Actualización de datos: UPDATE

```
Update empleado
Set nombre = 'JUAN_CASTRO'
Where codigo=1;
```

1.2. Modelo de datos

1.2.1. Definición

Un modelo es una abstracción de la realidad, por ejemplo modelos de aviones, modelos matemáticos, maquetas, etc. Es una representación de objetos y sucesos del 'mundo real', así como de sus asociaciones, que se concentra en aspectos esenciales e inherentes de una organización e ignora las propiedades accesorias.

Un modelo de datos se define como una colección integrada de conceptos integrada de conceptos para describir y manipular datos, las relaciones existentes entre los mismos y las restricciones aplicables a los datos, todo ello dentro de una organización. Debe proporcionar los conceptos básicos y las notaciones que permitan a los diseñadores de la base de datos y a los usuarios finales comunicar de forma precisa y no ambigua su compresión de los datos.

Consolida las visiones de diferentes personas involucradas en la realidad a modelar. Permite chequear consistencias y validar que todos los datos y asociaciones hayan sido identificadas.

Se usan fundamentalmente en la fase de análisis del problema, y por tanto, el diseño de un modelo conceptual parte de la especificación de requisitos.

El proposito del modelo conceptual es describir el contenido de la información de la base de datos, en vez de las estructuras de almacenamiento que se requerirán para manejar esa información. Se expresa mediante un lenguaje de alto nivel. Es un modelo de datos que describe un conjunto de conceptos de una realidad.

Características

- Expresividad:

  Representación de gran variedad de restricciones.

- Simplicidad:

  Fácil de comprender por los usuarios.

- Minimalidad:

  Ningún conceptos presente se puede expresar por otros componentes.

- Formalidad:

  Conceptos con interpretación única, precisa y bien definida.

Los modelos más usados para bases de datos son:

- Entidad/Relación(MER)
- Diagrama de clases de UML
- Semántico

1.2.2. Etapas básicas del diseño de una Base de datos

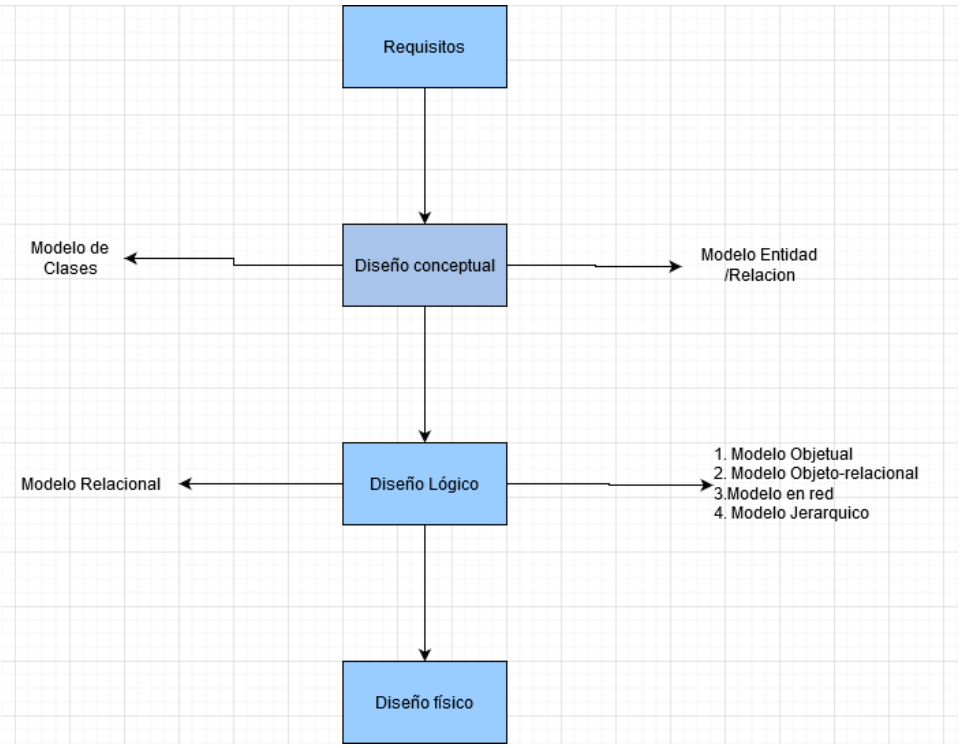


Figura 1.10: Etapas básicas del diseño de una base de datos

1.2.3. Introducción al modelo entidad-relación

- No existe un único MER, sino una familia de datos.
- Describe el 'mundo real' como un conjunto de entidades y relaciones entre ellas.
- Muy extendida a los métodos de diseño de bases de datos.

1.2.3.1. Esquema conceptual

Descripción concisa de los requisitos de información de los usuarios. Descripciones detalladas de:

- Tipos de datos.
- Relaciones entre datos.
- Restricciones que los datos deben cumplir.

Sin detalles de implementación

- Más fácil de entender.
- Comunicación con el usuario no técnico.

1.2.4. Conceptos básicos del modelo

1.2.4.1. Entidad(entity)

Cosa u objeto del mundo real con existencias propia y distinguible del resto. Es un objeto con existencia:

- Física o real (una persona, un libro, un empleado).
- Abstracta o conceptual (una asignatura, un viaje).

1. **Tipo de entidad.**

   El término entidad se utiliza tanto para denominar objetos similares de los que nos interesan los mismos atributos; es decir, que por ejemplo, se utiliza para designar tanto a un empleado concreto de una empresa como al conjunto de todos los empleados de la empresa. Ejemplos:

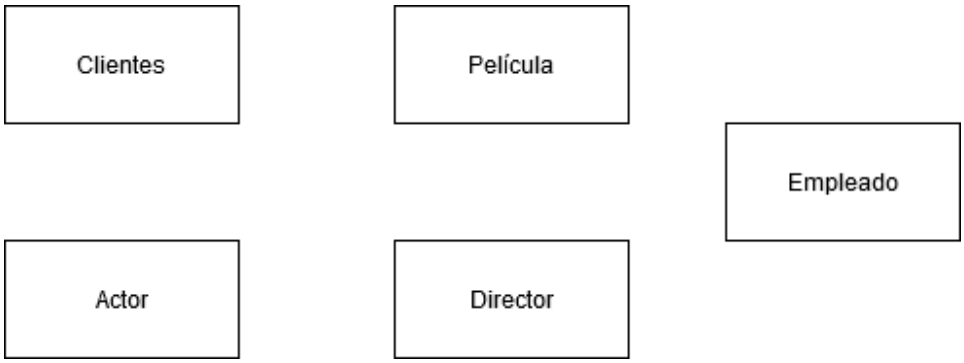


Figura 1.11: Tipos de entidad

2. **Instancia de un tipo de entidad.**

   También se conoce como:

   - Ocurrencia
   - Realización
   - Ejemplar
   - Entidad concreta o individual

3. **Entidad débil**

   Hay dos tipos de entidades Entidades fuertes (o regulareS) y débiles. Una entidad debil es una entidad cuya existencia depende de la existencia de otra entidad:

Figura 1.12: Notación

Figura 1.13: Ejemplo

4. **Dependencia en existencia**

   Si desaparece una instancia del tipo entidad regular deben desaparecer las instancias de la entidad debil que dependen de ella.

5. **Dependencia en identificación**

   - Además de la dependencia en existencia
   - Una instancia del tipo entidad debil no se puede identificar por si mismo
   - Su clave es (clave_entidad_regular,clave_parcial).

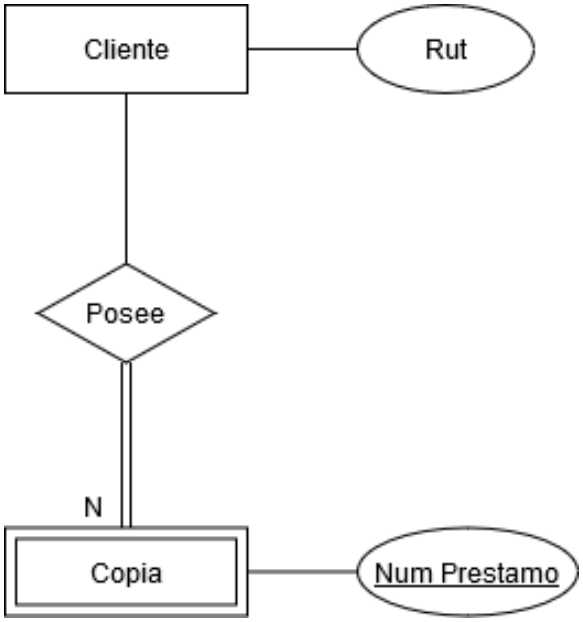


Figura 1.14: Ejemplo dependencia en existencia

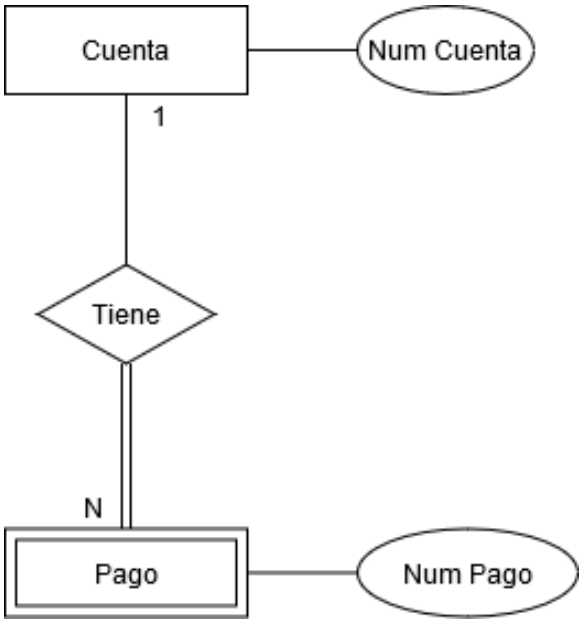


Figura 1.15: Ejemplo dependencia en identificación

1.2.4.2. Atributo

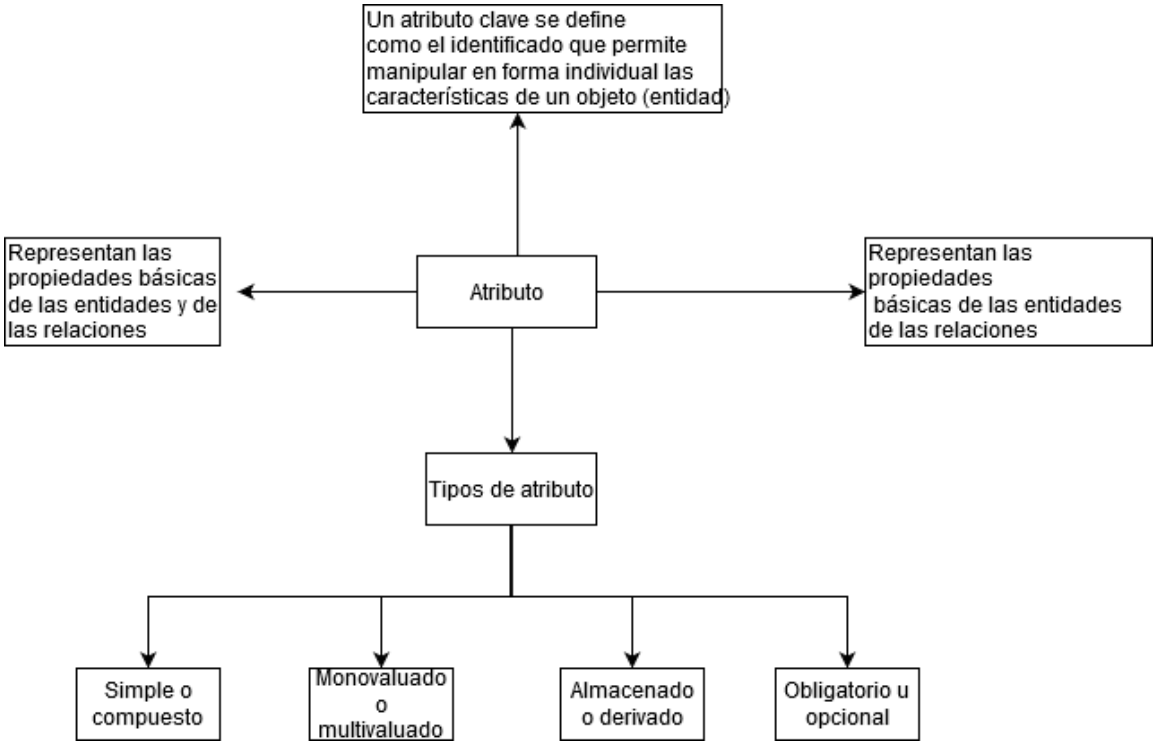


Figura 1.16: Atributo

1. **Atributos derivados**.

   Valor calculado a partir de otra información ya existente (atributos, entidades relacionadas). Son información redundante.

2. **Atributos almacenados**.

   Atributos que necesitan almacenarse

3. **Atributos compuestos**.

   Pueden dividirse en otros con significado propio.

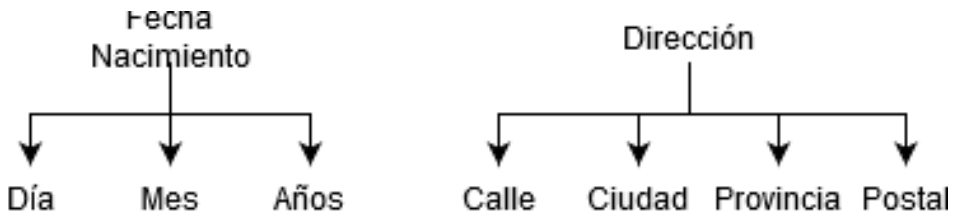


Figura 1.17: Atributo Compuesto

El valor compuesto consiste en una concatenación de valores de componentes.

4. **Atributos simples.**

   Elementos no divisibles, atómicos. Como por ejemplo 'Genero', 'Edad'.

5. **Atributo monovalorado.**

   Solo se le atribuye un valor para cada entidad.

6. **Atributos multivalorados**.

   Más de un valor para la misma entidad. Pueden tener límites superior e inferior del número de valores por entidad.

7. **Atributos opcionales (nulos)**

   El nulo (null value) es usado cuando:

   - Se desconoce el valor de un atributo para cierta entidad.
     - El valor existe pero falta:
       ALTURA[DE_UN_EMPLEADO]
     - No se sabe si el valor existe o no
       TELEFONO[DE_UN_EMPLEADO]
   - La entidad no tiene ningun valor aplicable para el atributo

8. **Atributos clave**

   Atributo con valor distinto para cada instancia de un tipo de entidad.

   RUT en EMPLEADO

   Una clave identifica de forma única cada entidad concreta (atributo identificador).

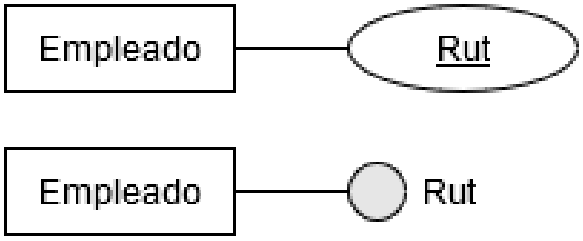


Figura 1.18: Atributo clave

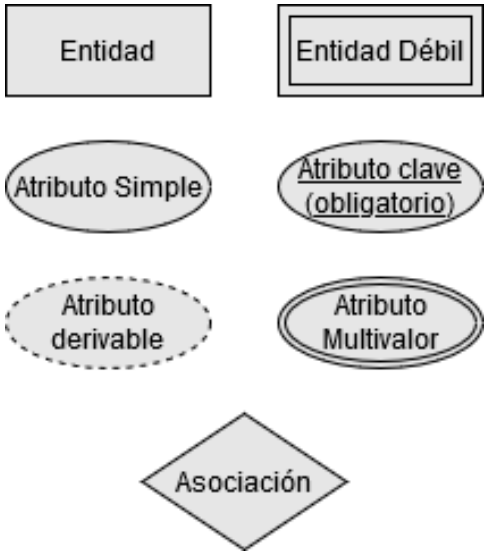


Figura 1.19: Notación

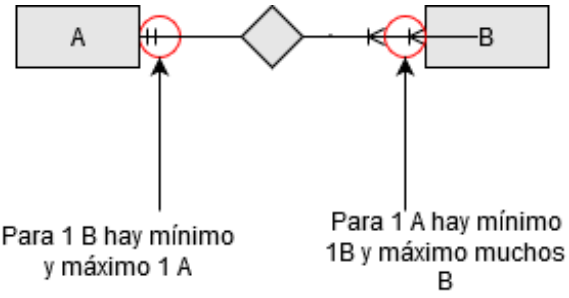


Figura 1.20: Ejemplo con cardinalidad

1.2.4.3. Dominio

Es un conjunto de valores. Cada atributo simple está asociado a un dominio que especifica sus valores válidos.

Dominio	Atributo	Descripción
Nombre	Nombres	Cadena de hasta 30 caracteres alfabéticos
Teléfono	Teléfonos	Cadena de hasta 9 caracteres numéricos
Altura	Medidas	Números reales entre 0 y 2,5 (metros)

Tabla 1.2: Ejemplos de dominios

1.2.4.4. Relación

Es denominado interrelación. Es una asociación, vínculo o correspondencia entre instancias de entidades relacionadas de alguna manera en el mundo real.

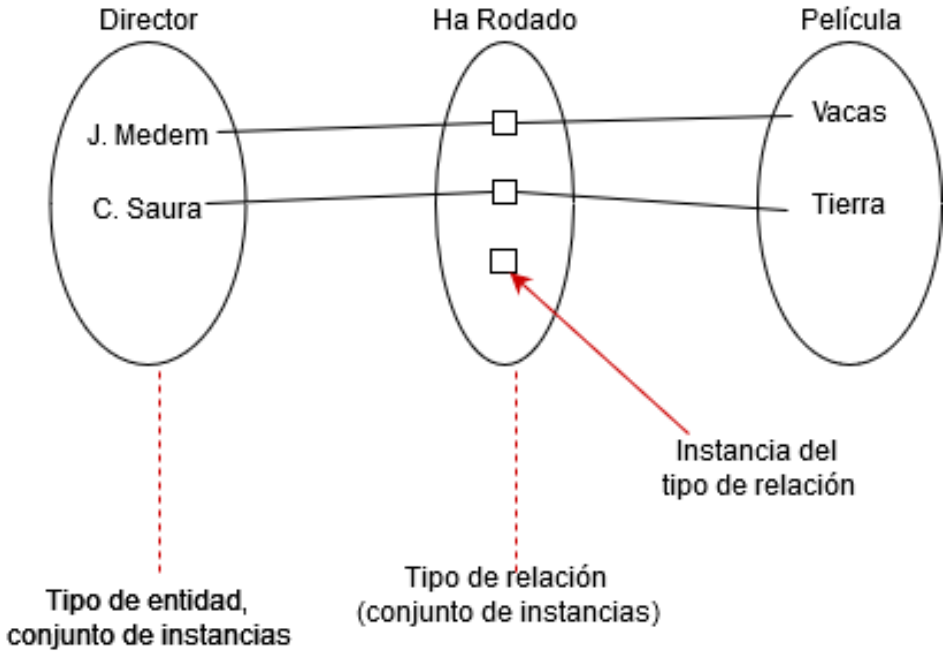


Figura 1.21: Ejemplo con relación

- Tipos de relación

  Estructura genérica o abstracción del conjunto de relaciones existentes entre dos o más tipos de entidad.

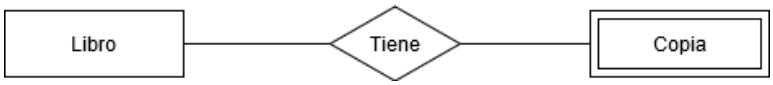


Figura 1.22: Notación

- Grado de un tipo de relación

  Número de tipos de entidad que participan en el tipo de relación.

  - Un ejemplo de una relación (o de grado 2) es la figura 1.22
  - Relación ternaria (o de grado 3):

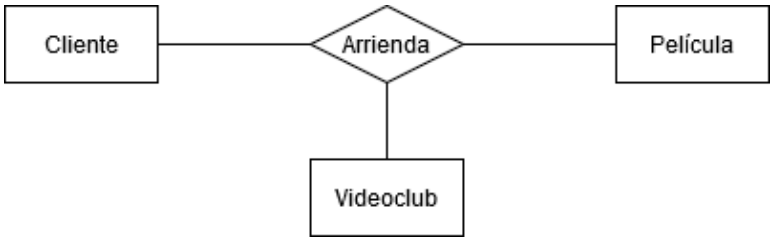


Figura 1.23: Ejemplo relación ternaria

  - Reflexiva (grado 1):

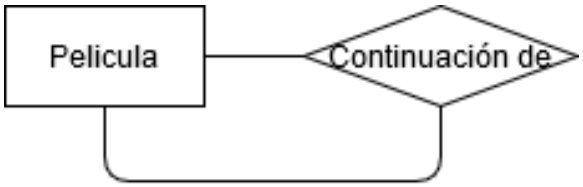


Figura 1.24: Ejemplo relación reflexiva

- Nombres de rol Todo tipo de entidad que participa en un tipo de relación juega un papel específico en la relación. Los nombres de rol se deben usar, sobre todo, en los tipos de relación reflexivos, para evitar ambiguedad.

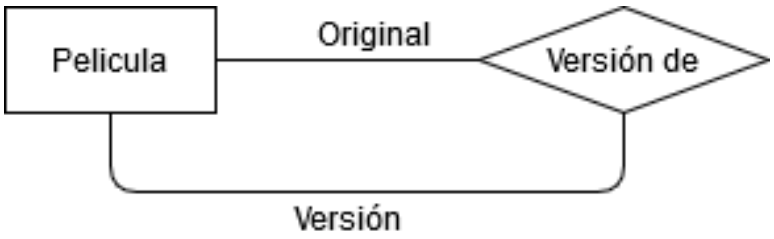


Figura 1.25: Ejemplo nombres de rol

- Restricciones estructurales sobre tipos de relación.

  Limitan las posibles combinaciones de entidades que pueden participar en las relaciones. Extraídas de la situación real que se modela:

  - Una película debe haber sido dirigido por una y solo un director.
  - Un director ha dirigido al menos una película y puede haber dirigido muchas.

  Clases de restricciones estructurales

  - Razón de participación
  - Razón de cardinalidad Número máximo de instancias de tipo de relación en las que participa un misma instancia de cada tipo de entidad.

Figura 1.26: Ejemplo razón de cardinalidad

    - La cardinalidad de 'ha rodado' es '1 a n'
    - 'Ha rodado' es de tipo '1 a n'

    Notación: Etiqueta en la lína que une entidad y relación

  - Razón de cardinalidad más comunes
    - 1:1 (uno a uno)
    - 1:n (uno a muchoS)
    - m:n (muchos a muchos)

- Cardinalidad de tipo entidad.

  Números mínimo y máximo de instancias del tipo de relación en las que puede intervenir una instancia del tipo de entidad.

  Notación:

  (min,max) en la línea que une entidad y relación

1.2.5. Extensiones del modelo

Permiten representar:

- Relaciones exclusivas entre sí.
- Jerarquías de especialización/generalización.
- Agregación de entidades

Especializaciones/generalizaciones(E/G).

- Caso especial de relación entre un tipo de entidad y varios otros tipos de entidad.
- La jerarquía o relación que se establece entre uno y otros corresponde a la noción de 'es un' o de 'es un tipo de'.
- Estas jerarquías pueden formarse por especialización o bien por generalización.

1.2.5.1. Subtipo de un tipo de entidad.

- Agrupación de instancias dentro de un tipo de entidad, que deben representarse explícitamente debido a su importancia para el diseño o aplicación:
  - Subtipos del tipo de entidad vehículo:
    - Camión
    - Turismo
    - Autobús
    - Ciclomotor
  - Subtipos del tipo de entidad empleado:
    - Secretario
    - Gerente

- Comercial

- El tipo de entidad que se especializa en otros se llama supertipo (vehículo, empleado). Es la relación que se establece entre un supertipo y cada uno de sus subtipos.

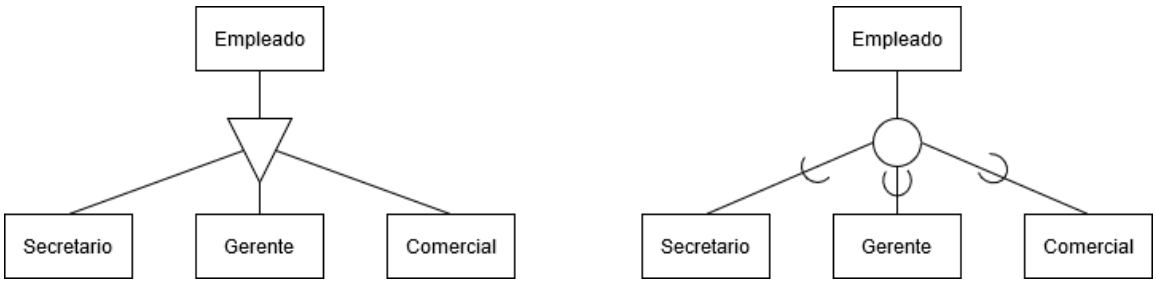


Figura 1.27: Notación Subtipo de entidad

- La extensión de un subtipo es un subconjunto de la extensión del supertipo.
  - Una instancia de subtipo también es instancia del supertipo y es la misma instancia, pero con un papel específico distinto.
  - Una instancia no puede existir solo por ser mimebro de un subtipo, también debe ser miembro del supertipo
  - Una instancia del supertipo puede no ser miembro de ningún subtipo.

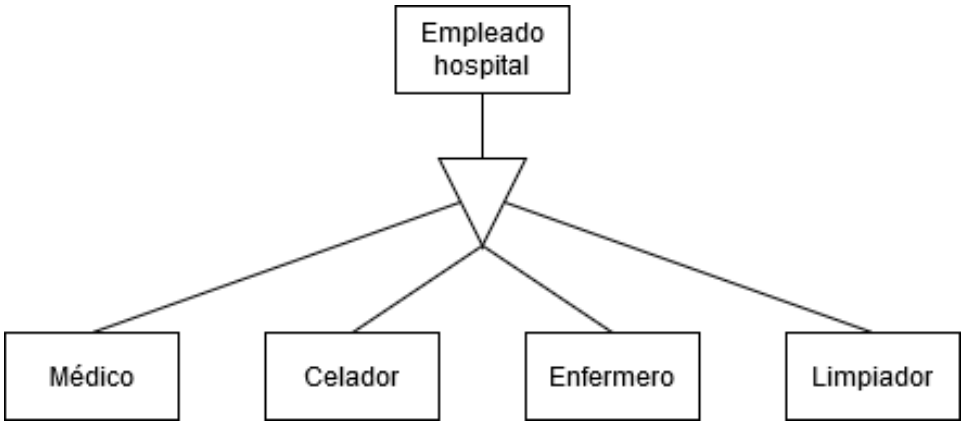


Figura 1.28: Notación Subtipo de entidad

- Un subtipo puede tener atributos propios (específicos) y participar en relaciones por separado.
- Un subtipo hereda todos los atributos del supertipo, y toda relación en la que participa el supertipo.

  Un subtipo, con sus atributos y relaciones específicos más los atributos y relaciones que hereda del supertipo, es un tipo de entidad por derecho propio.cx
- Varias especificaciones de un tipo de entidad, con base en diferentes discriminantes.

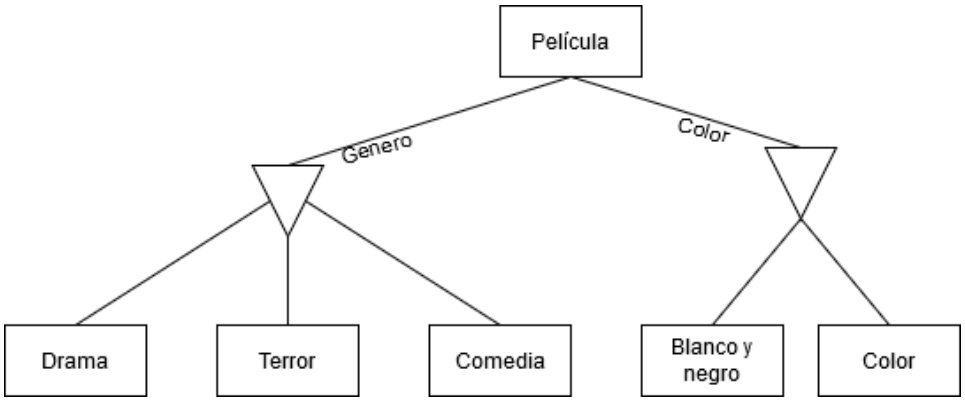


Figura 1.29: Notación Subtipo de entidad

- Conviene incluir relaciones subtipo/supertipo si hay
  - Atributos que solo tienen sentido para algunas instancias de un tipo y no para todos (atributos específicos).

    EspecialidadMedica «no es aplicable» a guardia
  - Tipos de relación en los que solo participan algunas entidades de un tipo y no todas (relaciones específicas).

    Relación supervisa entre guardia y sección_hospital

Figura 1.30: Ejemplo Convenio

- Generalización:
  - Énfasis en las similitudes
  - Cada instancia del supertipo es también una instancia de alguno de los subtipos.
- Especialización:
  - Énfasis en las diferencias
  - Alguna instancia del supertipo puede no ser instancia de ningún subtipo.
- ¿Qué instancias del subtipo pertenecen a cada subtipo?
  - Exclusión

    ¿A cuántos subtipos puede pertenecer (a la vez) una instancia del supertipo?
  - Completitud/Parcialidad

    ¿Debe toda instancia del supertipo pertenecer a algún subtipo?
- Subtipos exclusivos.

  Subtipos disjuntos o exclusivos si una instancia del supertipo puede ser miembro, como máximo, uno de los subtipos.

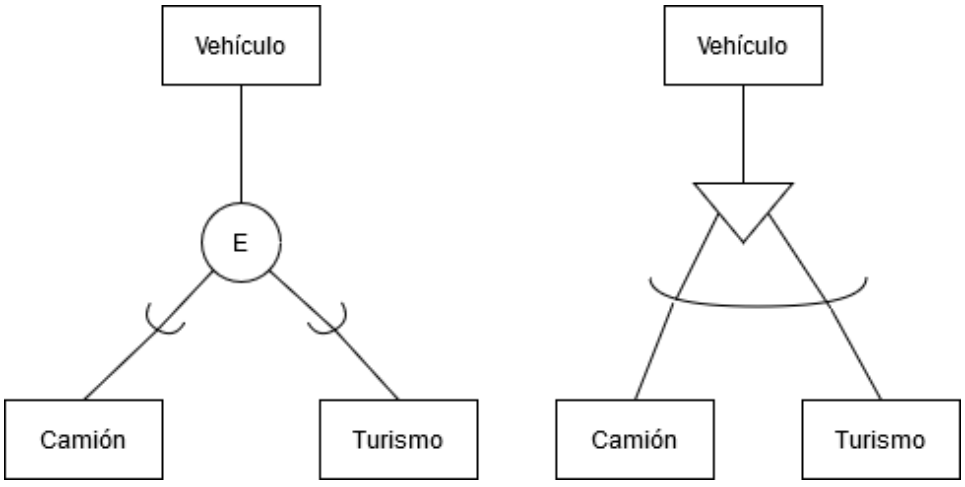


Figura 1.31: Subtipos exclusivos

- Subtipos solapados.

  Subtipos solapados si una instancia del supertipo puede ser, a la vez, miembro de más de un subtipol. Es la opción por defecto.

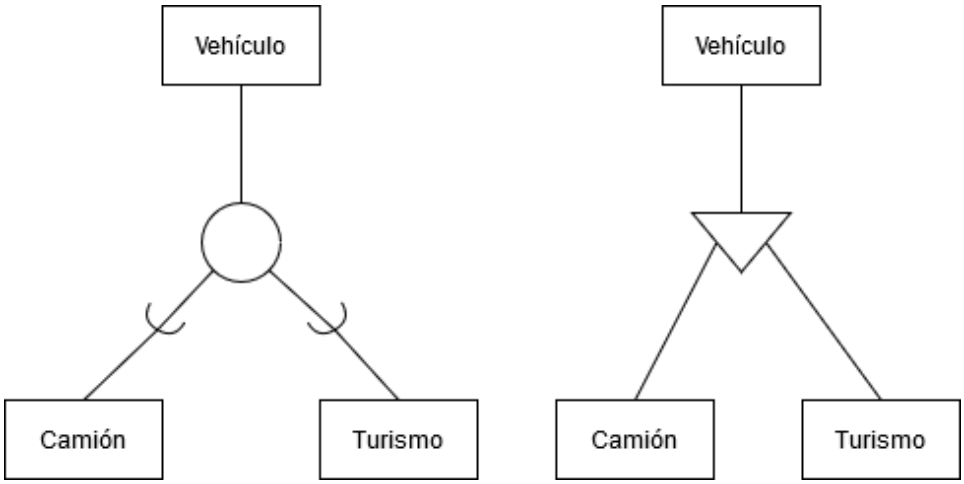


Figura 1.32: Subtipos solapados

- Especialización completa

  Especialización total (completa) indica que toda instancia del supertipo también debe ser instancia de algún subtipo.

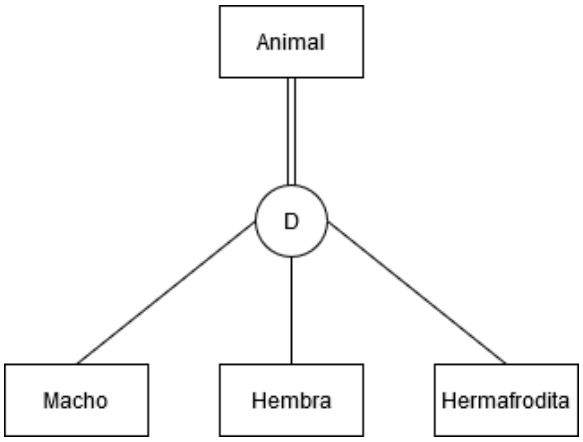


Figura 1.33: Especialización completa

- Especialización parcial.

  Especialización parcial indica que es posible que alguna instancia del supertipo no pertenezca a ninguno de los subtipos. Es la opción por defecto.

  La unión de las extensiones de los subtipos no es la extensión del supertipo en su totalidad.

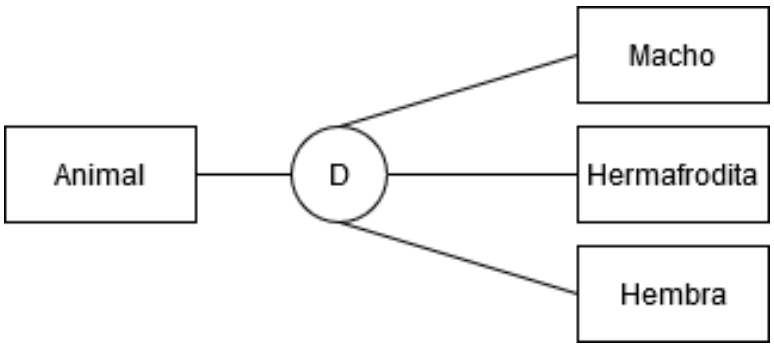
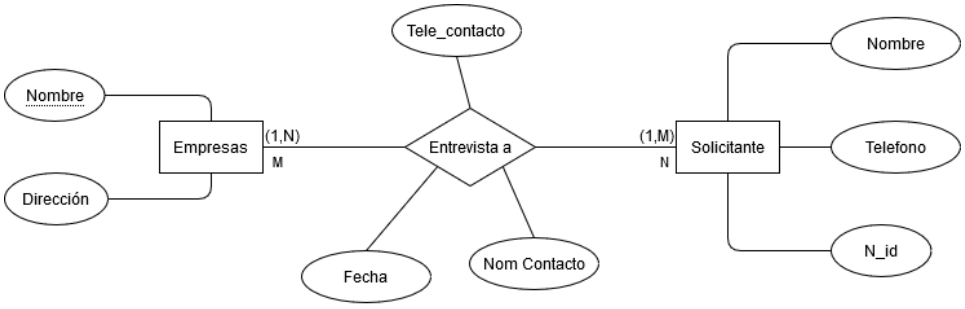


Figura 1.34: Especialización parcial

Esquema en el MER que almacena información sobre las entrevistas que una ETT organiza entre solicitantes de empleo y diferntes empresas.



1.3. Modelo de datos relacional

1.3.1. Representación

Se basa en el concepto de relación (diferentes a lo que es una relación en el modelo ER). Informalmente en el modelo relacional existe:

$$\text{Relación} = \text{tabla}$$

Se apoya en el algebra y el cálculo de relaciones. Generó los SGSD relaciones, pero es independiente de ellos, se puede implementar en cualquier SGBD.

1.3.2. Relación

Se define como un conjunto de atributos, cada uno de los cuales pertenece a un dominio y posee un nombre que la identifica. Se representa gráficamente como una tabla con columnas (atributos) y filas (tuplas).

Los valores que aparecen en cada una de las columnas pertenecen al conjunto de valores del dominio sobre el que está definido el atributo correspondiente.

1.3.3. Elementos del modelo relacional

Título	Año	Duración
Stars Wars	1997	120
El señor de los anillos	2001	180
Mar adentro	2004	90
El viaje de chihiro	2001	120

- 'Película' al nombre de la relación.
- 'Titulo', 'Año' y 'Duración' representan el nombre de cada atributo (columna).
- Cada fila de la relación 'Película' representa a una tupla de datos.

1.3.4. Propiedades de la relación

- La relación es el único elemento utilizado para representar tanto entidades como asociaciones entre ellas. Cada relación tiene un nombre, y éste es distinto del nombre de todas las demás.

- En una relación no hay filas (tuplas) repetidas, y las tuplas no están ordenadas.
- Las columnas de una relación tienen un nombre único dentro de la relación y no tiene orden.
- Cada celda es atómica (univaluada).

1.3.5. Notación para la relación

El esquema R se denota como:

$$\mathrm{R}(A_1, A_2, ..., A_n)$$

Donde R es el nombre de la relación, y $A_1, A_2, ..., A_n$ son los atributos de R.

Ejemplo:

EMPLEADO (CEDULA, NOMBRE, DIRECCION, SALARIO)

1.3.6. Atributos

Son características que describen una entidad o relación. Los atributos tienen asociados dominios.

Un dominio representa el conjunto de valores permitidos para un atributo, por ejemplo, los valores si y no, números enteros, etc. Los dominios sobre los que se definen los atirbutos son escalares, por lo que los valores de los atributos son atómicos.

1.3.7. Notación para dominio

- **Dominio.**

  El dominio del atributo A se denota como DOM(A).

- **Representación del dominio.**

  $$T[A] =< X > \rightarrow x \in \mathrm{DOM(A)}.$$

  Ejemplo.

  En algunos casos NULL *in* DOM(A), lo cual significa que el atributo A acepta valores nulos.

- **Una definición formal de relación.**

$$R(A_1, A_2, ..., A_n) \subseteq (DOM(A_1) \text{ x } DOM(A_2) \text{ x } .... \text{ x } DOM(A_n))$$

R es el subconjunto del producto cartesiano de los dominios de $A_1, A_2, ..., A_n$.

1.3.8. Tupla

- Cada instancia o fila registro de una relación es una tupla.
- Número de tuplas:

  Cardinalidad o extensión de la relación.

1.3.9. Tipos de relaciones

En un SGBD relacional hay dos tipos de relaciones.

- **Relaciones base.**

  Son relaciones reales que tienen nombre, y forman parte directa de la base de datos almacenada.

- **Vistas.**

  También denominadas relaciones virtuales, son relaciones con nombre y derivadas. Que son derivadas significa que se obtienen a partir de otras relaciones; se representan mediante su definición en terminos de esas otras relaciones. Las vistas no poseen datos almacenados propios, los datos que contienen corresponden a datos almacenados en relaciones base.

1.3.10. Claves

Las tuplas no pueden repetirse, y para asegurar esa propiedad, se usan los identificadores o claves. Se distinguen diferentes tipos de claves:

1.3.10.1. Clave candidata

Atributo o atributos que identifican de manera única una tupla dada. Deben cumplir unicidad y minimalidad (irreducibilidad).

Ejemplo.

N Matricula	N Motor	Marca	Modelo
CCA-341	91234908123	Toyota	Yaris
OFG-851	53489787679	Fiat	Fiorino
XTV-657	30752312386	Ford	Mustang
WGB-959	50934187123	Toyota	Avensis

Las claves candidatas son 'N Matricula' o 'N Motor'. Un automovil puede identificarse por el número de matrícula o por el número del motor.

1.3.10.2. Clave primaria

Es aquella clave candidata que se escoge para identificar las tuplas de una relación de modo único. Las demás quedan como claves alternativas o secundarias. Cuando una clave está formada por más de un atributo , se dice que es una clave compuesta.

Regla de integridad de las claves primarias.

'Ningún componente de la clave primaria acepta nulos'.

1.3.10.3. Clave secundaria

Un atributo o un conjunto de atributos de una relación cuyos valores coinciden con los valores de la clave primaria de alguna otra relación. Especifica de forma explícita la forma en que dos tablas se relacionan.

Por ejemplo, existen dos relaciones. Empleados y Departamentos.

ID	Primer nombre	Apellido	Departamento ID
100	Steven	King	90
101	Neena	Kokkar	90
102	Lex	De Haan	90
103	Jennifer	Whalen	10

Tabla 1.3: Relación 'empleado'

Departamento ID	Nombre dpto
10	Administración
20	Marketing
50	Ventas

Tabla 1.4: Relación 'Departamento'

Se designa 'Departamento ID' de la relación departamento como Primary Key de dicha relación. Por lo tanto, este mismo atributo en la relación 'empleado' será la Foreing Key de ella.

Otros tipos de clave:

- Superclave
- Natural
- Inteligente o semántica
- Artificial o subrogada
- Solapadas.

1.3.11. Reglas de integridad

Son las reglas semánticas que los datos almacenados en una estructura de BD deben cumplir para garantizar que son correctos.

- Restricción de valor único (UNIQUE)
- Estra restricción impide que un atributo tenga un valor repetido. Las claves primarias deben cumplir con esta restricción.

1.3.12. Restricción de dominio

Esta restricción exige que el valor que puede tomar un atributo este dentro del dominio definido. Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo.

1.3.12.1. Restricción de valor nulo

Cuando en una tupla un atributo es desconocido, se dice que es nulo. El nulo implica ausencia de información, bien porque al insertar la tupla se desconocía el valor del atributo, o bien porque para dicha tupla el atributo no tiene sentido.

Un atributo que no admite valores nulos es obligatorio. Todos los atributos que componen una clave primaria son obligatorios, esto se conoce como integridad de entidad.

Un atrinuto que admite valores NULL es un atributo opcional.

1.3.13. Regla de integridad referencial

'Ningún componente de una clave foránea puede contener valores que no están presentes en la clave primaria a la que hace referencia.'

- ¿Puede una clave foránea admitir nulos?

  Si, se debe especificar.

- ¿Cómo es el dominio de una clave foránea frente al dominio de la clave primaria a la que referencia?

  Deben tener el mismo dominio.

- ¿Qué pasa si la referencia ('Padre') de una clave foránea intenta ser cambiada?

  Posibles curso de acción.

  - Cascada:
    Se cambia la tupla referenciada y se propaga el cambiado a las tuplas que la referencian mediante la clave ajena.
  - Restringido:
    No se permite cambiar la tupla referenciada.
  - Nulificación:
    Poner NULL en las tuplas que la referenciaban (claves foránea), si acepta nulos.
  - Valor por defecto:
    Se cambia la tupla referenciada y en las tuplas que la referencian se pone el valor por defecto establecido.

1.3.14. Comparativo de terminos

Termino relación formal	Equivalente informal	Sistema de archivos
Relación	Tabla	Archivo
Tupla	Fila	Registro
Atributo	Columna	Campo
Instancia	Valor	Valor
Cardinalidad	#de filas	Sin equivalente
Grado	# de Grados	Sin equivalente
Clave primaria	Identificador unico	Sin equivalente
Clave foránea	Interrelación	Sin equivalente
Dominio	Valor válido	Valor válido

Tabla 1.5: Comparativo de terminos

Capítulo 2

Ayudantias

2.1. Ayudantía n1:

2.1.1. Structured Query Language (SQL)

El Lenguaje de consulta estructurada o SQL, está compuesto por:

- Data definition Languaje (DDL)
  El lenguaje de manipulación de datos, son un esquema de las relaciones, que se usan para generar objetos en una base de datos.
  - Crear elementos en la Base de datos (**CREATE**):
    Base de datos, esquema, vista, tabla, atributo (columna), procedimiento de almacenamiento.
  - Actualizar dato(**ALTER)**
  - Borrar dato (**DROP**).
- Data manipulation Languaje (DML)
  El lenguaje de manipulación de datos hace uso de los datos almacenados, haciendo modificación de los valores.
  - Insertar datos (**INSERT**)
  - Cambia valores (**UPDATE**)
  - Borrar datos (**DELETE**)
  - Consultar (**SELECT**)

  De alguna manera, manipular datos para obtener o recuperar información de la BD
- Data control Languaje (DCL) o lenguaje de control de datos.

2.1.2. ¿Qué es el SQL?

Es un lenguaje de programación diseñado para almacenar y recuperar datos almacenados en bases de datos relacionales.

- Estructura básica
  SELECT $A_1, A_2, A_3, ..., A_N$
  FROM $R_1, R_2, ..., R_N$
  WHERE condición

  - $A_1, A_2, A_3, ..., A_N$ es una lista de atributos
  - $R_1, R_2, ..., R_N$ lista de relaciones (tablas)
  - condición Es la condición que se aplica sobre los atributos de la clausula FROM

- SELECT: es la palabra clave que índica que la sentencia de SQL que queremos ejecutar es de selección.

- ALL: Indica que queremos seleccionar todos los valores. Es el valor por defecto y no suele especificarse.

- DISTINCT: Indica que queremos seleccionar solo los valores distintos.

- FROM: Indica la tabla (o tabla) desde la que queremos recuperar los datos. En el caso de que exista más de una tabla se denominaa la consulta 'consulta combinada' o 'join'. En las consultas combinadas es necesario aplicar una condición de combinacióna través de una clausula.

- WHERE: Especifica una condición que debe cumplirse para que los datos sean devueltos por la consulta. Admite los operadores lógicos AND y OR. También es utilizada para separar los registros seleccionados en grupos específicos.

- OR GROUP BY: Utilizada para determinar los registros seleccionados en grupos específicos.

- ORDER BY: Presenta el resultado ordenado por las columnas indicadas. El orden puede expresarse con ASC (orden ascendente) y DESC(orden descendente). El valor predertminado es ASC.

2.1.3. Conceptos

- **Tipos de datos**
  - Tipos de numéricos

Tipo de campo	Tamaño de almacenamiento
Tiny int	1 byte
Small int	2 bytes
Medium int	3 bytes
int	4 bytes
integer	4 bytes
Big int	8 bytes
Float (x)	4 u 8 bytes
Float	4 bytes
Double	8 bytes
Double precision	8 bytes
Real	8 bytes

Tabla 2.1: Tipos numéricos

  - Tipos de fichas
    - DATE: Tipo de fecha, almacena una fecha. El rango de valores va desde el 1 de enero de 1001 al 31 de diciembre de 9999.El formato de almacenamiento es de año-mes-día.
    - DATE TIME: Combinación de fecha y hora. El rango de valores va desde el 1 de enero del 1001 a las 0 horas, 0 minutos y 0 segundos al 31 de diciembre del 9999 a las 23 horas, 59 minutos y 59 segundos. El formato de almacenamiento es de año-mes-día horas:minutos:segundos.
    - TIME STAMP: Combinación de fecha y hora. El rango va desde el 1 de enero de 1970 al año 9037
  - Tipos de cadena

Tipo de campo	Tamaño de almacenamiento
CHAR(N)	N bytes
VARCHAR(N)	(N+1) bytes
TINY BLOB TINT TEXT	(Longitud+1) bytes
MEDIUM BLOB MEDIUM TEXT	(Longitud+3) bytes
LONG BLOB LONG TEXT	(Longitud+4) bytes
BLOB, TEXT	(Longitud+2) bytes

Tabla 2.2: Tipos de cadena

- **Restricción de datos**
  - Útiles para evitar datos inválidos en la Base de datos
  - NOT NULL: Evita que se inserten valores nulos en una columna. Se puede combinar con DEFAULT.
  - UNIQUE: Evita valores duplicados en una columna. Se implementa creando un índice. Es posible ingresar valores nulos.
  - PRIMARY KEY: Indica la columna que identifica de manera única a cada registro. Es una combinación de UNIQUE y NOT NULL. Al igual que con UNIQUE se crea automáticamente un índice y se agregael modificador NOT NULL a la columna.
- **Agregar registros**
  Supongamos que existe una tabla de persona.

```
INSERT INTO PERSONA VALUES('BON','SCOTT', 'QUILPUE',35);
INSERT INTO PERSONA VALUES('CHUCK','SCHULDINER','VALPARAISO',30);
INSERT INTO PERSONA VALUES('CLIFF','BURTON','QUILPUE',25);
```

NOMBRE	APELLIDO	CIUDAD	EDAD
BON	SCOTT	QUILPUE	35
CHUCH	SCHULDINER	VALPARAISO	30
CLIFF	BURTON	QUILPUE	25

Tabla 2.3: Ejemplo Agregar registro

- **Eliminar registros**
  Tomando en cuenta que usamos la tabla definida anteriormente como personas, ejecutamos:

  DELETE FROM PERSONA WHERE CIUDAD='VALPARAISO'

NOMBRE	APELLIDO	CIUDAD	EDAD
BON	SCOTT	QUILPUE	35
CLIFF	BURTON	QUILPUE	25

Tabla 2.4: Ejemplo eliminar registro

- **Mostrar registros**
  Si queremos mostrar todos los registros, debemos efectuar la siguiente línea:

  SELECT * FROM PERSONA;

  De esta manera, mostraremos todos los elementos que componen a la tabla

- **Mostrar los registros ordenados por edad**

  SELECT * FROM PERSONA ORDER BY EDAD;

NOMBRE	APELLIDO	CIUDAD	EDAD
CLIFF	BURTON	QUILPUE	25
CHUCH	SCHULDINER	VALPARAISO	30
BON	SCOTT	QUILPUE	35

Tabla 2.5: Mostrar los registros ordenados por edad

- **Mostrar registros ordenados por edad descendiente**

  SELECT * FROM PERSONA ORDER BY EDAD DESC;

NOMBRE	APELLIDO	CIUDAD	EDAD
BON	SCOTT	QUILPUE	35
CHUCH	SCHULDINER	VALPARAISO	30
CLIFF	BURTON	QUILPUE	25

Tabla 2.6: Mostrar los registros ordenados por edad descendiente

- **Consultas**

SELECT * FROM PERSONA WHERE CIUDAD='QUILPUE' AND EDAD <= 30;

NOMBRE	APELLIDO	CIUDAD	EDAD
CLIFF	BURTON	QUILPUE	25

Tabla 2.7: Consultas

- **Consultas sin obtener repetidos**

NOMBRE	APELLIDO	CIUDAD	EDAD
BON	SCOTT	QUILPUE	35
CHUCH	SCHULDINER	VALPARAISO	30
CLIFF	BURTON	QUILPUE	25

Tabla 2.8: Ejemplo de consultas sin repetidos

SELECT DISTINC CIUDAD FROM PERSONA ORDER BY CIUDAD;

CIUDAD
QUILPUE
VALPARAISO

Tabla 2.9: Resultado de consultas sin repetidos

- **Consultar por rango**

SELECT * FROM PERSONA WHERE EDAD BETWEEN 24 AND 30;

Que es equivalente a decir:

SELECT * FROM PERSONA WHERE EDAD >=25 AND EDAD <=30;

- **Consultas por comparaciones**

SELECT * FROM PERSONA WHERE CIUDAD LIKE '%v%';

NOMBRE	APELLIDO	CIUDAD	EDAD
BON	SCOTT	QUILPUE	35

Tabla 2.10: Ejemplo de consultas con comparaciones

COMPARACIONES	SINTAXIS
COMIENZA CON D	LIKE 'D%'
CONTIENE A D	LIKE '%D%'
D EN 2° LUGAR	LIKE '_D%'
NO CONTIENE CON D	NOT LIKE 'D%'

Tabla 2.11: Tabla de comparaciones

- **Otras operaciones importantes**

  - Editar un campo
    UPDATE MI_TABLA SET CAMPO1='NUEVO VALOR CAMPO1'
    WHERE CAMPO1='N';
  - Añadir columna
    ALTER TABLE TABLA1 ADD COLNUEVA VARCHAR(2);
  - Borrar columna
    ALTER TABLE TABLE_NAME DROP COLUMN COLUMN_NAME;

2.2. Ayudantía n2:

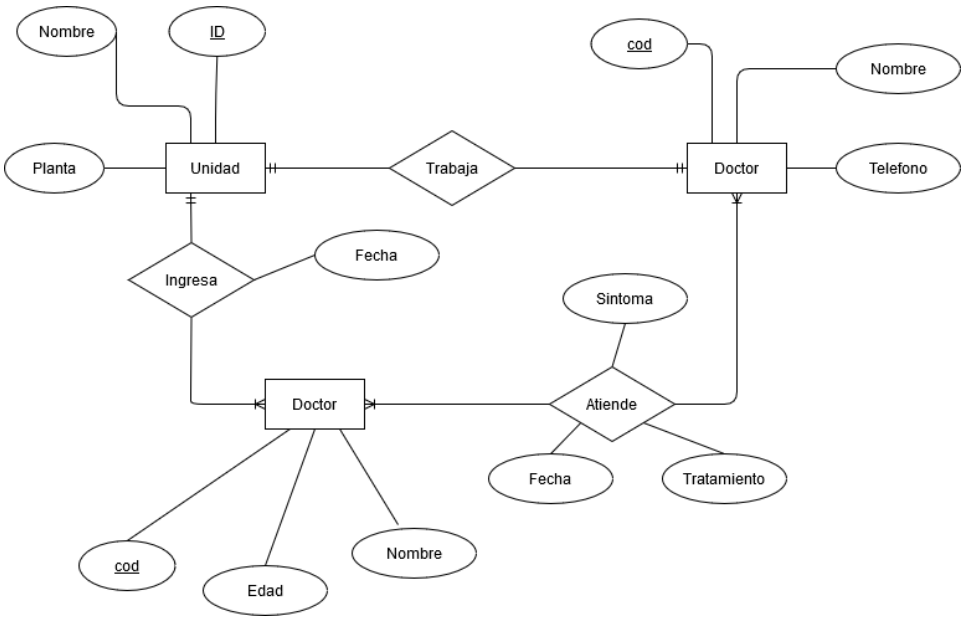
2.2.1. Pregunta 1

Una clínica desea mantener una base de datos con el historial de todos los pacientes que tiene ingresados.

La clínica está dividida en varias unidades, y cada una de las cuales tiene un identificador, su nombre y la planta en la que se encuentra. La unidad tiene un único doctor responsable, del cual se desea almacenar su código, el nombre y su especialidad.

Cuando llega un paciente, se le ingresa en una unidad, y se registra su número de la ss, nombre, edad y fecha de ingreso. Durante toda su estancia en la clínica, se anotan todas las intervenciones que realizan cada uno de los doctores, indicando la fecha, el síntoma observado y el tratamiento preescrito. Construya el MER.

Desarrollo:



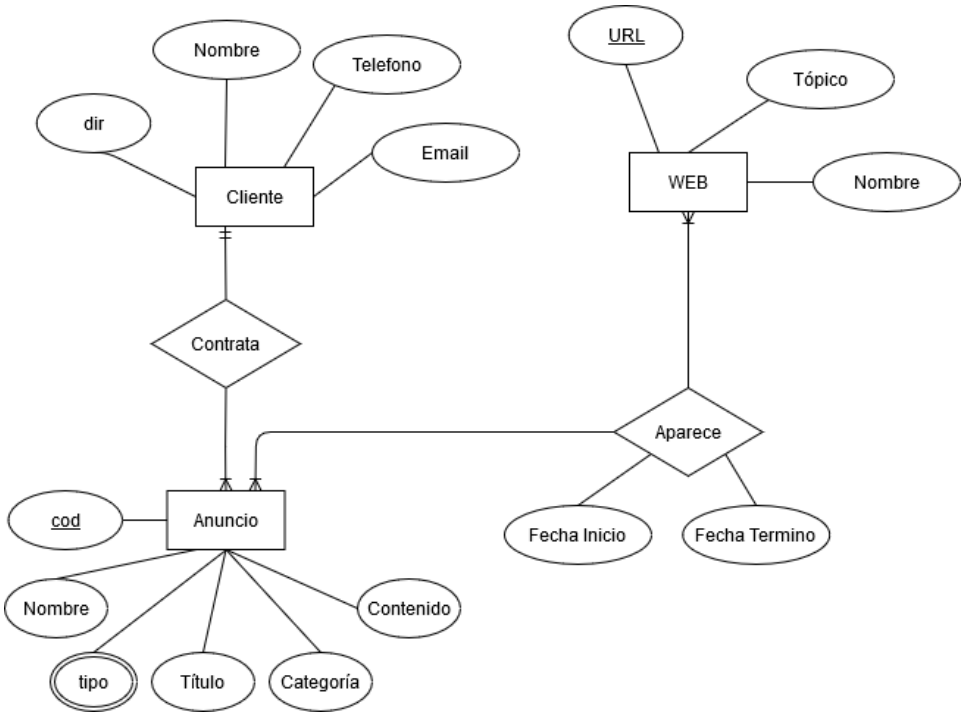
2.2.2. Pregunta 2

Una agencia de publicidad necesita una base de datos para registrar todas sus campañas en la web.

Sus clientes tienen un nombre, una dirección postal, el número de teléfono y una dirección de email. Cada cliente puede contratar varios anuncios. Los anuncios quedan identificados por un código y se caracterizan por un nombre, tipo (banner, popup, enlace patrocinado, etc), título, contenido, categoría (tipo de producto) y precio. Los anuncios pueden aparecer en más de una página web.

Cada web se caracteriza por su URL, nombre y tópico de interés. También se debe almacenar la fecha de inicio y de fin de la aparición del anuncio en la página web. Construya un MER para este caso.

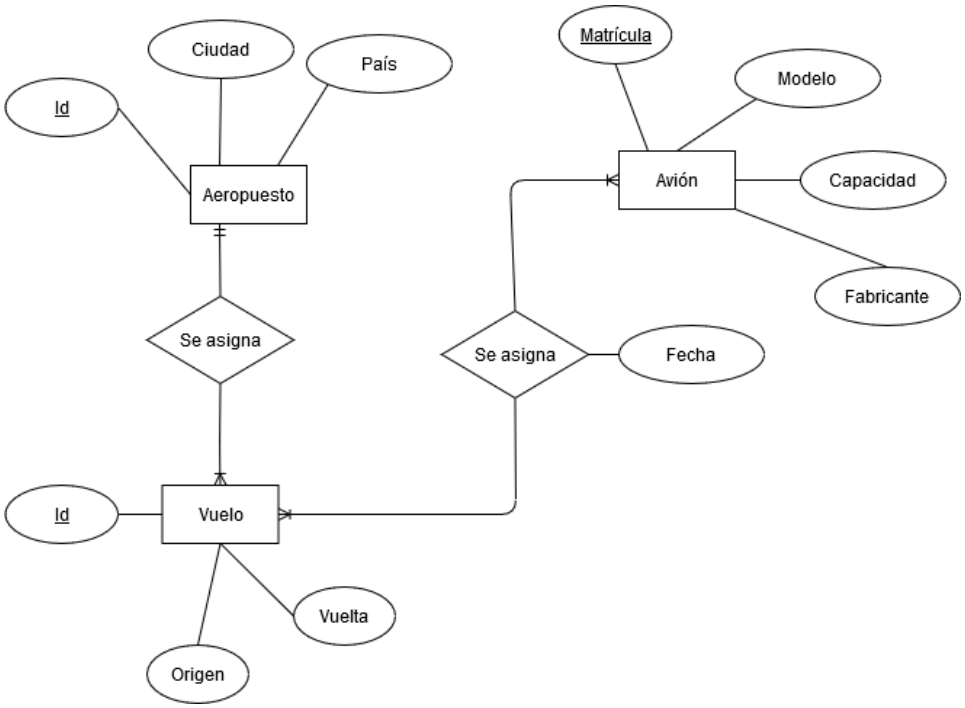
Desarrollo:



2.2.3. Pregunta 3

Una compañía aérea necesita una base de datos para registrar la información de sus vuelos. Los vuelos tienen un identificador único. Además, cada vuelo tiene asignado un aeropuerto de origen y uno de destino (no hay escala). Los aeropuertos están identificados por una siglas únicas. Además, de cada aeropuerto se guarda el nombre de la ciudad en la que está situada y el país. Cada vuelo es realizado por un avión. Los aviones tienen una matrícula que los identifica, el fabricante, un modelo e información sobre su capacidad (número máximo de pasajeros) y autonomía de vuelo (en horas). La asignación de aviones a vuelos no es único, así que no es necesario saber la frecha en la que un avión realizó cada uno de los vuelos asignados. Construya un MER para el caso.

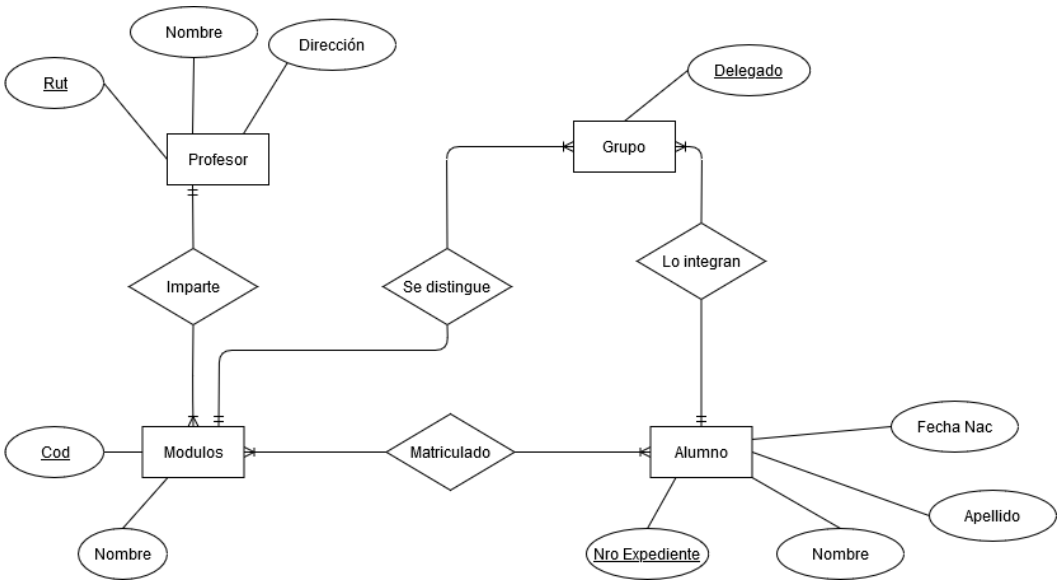
Desarrollo:



2.2.4. Pregunta 4

Se desea diseñar la base de datos de un instituto. En la base de datos, sea desea guardar los datos de los profesores del instituto (rut, nombre, dirección y teléfono). Los profesores imparten módulos, y cada módulo tiene un código y un nombre. Cada alumno está matrículado en uno o varios módulos. De cada alumno, se desea guardar el número de expediente, nombre, apellidos y fecha de nacimiento. Los profesores pueden impartir varios módulos, pero un módulo solo puede ser impartido por un profesor. Cada curso tiene un grupo de alumnos, uno de los cuales es el delegado del grupo.

Desarrollo:



2.3. Ayudantía n3:

2.3.1. Extensiones del modelo entidad relación

2.3.1.1. Relaciones exclusivas

Dos (o más) tipos de relación son exclusivos , respectos de un tipo de entidad que participa en ambas si cada instancia del tipo de entidad solo puede participar en uno de los tipos de relación.

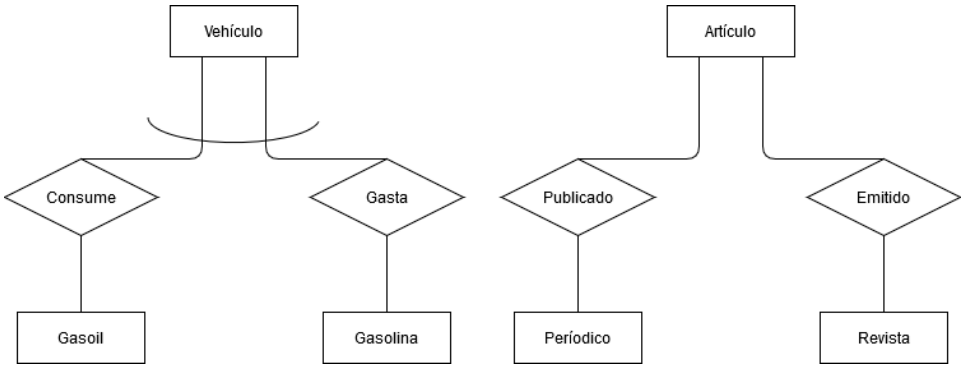


Figura 2.1: Relaciones exclusivas

2.3.2. Especialización y generalización

Las jerarquías pueden formarse por 'especialización' o 'generalización'. El tipo de entidad se 'especializa' en otra referida como supertipo (padre). Se lee como 'es un', 'es un tipo de'.

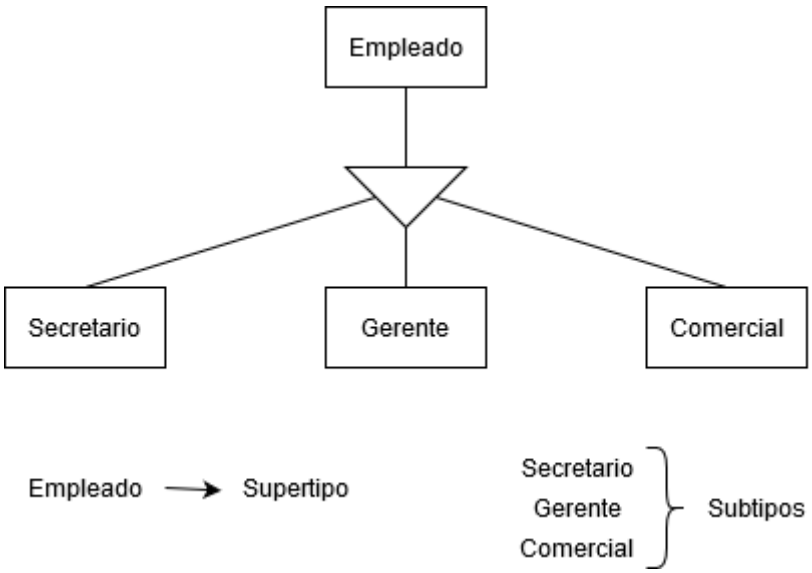


Figura 2.2: Ejemplo de Especialización/Generalización

Un subtipo (hijo) puede tener atributos propios (especificos) y participar en relaciones por separado. Además, hereda todos los atributos de su padre (supertipo) y toda relación en la que participa el supertipo (padre).

2.3.2.1. Tipos de especializaciones

- **Especialización total**

  Todas las entidades son de algún subtipo especializado. Todas las entidades 'empresa' se deben especializar en 'públicas' o 'privadas'.

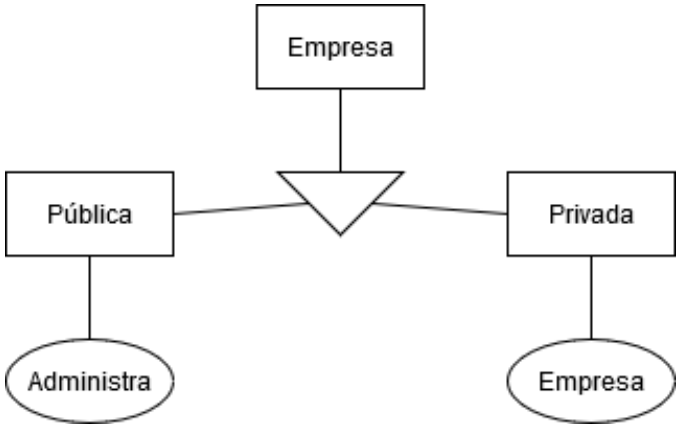


Figura 2.3: Ejemplo de Especialización total

- **Especialización parcial**

  Es posible que alguna instancia del supertipo no pertenezca a ninguno de sus subtipos.

  Por ejemplo, aparecen lácteos, frutas y verduras. Podrían alimentos que podrían ser considerados granos (cereales) o carne pero no aparecen en el modelo.

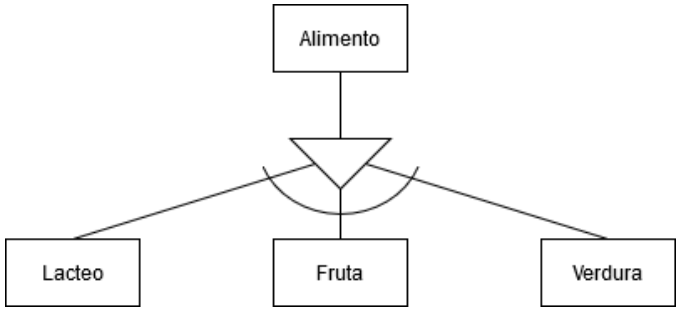


Figura 2.4: Ejemplo de Especialización parcial

- **Subtipos disjuntos (exclusivos)**

  Una instancia del supertipo puede ser miembro de como máximo uno de los subtipos (hijos).

  Por ejemplo, el vehículo es de turismo o es de camión (no puede ser ambos).

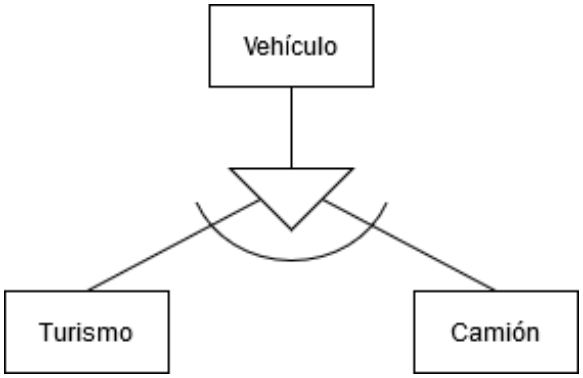


Figura 2.5: Ejemplo de Subtipos disjuntos

- **Subtipos solapados(superpuestos)**

  Una instancia del supertipo puede ser, a la vez, miembro de más de un subtipo. Por ejemplo, es la opción 'por defecto'. Una persona puede ser empleado y estudiante a la vez.

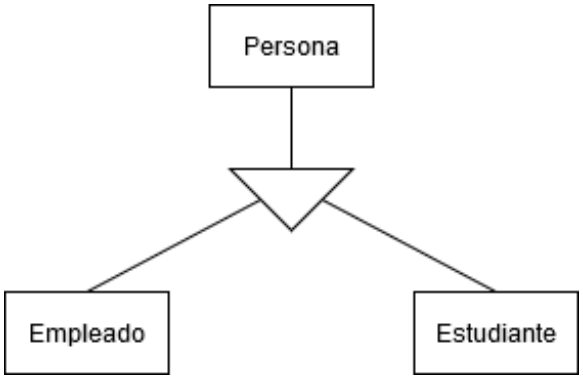


Figura 2.6: Ejemplo de Subtipos Solapados

Entonces, tenemos los siguientes casos:

- Casos cobertura total y exclusiva.

  Ejemplo: Todas las personas son empleados o clientes del banco, pero no ambos casos simultaneamente.

- Caso cobertura total y superpuesta.

Ejemplo: Todas las personas son empleados o clientes del banco, permitiendose que un empleado sea a su vez cliente.

- Caso cobertura parcial y exclusiva.

  Ejemplo: Hay personas, algunas de las cuales son empleados o clientes del banco, pero no ambas cosas simultáneamente.

- Caso cobertura parcial y superpuesta.

  Ejemplo: Algunas personas son empleados o clientes del banco, pudiendo ser ambas cosas.

2.3.2.2. Agregación

Concepto del todo y las partes.
Un todo se obtiene por la unión de diversas partes.

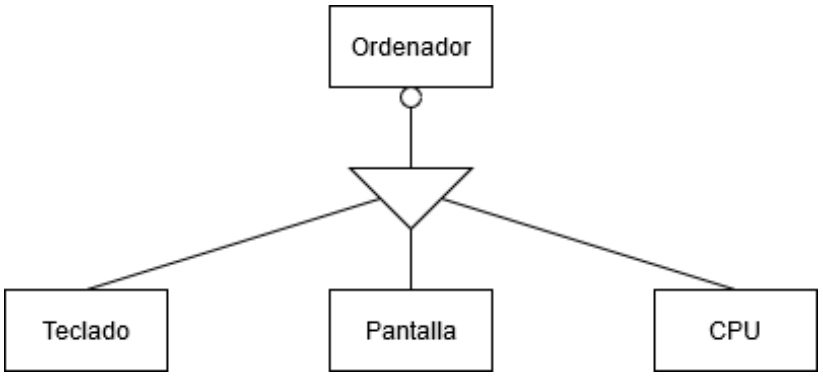


Figura 2.7: Ejemplo de Agregación

2.3.3. Resumen modelo relacional

2.3.3.1. Conceptos

- Modelo lógico
- Este modelo organiza y representa los datos en forma de tablas o relaciones.
- Se basa en el concepto de relación (Distinto a MER) item Informalmente: Relación = tabla

2.3.3.2. Atributo / Dominio / Ruta

ID	Nombre	Tarifa hr	Tipo oficio	ID Supervisor
1235	F.Aguilera	12,50	Electricista	1311
1412	A. Calvo	13,75	Fontanero	1540
2920	N. Marín	10,00	Carpintero	NULL

Tabla 2.12: Ejemplo Atributo/Dominio/Ruta

- **Atributo**: Elemento susceptible de tomar valores.
- **Dominio**: Conjunto de valores que puede tomar un atributo (se considera finito).
- **Tupla**: Cada uno de los elementos que contiene una instancia de la relación (filas)
- Cada relación debe tener una clave primaria.
- No se permiten atributos multivalores.
- No pueden haber relaciones N:M.

2.3.3.3. Transformaciones de las entidades

- **Atributos simples**

  Cada atributo del modelo entidad-relacional, será atributo del modelo relacional.

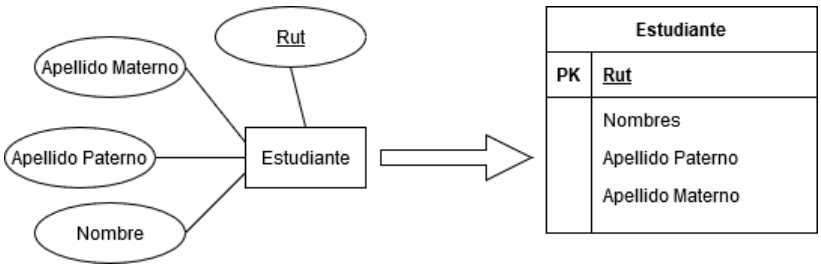


Figura 2.8: Atributo simple

- **Atributo multivaluado**

  Se debe crear una nueva tabla (descomponer) que contenga al atributo multivaluado junto al identificador de la entidad de la tabla al cual deriva.

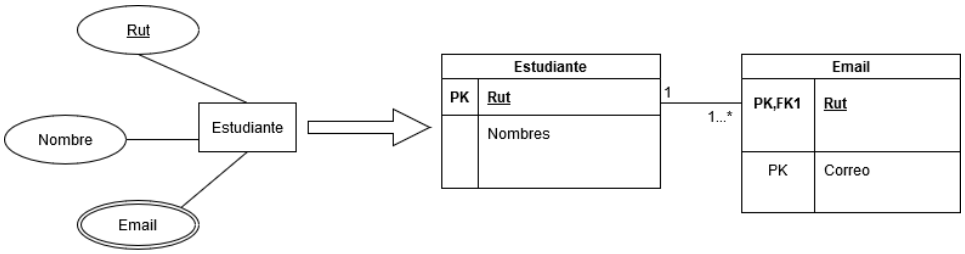


Figura 2.9: Atributo multivaluado

- **Atributo compuesto**

  1. Primera opción: Cada uno de los atributos que componen al atributo compuesto se incluirán en la tabla que pertenece.

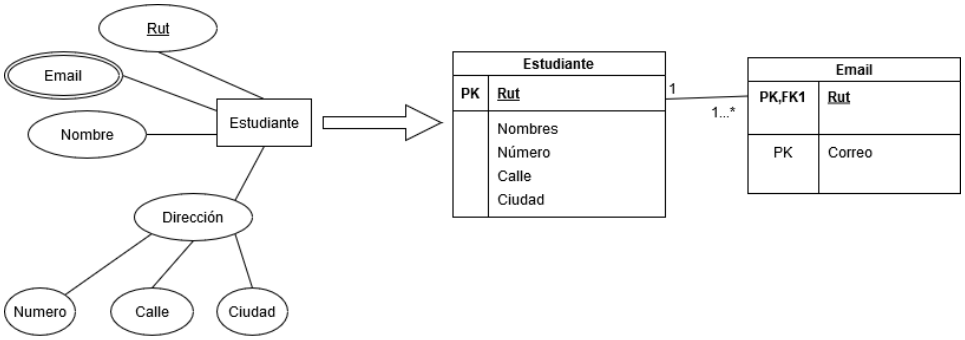


Figura 2.10: Atributo Compuesto opción 1

  2. Segunda opción: Se puede crear una tabla, donde la PK de esta corresponda a un ID del atributo compuesto, además de heredar la PK de la que contenía este atributo como FK.

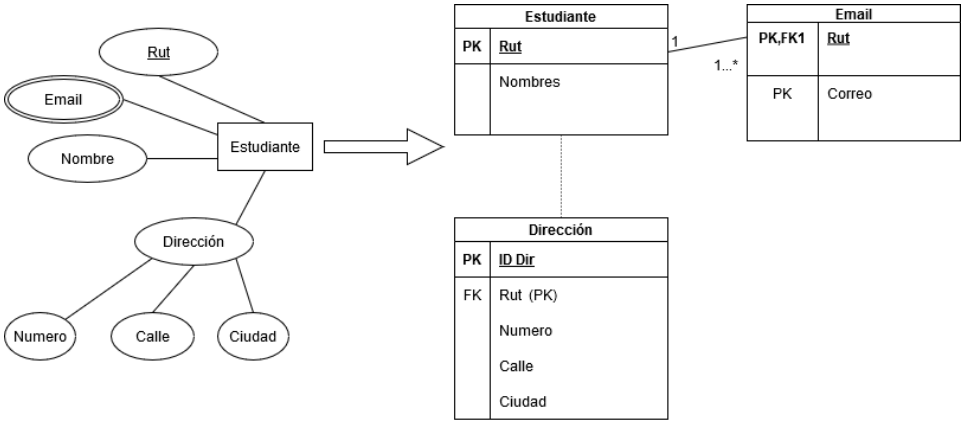


Figura 2.11: Atributo Compuesto opción 2

Hint: Si dirección hubiera sido compuesto y multievaluado, la única solución sería esta opción.

- **Asociación 1:N**.

  Se debe crear un atributo con el mismo dominio de la clave primaria en la entidad.

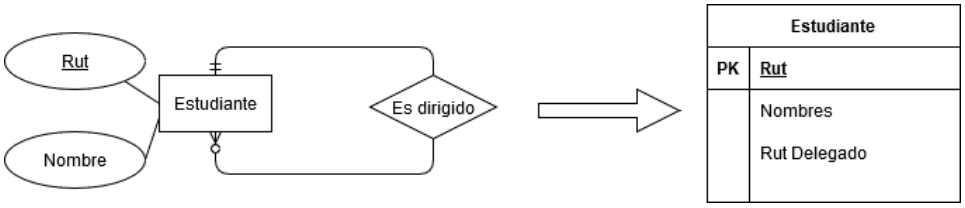


Figura 2.12: Asociación 1:N

- **Asociación N:M (con atributo)**

  Se crea una nueva tabla (vive con), la cual contendrá la clave primaria de la entidad padre (estudiante) y se debe crear un atributo que identifique la relación en cuestión, más los atributos que contenga esta asociación.

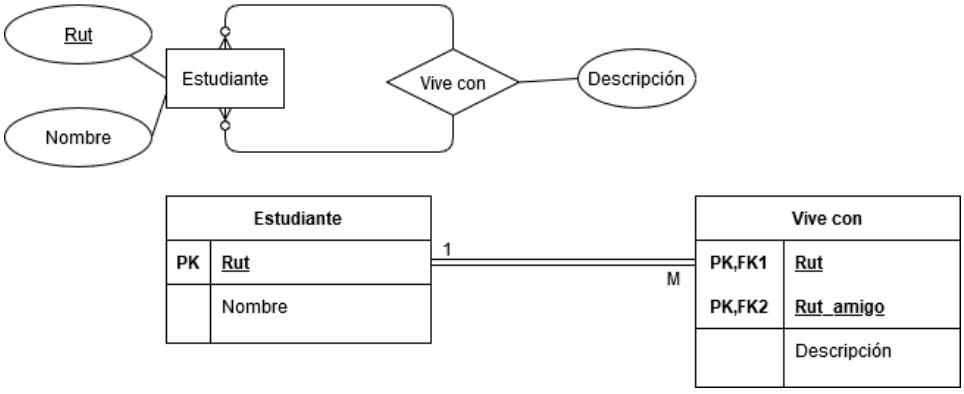


Figura 2.13: Asociación N:M

- **Asociación 1:1**

  1. Primera Opción: Se crea una nueva tabla (fuas) que contenga su atributo PK más el atributo foráneo de la tabla que 'tiene esta':

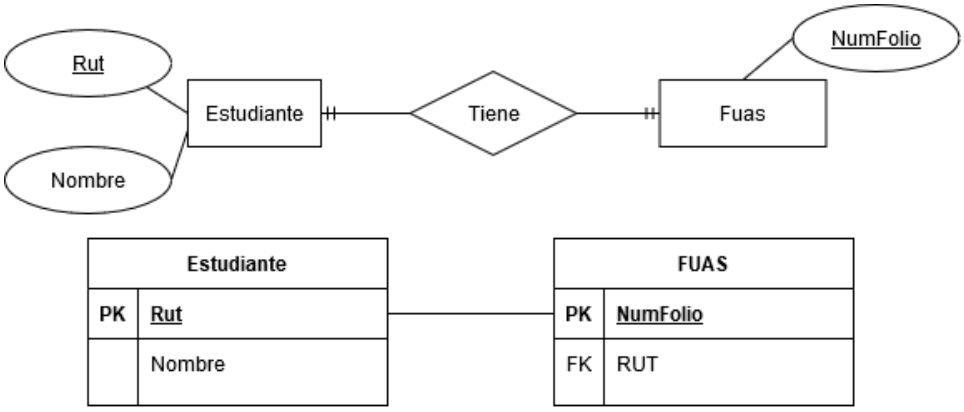


Figura 2.14: Asociación 1:1 - Primera opción

  2. Segunda opción: Se agrega el o los atributos que posee la entidad 'fuas' a la tabla estudiante.

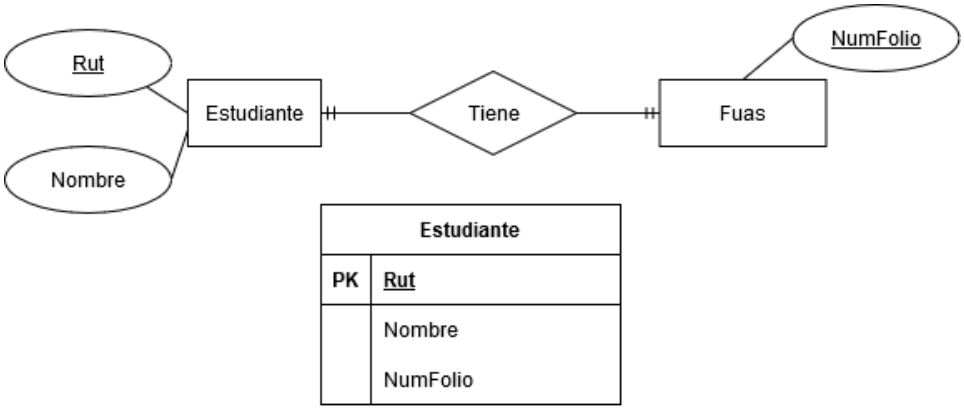


Figura 2.15: Asociación 1:1 - Segunda opción

- **Asociación 1:N (no débil)**

  El atributo que está en la asociación (semestre) deberá ser atributo de la tabla asignatura:

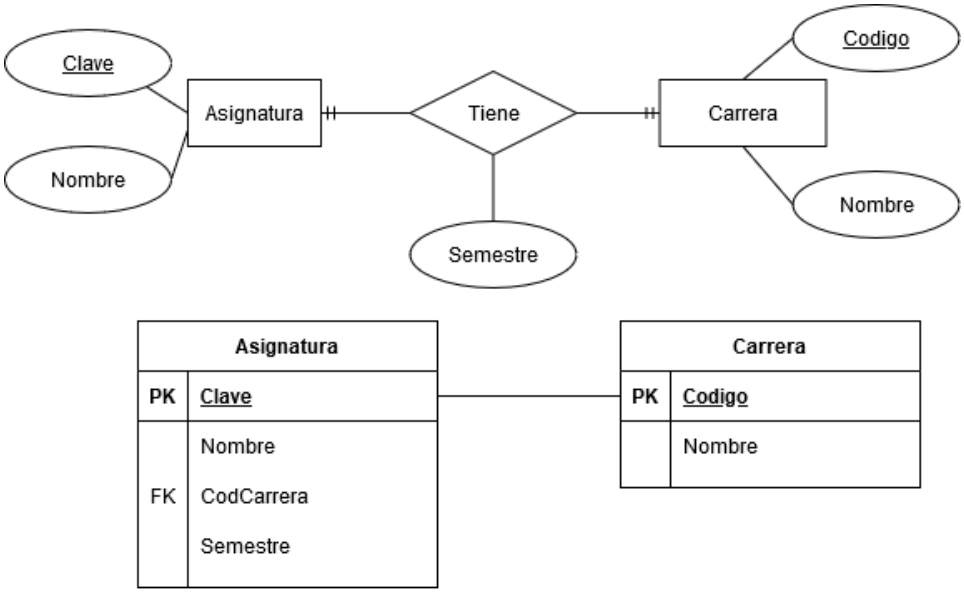


Figura 2.16: Asociación 1:N - No debil

- **Asociación N:M (Binaria)**

  Se cumple cuando hay una intersección entre dos entidades.

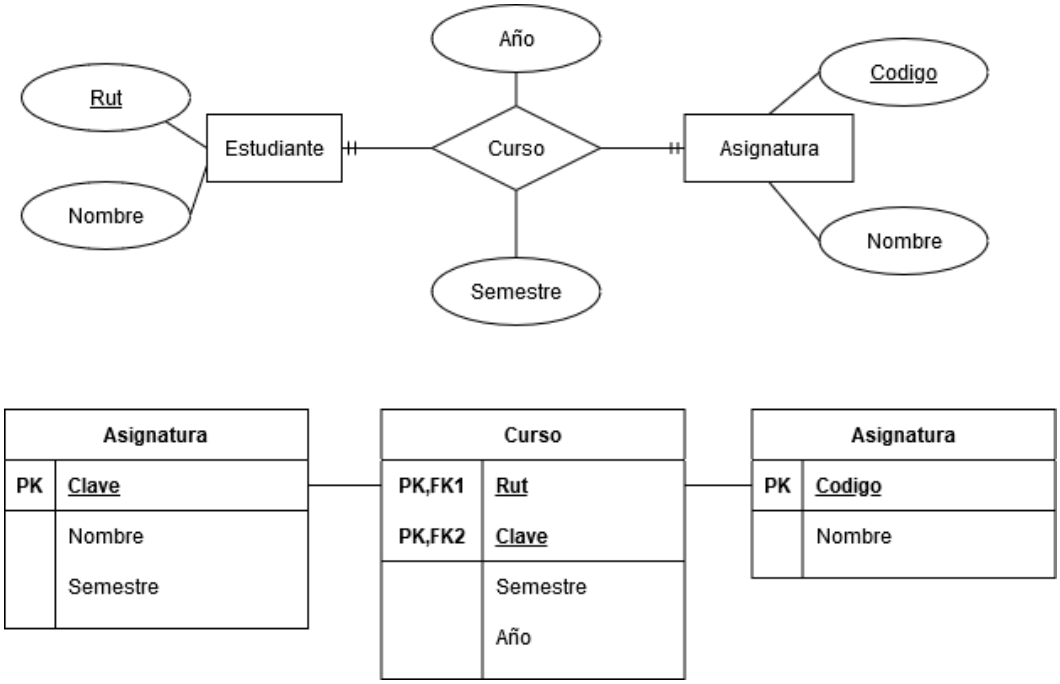


Figura 2.17: Asociación N:M - Binaria

- **Generalización ISA**

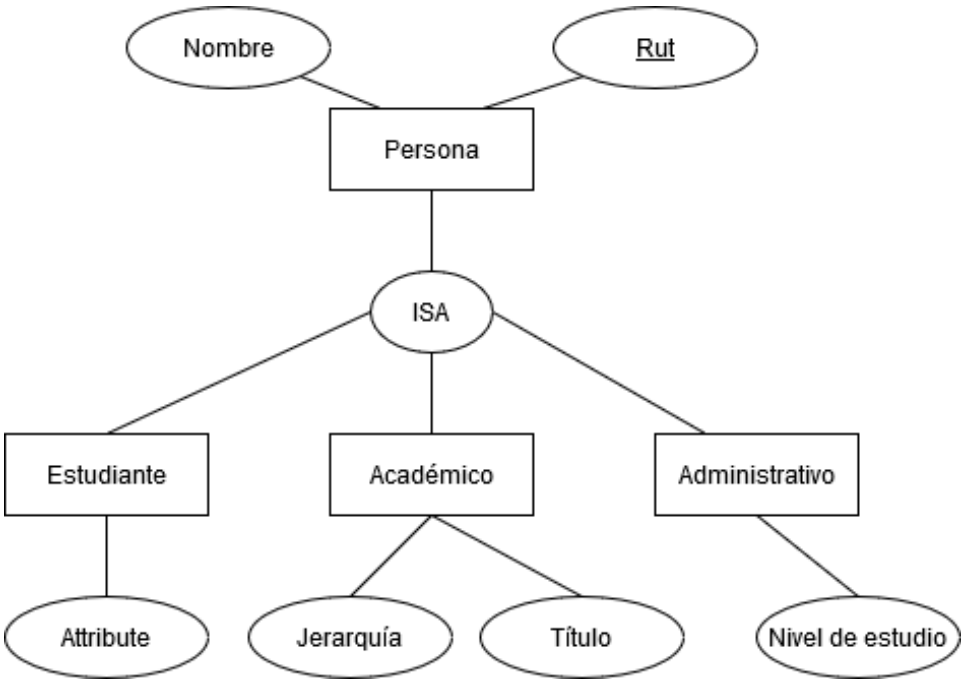


Figura 2.18: Generalización ISA

Da lo mismo si es excluyente o superpuesta (disyuntiva o solapada).

1. Opción 1: Se deberá crear una tabla por cada entidad en el MER.

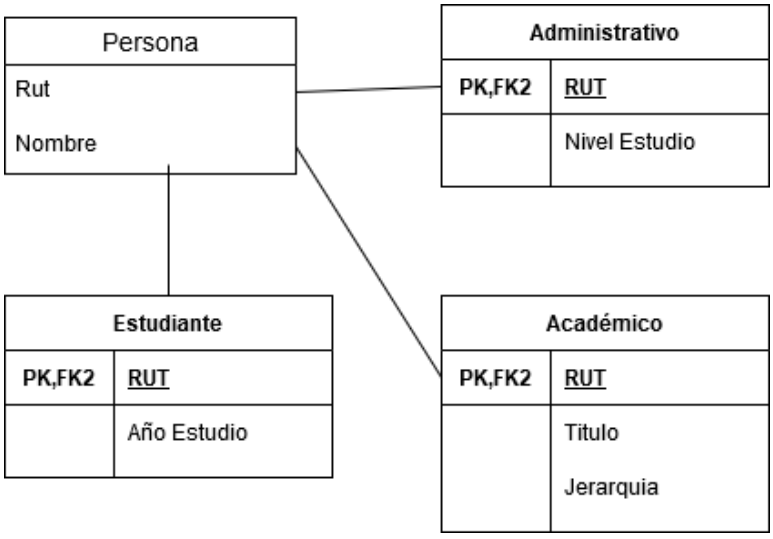


Figura 2.19: Generalización ISA - Opción 1

2. Opción 2: Se elimina la 'superclase' (persona) y se le agrega a cada subclase los atributos del padre.

Figura 2.20: Generalización ISA - Opción 2

- Si es ISA es excluyente (disyuntivo).
  Se crea una única tabla que contenga todos los atributos de la superclase y subclase, además de un atributo adicional que determine a que subclase pertenece los parámetros a ingresar (tipo).

Persona	
PK	Rut
	Nombre
	Año ingreso
	Título
	Jerarquía
	Nivel Estudio
	Tipo

Figura 2.21: ISA excluyente

- Si el ISA es superpuesto (solapado)
  Se crea una única tabla que contenga todos los atributos de la superclase y subclase, pero a diferencia de la anterior, no se debe crear un atributo adicional ya que esta puede aceptar datos nulos.

Persona	
PK	Rut
	Nombre
	Año ingreso
	Título
	Jerarquía
	Nivel Estudio

Figura 2.22: ISA superpuesta

2.4. Ayudantía n4:

2.4.1. Dependencias Funcionales (DF)

Concepto fundamental para la normalización de una relación. Son restricción de integridad sobre los datos. DF implica semántica intuitiva.

La dependencia funcional $x \rightarrow y$ asociada a una relación R indica que todas las tuplas de R que tengan un mismo valor en x deben tener un mismo valor en y.

x	y
x_1	y_{10}
x_2	y_5
x_3	y_6
x_4	y_{10}

Ejemplo:

Nombre	**Apellido**	**Función**	**Facultad**
Carlos	Zapata	Profesor	Escuela de sistemas
Ligia	Urrego	Investigador	Ingeniería Forestal
Carlos	Morales	Profesor	Matemáticas
Carlos	Zapata	Ingestigador	Escuela de sistemas

¿Cuáles de las siguientes afirmaciones son ciertas?

$nombre \rightarrow funcion$	NO	Ver 1° y 4° tupla
$nombre \rightarrow apellido$	NO	Ver 1° y 3° tupla
$nombre \rightarrow facultad$	NO	Ver 1° y 4° tupla
$(nombre, apellido) \rightarrow funcion$	NO	Ver 1° y 4° tupla
$(nombre, apellido) \rightarrow facultad$	SI	

2.4.2. Axiomas de Armstrong

Las axiomas de armstron son las siguientes:

1. Dependencia trivial:

   $A \rightarrow B \subset A$

2. Aumentación:

   $(A \rightarrow B)$ entonces $(A \cup C) \rightarrow (B \cup C)$

De manera más cómoda:

$(A \to B) \Rightarrow (AC) \to (BC)$

3. Transitividad

   $(A \to B) \wedge (B \to C) \Rightarrow (A \to C)$

4. Descomposición

   Si $(X \to YZ) \Rightarrow (X \to Y)$ y $(X \to Z)$

5. Unión

   Si $(X \to Y)$ y $(X \to Z) \Rightarrow (X \to YZ)$

6. Pseudotransitividad

   Si $(X \to Y)$ y $(WY \to Z) \Rightarrow (WX \to Z)$

2.4.3. Clausura de X bajo F(x+)

Dado F como un conjunto de dependencias funcionales, siendo x una de ellas se define x+ como el conjunto de atributos determinados funcionalmente por x.

Ejemplo:

Dado

EMP_PROY(_ID_E, _NUMEROP, NOMBREE, NOMBRERR, LUGARP)

F = {
_IDE –> NOMBREE ;
_NUMEROP –> NOMBRERR, LUGARP;
_IDE, _NUMEROP –> HORAS
}

_ID_E+ = { _ID_E, NOMBREE}

_NUMEROP+ = {_NUMEROP, NOMBRERR, LUGARP}

{_IDE, NUMEROP}+ = { _ID_E, _NUMEROP, NOMBREE, NOMBRERR, LUGAR}

2.4.4. Cobertura

Como determinamos si F cubre a E:

Para cada DF. $X \to Y \in E$, calculamos X+(F) y verificamos que X+ incluya los atributos en Y.

Ejemplo

Sea:

- $F = \{AB \to C, B \to D, D \to GC, CG \to H\}$
- $F_1 = \{D \to H, B \to C, AD \to GH\}$

¿F_1 cubre a F? No.

$$AB \to C$$
$$AB + (F_1) = \{A, B, C\} \quad \text{Contiene a C}$$

$$B \to D$$
$$B + (F_1) = \{B, C\} \quad \text{No contiene a D}$$

¿F cubre a F_1 ? Si.

$$D \to H$$
$$D + (F) = \{D, C, G, H\} \quad \text{Contiene a H}$$

$$B \to C$$
$$B + (F) = \{B, D, C, G, H\} \quad \text{Contiene a C}$$

$$AD \to GH$$
$$AD + (F) = \{A, D, C, G, H\} \quad \text{Contiene a GH}$$

2.4.5. Cobertura minima de F

Una cobertura mínima de F es un conjunto mínimo F min es equivalente a F. F es minimal si solo si:

- Toda DF en F tiene un solo atributo a la derecha.
- No podemos reemplazar ninguna DF $x \to A \in F$ por una DF $Y \to A$, donde $Y \subset X$, y seguir teniendo un conjunto de DF's equivalente a F.

Ejemplo:

$$DF : F = \{AB \to C, B \to D, D \to GC, CG \to H\ \}$$

Paso a Paso

1. Dejar solo un atributo a la derecha
   - $AB \to C$
   - $B \to D$
   - $D \to G$
   - $D \to C$
   - $CG \to H$
2. Eliminar redundancia a la izquierda. Analizar $AB \to C$ y $CG \to H$

$AB \to C$

A+ = {A}	No contiene a C, por tanto B no es redundante.
B+ = {B,D,C,G,H}	Contiene a C, por tanto A es redundante

Se elimina la dependencia funcional $AB \to C$.

$CG \to H$

C+ = {C}	No contiene a H, por tanto G no es redundante.
G+ = {G}	No contiene a H, por tanto C no es redundante.

3. Eliminar redundancia a la derecha.

F sin $B \to D$

B+ = {B}	No contiene a D, por tanto $B \to D$ no es redundante

F sin $D \to G$

D+ = {D,C}	No contiene a G, por tanto $B \to G$ no es redundante

F sin $D \to C$

D+ = {D,G}	No contiene a C, por tanto $D \to C$ no es redundante

F sin $CG \to H$

CG+ = {C,G}	No contiene a H, por tanto $CG \to H$ no es redundante

4. Finalmente, una cobertura mínima de $F = F_{min}$

$$F_{min} \text{ es } \{B \to D, D \to G, D \to C, CG \to H\ \}$$

2.5. Ayudantia n5:

2.5.1. Formas normales

Si una relación cumple una forma normal n, también cumple con las (n-1) formas normales. Mientras más alta la forma normal, es mucho mejor:

- 1NF, 2NF, 3NF: relaciones con 1 clave candidata.
- BCNF: relaciones con más de 1 clave candidata. (4NF, 5NF, 6NF)

2.5.1.1. Primera forma normal

Todos los atributos son atómicos (no hay multivalores).

Rol alumno	Nombre	Apellido	Teléfono
123	Alicia	Díaz	056-2-2586495
456	Juan	Perez	056-34-245678 056-34-245680
789	Pedro	González	056-58-233055

Tabla 2.13: Ejemplo Primera forma normal

Problema:
Asumiendo que la columna 'telefono' está definido en algún de n° telefónico, la representación de arribo no está en 1FN. La 1FN prohíbe a un campo contener más de un valor de su dominio de columna.
Solución:
Si tiene atributos multivaluados, se debe crear una nueva sub-relación.

Alumno(RolAlumno, nombre, apellido)
AlumnoFono(RolAlumno,telefono)

Note que ambos atributos quedan como PK.

2.5.1.2. Segunda forma normal

Eliminar las dependencias parciales: Parte de la clave determina un atributo no clave. Una relación está en 2NF si y solo si:

- Está en 1NF

- Todos los atributos no clave (si los hay) dependen funcionalmente por completo de la PK.

Cuando una relación R está en 1FN, y no tiene una clave compuesta, se encuentra automáticamente en 2FN.

Ejemplo.

Suponga la siguiente relación.

EMPLEADO(EMPLEADO,HABILIDAD, LUGAR_TRABAJO)

Con la siguiente dependencia funcional.

F = {
EMPLEADO, HABILIDAD –> LUGAR_TRABAJO;
EMPLEADO –> LUGAR_TRABAJO
}

VIOLA 2NF!!

Solución:

Se crea una nueva relación con la dependencia parcial. La PK de esa nueva relación serpa los atributos de la izquierda de la DF.

EMPLEADOHABILIDAD(EMPLEADO,HABILIDAD)

Empleado	**Habilidad**
Perez	Docencia
Perez	Investigación
Perez	Coordinación
Bravo	Limpieza ligera
Muñoz	Alquimia

EMPLEADO(EMPLEADO,LUGAR_TRABAJO)

Empleado	**Lugar de trabajo**
Perez	AV España 2250
Bravo	Av. Vicuña Mackenna 3864
Muñoz	Av. Vicuña Mackenna 3865
Herrera	Av. Vicuña Mackenna 3867

2.5.1.3. Tercera forma normal

Elimina las dependencias transitivas: Atributos no clave dependen de otros no clave. Una relación está en 3FN si y solo si:

- Está en 2FN
- Los atributos no clave (si no hay) no dependen de otros atributos no claves.

Problema:

R(torneo,año, ganador, fecha_nacimiento)

(torneo,año) → ganador
(ganador) → fecha_nacimiento

Torneo	**Año**	**Ganador**	Fecha Nac
Roland Garros	2010	Roger Federer	08/08/1981
Wimbledon	2011	Pete Sampras	18/08/1971
Open USA	2011	Roger Federer	08/08/1981
Roland Garros	2012	Rafael Nadal	03/06/1986

Solución:

- Para no violar 3FN, se crea (ganador, fecha_nac)

Ganador	**Fecha de nac**
Roger Federer	08/08/1981
Pete Sampras	12/08/1971
Rafael Nadal	03/06/1986

- La relación original queda:

(torneo,año,ganador)

Torneo	**Año**	**Ganador**
Roland Garros	2010	Roger Federer
Wimbledon	2011	Pete Sampras
Open USA	2011	Roger Federer
Roland Garros	2012	Rafael Nadal

Capítulo 3

Talleres de Bases de datos

3.1. Taller n1

1. Cree el diagrama Modelo Entidad relación con el set de datos entregado.

    - Utilice la interpretación de datos para poder entender y relacionar.
    - Modelar las diferentes entidades con sus atributos
    - En cada caso indicar el dominio de cada atributo y si es compuesto y/o multivaluado. Además, identificar la/las claves de cada entidad.
    - Utilizar la herramienta PgModeler de PostgreSQL.

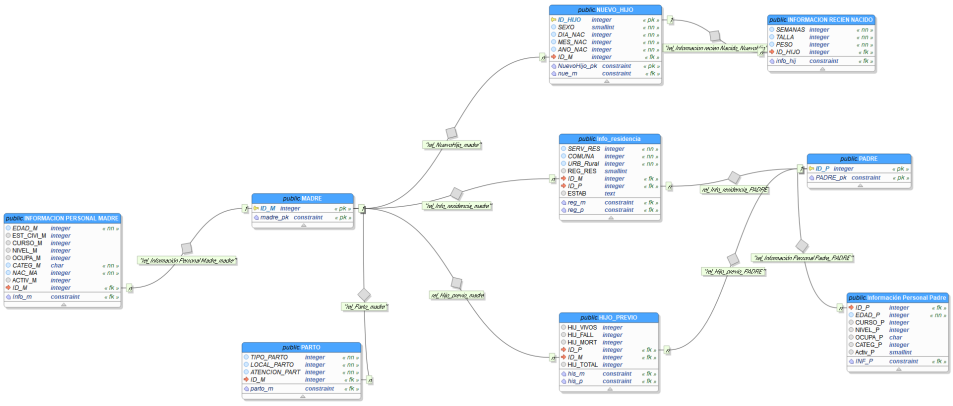


Figura 3.1: Modelo relacional propuesto

3.2. Taller n2

3.2.1. Modelo Entidad Relación

A continuación se expone el diagrama Entidad - Relación, resultado de las consideraciones mencionadas durante las clases, en caso de tener problemas al visualizar alguna de las figuras de este informe, junto con este informa se han adjuntado todas ellas en una mejor resolución.

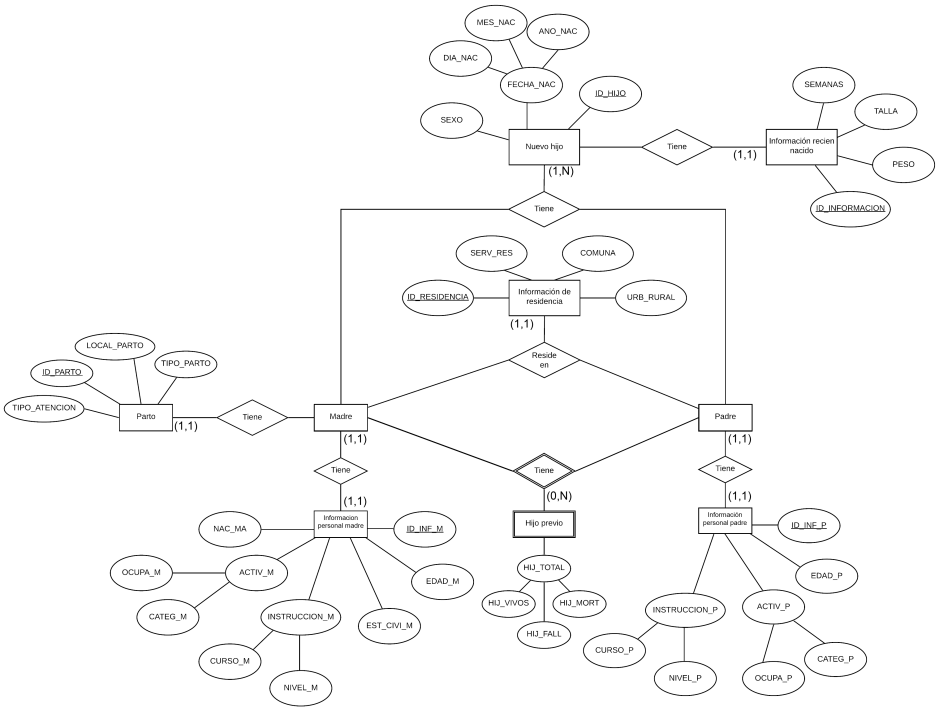


Figura 3.2: Modelo Relacional

3.2.1.1. Captura de pgModeler

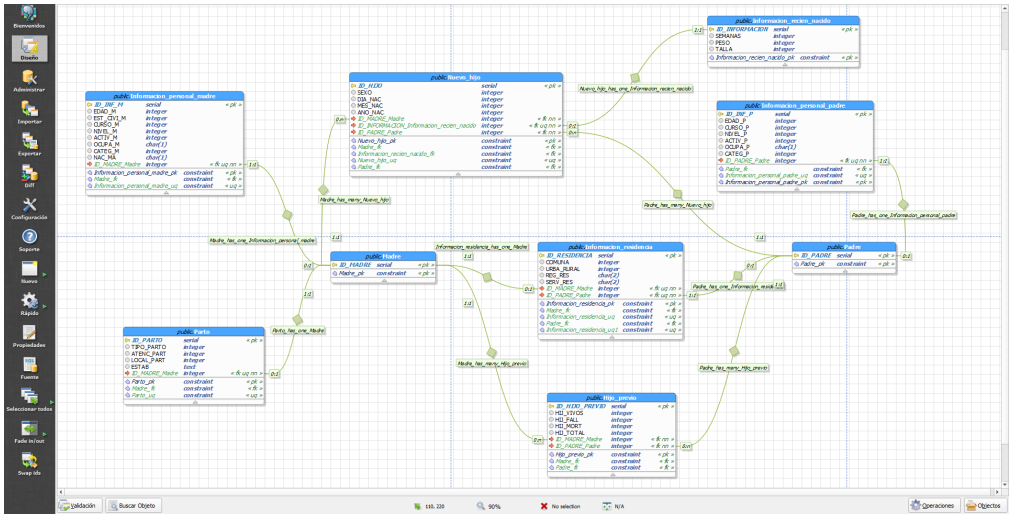


Figura 3.3: Captura de Pantalla pgModeler

3.2.1.2. Diagrama Relacional

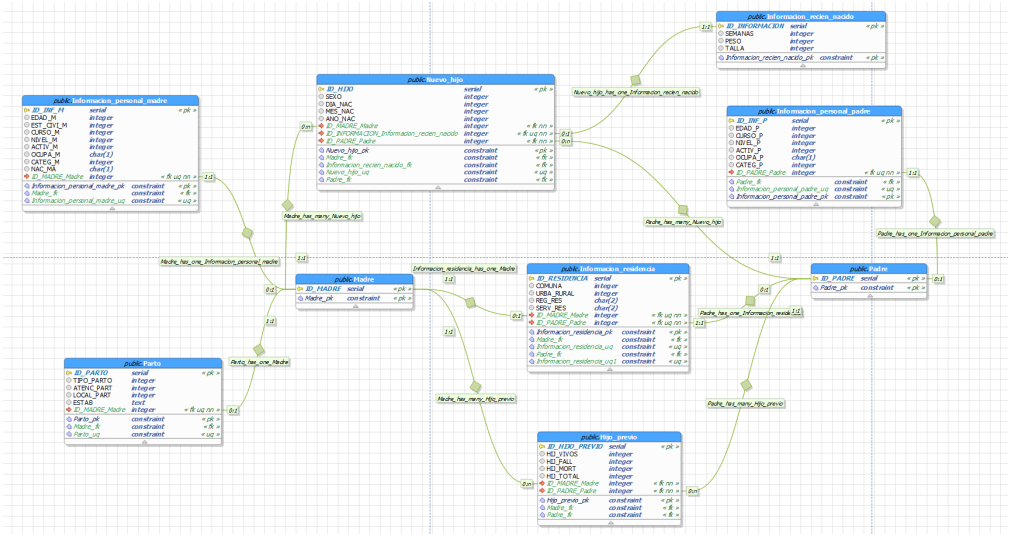


Figura 3.4: Modelo Relacional diseñado con pgModeler

3.2.1.3. Código Generado - Tablas

Tabla "Información recien nacido"

```
CREATE TABLE "Informacion_recien_nacido"(
        "ID_INFORMACION" serial NOT NULL,
        "SEMANAS" integer,
        "PESO" integer,
        "TALLA" integer,
        CONSTRAINT "Informacion_recien_nacido_pk"
        PRIMARY KEY ("ID_INFORMACION")
);

COMMENT ON COLUMN
"Informacion_recien_nacido"."PESO" IS 'PESO_EN_GRAMOS';

COMMENT ON COLUMN
"Informacion_recien_nacido"."TALLA" IS 'TALLA_EN_CM';
```

Tabla "Padre"

```
CREATE TABLE "Padre"(
        "ID_PADRE" serial NOT NULL,
        CONSTRAINT "Padre_pk" PRIMARY KEY ("ID_PADRE")
);
```

Tabla "Informacion personal padre"

```
CREATE TABLE "Informacion_personal_padre"(
        "ID_INF_P" serial NOT NULL,
        "EDAD_P" integer,
        "CURSO_P" integer,
        "NIVEL_P" integer,
        "ACTIV_P" integer,
        "OCUPA_P" char(1),
        "CATEG_P" integer,
        "ID_PADRE_Padre" integer NOT NULL,
        CONSTRAINT "Informacion_personal_padre_pk"
        PRIMARY KEY ("ID_INF_P")
);

COMMENT ON COLUMN "Informacion_personal_padre"."NIVEL_P"
IS '1= Superior
2= Medio
3= Secundario
4= Basico o Primario
5= Ninguno';

COMMENT ON COLUMN "Informacion_personal_padre"."ACTIV_P"
IS '0= Inactivo
 1= Activo
 2= Cesante o Desocupado
 3= Ignorado';

COMMENT ON COLUMN "Informacion_personal_padre"."OCUPA_P"
IS 'Actividad=0
 2= Labores de casa
```

```
3=␣Estudiante
4=␣Rentista
5=␣Jubilado
6=␣Invalido␣o␣recluido
7=␣Otros
8=␣Ninguno
Actividad␣=1␣(Gran␣Grupo␣de␣Actividad,␣CIUO␣88)
0=␣Fuerza␣armadas␣y␣del␣orden
1=␣Miembros␣del␣poder␣judicial
2=␣Profesionales,␣Cientificos␣o␣Intelectuales
3=␣Tecnicos␣y␣profesionales␣medios
4=␣Empleados␣de␣oficina
5=␣Trabajadores␣de␣servicios␣y␣vendedores
de␣Comercio␣y␣mercado
6=␣Agricultores␣y␣trabajadores␣calificados
agropecuarios␣y␣pesqueros
7=␣Oficiales,␣operarios␣y␣artesanos␣de␣artes
mecanicas␣y␣otros␣oficios
8=␣Operadores␣De␣instalacion,␣maquinas␣y␣montadores
9=␣Trabajadores␣no␣calificados
Actividad␣=2
X=␣Cesante␣o␣desocupado
Actividad␣=9
X=Ignorada';

COMMENT ON COLUMN "Informacion_personal_padre"."CATEG_P"
IS '␣Actividad␣=0
␣0=␣Inactivo
␣Actividad␣=1
␣1=␣Patron
␣2=␣Empleado
␣3=␣Obrero
␣4=␣Trabajador␣por␣cuenta␣propia
␣9=␣Ignorado
␣Actividad=2
␣9=␣Ignorado
␣Actividad=␣9
␣9=␣Ignorado';
```

Tabla "Madre"

```
CREATE TABLE "Madre"(
        "ID_MADRE" serial NOT NULL,
        CONSTRAINT "Madre_pk" PRIMARY KEY ("ID_MADRE")
);
```

Tabla "Informacion personal madre"

```
CREATE TABLE "Informacion_personal_madre"(
        "ID_INF_M" serial NOT NULL,
        "EDAD_M" integer,
        "EST_CIVI_M" integer,
        "CURSO_M" integer,
        "NIVEL_M" integer,
        "ACTIV_M" integer,
        "OCUPA_M" char(1),
        "CATEG_M" integer,
        "NAC_MA" char(1),
        "ID_MADRE_Madre" integer NOT NULL,
        CONSTRAINT "Informacion_personal_madre_pk"
        PRIMARY KEY ("ID_INF_M")
);

COMMENT ON COLUMN "Informacion_personal_madre"."EST_CIVI_M"
IS '1= Soltera
2= Casada
3= Viuda
4= Divorciado
5= Separado judicialmente
 6= Conviviente civil
 9= Ignorado';

COMMENT ON COLUMN "Informacion_personal_madre"."NIVEL_M"
IS '1= Superior
```

```
2=␣Medio
3=␣Secundario
4=␣Basico␣o␣Primario
5=␣Ninguno';

COMMENT ON COLUMN "Informacion_personal_madre"."ACTIV_M"
IS '0=␣Inactivo
1=␣Activo
2=␣Cesante␣o␣Desocupado
9=␣Ignorado';

COMMENT ON COLUMN "Informacion_personal_madre"."CATEG_M"
IS 'Actividad␣=0
␣0=␣Inactivo
␣Actividad␣=␣1
␣1=␣Patron
␣2=␣Empleado
␣3=␣Obrero
␣4=␣Trabajador␣por␣cuenta␣propia
␣9=␣Ignorado
␣Actividad=2
␣9=␣Ignorado
␣Actividad=9
␣9=␣Ignorado␣';
```

Tabla "Nuevo hijo"

```
CREATE TABLE "Nuevo_hijo"(
        "ID_HIJO" serial NOT NULL,
        "SEXO" integer,
        "DIA_NAC" integer,
        "MES_NAC" integer,
        "ANO_NAC" integer,
        "ID_MADRE_Madre" integer NOT NULL,
        "ID_INFORMACION_Informacion_recien_nacido" integer NOT NULL,
        "ID_PADRE_Padre" integer NOT NULL,
        CONSTRAINT "Nuevo_hijo_pk" PRIMARY KEY ("ID_HIJO")
);
```

Tabla "Informacion residencia"

```
CREATE TABLE "Informacion_residencia"(
        "ID_RESIDENCIA" serial NOT NULL,
        "COMUNA" integer,
        "URBA_RURAL" integer,
        "REG_RES" char(2),
        "SERV_RES" char(2),
        "ID_MADRE_Madre" integer NOT NULL,
        "ID_PADRE_Padre" integer NOT NULL,
        CONSTRAINT "Informacion_residencia_pk"
        PRIMARY KEY ("ID_RESIDENCIA")
);
```

Tabla "Hijo previo"

```
CREATE TABLE "Hijo_previo"(
        "ID_HIJO_PREVIO" serial NOT NULL,
        "HIJ_VIVOS" integer,
        "HIJ_FALL" integer,
        "HIJ_MORT" integer,
        "HIJ_TOTAL" integer,
        "ID_MADRE_Madre" integer NOT NULL,
        "ID_PADRE_Padre" integer NOT NULL,
        CONSTRAINT "Hijo_previo_pk" PRIMARY KEY ("ID_HIJO_PREVIO")
);
```

Tabla "Parto"

```
CREATE TABLE "Parto"(
        "ID_PARTO" serial NOT NULL,
        "TIPO_PARTO" integer,
        "ATENC_PART" integer,
        "LOCAL_PART" integer,
        "ESTAB" text,
        "ID_MADRE_Madre" integer NOT NULL,
        CONSTRAINT "Parto_pk" PRIMARY KEY ("ID_PARTO")
);


COMMENT ON COLUMN "Parto"."TIPO_PARTO" IS '1= Simple
2= Doble
 3= Triple
 4= Otro
 9=Ignorado';

COMMENT ON COLUMN "Parto"."ATENC_PART" IS '1=Medico
 2=Matrona
 3=Sin Atencion Profesional
 4=Otro Personal de salud
 9= Desconocido';

COMMENT ON COLUMN "Parto"."LOCAL_PART" IS '1=Hospital
 2=Casa Habitacion
 3=Otro
 9= Ignorado';
```

Para evitar confusiones cabe mencionar que tanto los Constraint Layout como los Commentary se encuentran dentro del código de creación de cada una de las tablas.

3.3. Taller n3

3.3.1. Pregunta 1

Transformar fichero a CSV UTF-8 Utilizaremos la instruccion COPY FROM para cargar los datos del fichero csv en las tablas de PostgreSQL.

Ejemplo de la instrucción:

```
COPY persons(first_name,last_name,dob,email)
FROM 'C:\tmp\persons.csv' DELIMITER ',' CSV HEADER;
```

Es importante guardar el archivo de datos csv, en un nuevo .csv que contenga formato 'UTF-8' y delimitado por comas, para que la instrucción SQL pueda ser exitosa.

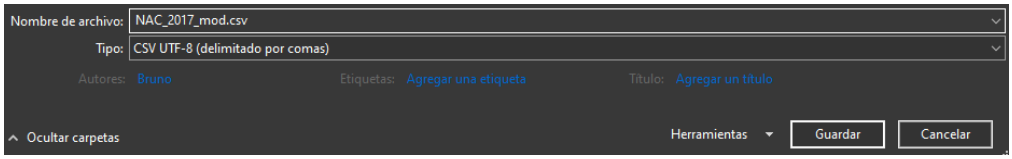


Figura 3.5: Guardar fichero .csv con codificación UTF-8 delimitado por comas

3.3.2. Script - SQL

A continuación, se detalla la creación de una tabla temporal, con los atributos a agregar desde el fichero CSV.

Los atributos serán almacenados como una cadena de caracteres en esta tabla temporal, y posteriormente serán asignados a las tablas correspondientes a dichos atributos.

```
CREATE TABLE temp_table(
        id serial NOT NULL,
        SEXO character varying(50),
        DIA_NAC character varying(50),
        MES_NAC character varying(50),
        ANO_NAC character varying(50),
        TIPO_PARTO character varying(50),
        ATENC_PART character varying(50),
        LOCAL_PART character varying(50),
        SEMANAS character varying(50),
        PESO character varying(50),
        TALLA character varying(50),
        EDAD_P character varying(50),
        CURSO_P character varying(50),
        NIVEL_P character varying(50),
        ACTIV_P character varying(50),
        OCUPA_P character varying(50),
        CATEG_P character varying(50),
        EDAD_M character varying(50),
        EST_CIVIL_M character varying(50),
        CURSO_M character varying(50),
        NIVEL_M character varying(50),
        ACTIV_M character varying(50),
        OCUPA_M character varying(50),
        CATEG_M character varying(50),
        COMUNA character varying(50),
        URBA_RURAL character varying(50),
        HIJ_VIVOS character varying(50),
        HIJ_FALL character varying(50),
        HIJ_MORT character varying(50),
        HIJ_TOTAL character varying(50),
```

```
        REG_RES character varying(50),
        SERV_RES character varying(50),
        ESTAB character varying(50),
        NAC_MA character varying(50),
        CONSTRAINT temp_table_pk PRIMARY KEY (id)
);
```

Las tablas son cargadas

```
COPY temp_table(
                    SEXO,
                    DIA_NAC,
                    MES_NAC,
                    ANO_NAC,
                    TIPO_PARTO,
                    ATENC_PART,
                    LOCAL_PART,
                    SEMANAS,
                    PESO,
                    TALLA,
                    EDAD_P,
                    CURSO_P,
                    NIVEL_P,
                    ACTIV_P,
                    OCUPA_P,
                    CATEG_P,
                    EDAD_M,
                    EST_CIVIL_M,
                    CURSO_M,
                    NIVEL_M,
                    ACTIV_M,
                    OCUPA_M,
                    CATEG_M,
                    COMUNA,
                    URBA_RURAL,
                    HIJ_VIVOS,
                    HIJ_FALL,
                    HIJ_MORT,
                    HIJ_TOTAL,
                    REG_RES,
                    SERV_RES,
```

```
                    ESTAB,
                    NAC_MA
                    )
FROM 'C:\NAC_2017_mod.csv' DELIMITER ';' CSV HEADER;
```

3.3.3. Asignar datos a tablas

3.3.3.1. Informacion recien nacido

```
/* Informacion recien nacido */
insert into "Informacion_recien_nacido" ("ID_INFORMACION",
                                         "PESO" ,
                                         "SEMANAS" ,
                                         "TALLA" )
select
    id,
    cast(PESO as INTEGER),
    cast(SEMANAS as INTEGER),
    cast(TALLA as INTEGER)
from temp_table;
```

3.3.3.2. Padre

```
/* Padre */
insert into "Padre" ("ID_PADRE") select id from temp_table;
```

3.3.3.3. Información del Padre

```
/* Informacion del padre */
INSERT INTO "Informacion_personal_padre"("ID_INF_P",
                                         "EDAD_P",
                                         "CURSO_P",
                                         "NIVEL_P",
                                         "ACTIV_P",
                                         "CATEG_P",
                                         "ID_PADRE_Padre",
                                         "OCUPA_P")
select
    id,
    cast(EDAD_P as INTEGER),
    CAST(CURSO_P as INTEGER),
    CAST(NIVEL_P as INTEGER),
```

```
        CAST(ACTIV_M as INTEGER),
        CAST(CATEG_M as INTEGER),
        id,
        CAST(OCUPA_P as CHAR(2))
    FROM temp_table;
```

3.3.3.4. Madre

```
    /* Creacion de la madre */
    insert into "Madre"("ID_MADRE") select id from temp_table;
```

3.3.3.5. Información de la Madre

```
    /* Informacion de la madre */
    INSERT INTO "Informacion_personal_madre" ("ID_INF_M",
                                              "EDAD_M",
                                              "EST_CIVI_M",
                                              "CURSO_M",
                                              "NIVEL_M",
                                              "ACTIV_M",
                                              "OCUPA_M",
                                              "CATEG_M",
                                              "NAC_MA",
                                              "ID_MADRE_Madre")
    select
        id,
        CAST(EDAD_M as INTEGER),
        CAST(EST_CIVIL_M AS INTEGER),
        cast(CURSO_M as INTEGER),
        cast(NIVEL_M as INTEGER),
        cast(ACTIV_M as INTEGER),
        cast(OCUPA_M as CHAR(1)),
        cast(CATEG_M as INTEGER),
        cast(NAC_MA as CHAR(1)),
        id
    from temp_table;
```

3.3.3.6. Parto

```
/* Parto */
insert into "Parto" ("ID_PARTO" ,
                     "TIPO_PARTO",
                     "ATENC_PART",
                     "LOCAL_PART",
                     "ESTAB",
                     "ID_MADRE_Madre" )
select
    id,
    CAST(TIPO_PARTO as INTEGER),
    CAST(ATENC_PART as INTEGER),
    CAST(LOCAL_PART as INTEGER),
    CAST(ESTAB as TEXT),
    id
from temp_table;
```

3.3.3.7. Nuevo Hijo

```
/* Nuevo hijo */
insert into "Nuevo_hijo" ("ID_HIJO",
                     "SEXO",
                     "DIA_NAC",
                     "MES_NAC",
                     "ANO_NAC",
                     "ID_MADRE_Madre",
                     "ID_INFORMACION_Informacion_recien_nacido",
                     "ID_PADRE_Padre" )
select
    id,
    CAST(SEXO as INTEGER),
    CAST(DIA_NAC as INTEGER),
    CAST(MES_NAC as INTEGER),
    CAST(ANO_NAC as INTEGER),
    id,
    id,
    id
```

```
from temp_table;
```

3.3.3.8. Informacion de residencia

```
insert into "Informacion_residencia" ("ID_RESIDENCIA",
                                      "COMUNA",
                                      "URBA_RURAL",
                                      "REG_RES",
                                      "SERV_RES",
                                      "ID_MADRE_Madre",
                                      "ID_PADRE_Padre" )
select
    id,
    CAST(COMUNA as INTEGER),
    CAST(URBA_RURAL as INTEGER),
    CAST(REG_RES as INTEGER),
    CAST(SERV_RES as INTEGER),
    id,
    id
from temp_table;
```

3.3.3.9. Hijo Previo

```
insert into "Hijo_previo" ("ID_HIJO_PREVIO",
                           "HIJ_VIVOS",
                           "HIJ_FALL",
                           "HIJ_MORT",
                           "HIJ_TOTAL",
                           "ID_MADRE_Madre",
                           "ID_PADRE_Padre" )
select
    id,
    CAST(HIJ_VIVOS as INTEGER),
    CAST(HIJ_FALL as INTEGER),
    CAST(HIJ_MORT as INTEGER),
    CAST(HIJ_TOTAL as INTEGER),
    id,
```

```
    id
from temp_table;
```

3.3.3.10. Eliminar tabla temporal

```
-- Finalizar dropeando la tabla temporal
drop table temp_table ;
```

3.3.4. Capturas del Pantalla

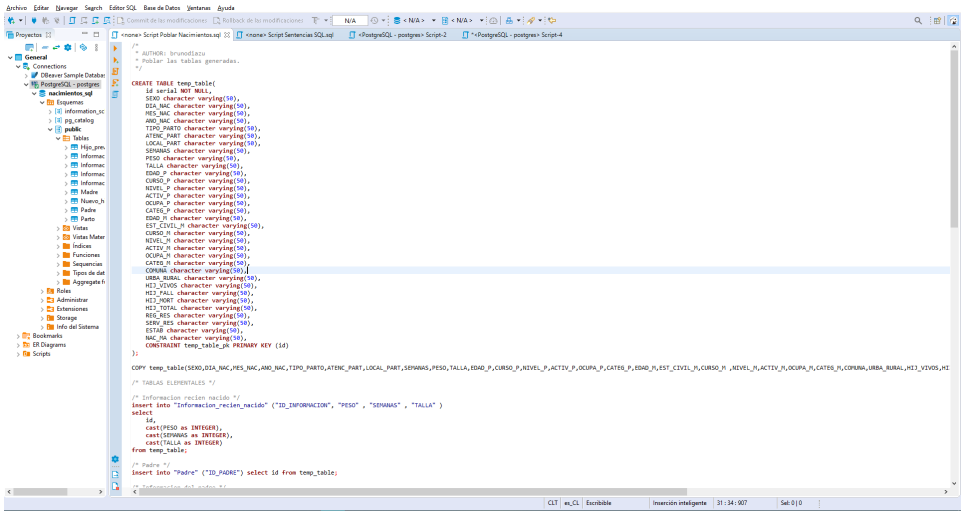


Figura 3.6: IDE DBeaver para la ejecución de sentencias SQL

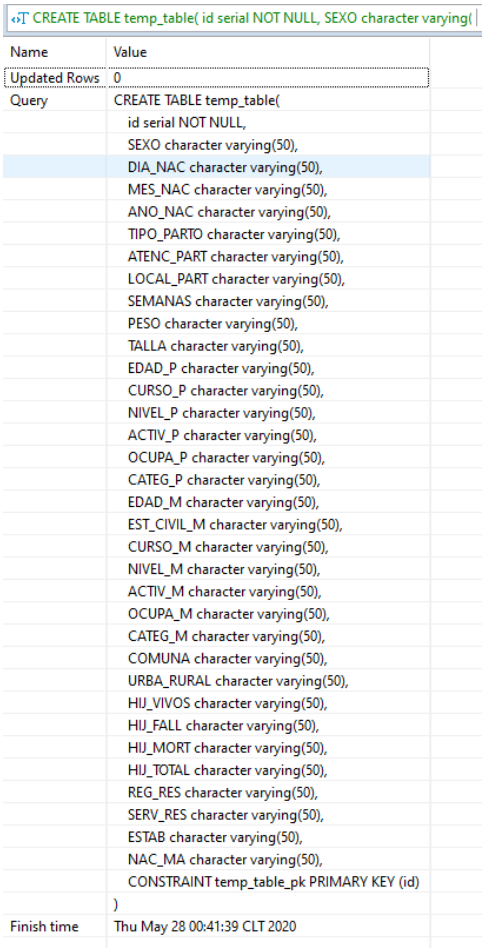


Figura 3.7: Resultado al ejecutar sentencia para crear tabla temporal

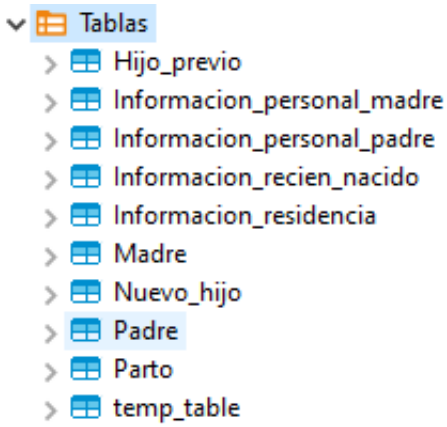


Figura 3.8: Resultado al ejecutar sentencia para crear tabla temporal (Vista Arbol)

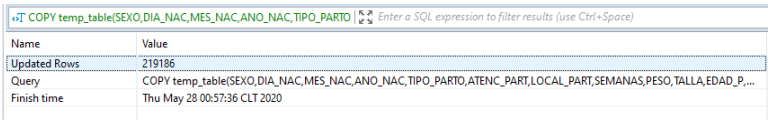


Figura 3.9: Resultado al ejecutar sentencia para poblar tabla temporal

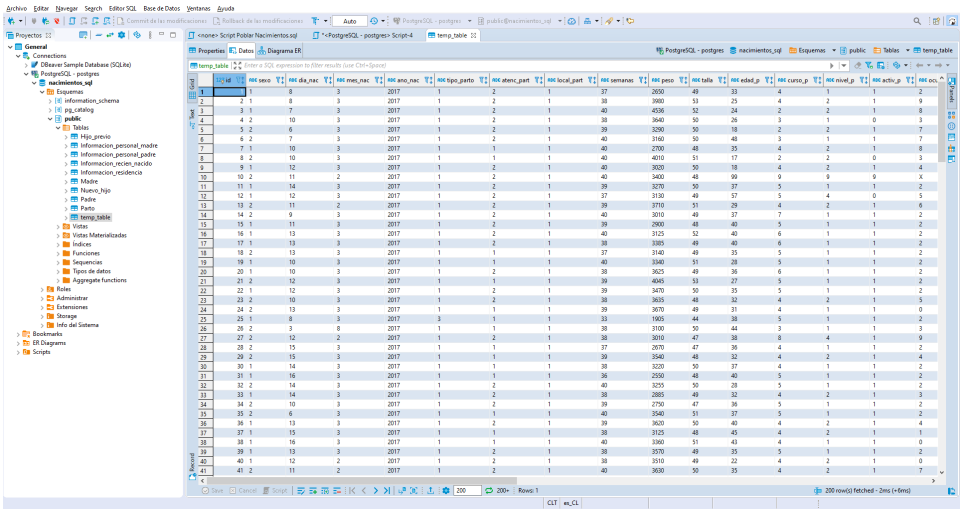


Figura 3.10: Datos cargados en la tabla temporal

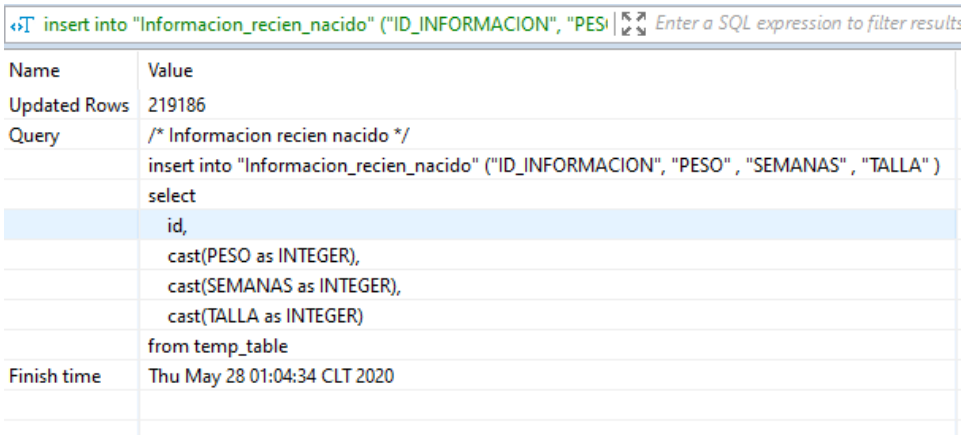


Figura 3.11: Poblar Tabla Informacion Recien Nacidos

Informacion_recien_nacido | Enter a SQL expression to filter results (use Ctrl+Space)

Grid | Text | Record

	ID_INFORMACION	SEMANAS	PESO	TALLA
1	1	37	2.650	49
2	2	38	3.980	53
3	3	40	4.536	52
4	4	38	3.640	50
5	5	39	3.290	50
6	6	40	3.160	50
7	7	40	2.700	48
8	8	40	4.010	51
9	9	40	3.020	50
10	10	40	3.400	48
11	11	39	3.270	50
12	12	37	3.130	49
13	13	39	3.710	51
14	14	40	3.010	49
15	15	39	2.900	48
16	16	40	3.125	52
17	17	38	3.385	49
18	18	37	3.140	49
19	19	40	3.340	51
20	20	38	3.625	49
21	21	39	4.045	53
22	22	39	3.470	50
23	23	38	3.635	48
24	24	39	3.670	49
25	25	33	1.905	44
26	26	38	3.100	50
27	27	38	3.010	47
28	28	37	2.670	47
29	29	39	3.540	48
30	30	38	3.220	50
31	31	36	2.550	48
32	32	40	3.255	50
33	33	38	2.885	49
34	34	39	2.750	47
35	35	40	3.540	51
36	36	39	3.620	50
37	37	38	3.125	48
38	38	40	3.360	51
39	39	38	3.570	49
40	40	38	3.510	49
41	41	40	3.630	50
42	42	40	3.380	50

Figura 3.12: Datos cargados en la Tabla Informacion Recien Nacidos

3.4. Taller n4

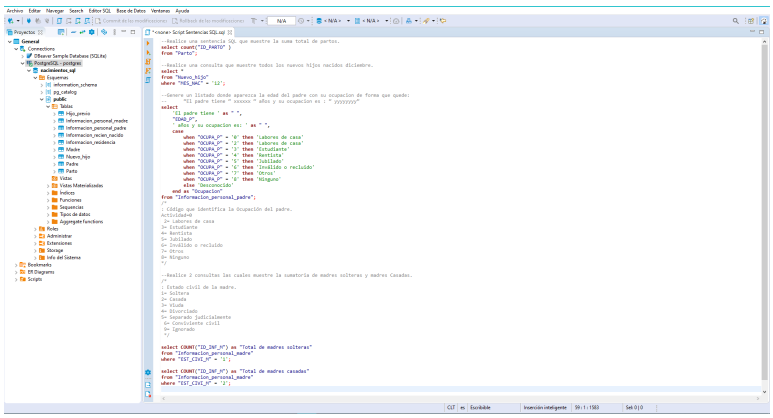


Figura 3.13: Captura de Pantalla pgModeler

3.4.1. Pregunta 1

Realice una sentencia SQL que muestre la suma total de partos.

```
--Realice una sentencia SQL que muestre la suma total de partos.
select count("ID_PARTO" )
from "Parto";
```

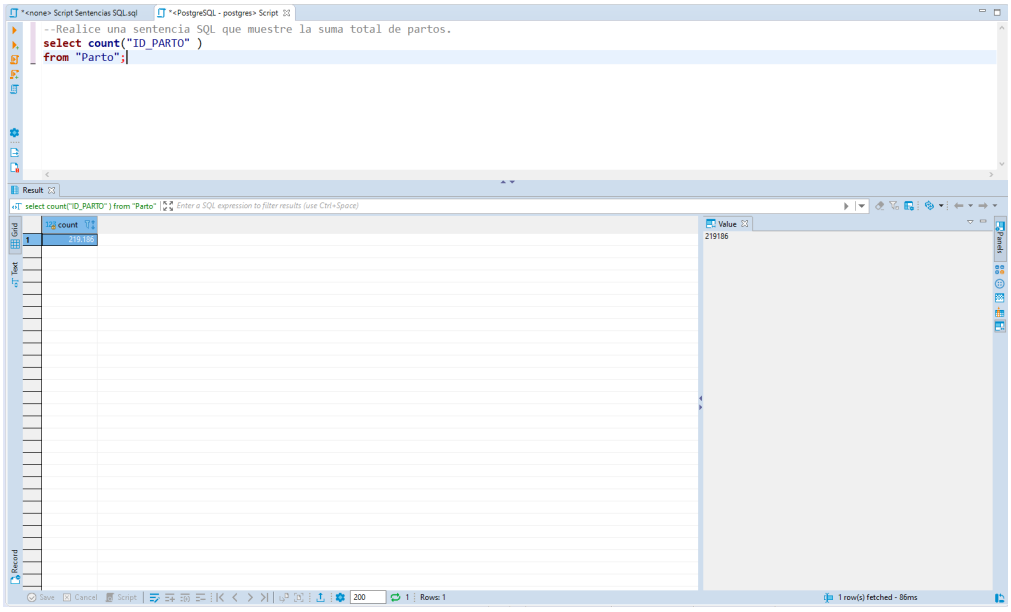


Figura 3.14: Ejecución de la sentencia SQL para la suma total de partos.

3.4.2. Pregunta 2

Realice una consulta que muestre todos los nuevos hijos nacidos diciembre

```
--Realice una consulta que muestre todos los nuevos hijos
--nacidos en diciembre.
select *
from "Nuevo_hijo"
where "MES_NAC" = '12';
```

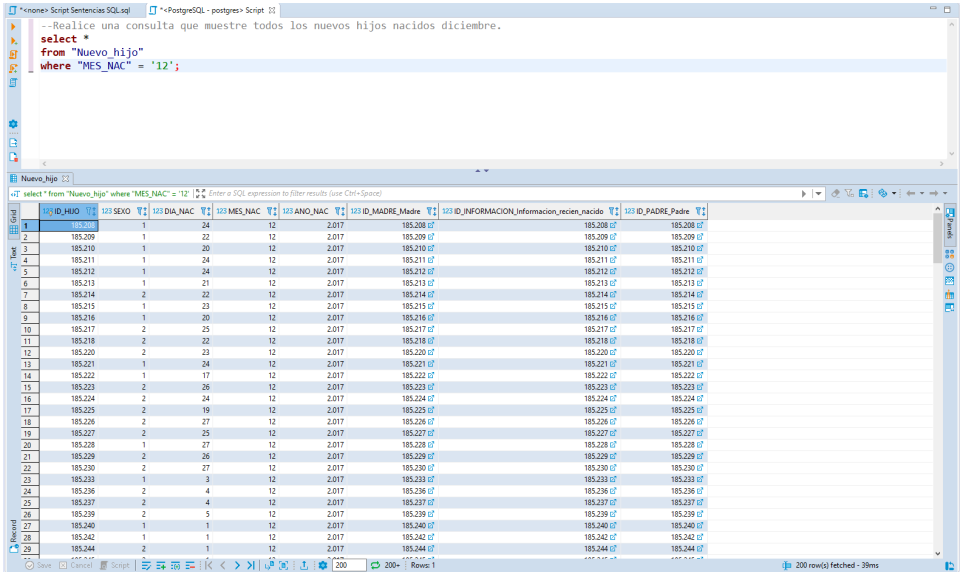


Figura 3.15: Ejecución de la sentencia SQL para los nacidos en diciembre.

3.4.3. Pregunta 3

Genere un listado donde aparezca la edad del padre con su ocupación.
de forma que quede: “El padre tiene “ xxxxxx “ años y su ocupacion es : “ yyyyyyyy”

```
    select
        'El padre tiene ' as " ",
        "EDAD_P",
        ' anos y su ocupacion es: ' as " ",
        case
                when "OCUPA_P" = '0' then 'Labores de casa'
                when "OCUPA_P" = '2' then 'Labores de casa'
                when "OCUPA_P" = '3' then 'Estudiante'
                when "OCUPA_P" = '4' then 'Rentista'
                when "OCUPA_P" = '5' then 'Jubilado'
                when "OCUPA_P" = '6' then 'Invalido o recluido'
                when "OCUPA_P" = '7' then 'Otros'
                when "OCUPA_P" = '8' then 'Ninguno'
                else 'Desconocido'
        end as "Ocupacion"
from "Informacion_personal_padre";
```

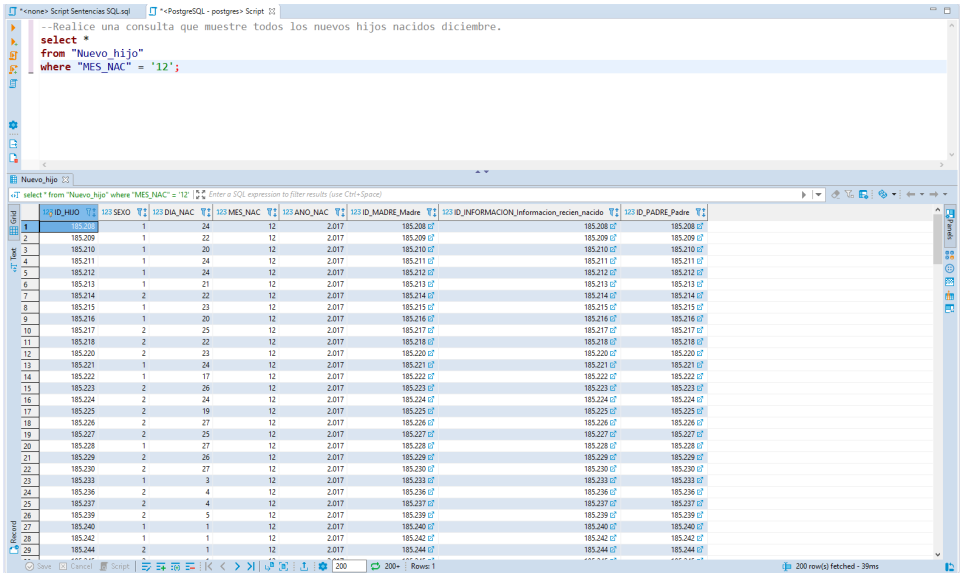


Figura 3.16: Ejecución de la sentencia SQL para las ocupaciones de los Padres.

3.4.4. Pregunta 4

Realice 2 consultas las cuales muestre la sumatoria de madres solteras y madres casadas.

```
select COUNT("ID_INF_M") as "Total de madres solteras"
from "Informacion_personal_madre"
where "EST_CIVI_M" = '1';

select COUNT("ID_INF_M") as "Total de madres casadas"
from "Informacion_personal_madre"
where "EST_CIVI_M" = '2';
```

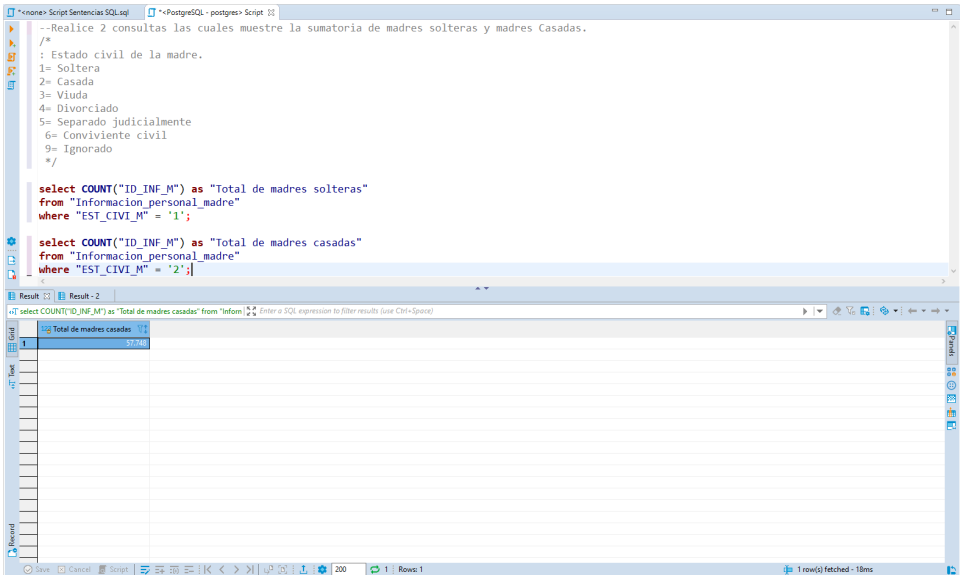


Figura 3.17: Ejecución de la sentencia SQL para la sumatoria de madres solteras.

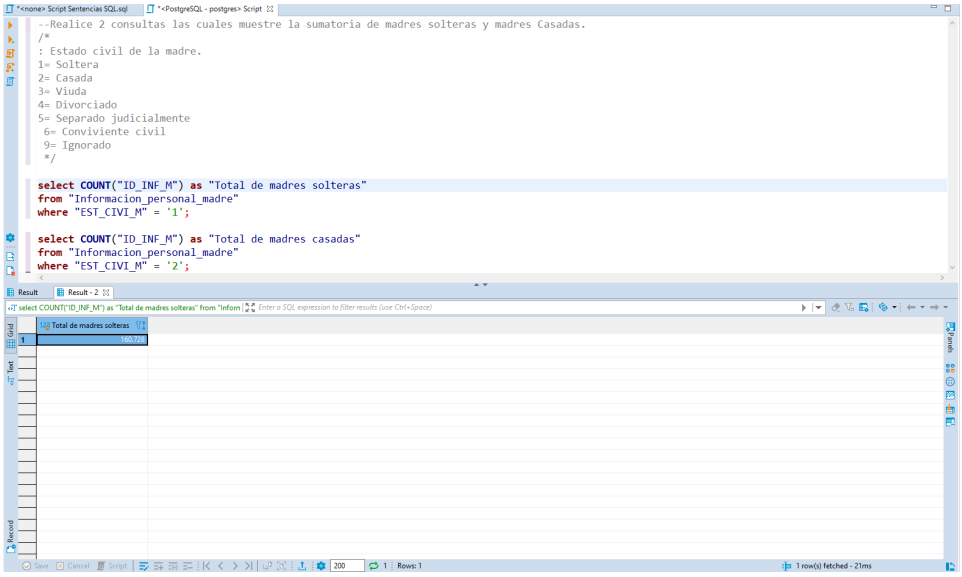


Figura 3.18: Ejecución de la sentencia SQL para la sumatoria de madres Casadas.

3.5. Taller n5

3.5.1. Capturas

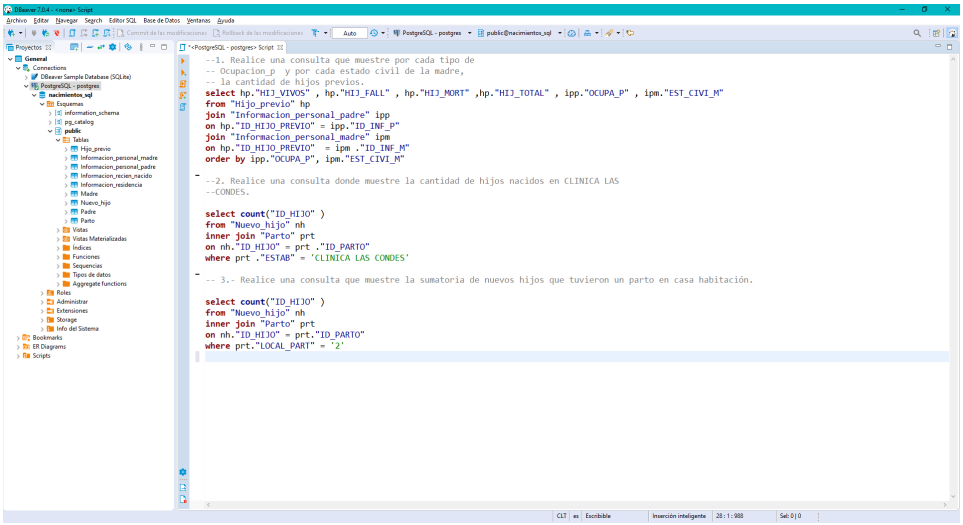


Figura 3.19: Captura de pantalla del entorno DBeaver.

3.5.2. Instrucciones SQL

3.5.2.1. Pregunta 1

Realice una consulta que muestre por cada tipo de OcupacionP y por cada estado civil de la madre, la cantidad de hijos previos.

```
--1. Realice una consulta que muestre por cada tipo de
-- Ocupacion_p  y por cada estado civil de la madre,
-- la cantidad de hijos previos.

select hp."HIJ_VIVOS" ,
       hp."HIJ_FALL" ,
       hp."HIJ_MORT" ,
       hp."HIJ_TOTAL" ,
       ipp."OCUPA_P" ,
       ipm."EST_CIVI_M"
from "Hijo_previo" hp
join "Informacion_personal_padre" ipp
on hp."ID_HIJO_PREVIO" = ipp."ID_INF_P"
join "Informacion_personal_madre" ipm
on hp."ID_HIJO_PREVIO"  = ipm ."ID_INF_M"
order by ipp."OCUPA_P", ipm."EST_CIVI_M"
```

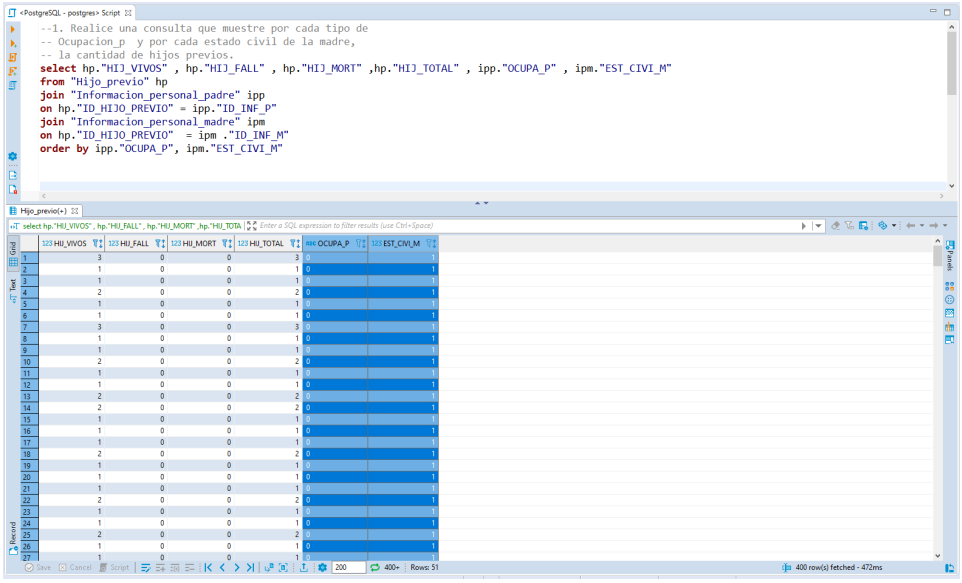


Figura 3.20: Ejecución de la sentencia SQL para los dos tipos.

3.5.2.2. Pregunta 2

Realice una consulta donde muestre la cantidad de hijos nacidos en 'CLINICA LAS CONDES'

```
--2. Realice una consulta donde muestre la cantidad
--de hijos nacidos en CLINICA LAS
--CONDES.

select count("ID_HIJO" )
from "Nuevo_hijo" nh
inner join "Parto" prt
on nh."ID_HIJO" = prt ."ID_PARTO"
where prt ."ESTAB" = 'CLINICA_LAS_CONDES'
```

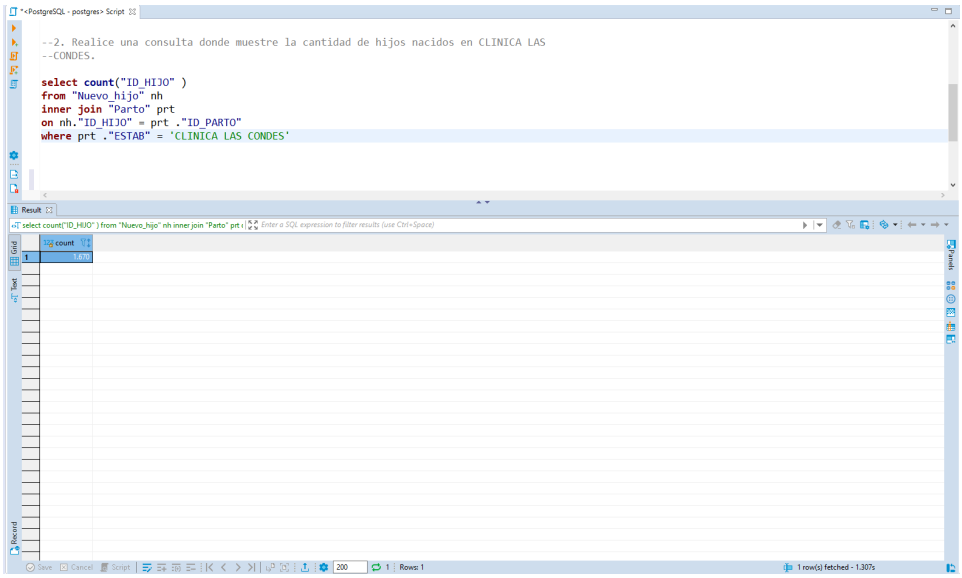


Figura 3.21: Ejecución de la sentencia SQL para la selección de nacimientos.

3.5.2.3. Pregunta 3

Realice una consulta que muestre la sumatoria de nuevos hijos que tuvieron un parto en casa habitación.

```
-- 3.- Realice una consulta que muestre la sumatoria de
--nuevos hijos que tuvieron un parto en casa habitacion.

select count("ID_HIJO" )
from "Nuevo_hijo" nh
inner join "Parto" prt
on nh."ID_HIJO" = prt."ID_PARTO"
where prt."LOCAL_PART" = '2'
```

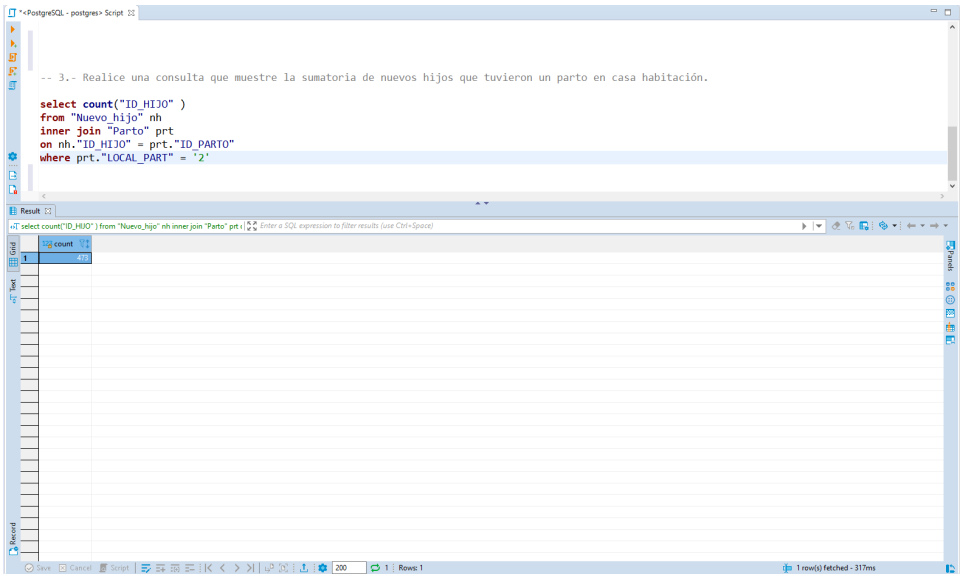


Figura 3.22: Ejecución de la sentencia SQL para la selección de nacimientos en casa habitación.

3.6. Taller n6

3.6.1. Pregunta 1

Consulta que muestre a los nacidos HOSPITAL CLAUDIO VICUNA con padres con ocupacion Jubilado

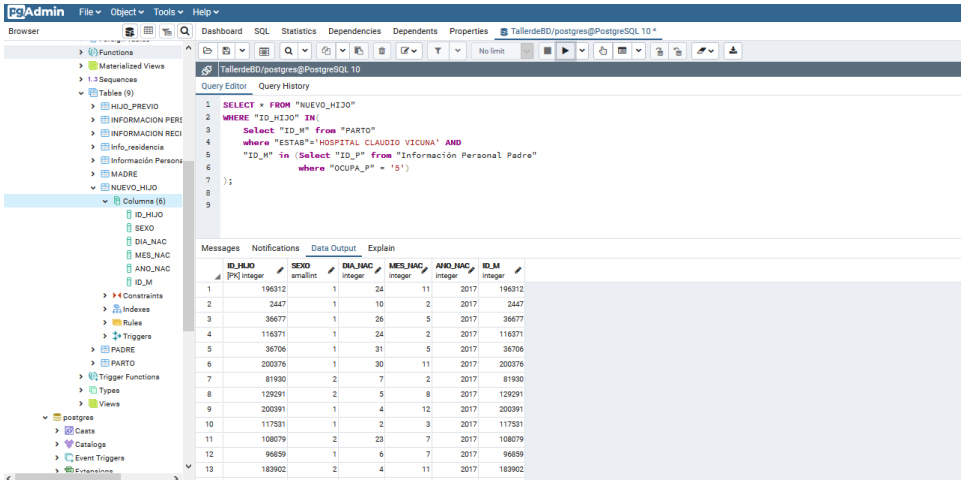


Figura 3.23: Ejecución de la sentencia SQL.

3.6.2. Pregunta 2

Consulta que indique la cantidad de soltero con resgitro de residencia en la region de valparaiso

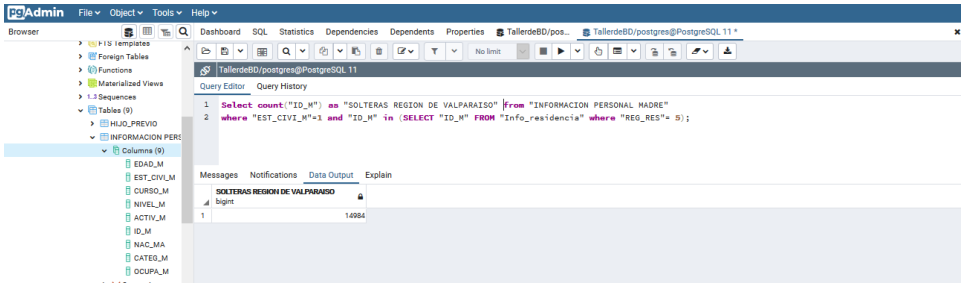


Figura 3.24: Ejecución de la sentencia SQL.

3.6.3. Pregunta 3

Consulta que muestre el nombre de las madres con hijos fallecidos por region y cual son sus totalidades por región

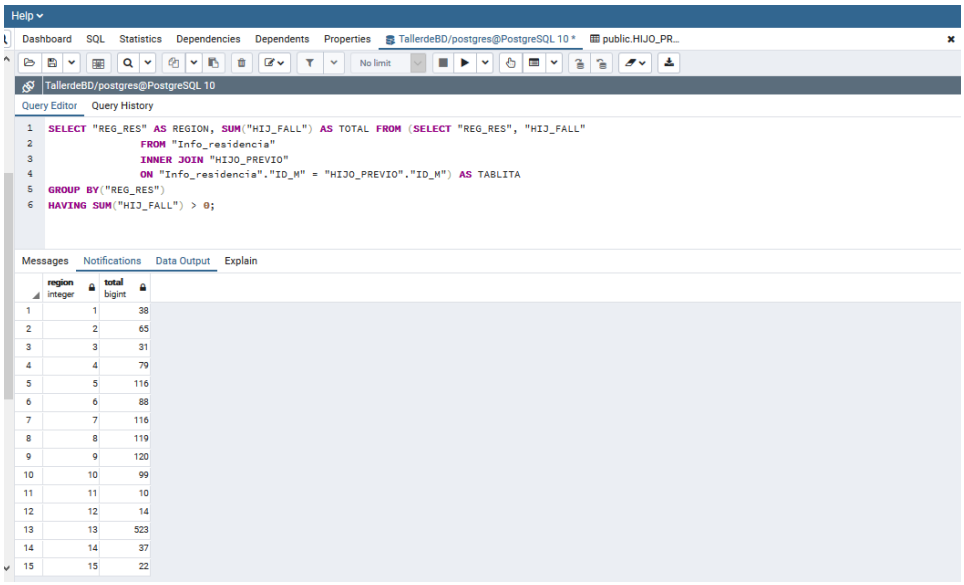


Figura 3.25: Ejecución de la sentencia SQL.

3.6.4. Pregunta 4

Consulta la cual muestre la sumatoria ya sea de la madre y el padre de hijos mortinatos.

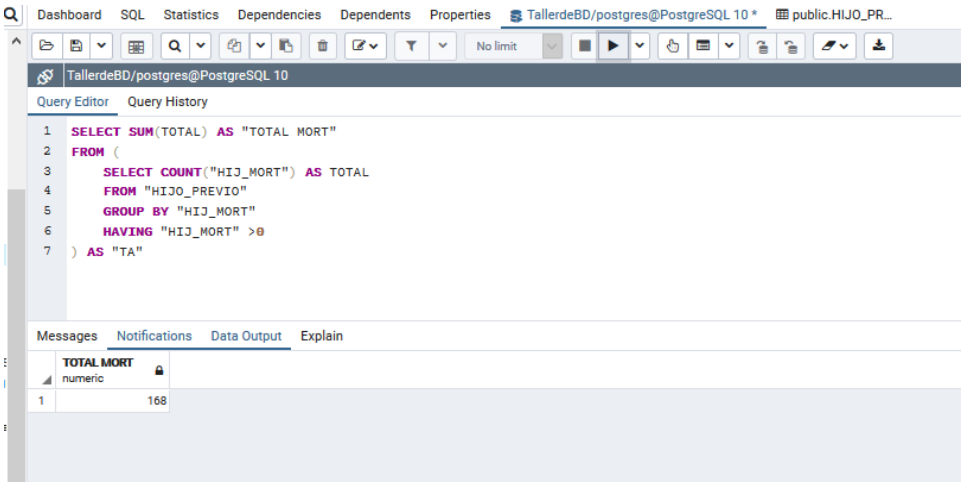


Figura 3.26: Ejecución de la sentencia SQL.

3.7. Taller n7

3.7.1. Pregunta 1

Realice una consulta en la cual separe por tipo de atencion de los nuevos hijos separado por por sexo

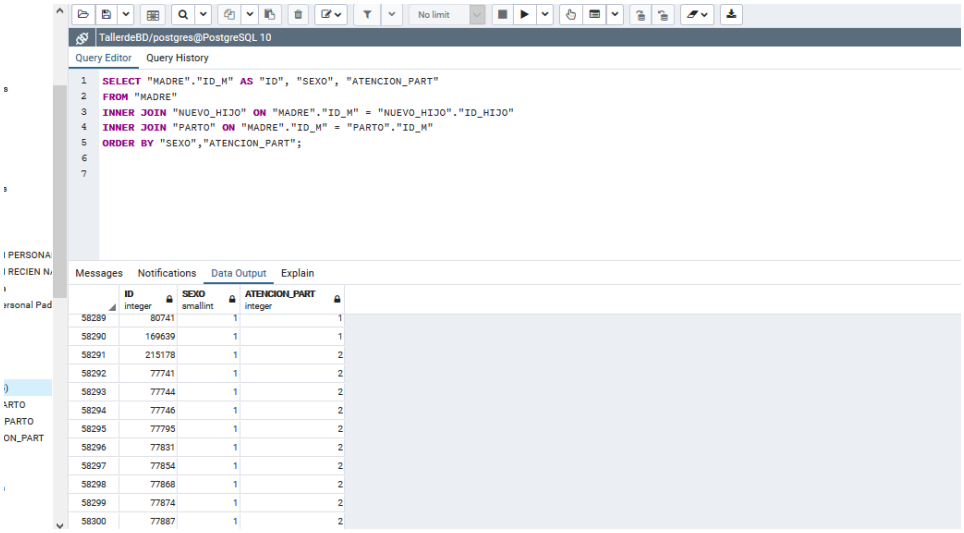


Figura 3.27: Ejecución de la sentencia SQL.

3.7.2. Pregunta 2

Realice una consulta en la cual dependiendo de la ocupacion de los madres que tipo de parto tienen .Se necesita saber la cantidad de cada uno

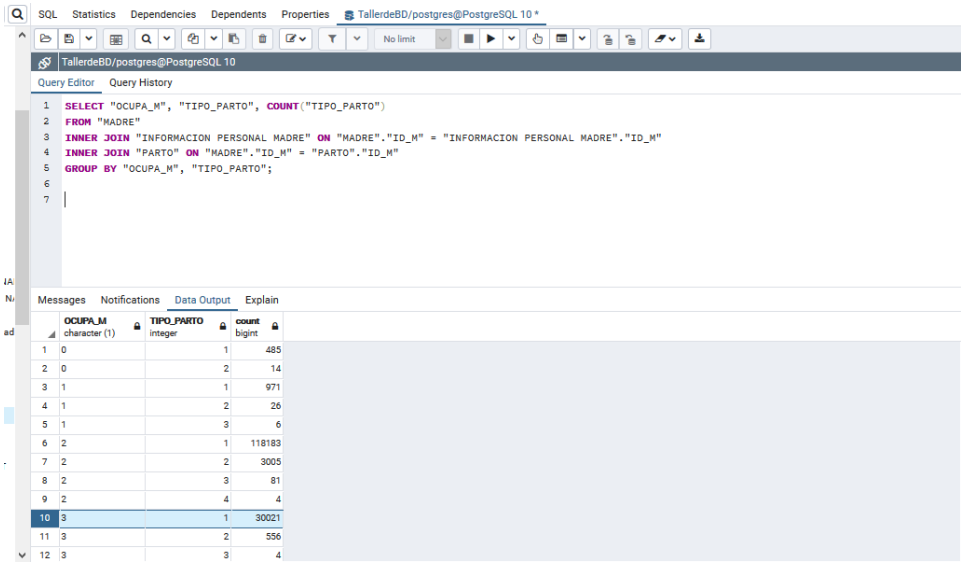


Figura 3.28: Ejecución de la sentencia SQL.

3.7.3. Pregunta 3

Realice una consulta en la cual dependiendo de la ocupacion de los padres que tipo de parto tienen Se necesita saber la cantidad de cada uno

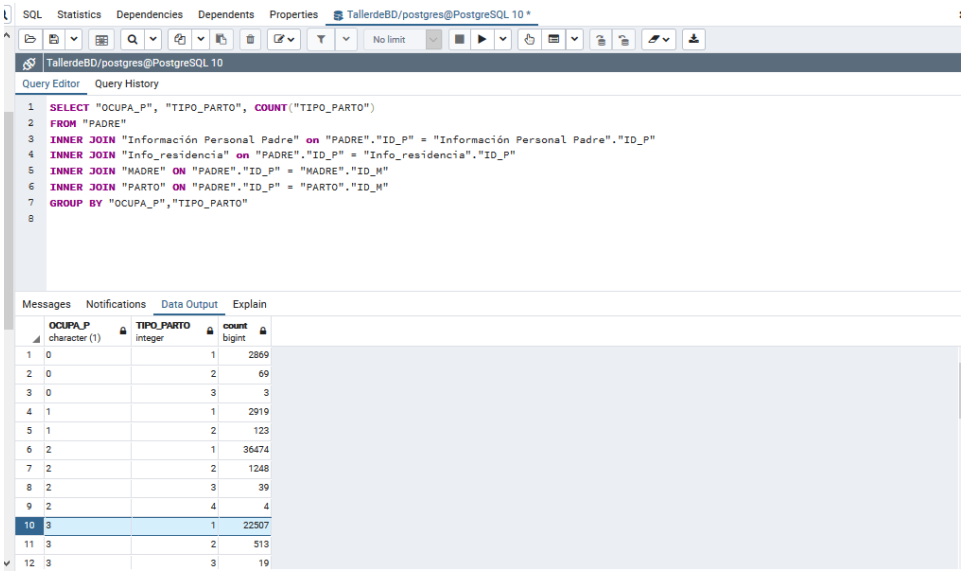


Figura 3.29: Ejecución de la sentencia SQL.

3.7.4. Pregunta 4

Necesito saber que edad son mas propenso a a tener partos tanto de madres y padres.

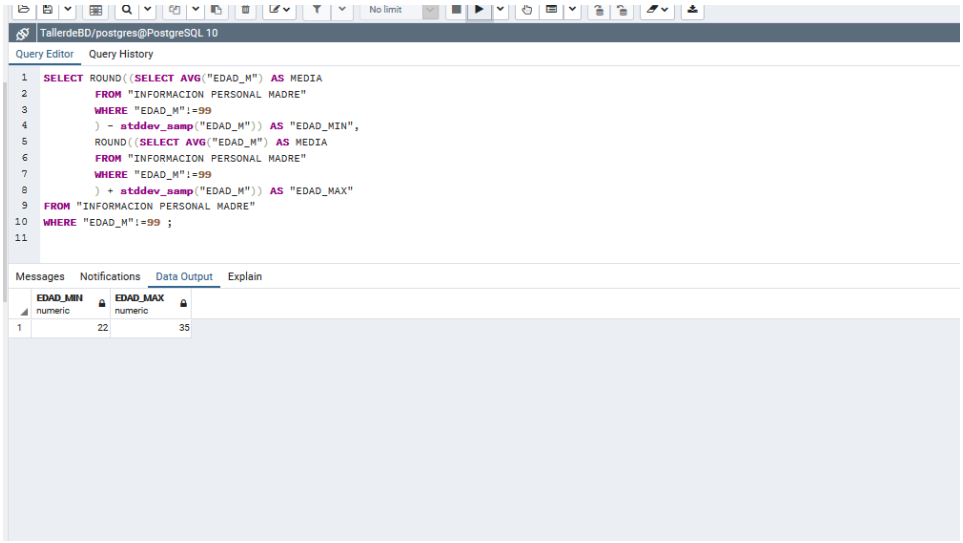


Figura 3.30: Ejecución de la sentencia SQL.

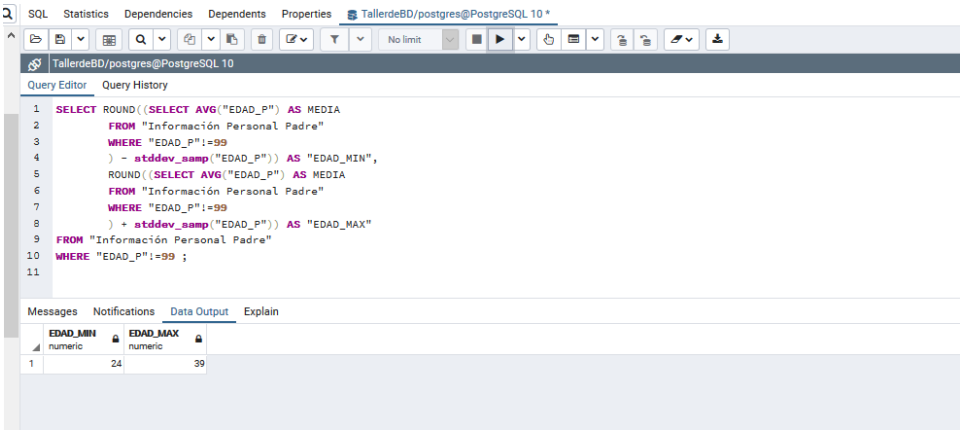


Figura 3.31: Ejecución de la sentencia SQL.

3.8. Taller n8

3.8.1. Pregunta 1

. Crear un manual paso a paso como se ejecuta el lenguaje plpsql en postgressql.

Antes de realizar cualquier acción relacionado al lenguaje plpsql, hay que realizar una verificación si es que se encuentra instalado, por ende, utilizamos el siguiente comando ante cualquier duda:

CREATE PROCEDURAL plpgsql;

Luego, podemos escribir los procedimientos que encontremos necesarios, en efecto:

```
CREATE OR REPLACE FUNCTION funcionadito(num INTEGER)
RETURNS integer AS $$
BEGIN RETURN NUMERO;
END;
$$LANGUAGE plpgsql;
```

3.8.2. Pregunta 2

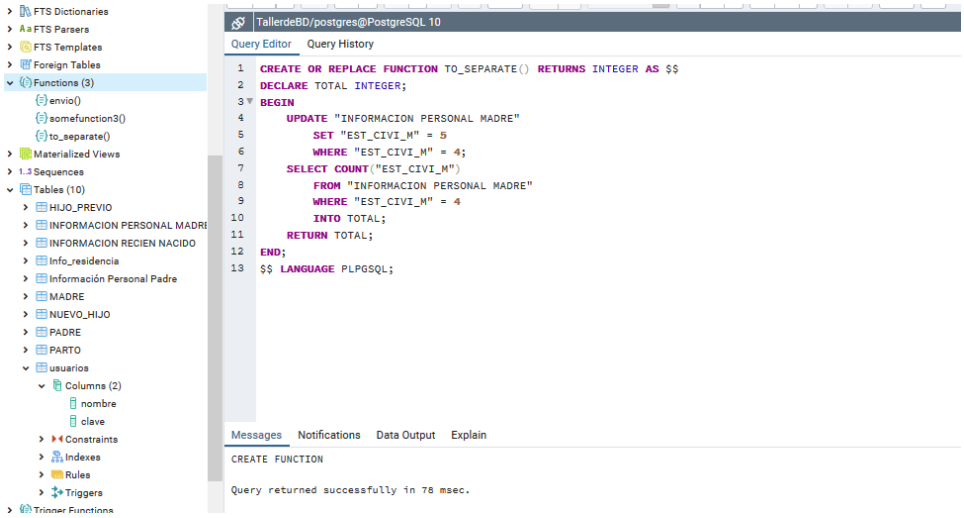


Figura 3.32: Ejecución de la sentencia SQL.

3.8.3. Pregunta 3

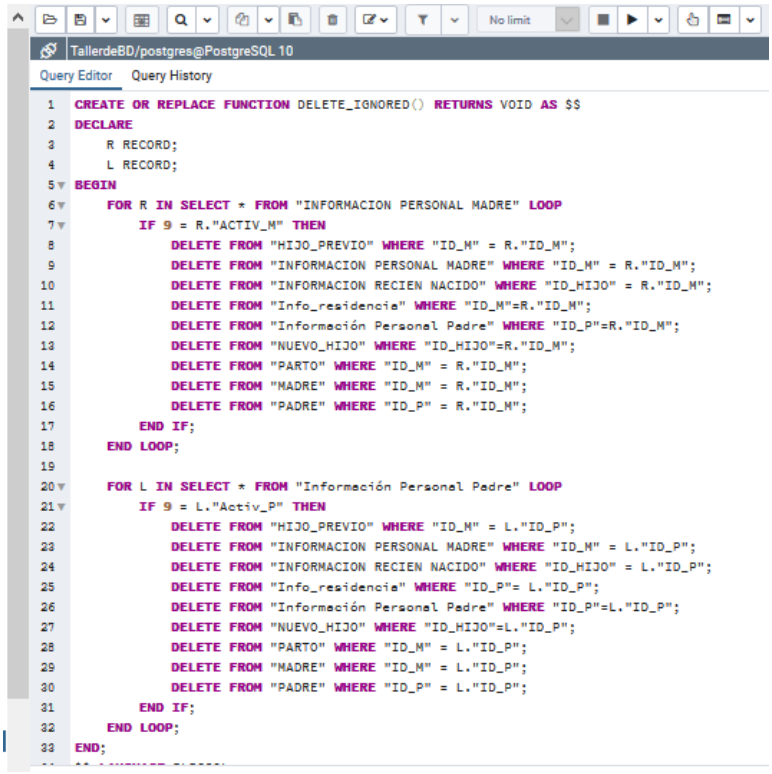


Figura 3.33: Ejecución de la sentencia SQL.

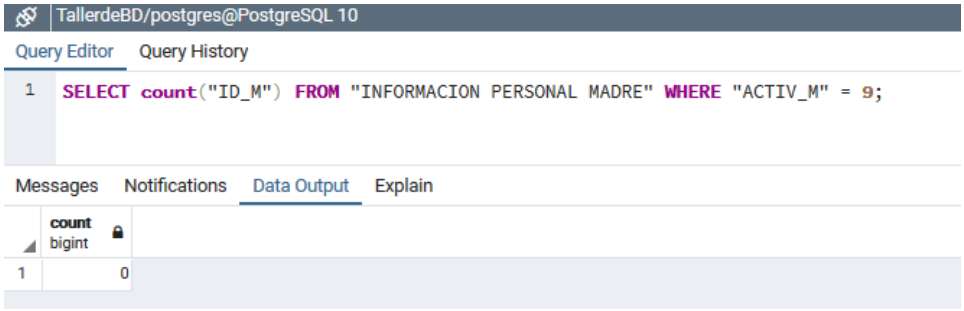


Figura 3.34: Ejecución de la sentencia SQL.

3.8.4. Pregunta 4

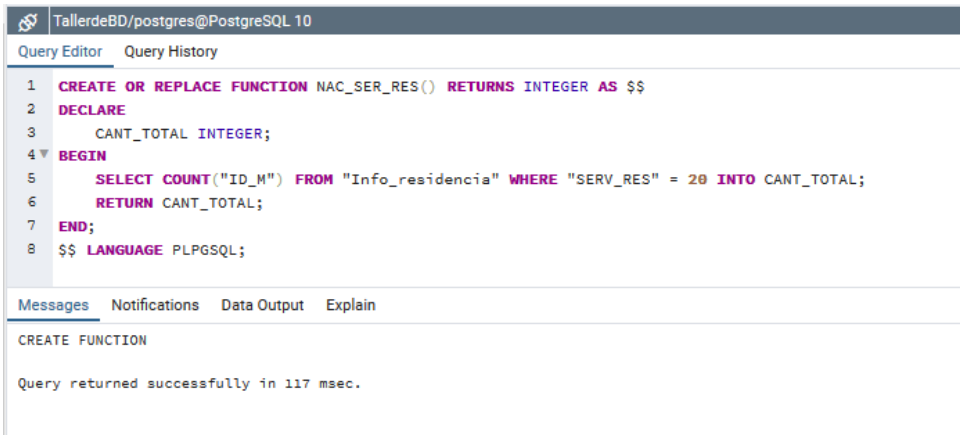


Figura 3.35: Ejecución de la sentencia SQL.

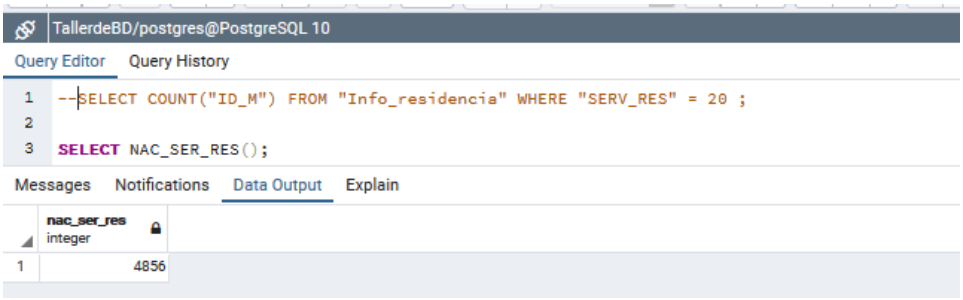


Figura 3.36: Ejecución de la sentencia SQL.

3.9. Taller n9

3.9.1. Pregunta 1

Cree un trigger a su criterio.

```
CREATE OR REPLACE FUNCTION SET_MORT() RETURNS TRIGGER AS $$
BEGIN
IF NEW.'HIJ_MORT'=99 THEN
SET NEW.'HIJ_MORT'=0;
END IF;
RETURN NEW;
END;
$$ LANGUAGE PLPGSQL;
CREATE TRIGGER SET_HIJ_MORT BEFORE INSERT ON 'HIJ_MORT' FOR EACH ROW
EXECUTE PROCEDURE SET_MORT();
```

En caso que se ingrese un parto donde los padres tienen 'HIJ_MORT' = 99, se cambia automaticamente este valor a 0.

```
CREATE OR REPLACE FUNCTION SET_MORT() RETURNS TRIGGER AS
$$
BEGIN
    IF NEW."HIJ_MORT" = 99 THEN
        SET NEW."HIJ_MORT" = 0;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER SET_HIJ_MORT BEFORE INSERT ON "HIJO_PREVIO" FOR EACH ROW
EXECUTE PROCEDURE SET_MORT();
```

Figura 3.37: Ejecución de la sentencia SQL.

3.9.2. Pregunta 2

Realice un trigger que elimine todos tipos de partos ignorados.

```
CREATE OR REPLACE FUNCTION elim_ignor()
RETURNS TRIGGER AS $ELIM$
DECLARE
BEGIN
DELETE FROM PARTO WHERE TIPO_PARTO = 9;
RETURN NULL;
END;
$ELIM$ LANGUAGE PLPGSQL;
CREATE TRIGGER IGNOR_ELIM AFTER DELETE ON 'PARTO' FOR EACH ROW
EXECUTE PROCEDURE ELIM_IGNOR();
```

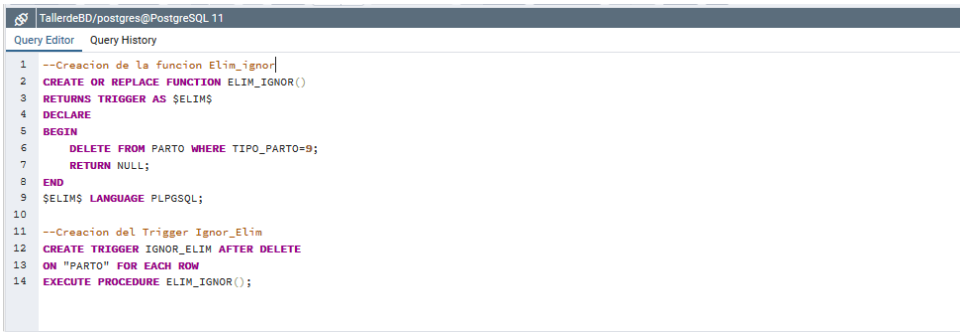


Figura 3.38: Ejecución de la sentencia SQL.

3.9.3. Pregunta 3

Realice un trigger que actualice todos los niveles de profesión del padre ignorado a básico.

```
CREATE OR REPLACE FUNCTION act_prof()
RETURNS TRIGGER AS $act$
DECLARE
BEGIN
UPDATE 'Informacion Personal Padre' SET 'NIVEL_P'=4 WHERE 'NIVEL_P'=5;
RETURN NULL;
END;
$ELIM$ LANGUAGE PLPGSQL;
CREATE TRIGGER PROF_ACT
AFTER UPDATE ON 'Informacion Personal Padre' FOR EACH ROW
EXECUTE PROCEDURE act_prof();
```

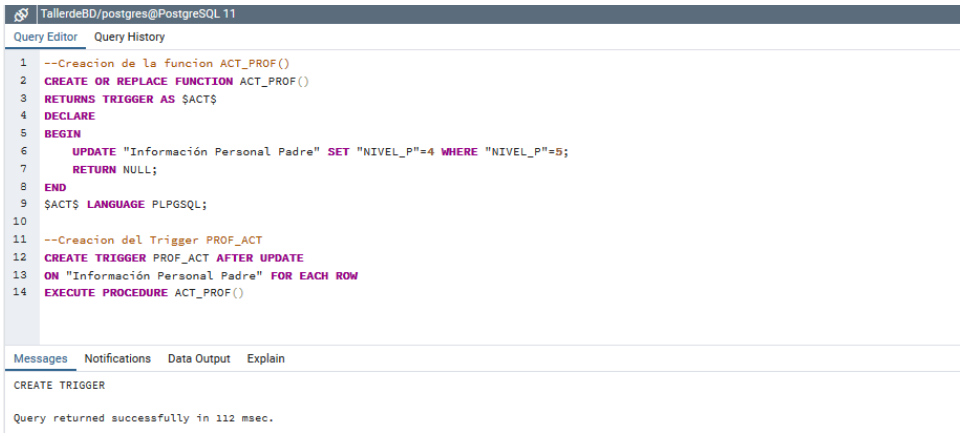


Figura 3.39: Ejecución de la sentencia SQL.

3.9.4. Pregunta 4

Realice un trigger que inserte información de una madre que tiene 22 años es casada con nivel de estudio superior y actividad activa

```
CREATE OR REPLACE FUNCTION ADD_MOM()
RETURNS TRIGGER AS $MOM$
DECLARE
BEGIN
INSERT INTO MADRE('ID_M') VALUES (0);
INSERT INTO 'INFORMACION_PERSONAL_MADRE'(EDAD_M, EST_CIVI, CURSO_M,
NIVEL_M, ACTIV_M, 'ID_M', NAC_M, CATEG_M, OCUPA_M)
VALUES (22,2,1,1,1,1,0,'C',7);
RETURN NULL;
END;
$MOM$ LANGUAGE PLPGSQL;
CREATE TRIGGER MOM_ADD AFTER UPDATE ON
 'INFORMACION_PERSONAL_MADRE' FOR EACH ROW
EXECUTE PROCEDURE ADD_MOM();
```

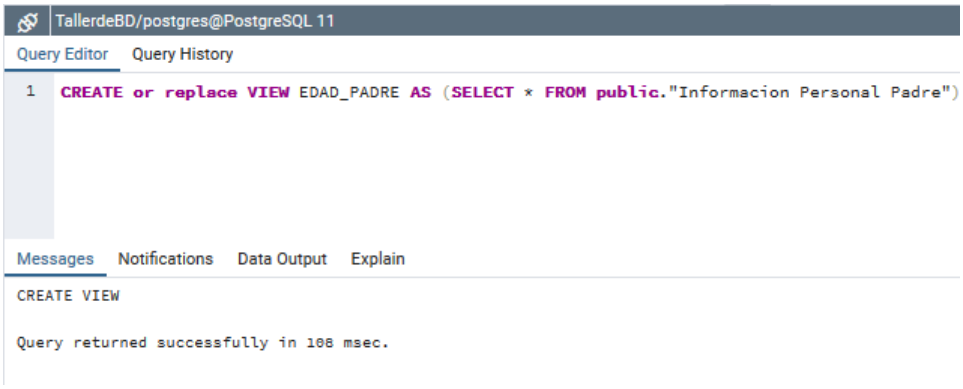


Figura 3.40: Ejecución de la sentencia SQL.

3.10. Taller n10

3.10.1. Pregunta 1

Desde SQL (no desde PGADMIN) cree un usuario con nombre INF3540 con password tallerbd.

CREATE USER inf3540 password 'tallerbd';

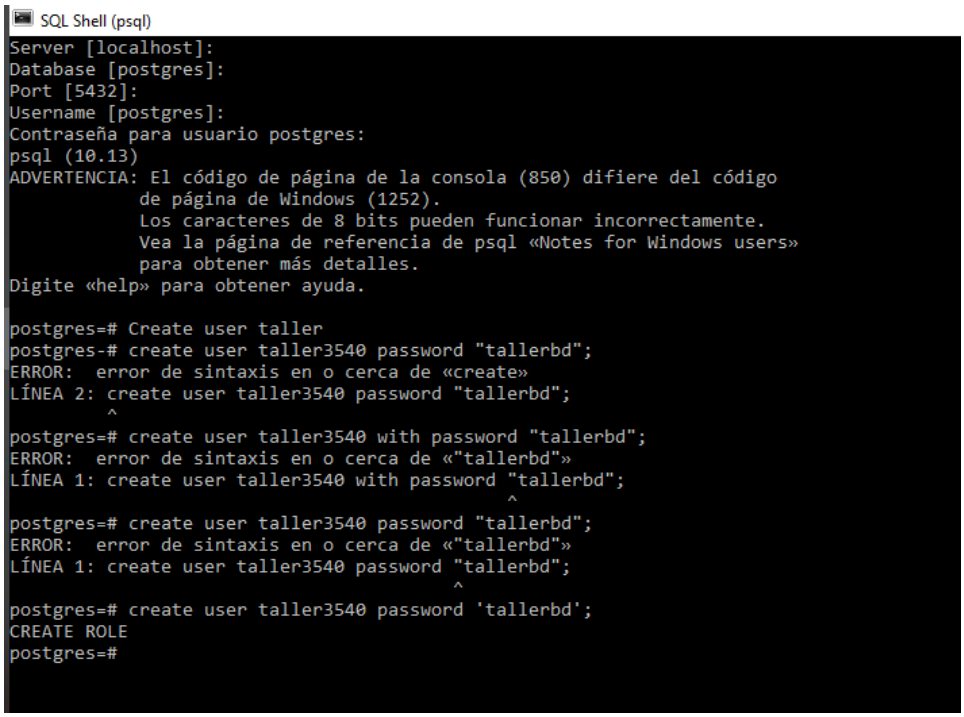


Figura 3.41: Crear un usuario desde SQL

Por un error de escritura el usuario se llamó taller350, pero luego fue cambiado al nombre solicitado por el enunciado del ejercicio.

3.10.2. Pregunta 2

Usando el psql, conéctese a Postgres usando el usuario creado

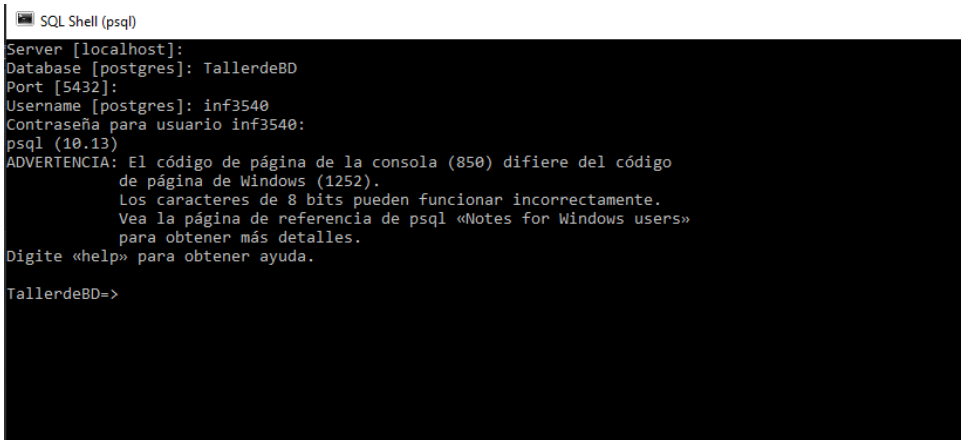


Figura 3.42: Conectandose a Postgres

Como se logra apreciar, el usuario fue cargado como 'inf350' arreglando lo mencionado en el ejercicio anterior.

3.10.3. Pregunta 3

Realice un select, inserción, eliminación de algún registro en la tabla ocupación de la madre.

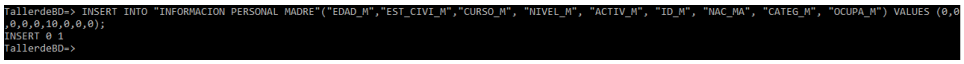


Figura 3.43: Comprobación de inserción

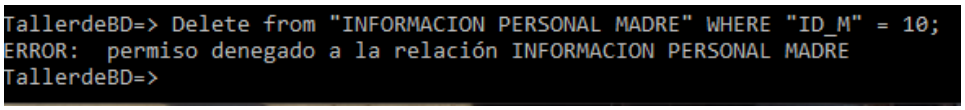


Figura 3.44: Comprobación de la eliminación

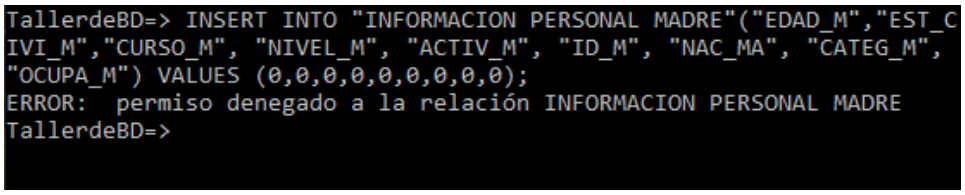


Figura 3.45: Comprobación de la consulta

3.10.4. Pregunta 4

Desde el SQL del PGADMIN conectado con usuario postgres, de accesos de lectura e inserción a la tabla tipo de atención, luego repita el paso 3.

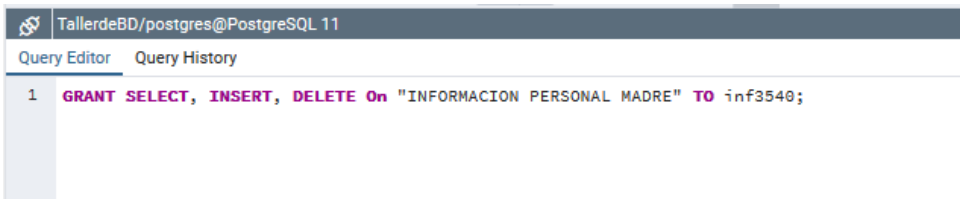


Figura 3.46: Comprobación de Permiso

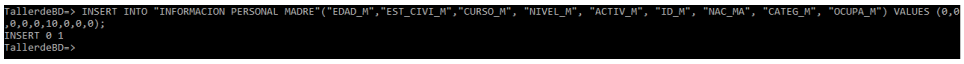


Figura 3.47: Comprobación de inserción

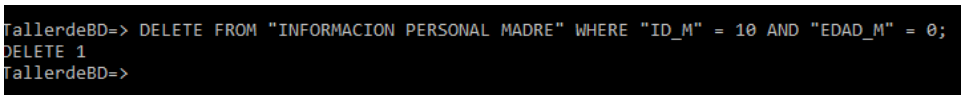


Figura 3.48: Comprobación de la eliminación

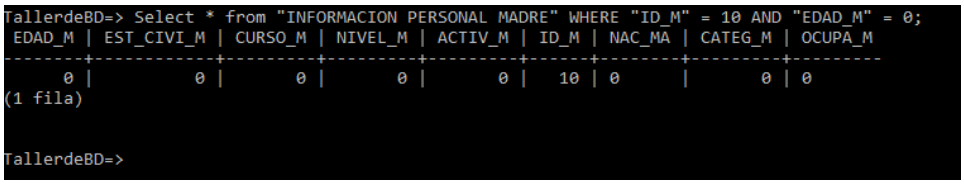


Figura 3.49: Comprobación de la consulta

3.10.5. Pregunta 5

De acceso de modificación solo al campo sexo nacido vivo

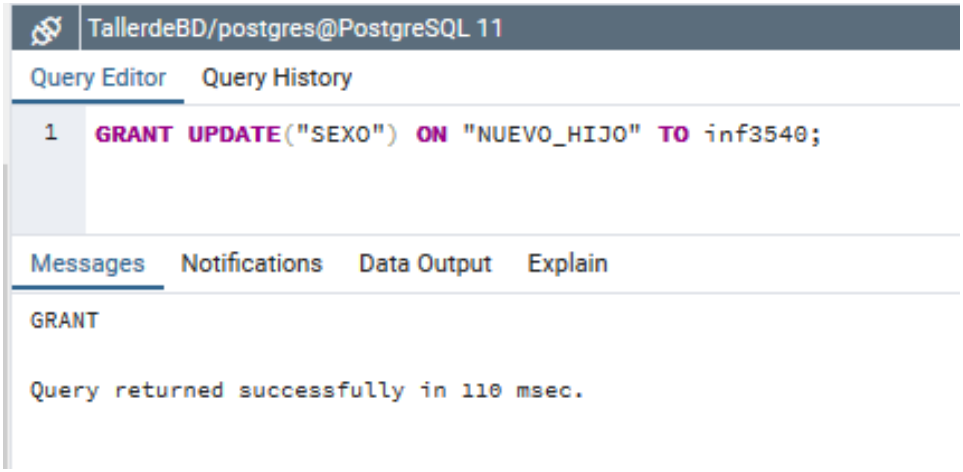
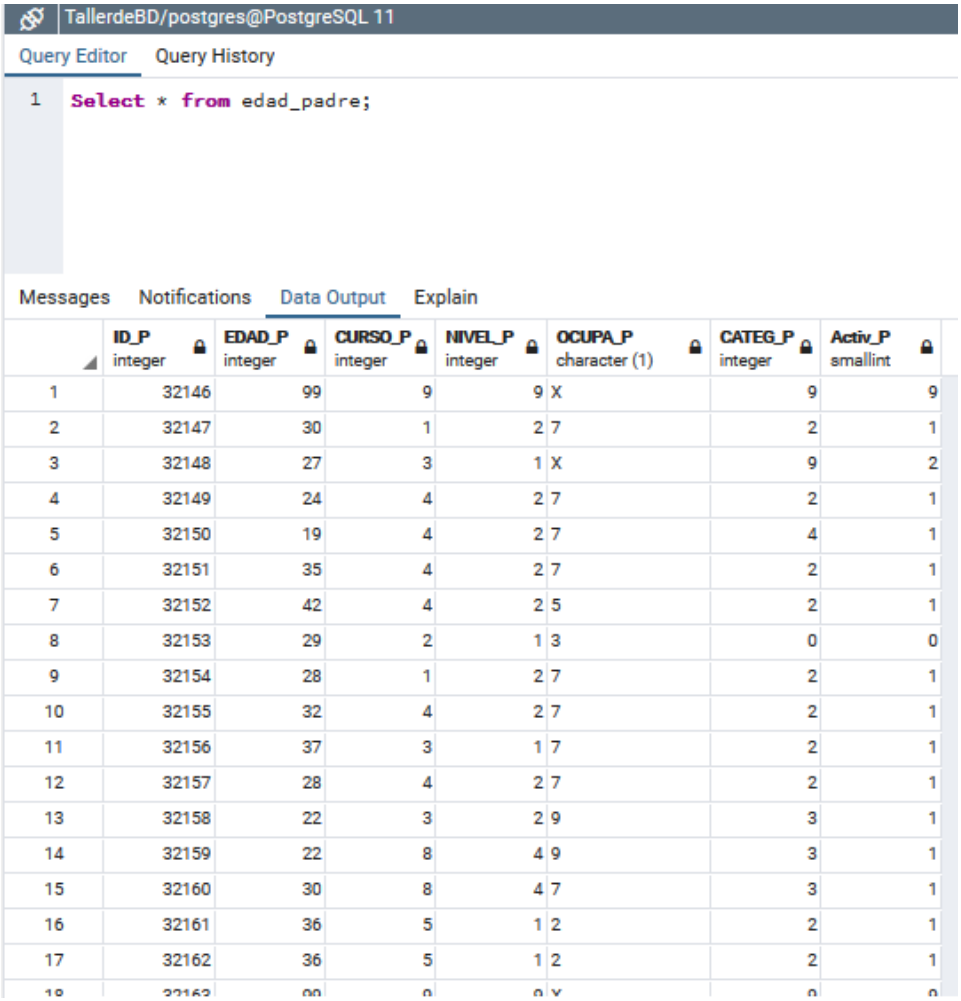


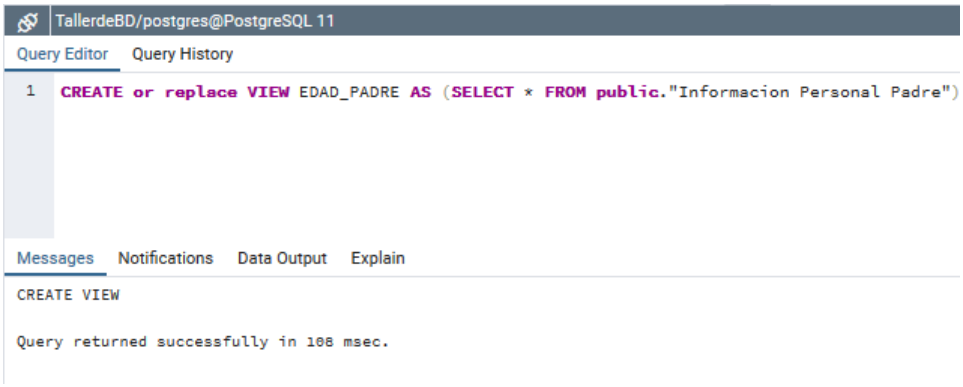
Figura 3.50: Acceso de modificación

3.10.6. Pregunta 6

Desde SQL de PGADMIN (usuario postgress), cree una vista



	ID_P integer	EDAD_P integer	CURSO_P integer	NIVEL_P integer	OCUPA_P character (1)	CATEG_P integer	Activ_P smallint
1	32146	99	9	9	X	9	9
2	32147	30	1	2	7	2	1
3	32148	27	3	1	X	9	2
4	32149	24	4	2	7	2	1
5	32150	19	4	2	7	4	1
6	32151	35	4	2	7	2	1
7	32152	42	4	2	5	2	1
8	32153	29	2	1	3	0	0
9	32154	28	1	2	7	2	1
10	32155	32	4	2	7	2	1
11	32156	37	3	1	7	2	1
12	32157	28	4	2	7	2	1
13	32158	22	3	2	9	3	1
14	32159	22	8	4	9	3	1
15	32160	30	8	4	7	3	1
16	32161	36	5	1	2	2	1
17	32162	36	5	1	2	2	1
18	32163	99	9	9	X	9	9


TallerdeBD/postgres@PostgreSQL 11
Query Editor
Query History
1
CREATE or replace VIEW EDAD_PADRE AS (SELECT * FROM public."Informacion Personal Padre")
Messages
Notifications
Data Output
Explain
CREATE VIEW
Query returned successfully in 108 msec.