CODER HOUSE

PROYECTO FINAL

COMISION:

SQL 31275

ESTUDIANTE:

JANDRY ANCHUNDIA

PROFESOR:

CAMILO ANDRES REDONDO

TUTORA

IARA CAMILA BATISTUTA

Contenido

Introducción

El presente informe corresponde al Proyecto Final del curso de SQL dictado en CoderHouse. En el mismo se desarrolló una base de datos que permitiría a los alumnos y profesores estar enterados sobre todos los procesos que se llevan a cabo dentro de la institución educativa. La información incluye: promedios, becas con la que cuenta un estudiante y fecha en la que se adquirió; y materias que imparte tal profesor de acorde a su especialidad etc.

Objetivo

El objetivo del proyecto final fue la creación de una base de datos relacional en SQL para llevar a cabo una correcta organización y seguimiento de todos los procesos que se efectúan dentro de una institución educativa superior.

Modelo de negocio

La base de datos de “Alumnos” por ser parte intrínseca de una entidad educativa requiere sí o sí el control de todas las actividades que realiza tanto su personal como sus estudiantados. Este proyecto no cuenta con un lugar determinante, por lo tanto, solo formó parte del desarrollo de todos los entregables propuestos en el curso. Las principales entidades que se fijaron fueron: alumno, profesor, materia, carrera, y becas.

Situación problemática

Alumnos fue creada con el propósito de estructurar y organizar eficientemente la información de una entidad de educación superior. A la vez, el desarrollo de esta base de datos permitirá extraer información importante de manera inmediata e íntegra.

Descripción de tablas

Se desea almacenar los datos principales de un alumno, la carrera que estudia, las materias que cursa, los profesores que le imparten clase y las becas que

posee. De igual manera se desea llevar un registro de las materias que imparte cada profesor.

TABLA:	Alumnno			
Campo	**Tipo**	**Longitud**	**Descripcion**	**Llave**
Matricula_alu (Pk)	INT	10	nro de identificacion del alumno	PK
Nombre_alu	VARCHAR	45	Nombre del alumno	
Edad_alu	INT	10	Edad con la que cuenta el alumno	
Semestre_alu	INT	10	Semestre en el que esta el alumno	
Genero_alu	VARCHAR	45	Sexo con el que se identifica el alumno	
Clave_C1 (Pk)	INT	10	nro para identidicar la carrera que el alumno cursa	FK

TABLA:	Carrera			
Campo	**Tipo**	**Longitud**	**Descripcion**	**Llave**
Clave_c (Pk)	INT	10	nro para identidicar la carrera que el alumno cursa	PK
Nombre_c	VARCHAR	45	Nombre de la carrera	
Duracion_c	INT	10	Años totales que demanda cursar dicha carrea	

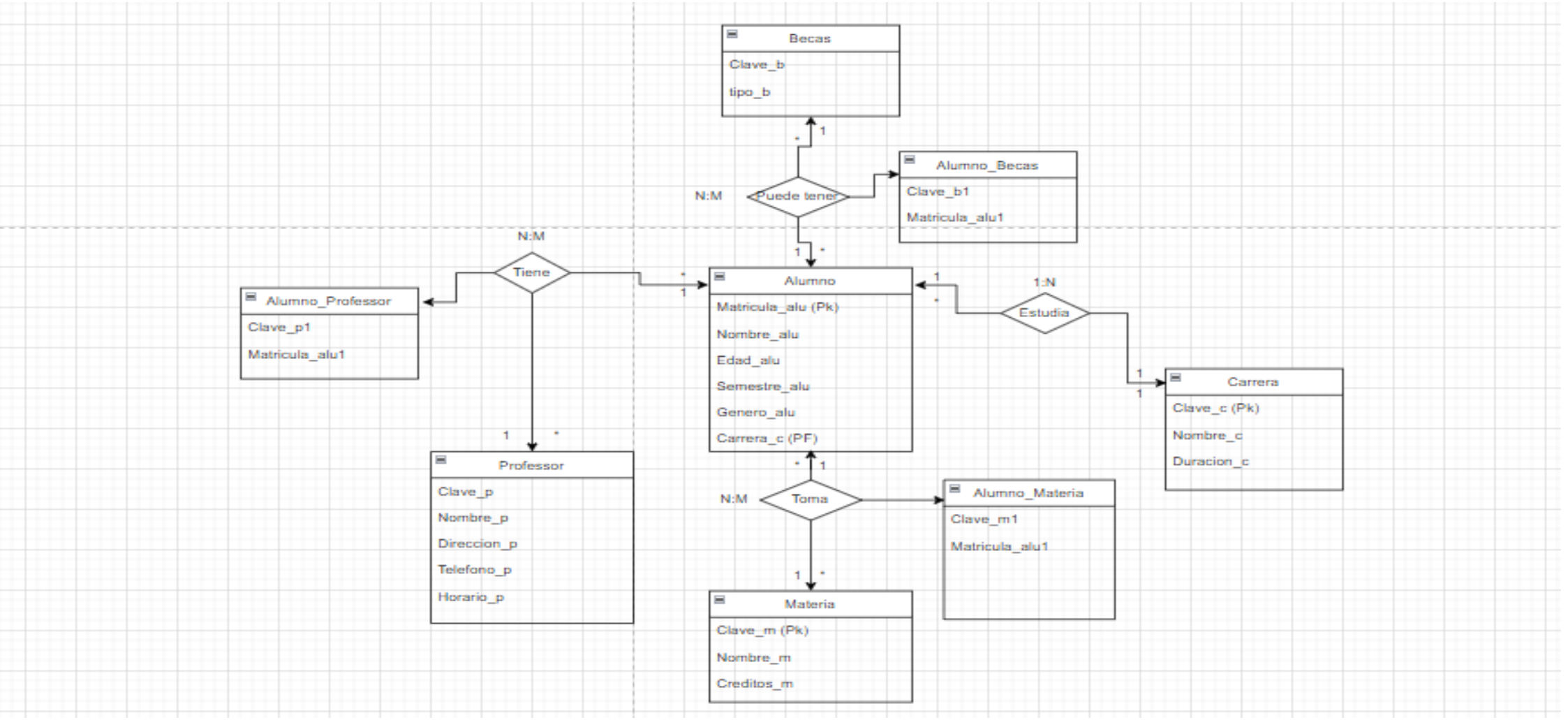
TABLA:	Profesor			
Campo	**Tipo**	**Longitud**	**Descripcion**	**Llave**
Clave_P (Pk)	INT	10	nro de identificacion del profesor	PK
Nombre_P	VARCHAR	45	Nombre del profesor	
Direccion_P	VARCHAR	100	Direccion del profesor	
Telefono_P	INT	15	Telefono con el que sepodra contactar al profesor	
Genero_P	VARCHAR	45	Sexo con el que se identifica el profesor	
Horario_P	INT	10	Hora en la que el profesor tendra que impartir su materia	
TABLA INTERMEDIA POR RELACION DE MUCHOS A MUCHOS				
Matricula_alu1 (Pk)	INT	10	nro de identificacion del alumno	FK
Clave_P1 (Pk)	INT	10	nro de identificacion del profesor	FK

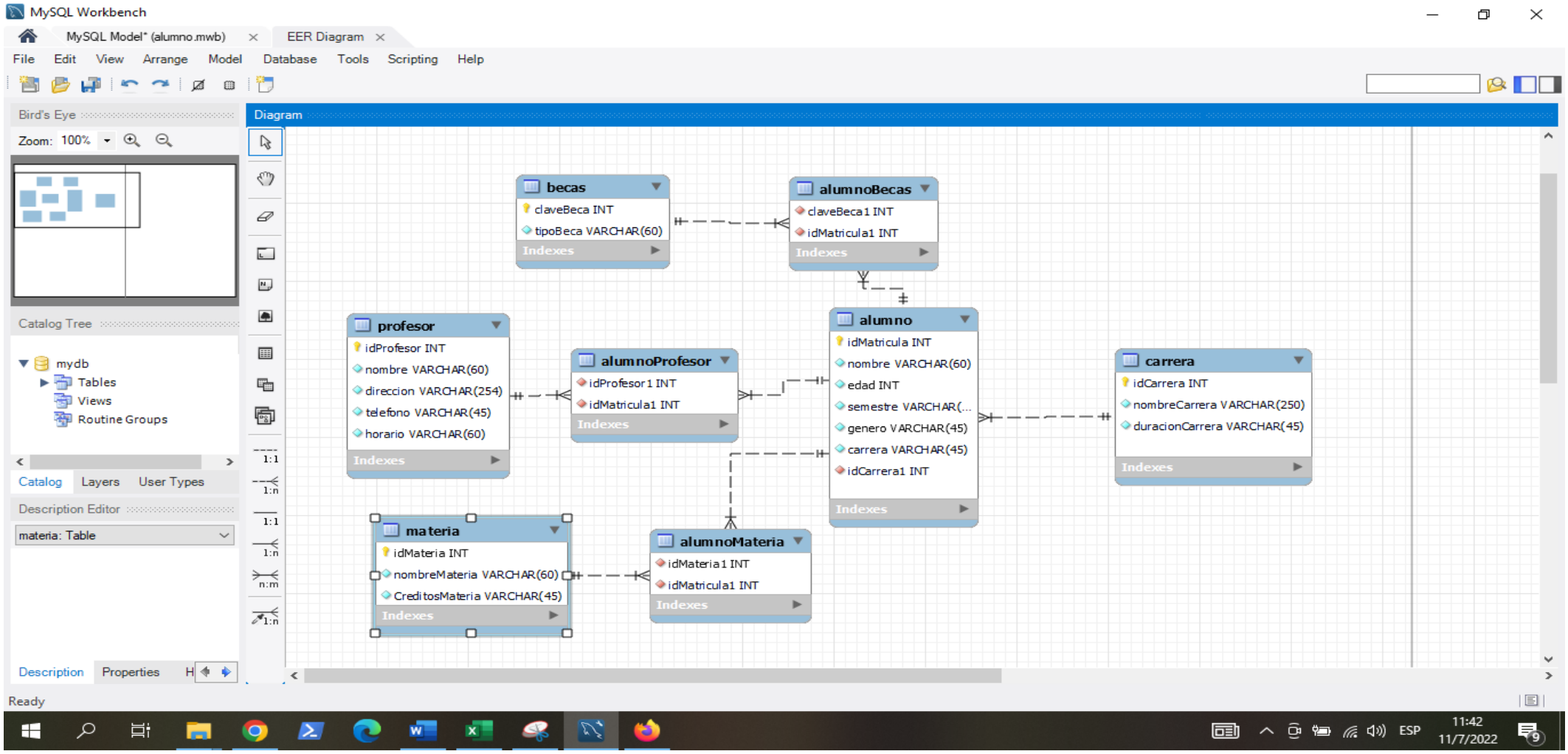
TABLA:	Materia			
Campo	**Tipo**	**Longitud**	**Descripcion**	**Llave**
Clave_M	INT	10	nro de identificacion de la materia	PK
Nombre_M	VARCHAR	50	Nombre de la materia	
Creditos_M	INT	10	Creditos con los que cuenta la materia. (De a corde a los creditos se completara el pensum para terminar con éxito la carrera)	
TABLA INTERMEDIA POR RELACION DE MUCHOS A MUCHOS				
Clave_M1	INT	10	nro de identificacion de la materia	FK
Clave_P1 (Pk)	INT	10	nro de identificacion del profesor	FK

TABLA:	Becas			
Campo	**Tipo**	**Longitud**	**Descripcion**	**Llave**
Clave_B	INT	10	nro de identificacion de la Beca a la cual el alumno esta postulando	PK
Tipo_B	VARCHAR	50	Tipo de beca	
Nombre_B	VARCHAR	50	De acuerdo al tipo de beca que escogio aquí se debe especificar el nombre.	
TABLA INTERMEDIA POR RELACION DE MUCHOS A MUCHOS				
Matricula_alu1 (Pk)	INT	10	nro de identificacion del alumno	FK
Clave_B1 (PK)	INT	10	nro de identificacion de la Beca a la cual el alumno esta postulando	FK

Diagramas Entidad – Relación

A continuación, se muestran dos imágenes del diagrama de Entidad – Relación del proyecto. La primera imagen fue realizada en una herramienta web y se realizó al comienzo del curso (luego se realizaron modificaciones sobre los campos de las tablas). La segunda imagen contiene el Diagrama E – R final y se realizó a partir de un Script en SQL en MySQL Workbench.




MySQL Workbench
MySQL Model* (alumno.mwb)
EER Diagram
File Edit View Arrange Model Database Tools Scripting Help
Bird's Eye
Zoom: 100%
Catalog Tree
mydb
Tables
Views
Routine Groups
Catalog Layers User Types
Description Editor
materia: Table
Description Properties
Diagram
becas
claveBeca INT
tipoBeca VARCHAR(60)
Indexes
alumnoBecas
claveBeca1 INT
idMatricula1 INT
Indexes
profesor
idProfesor INT
nombre VARCHAR(60)
direccion VARCHAR(254)
telefono VARCHAR(45)
horario VARCHAR(60)
Indexes
alumnoProfesor
idProfesor1 INT
idMatricula1 INT
Indexes
alumno
idMatricula INT
nombre VARCHAR(60)
edad INT
semestre VARCHAR(...
genero VARCHAR(45)
carrera VARCHAR(45)
idCarrera1 INT
Indexes
carrera
idCarrera INT
nombreCarrera VARCHAR(250)
duracionCarrera VARCHAR(45)
Indexes
materia
idMateria INT
nombreMateria VARCHAR(60)
CreditosMateria VARCHAR(45)
Indexes
alumnoMateria
idMateria1 INT
idMatricula1 INT
Indexes
Ready
ESP
11:42
11/7/2022

Creación de tablas en SQL

En este apartado se procedió a la creación de las tablas del proyecto. En este caso fueron 4 tablas principales y 3 intermedias. Se adjunta en el documento un Script SQL con la creación de estas (en el script de la carpeta se encuentra el resto).

```sql
-- ----------------------------------------
-- Creacion de tabla de Carrera
-- ----------------------------------------
create table if not exists carrera (
    id              int             not null auto_increment,
    nombreCarrera   varchar(150)    not null,
    duracionCarrera varchar(45)     not null,
    primary key (id)


) engine = innodb;
```

```sql
-- ----------------------------------------
-- Creacion de tabla de Becas
-- ----------------------------------------
create table if not exists becas (
    id          int             not null auto_increment,
    tipoBeca    varchar(150)    not null,
    primary key (id)

) engine = innodb;
```

```sql
-- -----------------------------------------
-- Creacion de tabla intermedia de alumnoBecas
-- -----------------------------------------
create table if not exists alumnoBecas (
    id              int             not null auto_increment,
    idBecas         int             not null,
    idAlumno        int             not null,
    fechaInicio     date            not null,
    primary key (id),

    constraint fkAlumnoBecasBecas
        foreign key (idBecas)
        references becas (id)
        on delete no action
        on update no action,

    constraint fkAlumnoBecasAlumno
        foreign key (idAlumno)
        references alumno (id)
        on delete no action
        on update no action

) engine = innodb;
```

```sql
-- -------------------------------------
-- Creacion de tabla de Alumno
-- -------------------------------------
create table if not exists alumno (

id          int         not null    auto_increment,
nombre      varchar(60) not null,
edad        int         not null,
semestre    varchar(45) not null,
genero      varchar(45) not null,
idCarrera   int         not null,
primary key (id),

constraint fkAlumnoCarrera
        foreign key (idCarrera)
        references carrera (id)
        ON DELETE NO ACTION
        ON UPDATE NO ACTION
) ENGINE = InnoDB;
```

Inserción de datos en SQL

Se procedió a realizar la inserción de datos dentro de las tablas. Algunos registros fueron realizados manualmente otros mediante importación de archivos CSV. Se adjunta en el documento un Script SQL con la inserción de los registros en las tablas. Los datos que se incorporan al proyecto corresponden a alumno, profesor, materia, becas, carrera. El proceso que se realizó fue el siguiente:

```sql
-- INSERSION DE DATOS EN LA TABLA DE CARRERA
insert into carrera (id, nombreCarrera, duracionCarrera) values (1, 'Ingeniería en sistemas', '5 años');
insert into carrera (id, nombreCarrera, duracionCarrera) values (2, 'Ciencias Agropecuarias', '4 años');
insert into carrera (id, nombreCarrera, duracionCarrera) values (3, 'Ingeniería en Telecomunicaciones', '5 años')
select * from carrera;
```

```sql
-- INSERSION DE DATOS EN LA TABLA DE ALUMNOS

insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Jandry Anchundia', 23, 'DECIMO', 'MASCULINO', 1);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Jhon Rosado', 23, 'DECIMO', 'MASCULINO', 1);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Jose Rolando', 23, 'DECIMO', 'MASCULINO', 1);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Ben Mell', 23, 'DECIMO', 'MASCULINO', 1);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Jose David', 23, 'DECIMO', 'MASCULINO', 1);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Emanuel Delgado', 23, 'DECIMO', 'MASCULINO', 1);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Jhon Rosado', 23, 'DECIMO', 'MASCULINO', 2);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Robert Rosado', 23, 'DECIMO', 'MASCULINO', 1);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Orlando Rosado', 23, 'DECIMO', 'MASCULINO', 3);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Emanuel Delgado', 23, 'DECIMO', 'MASCULINO', 1);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Jhon Rosado', 23, 'DECIMO', 'MASCULINO', 2);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Robert Rosado', 23, 'DECIMO', 'MASCULINO', 2);
insert into alumno (id, nombre, edad, semestre, genero, idCarrera  ) values (0, 'Orlando Rosado', 23, 'DECIMO', 'MASCULINO', 2);
select * from alumno;
```

```
243 •  insert into alumnoMateria (id, idAlumno, idMateria, promedioParcial_1, promedioParcial_2) values(0, 2, 3, 3, 9);
244 •  insert into alumnoMateria (id, idAlumno, idMateria, promedioParcial_1, promedioParcial_2) values(0, 1, 1, 7, 9);
245 •  insert into alumnoMateria (id, idAlumno, idMateria, promedioParcial_1, promedioParcial_2) values(0, 3, 1, 7, 9);
246 •  insert into alumnoMateria (id, idAlumno, idMateria, promedioParcial_1, promedioParcial_2) values(0, 1, 3, 2, 9);
247 •  insert into alumnoMateria (id, idAlumno, idMateria, promedioParcial_1, promedioParcial_2) values(0, 2, 3, 7, 9);
248
```

```
141     -- INSERSION DE DATOS SOBRE EL TIEMPO EN QUE LLEVA SIENDO BENEFICARIO DE LA BECA
142
143 •   insert into alumnoBecas (id, idBecas, idAlumno, fechaInicio ) values(0, 1, 1, '2019-12-31');
144 •   insert into alumnoBecas (id, idBecas, idAlumno, fechaInicio ) values(0, 1, 2, '2019-12-31');
145 •   insert into alumnoBecas (id, idBecas, idAlumno, fechaInicio ) values(0, 1, 3, '2019-12-31');
146 •   insert into alumnoBecas (id, idBecas, idAlumno, fechaInicio ) values(0, 1, 3, '2019-12-31');
147 •   insert into alumnoBecas (id, idBecas, idAlumno, fechaInicio ) values(0, 2, 4, '2019-12-31');
148 •   insert into alumnoBecas (id, idBecas, idAlumno, fechaInicio ) values(0, 3, 5, '2019-12-31');
149 •   insert into alumnoBecas (id, idBecas, idAlumno, fechaInicio ) values(0, 2, 6, '2019-12-31');
150 •   insert into alumnoBecas (id, idBecas, idAlumno, fechaInicio ) values(0, 1, 7, '2019-12-31');
151 •   select *  from alumnoBecas;
152
```

```
85 •   insert into becas (id, tipoBeca) values (2, 'ALUMNO DESTACADO');
```

```
176     -- INSERSION DE DATOS EN LA TABLA DE PROFESOR
177 •   insert into profesor (id, nombreProfesor, direccion, genero, telefono, horario  ) values (0, 'ELMER Anchundia', '9 de Octubre', 'MASCULINO', '0969849494', 'DE 7AM A 1PM');
178 •   insert into profesor (id, nombreProfesor, direccion, genero, telefono, horario  ) values (0, 'DAVID Anchundia', '9 de Octubre', 'MASCULINO', '0969849494', 'DE 7AM A 1PM');
179 •   insert into profesor (id, nombreProfesor, direccion, genero, telefono, horario  ) values (0, 'JOSÉ Anchundia', '9 de Octubre', 'MASCULINO', '0969849494', 'DE 7AM A 1PM');
180 •   insert into profesor (id, nombreProfesor, direccion, genero, telefono, horario  ) values (0, 'MIGUEL Anchundia', '9 de Octubre', 'MASCULINO', '0969849494', 'DE 7AM A 1PM');
181 •   insert into profesor (id, nombreProfesor, direccion, genero, telefono, horario  ) values (0, 'Jandry Anchundia', '9 de Octubre', 'MASCULINO', '0969849494', 'DE 7AM A 1PM');
182 •   insert into profesor (id, nombreProfesor, direccion, genero, telefono, horario  ) values (0, 'Jandry Anchundia', '9 de Octubre', 'MASCULINO', '0969849494', 'DE 7AM A 1PM');
183 •   insert into profesor (id, nombreProfesor, direccion, genero, telefono, horario  ) values (0, 'Jandry Anchundia', '9 de Octubre', 'MASCULINO', '0969849494', 'DE 7AM A 1PM');
184 •   insert into profesor (id, nombreProfesor, direccion, genero, telefono, horario  ) values (0, 'Jandry Anchundia', '9 de Octubre', 'MASCULINO', '0969849494', 'DE 7AM A 1PM');
```

Creación de Vistas

Una vista es una tabla virtual que se genera a partir de la ejecución de una o más consultas SQL, aplicada sobre una o más tablas. En este apartado se realizaron diversas Vistas de consultas a la base de datos del proyecto. Se adjunta en el documento un Script SQL con las siguientes vistas:

```
4     -- ---------------------------------------
5     -- Creacion de vitas
6     -- ---------------------------------------
7     -- Vista sobre la carrera que sigue un alumno
8     create or replace view alumnoCarreras as
9     select alumno.nombre, carrera.duracionCarrera, carrera.nombreCarrera
10    from alumno
11    inner join carrera on alumno.idCarrera = carrera.id;
12    select * from alumnoCarreras ;
15     -- Vista sobre informacion del alumno
16    create or replace view informacionAlumnos as
17    select id, nombre, genero, edad from alumno;
18    select * from informacionAlumnos;
19
20
36    -- Vista sobre los profesores y los alumnos
37    create or replace view alumnoProfesor as
38    select nombre as nombreAlumno, nombreProfesor
39    from alumno, profesor;
40    select * from alumnoProfesor;
41
```

```sql
-- Vista sobre los alumnos que tienen becas
create or replace view informacionAlumnosBeca as
select alumno.nombre, becas.tipoBeca
from alumno, becas
inner join alumnobecas on alumnobecas.idBecas = alumnoBecas.idAlumno;
select * from informacionAlumnosBeca;


-- vista de las materias

create or replace view informacionMateria as
select nombreMateria, creditosMateria
from materia;
select * from  informacionMateria;

```

Creación de Funciones

Las funciones personalizadas o almacenadas de Mysql permiten procesar y manipular datos de forma procedural y eficiente. Dichos datos son enviados a través de uno o más parámetros, al momento de invocar la función, y devueltos como resultado por esta misma. Se adjunta en el documento un Script SQL con las siguientes funciones.

```sql
-- Funcion que ayude a concer cuantos alumnos hay con determinada letra
drop function if exists fn_numeroLetras;
delimiter $$
use alumnos $$
create function fn_numeroLetras (letra char) returns int
no sql
begin
    declare numero int;
    select count(*) into numero
    from alumno
    where nombre like concat('%',letra,'%');
    return numero;
end$$
delimiter $$;

select * from alumno;
select alumnos.fn_numeroLetras('a');
```

```sql
-- Funcion que perimite calcular el promedio de los 2 parciales de los alumnos
drop function if exists fn_calcularPromedioParciales;
delimiter $$
use alumnos $$
create function fn_calcularPromedioParciales (p_parcial1 int, p_parcial2 int) returns float
no sql
begin
declare resultado float;
set resultado = (p_parcial1 + p_parcial2) / 2;
return resultado;
end$$
delimiter $$;

select alumnos.fn_calcularPromedioParciales(7, 7);
```

Procedimientos Almacenados en SQL

Un Procedimiento Almacenado o Stored Procedure es un programa almacenado físicamente en una base de datos, creado para cumplir tareas específicas. Permite también establecer niveles de seguridad y manipular operaciones complejas o extensas del lado del servidor, evitando un ida y vuelta de datos que termine sobrecargando una red o servidor. Se adjunta en el documento un Script SQL con los siguientes Procesos almacenados que se ejecutan sobre la base de datos del proyecto.

```sql
-- Este Script permite la actualizacion de los alumnos
use alumnos;
create procedure actualizaAlumnos(nombreNuevo varchar(60), edadNueva int, idAlumno int )
update alumno set  nombre = nombreNuevo, edad = edadNueva where id = idAlumno;

call alumnos.actualizaAlumnos('Jose Juaquin', 200, 1);
select * from alumno;
```

```sql
#PS 2: Permite al usuario seleccionar la columna por la que quiere realizar
-- el ordenamiento, el orden (ASCENDENTE O DESCENDENTE) y finalmente la tabla que desea consultar.
DELIMITER $$
CREATE PROCEDURE ordenamientoTablas(IN campo VARCHAR(255),
IN tipo_ordenamiento ENUM('ASC','DESC'), IN tabla VARCHAR(255))
BEGIN
    IF campo <> '' THEN
        SET @ordenar = CONCAT(' ORDER BY ', campo);
    ELSE
        SET @ordenar = '';
    END IF;
    IF tipo_ordenamiento <> '' THEN
        SET @tipo = CONCAT(' ', tipo_ordenamiento);
    ELSE
        SET @tipo = '';
    END IF;
    SET @clausula = CONCAT('SELECT * FROM ',Tabla,@ordenar, @tipo);
    PREPARE ejecutarSQL FROM @clausula;
    EXECUTE ejecutarSQL;
    DEALLOCATE PREPARE ejecutarSQL;
END;
DELIMITER$$
```

Triggers en SQL

Un Trigger puede definirse como una aplicación o programa almacenado en el servidor (de base de datos) creado para ejecutarse de forma automática, cuando uno o más eventos específicos ocurren en la base de datos. Se adjunta en el documento un Script SQL con los siguientes Triggers.

```sql
#TRIGGER 1: Este trigger genera un registro luego de que se insertan
-- datos correspondientes a la tabla

CREATE TABLE IF NOT EXISTS Auditoria_Alumno(
                                        id_auditoria INT AUTO_INCREMENT PRIMARY KEY,
                                        fecha DATETIME NOT NULL,
                                        usuario VARCHAR(255));


CREATE TRIGGER `Aft_Ins_Alumno`AFTER INSERT ON `alumno` FOR EACH ROW
INSERT INTO `Auditoria_Alumno` (fecha, usuario)
VALUES (NOW(), USER(), 'INSERT');
```

```sql
#TRIGGER 2: El siguiente trigger guarda en una tabla "auditoria_becas"
-- las nuevas becas que se ingresen al sistema, guardando también la fecha y usuario encargado

CREATE TABLE IF NOT EXISTS Auditoria_Becas(
                                        ID_Beca INT auto_increment primary key,
                                        NombreBeca VARCHAR(255) NOT NULL,
                                        Fecha DATETIME NOT NULL,
                                        Usuario VARCHAR(255));

CREATE TRIGGER `Bef_Ins_Beca`
BEFORE INSERT ON `becas` FOR EACH ROW
INSERT INTO `Auditoria_Becas` (ID_Beca, NombreBeca)
VALUES (NEW.tipoBeca, NOW(), USER(), 'Insert');
```

Data Control Language

El Lenguaje de Control de Datos (DCL) permite definir diferentes usuarios dentro del motor de base de datos Mysql, y establecer para cada uno de ellos, permisos totales, parciales, o negar el acceso sobre los diferentes Objetos que conforman la Base de Datos.
Para el proyecto se utilizó este lenguaje para la creación de dos usuarios, con diferentes permisos:

• Otorgar permisos de solo lectura a todas las tablas al Usuario1.

• Otorgar permisos de lectura, inserción y modificación a todas las tablas al Usuario2.

• Ambos usuarios no pueden eliminar registros.

Se adjunta un Script SQL con la creación de los dos usuarios y los permisos correspondientes a cada uno:

```
1  USE mysql;
2  SELECT * FROM USER;
3  #Creación de los usuarios
4  CREATE USER Usuario1@localhost IDENTIFIED BY 'Test2021';
5  CREATE USER Usuario2@localhost IDENTIFIED BY 'Test2022';
6
7  #Otorgar permisos de solo lectura a todas las tablas al Usuario1:
8
9  GRANT SELECT ON *.* TO Usuario1@localhost;
10
11 #Otorgar permisos de lectura, inserción y modificación a todas las tablas al Usuario2:
12
13 GRANT SELECT, INSERT, UPDATE ON *.* TO Usuario2@localhost;
14
15 #Aseguramos que ambos usuarios no pueden eliminar registros:
16
17 REVOKE DELETE ON *.* FROM Usuario1@localhost, Usuario2@localhos;
18
19 SELECT * FROM USER;
```

Transaction Control Language

Se conoce como Transaction Control Language (o TCL) a una parte del lenguaje Transact-SQL que administra las transacciones en una base de datos.

TCL es utilizado para administrar cada uno de los cambios que se generan en una o más tablas mediante las cláusulas DML.

```
1   #Eliminar registros y recuperarlos
2
3   START TRANSACTION;
4       DELETE FROM alumnos.alumnomateria WHERE idMateria = 1;
5       DELETE FROM alumnos.alumnomateria WHERE idMateria  = 2;
6       DELETE FROM alumnos.alumnomateria WHERE idMateria  = 3;
7       DELETE FROM alumnos.alumnomateria WHERE idMateria  = 4;
8       DELETE FROM alumnos.alumnomateria WHERE idMateria  = 5;
9
10
11  #Para deshacer las eliminaciones ejecuto:
12  ROLLBACK;
13
14  #Eliminar registros definitivamente
15
16  START TRANSACTION;
17      DELETE FROM alumnos.alumnobecas WHERE idAlumno = 1;
18      DELETE FROM alumnos.alumnobecas WHERE idAlumno = 2;
19      DELETE FROM alumnos.alumnobecas WHERE idAlumno = 3;
20      DELETE FROM alumnos.alumnobecas WHERE idAlumno = 4;
21      DELETE FROM alumnos.alumnobecas WHERE idAlumno = 5;
22  #Para confirmar definitivamente las eliminaciones ejecuto:
23  COMMIT;
24
```

```
28  START TRANSACTION;
29      INSERT INTO alumnos.alumnomateria (`id`,`idAlumno`,`idMateria`,`promedioParcial_1`,`promedioParcial_2`)
30      VALUES ('0','1','2',10,10);
31      INSERT INTO alumnos.alumnomateria (`id`,`idAlumno`,`idMateria`,`promedioParcial_1`,`promedioParcial_2`)
32      VALUES ('0','2','2',10,10);
33      INSERT INTO alumnos.alumnomateria (`id`,`idAlumno`,`idMateria`,`promedioParcial_1`,`promedioParcial_2`)
34      VALUES ('0','3','2',10,10);
35      INSERT INTO alumnos.alumnomateria (`id`,`idAlumno`,`idMateria`,`promedioParcial_1`,`promedioParcial_2`)
36      VALUES ('0','4','2',10,10);
37
38  SAVEPOINT CuatroRegistros;
39      INSERT INTO alumnos.alumnobecas (`id`,`idBecas`,`idAlumno`,`fechaInicio`)
40      VALUES (0,1,2,'2020-12-20');
41      INSERT INTO alumnos.alumnobecas (`id`,`idBecas`,`idAlumno`,`fechaInicio`)
42      VALUES (0,1,2,'2020-12-20');
43      INSERT INTO alumnos.alumnobecas (`id`,`idBecas`,`idAlumno`,`fechaInicio`)
44      VALUES (0,1,2,'2020-12-20');
45      INSERT INTO alumnos.alumnobecas (`id`,`idBecas`,`idAlumno`,`fechaInicio`)
46      VALUES (0,1,2,'2020-12-20');
47      INSERT INTO alumnos.alumnobecas (`id`,`idBecas`,`idAlumno`,`fechaInicio`)
48      VALUES (0,1,2,'2020-12-20');
49      INSERT INTO alumnos.alumnobecas (`id`,`idBecas`,`idAlumno`,`fechaInicio`)
50      VALUES (0,1,2,'2020-12-20');
51      INSERT INTO alumnos.alumnobecas (`id`,`idBecas`,`idAlumno`,`fechaInicio`)
```

Backup y restauración

En este apartado lo que se realizó fue una copia de los datos originales de la base de datos que se realiza con el fin de disponer de un medio para recuperarlos en caso de una falla o pérdida. El gestor de bases de datos MySQL permite dos métodos para realizar backups de la información:

```
1
2     # ENTREGABLE 11: BACKUP Y RESTAURACIÓN
3
4     # Las tablas que se seleccionaron para realizar el backup son las siguientes:
5     # "Alumno", "Carrera", "Becas", "Pofesores", "Materia"
6
7     -- MySQL dump 10.13  Distrib 8.0.29, for Win64 (x86_64)
8     --
9     -- Host: localhost    Database: alumnos
10    -- ------------------------------------------------------
11    -- Server version   8.0.29
12
13    /*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
14    /*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
15    /*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
16    /*!50503 SET NAMES utf8 */;
17    /*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
18    /*!40103 SET TIME_ZONE='+00:00' */;
19    /*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
20    /*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;
21    /*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
```

```
28    LOCK TABLES `alumno` WRITE;
29    /*!40000 ALTER TABLE `alumno` DISABLE KEYS */;
30    INSERT INTO `alumno` VALUES (1,'Jose Juaquin',200,'DECIMO','MASCULINO',1),(2,'Jhon Rosado',23,'DECIMO','MASCULINO',
31    /*!40000 ALTER TABLE `alumno` ENABLE KEYS */;
32    UNLOCK TABLES;
33
34    --
35    -- Dumping data for table `becas`
36    --
37
38    LOCK TABLES `becas` WRITE;
39    /*!40000 ALTER TABLE `becas` DISABLE KEYS */;
40    INSERT INTO `becas` VALUES (1,'SOCIOECONIMICA'),(2,'ALUMNO DESTACADO'),(3,'TRANSPORTE');
41    /*!40000 ALTER TABLE `becas` ENABLE KEYS */;
42    UNLOCK TABLES;
43
44    --
45    -- Dumping data for table `carrera`
46    --
47
48    LOCK TABLES `carrera` WRITE;
```

Herramientas y tecnologías utilizadas

A continuación, se realiza un listado de las principales tecnologías empleadas para la realización de este proyecto:

- **draw.io**, es una aplicación web para generar diagramas de flujo, gráficos y esquemas sin tener necesidad incluso de instalar ningún software: Se utilizó para la creación de los esquemas y llenarlos de descripciones que pudieran facilitar la comprensión
- Microsoft Excel: se empleó para realizar transformaciones a los datos, depurar y prepararlos para luego poder ser importados a la base de datos en MySQL.
- MySQL Workbench: se utilizó este gestor de bases de datos para la realización del Schema del proyecto, sus tablas y registros. Como así también las vistas, procesos almacenados y triggers entre otras aplicaciones que nos permitió desarrollar el programa a lo largo del proyecto.