



FACULTAD DE CIENCIAS
FUNDAMENTOS DE BASES DE DATOS

Proyecto Final

Semestre 2024 – 1

Alumno

Marco Silva Huerta

Profesor:

Víctor Manuel Corza Vargas

Ayudantes:

Alexis Hernández Castro

Diana Irma Canchola Hernández

Gibrán Aguilar Zuñiga

Oscar José Hernández Sánchez

11 de Diciembre de 2023

Índice

1. Escuela

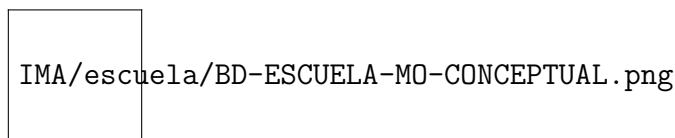
1.1. Requerimientos

Saludos desde Hogwarts Escuela de Magia y Hechicería. Después de revisar exhaustivamente sus habilidades y experiencia en el mundo mágico de la programación, nos complace informarles que han sido seleccionados para crear y gestionar la base de datos mágica de nuestra venerable institución.

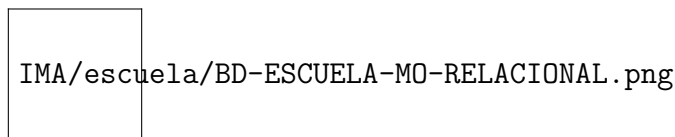
Principales restricciones

- La clave primaria en cada tabla debe ser única y no nula.
- Las claves foráneas deben hacer referencia a claves primarias existentes en las tablas referenciadas.
- La columna de Fecha de Nacimiento en las tablas Alumnos y Maestros debe contener fechas válidas.
- La columna de Género en las tablas Alumnos y Maestros debe contener valores específicos: 'Masculino' o 'Femenino'.
- La columna de Casa en la tabla Alumnos debe contener valores específicos: 'Gryffindor', 'Hufflepuff', 'Ravenclaw' o 'Slytherin'.
- La columna de Año Académico en la tabla Cursos debe contener valores específicos: '1ro', '2do', '3ro', '4to', '5to', '6to'.
- La columna de Nombre y Apellido en las tablas Alumnos y Maestros debe contener texto no nulo y no vacío.
- La tabla Asignaturas debe tener una relación adecuada entre las claves foráneas y la información del profesor y el curso.
- La columna de Localidad en la tabla Ubicacion debe contener valores no nulos y no vacíos.
- Cada año escolar tiene un curso asociado

1.2. Modelo conceptual



1.3. Modelo relacional



1.4. Script de creación

```
1  -- Solo los titulos de las tablas para escuela
2  CREATE TABLE UBICACION
3  (
4
5  );
6
7  CREATE TABLE MAESTROS
8  (
9
10 );
11
12 CREATE TABLE CURSOS
13 (
14
15 );
16
17 CREATE TABLE ASIGNATURAS
18 (
19
20 );
21
22 CREATE TABLE ALUMNOS
23 (
24
25 );
```

Listing 1: Tablas para la BdDatos

1.5. Script de Insert

→ Se deben generar 100 registros para cada tabla.

→ Si para el buen funcionamiento de la base de datos se requieren más de 100 registros o menos de 100 registros en una tabla, se debe explicar claramente la razón, sólo en este caso sí se debe incluir un apartado en el reporte final.

Tablas:

- UBICACION: Si es posible llegar a 100 registros de ubicación
- MAESTROS: No es posible llegar a 100 porque la tabla solo contempla 24 maestros registrados
- CURSOS: No es posible llegar a 100 porque solo se contemplan 6 cursos para la escuela
- ASIGNATURAS: No es posible llegar a 100 porque hay 7 asignaturas por cada curso, son 24 asignaturas
- Alumnos: Si es posible llegar a 100 registros de alumnos

1.6. Funcionamiento Restricciones


Evidencia del funcionamiento de al menos 4 restricciones de integridad referencial.

Restricción 01

- Tablas involucradas en la restricción: ALUMNOS y UBICACION
- FK de la tabla que referencia y PK de la tabla referenciada: ubicacionID en ALUMNOS es una clave foránea que referencia a ubicacionID en UBICACION
- Justificación del trigger de integridad referencial elegido: Esta restricción asegura que cada alumno esté asociado con una ubicación válida en la tabla UBICACION. Si se intenta eliminar una ubicación que está siendo referenciada por un alumno, o se intenta actualizar el ubicacionID de una ubicación a un valor que no existe en la tabla ALUMNOS, la operación será rechazada.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM UBICACION WHERE ubicacionID = 'OXFU';
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.


IMA/escuela/H-6-01.png

Restricción 02

- Tablas involucradas en la restricción: CURSOS y MAESTROS
- FK de la tabla que referencia y PK de la tabla referenciada: maestroID en CURSOS es una clave foránea que referencia a maestroID en MAESTROS
- Justificación del trigger de integridad referencial elegido: Esta restricción asegura que cada curso esté asociado con un maestro válido en la tabla MAESTROS. Si se intenta eliminar un maestro que está siendo referenciado por un curso, o se intenta actualizar el maestroID de un maestro a un valor que no existe en la tabla CURSOS, la operación será rechazada.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM MAESTROS WHERE maestroID = 'GG001';
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

IMA/escuela/H-6-02.png

Restricción 03

- Tablas involucradas en la restricción: ASIGNATURAS y CURSOS
- FK de la tabla que referencia y PK de la tabla referenciada: cursoID en ASIGNATURAS es una clave foránea que referencia a cursoID en CURSOS
- Justificación del trigger de integridad referencial elegido: Esta restricción asegura que cada asignatura esté asociada con un curso válido en la tabla CURSOS. Si se intenta eliminar un curso que está siendo referenciado por una asignatura, o se intenta actualizar el cursoID de un curso a un valor que no existe en la tabla ASIGNATURAS, la operación será rechazada.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM CURSOS WHERE cursoID = 'A1IM';
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

IMA/escuela/H-6-03.png

Restricción 03

- Tablas involucradas en la restricción: ALUMNOS y CURSOS
- FK de la tabla que referencia y PK de la tabla referenciada: cursoID en ALUMNOS es una clave foránea que referencia a cursoID en CURSOS
- Justificación del trigger de integridad referencial elegido: Esta restricción asegura que cada alumno esté asociado con un curso válido en la tabla CURSOS. Si se intenta eliminar un curso que está siendo referenciado por un alumno, o se intenta actualizar el cursoID de un curso a un valor que no existe en la tabla ALUMNOS, la operación será rechazada.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
UPDATE ALUMNOS  
SET cursoID = 'A7SA'  
WHERE alumnoID = 'S29';
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

IMA/escuela/H-6-04.png

1.7. Funcionamiento Restricciones check

Evidencia del funcionamiento de al menos 3 restricciones check para “atributos” de varias tablas.


- Tabla elegida: ALUMNOS
- Atributo elegido: casa
- Descripción: Esta restricción asegura que el valor de la columna Casa sea uno de los cuatro nombres de las casas de Hogwarts: 'Hufflepuff', 'Ravenclaw', 'Gryffindor', 'Slytherin'.
- Instrucción para la creación de la restricción:

```
ALTER TABLE ALUMNOS
ADD CONSTRAINT casaCasa CHECK (Casa IN ('Hufflepuff',
                                         'Ravenclaw',
                                         'Gryffindor',
                                         'Slytherin'));
```

- Instrucción que permita evidenciar que la restricción esta funcionando:

```
INSERT INTO ALUMNOS (alumnoID, Nombre, Apellido, Casa, fechaNacimiento,
                    Genero, ubicacionID, cursoID)
VALUES ('nuevoID', 'Nombre', 'Apellido', 'CasaIncorrecta',
        '2000-01-01', 'Genero', 'ubicacionID', 'cursoID');
```

- Captura de pantalla

IMA/escuela/H-7-01.png

- Tabla elegida: MAESTROS
- Atributo elegido: fechaNacimiento
- Descripción: Esta restricción asegura que el valor de la columna fechaNacimiento sea una fecha válida y no sea en el futuro.
- Instrucción para la creación de la restricción:

```
ALTER TABLE MAESTROS
ADD CONSTRAINT fechaMaestro CHECK (fechaNacimiento <= CURRENT_DATE);
```

- Instrucción que permita evidenciar que la restricción esta funcionando:

```
INSERT INTO MAESTROS
    (maestroID, Nombre, Apellido, Especialidad, fechaNacimiento,
     Genero, ubicacionID)
VALUES ('nuevoID', 'Nombre', 'Apellido', 'Especialidad',
        '3000-01-01', 'Genero', 'ubicacionID');
```

- Tabla elegida: ALUMNOS
- Atributo elegido: fechaNacimiento
- Breve descripción de la restricción: Esta restricción asegura que el valor de la columna fechaNacimiento sea una fecha válida y no sea en el futuro.
- Instrucción para la creación de la restricción:

```
ALTER TABLE ALUMNOS
ADD CONSTRAINT fechaAlumno CHECK (fechaNacimiento <= CURRENT_DATE);
```

- Instrucción que permita evidenciar que la restricción esta funcionando:

```
INSERT INTO ALUMNOS (alumnoID, Nombre, Apellido, Casa, fechaNacimiento,
                     Genero, ubicacionID, cursoID)
VALUES ('nuevoID', 'Nombre', 'Apellido', 'Hufflepuff', '3000-01-01',
        'Genero', 'ubicacionID', 'cursoID');
```

1.8. Creación de dominios personalizados

Evidencia de la creación de al menos tres dominios personalizados. Se deben utilizar restricciones check en la creación de los tres dominios.

- Tabla elegida: ALUMNOS
- Atributo elegido: Casa
- Descripción: El dominio Casa representa las cuatro casas de la escuela Hogwarts de Magia y Hechicería. Las cuatro casas son: Hufflepuff Ravenclaw Gryffindor Slytherin
- Instrucción para la creación del dominio personalizado.

```
1 CREATE DOMAIN Casa
2 AS CHAR(20)
3 CHECK (
4     Casa IN ('Hufflepuff', 'Ravenclaw',
5             'Gryffindor', 'Slytherin')
6 );
```

Listing 2: Tablas para la BdDatos

- Tabla elegida: MAESTROS
- Atributo elegido: genero
- Descripción: El dominio Genero representa el género de un maestro. Los valores permitidos son: Masculino, Femenino
- Instrucción para la creación del dominio personalizado.

```
1      CREATE DOMAIN Genero
2      AS CHAR(20)
3      CHECK (
4          Genero IN ('Masculino', 'Femenino')
5      );
```

Listing 3: Tablas para la BdDatos

- Tabla elegida: MAESTROS
- Atributo elegido: especialidad
- Descripción: El dominio Especialidad representa la especialidad de un maestro. La restricción check propuesta indica que el valor de este atributo debe ser una cadena de caracteres de al menos 3 caracteres y no más de 50 caracteres.
- Instrucción para la creación del dominio personalizado.

```
1      USE [EscuelaDeMagia]
2      GO
3
4      CREATE DOMAIN [Especialidad] AS VARCHAR(50)
5      CHECK (LENGTH(VALUE) >= 3 AND LENGTH(VALUE) <= 50)
6      GO
```

Listing 4: Tablas para la BdDatos

1.9. Restricciones para tuplas

Evidencia del funcionamiento de al menos 2 restricciones para “tuplas” en diferentes tablas (Unidad 8 Integridad, tema Specifying Constraints on Tuples Using CHECK)

- Tabla elegida: ALUMNOS
- Breve descripción de la restricción: Esta restricción asegura que los alumnos no puedan ser mayores de 100 años. Para ello, se verifica que la fecha de nacimiento sea posterior al año actual menos 100.
- Instrucción para la creación de la restricción:

```
ALTER TABLE ALUMNOS
ADD CONSTRAINT edadAlumno
CHECK (fechaNacimiento > DATE_SUB(CURRENT_DATE, INTERVAL 100 YEAR));
```

- Instrucción Insert o Update que permita evidenciar que la restricción esta funcionando:

```
INSERT INTO ALUMNOS (alumnoID, Nombre, Apellido, Casa, fechaNacimiento, Genero, ubic
VALUES ('nuevoID', 'Nombre', 'Apellido', 'Hufflepuff', '1900-01-01', 'Genero', 'ubic
```

- Tabla elegida: ‘MAESTROS’
- Breve descripción de la restricción: Esta restricción asegura que los maestros no puedan ser menores de 25 años. Para ello, se verifica que la fecha de nacimiento sea anterior al año actual menos 25.
- Instrucción para la creación de la restricción:

```
ALTER TABLE MAESTROS
ADD CONSTRAINT edadMaestro
CHECK (fechaNacimiento < DATE_SUB(CURRENT_DATE, INTERVAL 25 YEAR));
```

- Instrucción Insert o Update que permita evidenciar que la restricción esta funcionando:

```
INSERT INTO MAESTROS (maestroID, Nombre, Apellido, Especialidad, fechaNacimiento, Gene
VALUES ('nuevoID', 'Nombre', 'Apellido', 'Especialidad', '2000-01-01', 'Genero', 'ubic
```

1.10. Consultas

Plantea 3 consultas que consideres relevantes para la base de datos propuesta. Para cada consulta planteada, incluir en el reporte los siguientes incisos:

Consulta 01

- Redacción clara de la consulta: Obtener una lista de todos los alumnos que están en la casa Gryffindor.
- Código en lenguaje SQL de la consulta.

```
1      SELECT Nombre, Apellido
2      FROM ALUMNOS
3      WHERE Casa = 'Slytherin';
```

Listing 5: Tablas para la BdDatos


IMA/escuela/consu01.png

Consulta 02

- Redacción clara de la consulta: Obtener una lista de todos los cursos que son impartidos por un maestro específico.
- Código en lenguaje SQL de la consulta.

```
1      SELECT nombreCurso
2      FROM CURSOS
3      WHERE maestroID = 'GG001';
```

Listing 6: Tablas para la BdDatos




IMA/escuela/consu02.png

Consulta 03

- Redacción clara de la consulta: Obtener una lista de todos los maestros que viven en una ubicación específica.
- Código en lenguaje SQL de la consulta.

```
1      SELECT Nombre, Apellido
2      FROM MAESTROS
3      WHERE ubicacionID = 'NEWC';
```

Listing 7: Tablas para la BdDatos



IMA/escuela/consu03.png

1.11. Vistas

Plantea 3 vistas que consideres relevantes para la base de datos propuesta. Para cada vista planteada, incluir en el reporte los siguientes incisos:

Vista 01

- Redacción clara de la vista planteada: Una vista que muestre todos los alumnos junto con la información de su ubicación.
- Código en lenguaje SQL que permita crear la vista solicitada.


```
1 CREATE VIEW AlumnosUbicacion AS
2 SELECT ALUMNOS.Nombre, ALUMNOS.Apellido, UBICACION.Localidad,
   UBICACION.Pais
3 FROM ALUMNOS
4 JOIN UBICACION ON ALUMNOS.ubicacionID = UBICACION.ubicacionID
   ;
```

Listing 8: Tablas para la BdDatos

■ USO:

```
1 SELECT * FROM AlumnosUbicacion;
```

Listing 9: Tablas para la BdDatos

 IMA/escuela/vista01.png

Vista 02

- Redacción clara de la vista planteada: Una vista que muestre todos los cursos junto con el nombre del maestro que los imparte.
- Código en lenguaje SQL que permita crear la vista solicitada.


```
1 CREATE VIEW CursosMaestros AS
2 SELECT CURSOS.nombreCurso, MAESTROS.Nombre, MAESTROS.Apellido
3 FROM CURSOS
4 JOIN MAESTROS ON CURSOS.maestroID = MAESTROS.maestroID;
```

Listing 10: Tablas para la BdDatos

■ USO:

```
1 SELECT * FROM CursosMaestros;
```

Listing 11: Tablas para la BdDatos

 IMA/escuela/vista02.png

Vista 03

- Redacción clara de la vista planteada: Una vista que muestre todas las asignaturas junto con el nombre del curso al que pertenecen.
- Código en lenguaje SQL que permita crear la vista solicitada.


```
1 CREATE VIEW AsignaturasCursos AS
2 SELECT ASIGNATURAS.nombreAsignatura, CURSOS.nombreCurso
3 FROM ASIGNATURAS
4 JOIN CURSOS ON ASIGNATURAS.cursoID = CURSOS.cursoID;
```

Listing 12: Tablas para la BdBdatos

■ USO:

```
1 SELECT * FROM AsignaturasCursos;
```

Listing 13: Tablas para la BdBdatos

 IMA/escuela/vista03.png

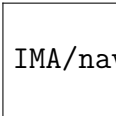
2. Naves

2.1. Requerimientos

La empresa Galactic Travel Solutions ha solicitado el diseño de una base de datos para gestionar los viajes galácticos en el año 4000. El objetivo es administrar eficientemente la información relacionada con pasajeros, naves, planetas, rutas y choferes para ofrecer servicios de transporte interplanetario de manera segura y organizada.

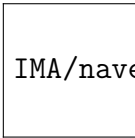
- Cada pasajero debe estar asociado a una única nave, y cada nave debe tener asignada una ruta específica.
- Los planetas deben tener información única y estar relacionados con las rutas para garantizar la coherencia de los viajes.
- Las relaciones entre las tablas deben ser mantenidas para asegurar la integridad referencial. Por ejemplo, un chofer debe estar asignado a una nave existente.
- Los identificadores primarios y foráneos deben seguir una convención clara y única para facilitar la comprensión y mantenimiento de la base de datos.
- Se deben establecer restricciones de integridad para prevenir operaciones que puedan comprometer la consistencia de los datos, como eliminar un planeta asociado a una ruta activa.

2.2. Modelo conceptual



IMA/nave/BD-NAVE-MO-CONCEPTUAL.png

2.3. Modelo relacional



IMA/nave/BD-NAVE-MO-RELACIONAL.png

2.4. Script de creación

```
1  -- Solo los titulos de las tablas para naves
2  CREATE TABLE PLANETAS
3  (
4
5  );
6
7  CREATE TABLE RUTAS
8  (
9
10 );
11
```

```
12
13 CREATE TABLE NAVES
14 (
15
16 );
17
18 CREATE TABLE CHOFERES
19 (
20
21 );
22
23 CREATE TABLE PASAJEROS
24 (
25
26 );
```

Listing 14: Tablas para la BdDatos

2.5. Script de Insert

- Se deben generar 100 registros para cada tabla.
- Si para el buen funcionamiento de la base de datos se requieren más de 100 registros o menos de 100 registros en una tabla, se debe explicar claramente la razón, sólo en este caso sí se debe incluir un apartado en el reporte final.

Tablas:

- PLANETAS: no es posible porque nos concentramos en los 8 planetas del sistema solar
- RUTAS: no es posible porque se toman las posibles rutas entre los 8 planetas, 28
- NAVES: no es posible porque se toman 28 naves para las 28 rutas
- CHOFERES: no es posible porque se toman 28 choferes para las 28 rutas

Es importante resaltar que para respetar las secuencias y no tener problemas con los ID se tuvo que realizar así la tabla

- PASAJEROS: Si es posible llegar a los 100 registros.

2.6. Funcionamiento Restricciones

Evidencia del funcionamiento de al menos 4 restricciones de integridad referencial.


Restricción 01

- Tablas involucradas en la restricción: NAVES y RUTAS
- FK de la tabla que referencia y PK de la tabla referenciada: rutaID en NAVES (FK) hace referencia a rutaID en RUTAS (PK)

- Justificación del trigger de integridad referencial elegido: Esta restricción asegura que cada rutaID en la tabla NAVES corresponda a un rutaID válido en la tabla RUTAS. Esto es crucial para mantener la coherencia de los datos, ya que no tendría sentido tener una nave asignada a una ruta que no existe.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM RUTAS WHERE rutaID = 'R1';
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.


IMA/nave/6-01.png

Restricción 02

- Tablas involucradas en la restricción: CHOFERES y NAVES
- FK de la tabla que referencia y PK de la tabla referenciada: naveID en CHOFERES (FK) hace referencia a naveID en NAVES (PK)
- Justificación del trigger de integridad referencial elegido: Esta restricción asegura que cada naveID en la tabla CHOFERES corresponda a un naveID válido en la tabla NAVES. Esto es crucial para mantener la coherencia de los datos, ya que no tendría sentido tener un chofer asignado a una nave que no existe.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM NAVES WHERE naveID = 'N19RR';
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

IMA/nave/6-02.png

Restricción 03

- Tablas involucradas en la restricción: PASAJEROS y NAVES
- FK de la tabla que referencia y PK de la tabla referenciada: naveID en PASAJEROS (FK) hace referencia a naveID en NAVES (PK)
- Justificación del trigger de integridad referencial elegido: Esta restricción asegura que cada naveID en la tabla PASAJEROS corresponda a un naveID válido en la tabla NAVES. Esto es crucial para mantener la coherencia de los datos, ya que no tendría sentido tener un pasajero asignado a una nave que no existe.

→ Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM NAVES WHERE naveID = 'N7SR';
```

→ Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

IMA/nave/6-03.png

Restricción 04

→ Tablas involucradas en la restricción: RUTAS y PLANETAS

→ FK de la tabla que referencia y PK de la tabla referenciada: origen y destino en RUTAS (FK) hacen referencia a planetaID en PLANETAS (PK)

→ Justificación del trigger de integridad referencial elegido: Esta restricción asegura que cada origen y destino en la tabla RUTAS corresponda a un planetaID válido en la tabla PLANETAS. Esto es crucial para mantener la coherencia de los datos, ya que no tendría sentido tener una ruta que comienza o termina en un planeta que no existe.

→ Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM PLANETAS WHERE planetaID = 'Tierra';
```

→ Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

IMA/nave/6-04.png

2.7. Funcionamiento Restricciones check

Evidencia del funcionamiento de al menos 3 restricciones check para “atributos” de varias tablas.

Restricción 01


- Tabla elegida: PLANETAS
- Atributo elegido: distanciaSol
- Descripción: Esta restricción asegura que la distancia de un planeta al sol sea siempre positiva.
- Instrucción para la creación de la restricción:

```
ALTER TABLE PLANETAS  
ADD CONSTRAINT distanciaSol CHECK (distanciaSol > 0);
```

- Instrucción que permita evidenciar que la restricción está funcionando:

```
INSERT INTO PLANETAS (planetaID, estacion, distanciaSol,  
                      gravedad, poblacion)  
VALUES ('Pluton', 'Deep', -100, 9.8, 1000000);
```

- Captura de pantalla

IMA/nave/7-01.png

Restricción 02


- Tabla elegida: NAVES
- Atributo elegido: capacidad
- Descripción: Esta restricción asegura que la capacidad de una nave sea siempre positiva.
- Instrucción para la creación de la restricción:

```
ALTER TABLE NAVES  
ADD CONSTRAINT capacidad CHECK (capacidad > 0);
```

- Instrucción que permita evidenciar que la restricción está funcionando:

```
INSERT INTO NAVES (naveID, modelo, capacidad, velocidadMax, rutaID)  
VALUES ('N002', 'Modelo1', -50, 1000, 'R001');
```

- Captura de pantalla

IMA/nave/7-02.png

Restricción 03


- Tabla elegida: CHOFERES
- Atributo elegido: edad
- Descripción: Esta restricción asegura que la edad de un chofer sea siempre positiva y menor que 100.
- Instrucción para la creación de la restricción:

```
ALTER TABLE CHOFERES  
ADD CONSTRAINT edad CHECK (edad > 0 AND edad < 100);
```

- Instrucción que permita evidenciar que la restricción está funcionando:

```
INSERT INTO CHOFERES (choferID, nombre, edad, genero, nacionalidad, naveID)  
VALUES ('C002', 'Chofer1', -30, 'M', 'Noruega', 'N001');
```

- Captura de pantalla

IMA/nave/7-03.png

2.8. Creación de dominios personalizados

Evidencia de la creación de al menos tres dominios personalizados. Se deben utilizar restricciones check en la creación de los tres dominios.

Dominio 01

- Tabla elegida: PLANETAS
- Atributo elegido: distanciaSol
- Descripción: Este dominio personalizado asegura que la distancia de un planeta al sol sea siempre positiva y menor que 1000000 (un millón de unidades de distancia).
- Instrucción para la creación del dominio personalizado.

```
1  distanciaSol INT CHECK (distanciaSol > 0 AND distanciaSol <  
    1000000),
```

Listing 15: Dominio 01

Dominio 02

- Tabla elegida: NAVES
- Atributo elegido: capacidad
- Descripción: Este dominio personalizado asegura que la capacidad de una nave sea siempre positiva y menor que 1000.
- Instrucción para la creación del dominio personalizado.

```
1  CREATE DOMAIN capacidad AS INT CHECK (VALUE > 0 AND VALUE <  
    300);
```

Listing 16: Dominio 02

Dominio 03

- Tabla elegida: CHOFERES
- Atributo elegido: edad
- Descripción: Este dominio personalizado asegura que la edad de un chofer sea siempre positiva y menor que 100.
- Instrucción para la creación del dominio personalizado.

```
1 CREATE DOMAIN EdadChofer AS INT CHECK (VALUE > 0 AND VALUE < 100);
```

Listing 17: Dominio 03

2.9. Restricciones para tuplas

Evidencia del funcionamiento de al menos 2 restricciones para tuplas en diferentes tablas (Unidad 8 Integridad, tema Specifying Constraints on Tuples Using CHECK)


Restricción 01

- Tabla elegida: NAVES
- Breve descripción de la restricción: Esta restricción asegura que la capacidad de una nave siempre sea menor que su velocidadMax.
- Instrucción para la creación de la restricción:

```
ALTER TABLE NAVES  
ADD CONSTRAINT capacidad_velocidad CHECK (capacidad < velocidadMax);
```

- Instrucción Insert o Update que permita evidenciar que la restricción está funcionando:

```
INSERT INTO NAVES (naveID, modelo, capacidad, velocidadMax, rutaID)  
VALUES ('N003', 'Modelo2', 2000, 1000, 'R001');
```

 IMA/nave/tu-01.png

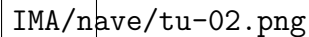
Restricción 02

- Tabla elegida: RUTAS
- Breve descripción de la restricción: Esta restricción asegura que el origen y el destino de una ruta sean diferentes.
- Instrucción para la creación de la restricción:

```
ALTER TABLE RUTAS
ADD CONSTRAINT origen_destino CHECK (origen <> destino);
```

- Instrucción Insert o Update que permita evidenciar que la restricción está funcionando:

```
INSERT INTO RUTAS (rutaID, origen, destino, distancia, duracion)
VALUES ('R003', 'P001', 'P001', 1000, 10);
```



2.10. Consultas


Plantea 3 consultas que consideres relevantes para la base de datos propuesta. Para cada consulta planteada, incluir en el reporte los siguientes incisos:

Consulta 01

- Redacción clara de la consulta: Obtener la lista de todas las naves y sus respectivas rutas.
- Código en lenguaje SQL de la consulta.

```
1 SELECT N.naveID, N.modelo, R.rutaID, R.origen, R.destino
2 FROM NAVES N
3 INNER JOIN RUTAS R ON N.rutaID = R.rutaID;
```

Listing 18: Consulta de naves y rutas



Consulta 02

- Redacción clara de la consulta: Obtener la lista de todos los choferes y las naves que están manejando.
- Código en lenguaje SQL de la consulta.

```
1 SELECT C.choferID, C.nombre, N.naveID, N.modelo
2 FROM CHOFERES C
3 INNER JOIN NAVES N ON C.naveID = N.naveID;
```

Listing 19: Consulta de choferes y naves



Consulta 03

- Redacción clara de la consulta: Obtener la lista de todos los pasajeros y las naves en las que están viajando.
- Código en lenguaje SQL de la consulta.

```
1  SELECT PASAJEROS.nombre, PASAJEROS.apellido, NAVES.modelo
2  FROM PASAJEROS
3  INNER JOIN NAVES ON PASAJEROS.naveID = NAVES.naveID;
```

Listing 20: Tablas para la BdDatos

IMA/nave/con3.png

2.11. Vistas

Plantea 3 vistas que consideres relevantes para la base de datos propuesta. Para cada vista planteada, incluir en el reporte los siguientes incisos:

Vista 01

- Redacción clara de la vista planteada: Esta vista mostrará todas las naves junto con sus rutas respectivas.
- Código en lenguaje SQL que permita crear la vista solicitada.

```
1  CREATE VIEW Naves_Rutas AS
2  SELECT NAVES.naveID, NAVES.modelo, RUTAS.origen, RUTAS.destino
3  FROM NAVES
4  INNER JOIN RUTAS ON NAVES.rutaID = RUTAS.rutaID;
```

Listing 21: Tablas para la BdDatos

- USO:

```
1  SELECT * FROM Naves_Rutas;
```

Listing 22: Tablas para la BdDatos

IMA/nave/V1.png

Vista 02

- Redacción clara de la vista planteada: Esta vista mostrará todos los choferes junto con las naves que están manejando.
- Código en lenguaje SQL que permita crear la vista solicitada.

```
1      CREATE VIEW Choferes_Naves AS
2      SELECT CHOFERES.nombre, NAVES.modelo
3      FROM CHOFERES
4      INNER JOIN NAVES ON CHOFERES.naveID = NAVES.naveID;
```

Listing 23: Tablas para la Bddatos

- USO:

```
1      SELECT * FROM Choferes_Naves;
```

Listing 24: Tablas para la Bddatos



IMA/nave/V2.png

Vista 03

- Redacción clara de la vista planteada: Esta vista mostrará todos los pasajeros junto con las naves en las que están viajando.
- Código en lenguaje SQL que permita crear la vista solicitada.

```
1      CREATE VIEW Pasajeros_Naves AS
2      SELECT PASAJEROS.nombre, PASAJEROS.apellido, NAVES.modelo
3      FROM PASAJEROS
4      INNER JOIN NAVES ON PASAJEROS.naveID = NAVES.naveID;
```

Listing 25: Tablas para la Bddatos

- USO:

```
1      SELECT * FROM Pasajeros_Naves;
```

Listing 26: Tablas para la Bddatos



IMA/nave/V3.png

3. Estadios

3.1. Requerimientos

La FIFA nos ha encargado el desarrollo de un sistema para la administración de la venta de boletos en estadios de fútbol para los partidos del Mundial de 2026. El sistema deberá cubrir los cuatro estadios de fútbol de México que albergarán partidos del torneo: el Estadio Azteca, el Estadio Corregidora, el Estadio Hidalgo y el Estadio Jalisco.

Restricciones de los datos

- El número de boleto, el nombre del estadio, el nombre de la sección (dentro de un estadio), el número de transacción, el nombre del equipo y el id del partido deben ser únicos.
- Las fechas del partido y de la transacción deben ser válidas.
- La fecha y hora del partido deben estar en el futuro.
- El estadio debe ser uno de los cuatro estadios de México.
- La sección y el asiento deben ser válidos dentro del estadio.
- El precio y la capacidad del estadio y de la sección deben ser números positivos.
- El estado de venta debe ser Vendido o Disponible.
- El nombre y la dirección del cliente, así como el país del equipo, deben ser cadenas no vacías.
- El teléfono del cliente y el número de tarjeta de crédito deben ser válidos.
- El correo electrónico del cliente debe ser una dirección de correo electrónico válida.
- La ubicación del estadio debe ser una cadena que describa la ciudad o lugar donde se encuentra.
- Debe haber al menos una sección asociada a cada estadio y al menos dos equipos participando en cada partido.
- El estadio asociado al partido y el cliente y el boleto asociados a la transacción deben existir en sus respectivas tablas.
- El precio en la transacción debe ser igual al precio del boleto multiplicado por la cantidad de boletos en la transacción.
- El id del equipo en el partido debe hacer referencia a un equipo existente en la tabla Equipos.
- El id del partido en el equipo partido debe hacer referencia a un partido existente en la tabla Partido.
- La combinación única de id del equipo y id del partido debe asegurar que un equipo no participe más de una vez en el mismo partido.

3.2. Modelo conceptual



IMA/estadio/BD-ESTADIO-MO-CONCEPTUAL.png

3.3. Modelo relacional



IMA/estadio/BD-ESTADIO-MO-RELACIONAL.png

3.4. Script de creación

- Script completo y sin errores para la creación de todos los elementos que conforman el esquema de la base de datos.
- El Script debe estar diseñado para la versión 14 de Postgres.
- Deben estar contempladas todas las llaves primarias, llaves candidatas y llaves foráneas; todas las llaves foráneas deben contar con un trigger de integridad referencial (SET NULL, CASCADE o SET DEFAULT).

```
1  -- Solo los títulos de las tablas
2  CREATE TABLE Clientes(
3
4  );
5
6  CREATE TABLE Estadios(
7
8  );
9
10 CREATE TABLE SeccionesEstadio(
11
12 );
13
14 CREATE TABLE Equipos(
15
16 );
17
18 CREATE TABLE Partidos(
19
20 );
21
22 CREATE TABLE Boletos (
23
24 );
25
```

```
26 CREATE TABLE Transacciones(  
27  
28 );
```

Listing 27: Tablas para la BdBdatos

3.5. Script de Insert

- Se deben generar 100 registros para cada tabla.
- Si para el buen funcionamiento de la base de datos se requieren más de 100 registros o menos de 100 registros en una tabla, se debe explicar claramente la razón, sólo en este caso sí se debe incluir un apartado en el reporte final.

Tablas:

- Clientes: Si es posible llegar a 100 registros
- Estadios: La tabla únicamente contiene la información de los 4 estadios que albergan el mundial, por lo que no es posible llegar a 100 registros
- SeccionesEstadio: Si es posible llegar a 100 registros
- Equipos: La tabla únicamente contiene la información de las 32 selecciones clasificadas al mundial por lo que no es posible llegar a 100 registros
- Partidos: La tabla solo contiene los 24 partidos de fase de grupos por lo que no es posible llegar a 100 registros
- Boletos: Si es posible llegar a 100 registros
- Transacciones: Si es posible llegar a 100 registros

3.6. Funcionamiento restricciones

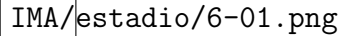
Evidencia del funcionamiento de al menos 4 restricciones de integridad referencial.

Restricción 01

- Tablas involucradas en la restricción: Partidos y Estadios
- FK de la tabla que referencia y PK de la tabla referenciada: FK: id Estadio en la tabla Partidos. PK id Estadio en la tabla Estadios
- Justificación del trigger de integridad referencial elegido: : Esta restricción asegura que no puedes tener un partido en un estadio que no exista en la tabla Estadios.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM Estadios WHERE id_Estadio = 1;
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

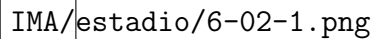
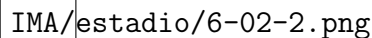
IMA/estadio/6-01.png

Restricción 02

- Tablas involucradas en la restricción: Boletos y Equipos
- FK de la tabla que referencia y PK de la tabla referenciada: Las claves foráneas en Boletos que hacen referencia a Equipos son id equipoLocal y id equipoVisita, y la clave primaria en Equipos es id Equipo.
- Justificación del trigger de integridad referencial elegido: Se utiliza la opción ON DELETE SET NULL. Esto significa que si se elimina un registro en la tabla Equipos, entonces el id equipoLocal y/o id equipoVisita correspondiente en la tabla Boletos se establecerá en NULL. Esto asegura la integridad de los datos.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM Equipos WHERE id_Equipo = 'SEN';
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.



IMA/estadio/6-02-1.pngIMA/estadio/6-02-2.png

Restricción 03

- Tablas involucradas en la restricción: Transacciones y Clientes
- FK de la tabla que referencia y PK de la tabla referenciada: La clave foránea en Transacciones que hace referencia a Clientes es id Cliente, y la clave primaria en Clientes es id Cliente.
- Justificación del trigger de integridad referencial elegido: Se utiliza la opción ON DELETE CASCADE. Esto significa que si se elimina un registro en la tabla Clientes, entonces todos los registros correspondientes en la tabla Transacciones también se eliminarán. Esto asegura la integridad de los datos.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM Clientes WHERE id_Cliente = 'CLI100';
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.



 IMA/estadio/6-03-1.png IMA/estadio/6-03-2.png

Restricción 04

- Tablas involucradas en la restricción: Transacciones y Boletos
- FK de la tabla que referencia y PK de la tabla referenciada: La clave foránea en Transacciones que hace referencia a Boletos es id Boleto, y la clave primaria en Boletos es num boleto.
- Justificación del trigger de integridad referencial elegido: Se utiliza la opción ON DELETE CASCADE. Esto significa que si se elimina un registro en la tabla Boletos, entonces todos los registros correspondientes en la tabla Transacciones también se eliminarán. Esto asegura la integridad de los datos.
- Instrucción UPDATE o DELETE que permita evidenciar que la restricción está funcionando.

```
DELETE FROM Boletos WHERE num_boleto = 2;
```

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

 IMA/estadio/6-04-1.png IMA/estadio/6-04-2.png

3.7. Funcionamiento Restricciones check

Evidencia del funcionamiento de al menos 3 restricciones check para *atributos* de varias tablas.

Check 01

- Tabla elegida: Clientes
- Atributo elegido: tarjetaCredito
- Breve descripción de la restricción: La restricción garantiza que el número de tarjeta de crédito tenga exactamente 16 dígitos.
- Instrucción para la creación de la restricción.

```
1 ALTER TABLE Clientes
2     ADD CONSTRAINT CHK_TarjetaCredito
3     CHECK (LENGTH(tarjetaCredito) = 16);
```

Listing 28: Tablas para la BdDatos

- Instrucción que permita evidenciar que la restricción esta funcionando.

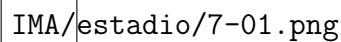
```

1      INSERT INTO Clientes
2          (id_Cliente, nombre_Cliente, direccion_Cliente,
3           telefono_Cliente, email_Cliente, tarjetaCredito)
4      VALUES ('CIL120', 'Juan_Luna', 'Calle_Falsa_123',
5              '55555555', 'juan.perez@example.com', '
              123456789012345');

```

Listing 29: Tablas para la BdDatos

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.



Check 02

- Tabla elegida: Estadios
- Atributo elegido: capacidad
- Breve descripción de la restricción: La restricción garantiza que la capacidad del estadio sea mayor a 0.
- Instrucción para la creación de la restricción.

```

1      ALTER TABLE Estadios
2      ADD CONSTRAINT CHK_Capacidad CHECK (capacidad > 0);

```

Listing 30: Tablas para la BdDatos

- Instrucción que permita evidenciar que la restricción esta funcionando.

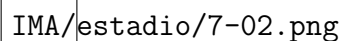
```

1      INSERT INTO Estadios (id_Estadio, nombre_Estadio, capacidad,
2      ubicacion) VALUES
      (5, 'Anfield', 0, 'San_Paulo');

```

Listing 31: Tablas para la BdDatos

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.



Check 03

- Tabla elegida: Boletos
- Atributo elegido: estado
- Breve descripción de la restricción: Garantiza que el estado del boleto sea Vendido o Disponible.
- Instrucción para la creación de la restricción.

```

1 ALTER TABLE Boletos
2 ADD CONSTRAINT CHK_Estado CHECK (estado IN ('Vendido', '
    Disponible'));

```

Listing 32: Tablas para la BdDatos

- Instrucción que permita evidenciar que la restricción esta funcionando.

```

1 INSERT INTO Boletos (num_boleto, id_Partido, id_equipoLocal,
2 id_equipoVisita,
3 estado, precio, nombre_Seccion)
4 VALUES (1, 'P123', 'E123', 'E124', 'Reservado',
    100, 'Seccion_1');

```

Listing 33: Tablas para la BdDatos

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

IMA/estadio/7-03.png

3.8. Creación de dominios personalizados

Evidencia de la creación de al menos tres dominios personalizados. Se deben utilizar restricciones check en la creación de los tres dominios.

Dominio 01

- Tabla elegida: Clientes
- Atributo elegido: tarjetaCredito
- Breve descripción del dominio y de la restricción check propuesta: Un dominio personalizado para el número de tarjeta de crédito que debe tener exactamente 16 dígitos.
- Instrucción para la creación del dominio personalizado.

```

1 CREATE DOMAIN TarjetaCredito AS VARCHAR(16)
2 CHECK (LENGTH(VALUE) = 16);

```

Listing 34: Tablas para la BdDatos

Instrucción para usar el dominio personalizado en la tabla:

```
1 ALTER TABLE Clientes
2 ALTER COLUMN tarjetaCredito TYPE TarjetaCredito;
```

Listing 35: Tablas para la Bddatos

- Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.

Dominio 02

- Tabla elegida: Equipos
- Atributo elegido: pais
- Breve descripción del dominio y de la restricción check propuesta: Un dominio personalizado para el país del equipo que no debe estar vacío.
- Instrucción para la creación del dominio personalizado.

```
1 CREATE DOMAIN PaisEquipo AS VARCHAR(255)
2 CHECK (VALUE <> '');
```

Listing 36: Tablas para la Bddatos

Instrucción para usar el dominio personalizado en la tabla:

```
1 ALTER TABLE Equipos
2 ALTER COLUMN pais TYPE PaisEquipo;
```

Listing 37: Tablas para la Bddatos

- Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.

Dominio 03

- Tabla elegida: Partidos
- Atributo elegido: id Estadio
- Breve descripción del dominio y de la restricción check propuesta: Un dominio personalizado para el id del estadio que debe estar entre 1 y 4.
- Instrucción para la creación del dominio personalizado.

```
1 CREATE DOMAIN IdEstadio AS INT
2 CHECK (VALUE IN (1, 2, 3, 4));
```

Listing 38: Tablas para la Bddatos

Instrucción para usar el dominio personalizado en la tabla:

```

1 ALTER TABLE Partidos
2 ALTER COLUMN id_Estadio TYPE IdEstadio;

```

Listing 39: Tablas para la BdBdatos

- Captura de pantalla de la estructura de la tabla donde se muestre el dominio personalizado en uso.

3.9. Restricciones para tuplas

Evidencia del funcionamiento de al menos 2 restricciones para “tuplas” en diferentes tablas (Unidad 8 Integridad, tema Specifying Constraints on Tuples Using CHECK)

Restricción 01

- Tabla elegida: Partidos
- Breve descripción de la restricción: La restricción garantiza que el id del estadio sea válido solo si la fecha y hora del partido están en el futuro.
- Instrucción para la creación de la restricción.

```

1 ALTER TABLE Partidos
2 ADD CONSTRAINT
3     CHK_FechaEstadio CHECK ((fecha_hora > CURRENT_TIMESTAMP AND
4     id_Estadio IN (1, 2, 3, 4)) OR (fecha_hora <=
5     CURRENT_TIMESTAMP));

```

Listing 40: Tablas para la BdBdatos

- Instrucción Insert o Update que permita evidenciar que la restricción esta funcionando.

```

1 INSERT INTO Partidos (id_Partido, fecha_hora, id_Estadio,
2     nombre_Arbitro)
3 VALUES ('P124', '2022-12-31 20:00:00', 5, 'Arbitro Prueba');
4 -- falla porque la fecha y hora estan en el futuro pero
5 -- el id del estadio no es valido

```

Listing 41: Tablas para la BdBdatos

- Captura de pantalla con el resultado de la instrucción que muestre que la restricción está funcionando.

Restricción 02

- Tabla elegida: Transacciones
- Breve descripción de la restricción: La restricción garantiza que el total de la transacción sea igual al precio del boleto multiplicado por la cantidad de boletos.


```

1  ALTER TABLE Transacciones
2  ADD CONSTRAINT CHK_TotalPrecioCantidad CHECK (total_Transaccion
    =
3      (SELECT precio
4       FROM Boletos
5       WHERE num_boleto = id_Boleto) *
        cantidad_Boletos);

```

Listing 42: Tablas para la BdDatos

- Instrucción Insert o Update que permita evidenciar que la restricción esta funcionando.

```

1  INSERT INTO Transacciones (num_Transaccion, fecha_transaccion,
    id_Cliente,
2      id_Boleto, cantidad_Boletos,
        total_Transaccion)
3  VALUES (2, CURRENT_TIMESTAMP, 'C123', 1, 2, 150);

```

Listing 43: Tablas para la BdDatos

3.10. Consultas

Plantea 3 consultas que consideres relevantes para la base de datos propuesta. Para cada consulta planteada, incluir en el reporte los siguientes incisos:

Consulta 01

- Redacción clara de la consulta: ¿Cuántos boletos se han vendido para cada partido?
- Código en lenguaje SQL de la consulta.

```

1  SELECT id_Partido, COUNT(*) as Boletos_Vendidos
2  FROM Boletos
3  WHERE estado = 'Vendido'
4  GROUP BY id_Partido;

```

Listing 44: Tablas para la BdDatos

Consulta 02

- Redacción clara de la consulta: ¿Cuál es el total de transacciones realizadas por cada cliente?
- Código en lenguaje SQL de la consulta.

```

1  SELECT id_Cliente, COUNT(*) as Total_Transacciones
2  FROM Transacciones
3  GROUP BY id_Cliente;

```

Listing 45: Tablas para la BdDatos

Consulta 03

- Redacción clara de la consulta: ¿Cuál es la capacidad total de cada estadio, sumando todas sus secciones?
- Código en lenguaje SQL de la consulta.

```
1      SELECT id_Estadio, SUM(capacidad_Seccion) as Capacidad_Total
2      FROM SeccionesEstadio
3      GROUP BY id_Estadio;
```

Listing 46: Tablas para la BdDatos

3.11. Vistas

Plantea 3 vistas que consideres relevantes para la base de datos propuesta. Para cada vista planteada, incluir en el reporte los siguientes incisos:

Consulta 01

- Redacción clara de la vista planteada: Una vista que muestre la cantidad de boletos vendidos para cada partido.
- Código en lenguaje SQL que permita crear la vista solicitada.

```
1      CREATE VIEW BoletosVendidosPorPartido AS
2      SELECT id_Partido, COUNT(*) as Boletos_Vendidos
3      FROM Boletos
4      WHERE estado = 'Vendido'
5      GROUP BY id_Partido;
```

Listing 47: Tablas para la BdDatos

Consulta 02

- Redacción clara de la vista planteada: Una vista que muestre el total de transacciones realizadas por cada cliente.
- Código en lenguaje SQL que permita crear la vista solicitada.

```
1      CREATE VIEW TotalTransaccionesPorCliente AS
2      SELECT id_Cliente, COUNT(*) as Total_Transacciones
3      FROM Transacciones
4      GROUP BY id_Cliente;
```

Listing 48: Tablas para la BdDatos

Consulta 03

- Redacción clara de la vista planteada: Una vista que muestre la capacidad total de cada estadio, sumando todas sus secciones.
- Código en lenguaje SQL que permita crear la vista solicitada.

```
1      CREATE VIEW CapacidadTotalPorEstadio AS
2      SELECT id_Estadio, SUM(capacidad_Seccion) as
           Capacidad_Total
3      FROM SeccionesEstadio
4      GROUP BY id_Estadio;
```

Listing 49: Tablas para la BdDatos