Presentación y Descripción del Programa: Gestión de Inscripciones a Mesas de Exámenes

ENTREGA FINAL

Presentación:

Este programa busca facilitar la gestión de inscripciones de alumnos a mesas de exámenes. Con esta base de datos, se busca optimizar el proceso de inscripción, brindando datos específicos para administrar las solicitudes de los alumnos y organizar eficientemente las mesas de exámenes.

Presentación técnica:

El programa es presentado modularmente en diferentes scripts, que puede ser importado manualmente o puede ser ejecutado mediante el archivo ***project.sh (bash, source).*** Creará ***procedures, functions, triggers, users, views, estructura de la db y datos de la db.***

Además, se presenta el ***DUMP*** para hacer un ***IMPORT*** de la database completa ***(estructura, datos y objetos)***.

Descripción del Programa:

- **Objetivo principal:** Obtener una vista de las inscripciones que detalle la fecha del examen, la materia a evaluar, el docente a cargo y el alumno registrado.
- **Funcionalidades:**
    1. Registro de alumnos: Permite añadir nuevos alumnos al sistema con sus datos básicos.
    2. Gestión de mesas de exámenes: Permite crear nuevas mesas de exámenes, asignar fechas, horarios y asignaturas.
    3. Inscripción a mesas de exámenes: Los alumnos podrán inscribirse en las mesas de exámenes disponibles según sus necesidades académicas.
    4. Consulta de inscripciones: Permite a los administradores visualizar las inscripciones realizadas por los alumnos.
    5. Gestión de usuarios: Administración de cuentas de usuarios con diferentes niveles de acceso.
- **Beneficios:**
    - Optimización del proceso de inscripción a exámenes.
    - Mayor organización y control de las mesas de exámenes.

Tablas:

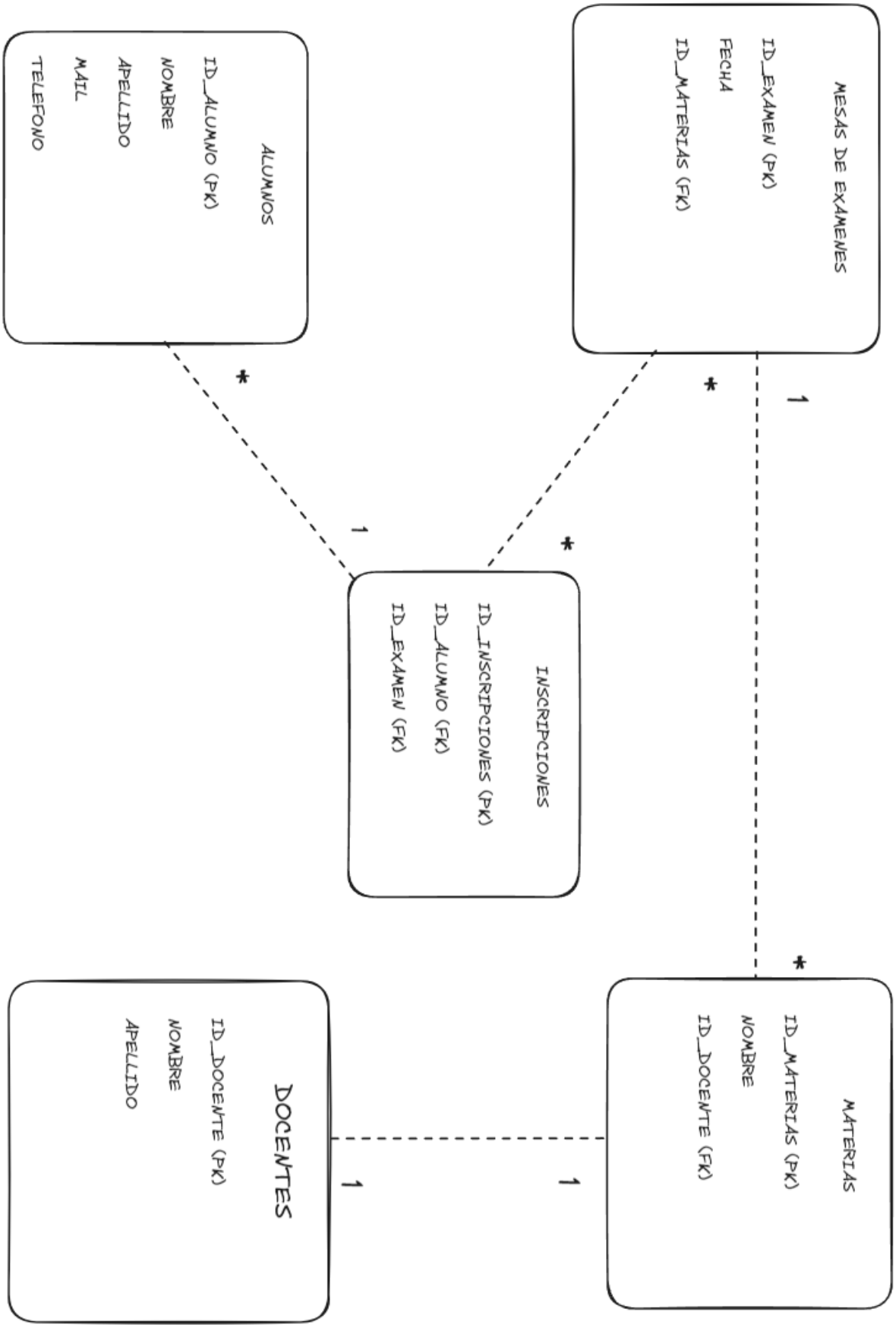
ALUMNOS					
Datos	TYPE	Primary Key	Foreign Key	AUTO_INCREMENT	NOT NULL
Id_alumno	**INT**	X		X	
Nombre	**VARCHAR(50)**				X
Apellido	**VARCHAR(50)**				X
Email	**VARCHAR(100)**				
Teléfono	**VARCHAR(20)**				

DOCENTE					
Datos	TYPE	Primary Key	Foreign Key	AUTO_INCREMENT	NOT NULL
Id_docente	**INT**	X		X	
Nombre	**VARCHAR(50)**				X
Apellido	**VARCHAR(50)**				X
MATERIAS					
Datos	TYPE	Primary Key	Foreign Key	AUTO_INCREMENT	NOT NULL
Id_materias	**INT**	X		X	
Nombre	**VARCHAR(50)**				X
Id_docente			X		

MESAS DE EXAMENES					
Datos	TYPE	Primary Key	Foreign Key	AUTO_INCREMENT	NOT NULL
Id_examen	**INT**	X		X	
fecha	**DATETIME**				X
Id_materias			X		X

INSCRIPCIONES					
Datos	TYPE	Primary Key	Foreign Key	AUTO_INCREMENT	NOT NULL
Id_inscripciones	**INT**	X		X	
Id_alumno			X		X
Id_examen			X		X

DIAGRAMA ENTIDAD-RELACIÓN



Vistas:

- **Vista de alumnos inscriptos:**
  DESCRIPCIÓN: Constancia de inscripción de examen.
  COLUMNAS: id de alumno, nombre, apellido, id de examen, fecha de examen.
- **Vista de materias-docente:**
  DESCRIPCIÓN: Asignación de docentes a materias.
  COLUMNAS: id de materia, nombre de materia y nombre de docente.
- **Vista de mesa de examen:**
  DESCRIPCIÓN: Constancia detallada del examen a tomar, pensado para consultas de la institución.
  COLUMNAS: id de examen, fecha de examen, nombre de la materia y nombre de docente.

Funciones:

- ***VERIFICACIÓN DE INSCRIPCIÓN:***
  DESCRIPCIÓN: Determina si el alumno consultado está inscripto en la materia indicada, mediante parámetros de id de alumno e id de materia.
  PARAMETROS DE ENTRADA: id de alumno e id de materia.
  RETORNO: Lo determina un condicional IF
    - ✓ ***Está inscripto.***
    - ✓ ***No está Inscripto.***

  EJEMPLO DE USO:

  ```
  SELECT alumno_inscripto_materia(3, 4);
  ```
- ***CANTIDAD DE INSCRIPTOS:***
  DESCRIPCIÓN: Determina número de alumnos inscriptos en un examen, mediante el id del examen.
  PARAMETROS DE ENTRADA: id de examen.
  RETORNO: Devuelve el número de inscriptos.
  EJEMPLO DE USO:

  ```
  SELECT cantidad_inscriptos_examen(1);
  ```

Stored Procedures:

- ***CANTIDAD DE EXAMEN INSCRIPTO:***
  DESCRIPCIÓN: Determina número de exámenes que se inscribe el alumno indicado como parámetro de entrada.
  PARAMETROS DE ENTRADA: id de alumno.
  EJEMPLO DE USO:

  ```
  CALL obtener_alumno_examenes(1);
  ```

- ***ORDEN DE EXAMENES:***

  DESCRIPCIÓN: Ordena según la fecha del examen e indica la cantidad de inscriptos.

  PARAMETROS DE ENTRADA: *NO CONTIENE.*

  EJEMPLO DE USO:

  ```
  CALL orden_examenes();
  ```

Triggers:

- ***REGISTRO DE CREACION DE ALUMNOS:***

  DESCRIPCIÓN: Registra la inserción de alumnos en la tabla ***log_alumnos.***

  DETALLES:

  - ✓ *Tabla afectada: alumnos.*
  - ✓ *Acción: Insert*
  - ✓ *Información registrada: fecha de acción, nombre, apellido, correo, teléfono, mensaje de acción.*

  EJEMPLO DE USO:

  - ✓ *Se inserta un nuevo alumno.*
  - ✓ *El trigger registra la acción en la tabla* ***log_alumnos*** *con los detalles correspondientes.*

- ***SEGURIDAD DE ELIMINACIÓN DE MESAS DE EXAMEN:***

  DESCRIPCIÓN: *Evita la eliminación de mesas de examen con alumnos inscriptos.*

  DETALLES:

  - ✓ *Tabla afectada: mesasexamen.*
  - ✓ *Acción: Delete*
  - ✓ *Alerta: SQLSTATE "45000" - "No se puede eliminar la mesa de examen. Hay alumnos inscriptos"*

  EJEMPLO DE USO:

  - ✓ *Intentamos:*

    ```
    DELETE FROM MesasExamen
    WHERE id_examen =1;
    ```

  - ✓ *El trigger evita la acción y envía el mensaje de alerta.*

Users y permisos:

Creación de usuarios con diferentes permisos de manipulación de la base de datos.
Permisos:

- *Alumnos:* ***SELECT e INSERT***
- *Docentes:* ***Todos los privilegios.***

TRANSACTIONS (commit, rollback, savepoints):

*El script de inserción de datos (database_populate.sql) contiene **savepoints** en cada inserción para poder realizar **rollbacks** específicos.*

*En las líneas comentadas se pueden realizar tales acciones asi también como un **release** de tales **savepoints.***

*En el final del script se encuentra el **commit** correspondiente a la transactions.*