UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS.
FUNDAMENTOS DE BASES DE DATOS.

Práctica 06
Mantenimiento de Llaves Foráneas.

Equipo: ChiquesSQL

Ivana Ix Chel Bonilla Negrete
315131994

Dylan Enrique Juarez Martinez
422117180

Daniel Rojo Mata
314297967

Profesor:

Gerardo Áviles Rosas

Ayudantes de Teoría:

Gerardo Uriel Soto Miranda
Valeria Fernanda Manjarrez Angeles

Ayudantes de Laboratorio:

Ricardo Badillo Macías
Rocío Aylin Huerta González

I. Preguntas Generales

1. ¿Qué es una política de mantenimiento de llaves foráneas?

   Una política de mantenimiento de llaves foráneas es un conjunto de reglas y procedimientos para garantizar la integridad y consistencia de las relaciones entre tablas en una base de datos, especialmente cuando se usan llaves foráneas (`foreign keys`).

   En `PostgreSQL`, las llaves foráneas son restricciones que establecen relaciones entre una tabla y otra. Estas restricciones garantizan que el valor en la columna de una tabla (la llave foránea) corresponda a un valor existente en la columna de la tabla relacionada (la llave primaria).

   Algunas posibles políticas son las siguientes:

   - **RESTRICT**: Esta opción impide que se borren o modifiquen filas en la tabla de origen que contienen la clave foránea. Esto significa que los valores no pueden ser alterados a lo largo de su ciclo de vida para evitar que se rompan las relaciones con otras tablas que hacen referencia a estas claves foráneas. Si se quiere eliminar un registro, primero se debe romper la relación entre tablas.
   - **CASCADE**: Si se eliminan o modifican valores en la tabla primaria, estas modificaciones se propagan automáticamente a las tablas relacionadas que contienen claves foráneas. Esto garantiza que los cambios se reflejen en toda la base de datos para mantener la coherencia.
   - **SET NULL**: Con esta opción, si se elimina o se modifica un valor en la tabla principal, las claves foráneas en las tablas relacionadas se configuran a NULL. Es útil cuando se quiere evitar eliminar datos pero dejar una señal de que el valor original ya no está disponible.
   - **NO ACTION:** Cuando se borra o se modifica una clave primaria, las tablas relacionadas con claves foráneas no se ven afectadas. Esto puede ser útil si las tablas relacionadas deben manejarse de forma independiente, sin cambiar cuando la clave de referencia se modifica o elimina.
   - **SET DEFAULT:** Esta opción permite asignar un valor predeterminado a las claves foráneas en las tablas relacionadas cuando se elimina o modifica una clave primaria. Esto puede ser útil para evitar que las relaciones queden incompletas y para mantener un valor conocido como reemplazo cuando se modifica la tabla principal.

2. Para cada política que investigaron, ¿cómo se indica en SQL? y ¿cuál es su objeto y su funcionamiento? (juntas las preguntas 2 y 3)

   Las políticas se especifican cuando se crean o modifican tablas y sus restricciones de llaves foráneas. Algunas principales políticas que se pueden indicar y cómo hacerlo en `PostgreSQL`.

   - `CASCADE`: Si se elimina o actualiza una fila en la tabla primaria, las filas relacionadas también se eliminan o actualizan. Se usa para mantener consistencia en cascada.

     ```
     1         ALTER TABLE child_table
     2         ADD CONSTRAINT fk_example
     3         FOREIGN KEY (foreign_key_column)
     4         REFERENCES parent_table (primary_key_column)
     5         ON DELETE CASCADE;
     ```

     - `ON DELETE CASCADE`: Esta cláusula define el comportamiento cuando se elimina una fila de la tabla de referencia (en este caso, `parent_table`). Con la opción `CASCADE`, si se elimina una fila en `parent_table`, las filas correspondientes en `child_table` también se eliminarán

automáticamente. Esto ayuda a mantener la integridad referencial evitando datos huérfanos o inconsistencias.

- `SET NULL`: Si se elimina o actualiza una fila en la tabla primaria, el valor en la tabla foránea se establece en `NULL`.

```
1        ALTER TABLE child_table
2        ADD CONSTRAINT fk_example
3        FOREIGN KEY (foreign_key_column)
4        REFERENCES parent_table (primary_key_column)
5        ON DELETE SET NULL;
```

  - `ON DELETE SET NULL`: Esta cláusula especifica el comportamiento cuando se elimina una fila de la tabla de referencia (`parent_table`). Con esta opción, si se elimina una fila de `parent_table`, la columna clave externa correspondiente en `child_table` se establece en `NULL`. Esto permite mantener la integridad referencial sin eliminar automáticamente las filas relacionadas, lo que puede ser útil para evitar datos huérfanos sin perder completamente el registro en la tabla `child_table`.

- `SET DEFAULT`: Si se elimina o actualiza una fila en la tabla primaria, la llave foránea se establece en su valor por defecto.

```
1        ALTER TABLE child_table
2        ADD CONSTRAINT fk_example
3        FOREIGN KEY (foreign_key_column)
4        REFERENCES parent_table (primary_key_column)
5        ON DELETE SET DEFAULT;
```

  - `ON DELETE SET DEFAULT`: Esta cláusula define el comportamiento cuando se elimina una fila de la tabla de referencia (`parent_table`). Con esta opción, si se elimina una fila de `parent_table`, la columna clave externa correspondiente en `child_table` se establece en su valor por defecto. Esto es útil para mantener relaciones consistentes sin eliminar las filas en la tabla secundaria.

- `RESTRICT`: Impide la eliminación o actualización en la tabla primaria si hay filas relacionadas en la tabla foránea.

```
1        ALTER TABLE child_table
2        ADD CONSTRAINT fk_example
3        FOREIGN KEY (foreign_key_column)
4        REFERENCES parent_table (primary_key_column)
5        ON DELETE RESTRICT;
```

  - `ON DELETE RESTRICT`: Esta cláusula impide la eliminación de una fila en `parent_table` si hay filas relacionadas en `child_table`. Con esta política, se garantiza la integridad referencial al evitar datos huérfanos, asegurando que no se eliminen registros necesarios para otras tablas.

- `NO ACTION`: Similar a RESTRICT, pero la verificación se realiza al final de la transacción.

```
ALTER TABLE child_table
ADD CONSTRAINT fk_example
FOREIGN KEY (foreign_key_column)
REFERENCES parent_table (primary_key_column)
ON DELETE NO ACTION;
```

  - `ON DELETE NO ACTION`: Esta política es similar a `RESTRICT`, pero la verificación de restricciones ocurre al final de la transacción. Puede ser útil en operaciones donde las restricciones referenciales se deben comprobar después de todas las modificaciones para permitir actualizaciones o eliminaciones condicionales.

3. Para cada política que investigaron, ¿cuáles son sus ventajas y desventajas?

- `CASCADE`:
  - **Ventajas**:
    - Mantiene la integridad referencial al eliminar automáticamente las filas relacionadas cuando se elimina la fila primaria.
    - Simplifica la administración de datos, ya que las eliminaciones en cascada reducen la necesidad de limpiar manualmente las tablas relacionadas.
  - **Desventajas**:
    - Puede llevar a eliminaciones no deseadas si no se tiene cuidado, eliminando varias filas de forma involuntaria.
    - Riesgo de pérdida de datos significativo si no se comprenden las relaciones entre tablas.

- `SET NULL`:
  - **Ventajas**:
    - Mantiene la integridad referencial sin eliminar datos relacionados, estableciendo la llave foránea en `NULL` cuando se elimina la fila primaria.
    - Evita la eliminación de datos en la tabla secundaria, permitiendo mayor flexibilidad.
  - **Desventajas**:
    - Puede llevar a datos huérfanos o registros incompletos en la tabla secundaria.
    - La semántica de `NULL` puede no ser clara en algunos contextos, llevando a confusión o errores.

- `SET DEFAULT`:
  - **Ventajas**:
    - Mantiene la integridad referencial al restablecer la llave foránea a un valor por defecto cuando se elimina la fila primaria.
    - Permite conservar registros en la tabla secundaria sin eliminarlos completamente.
  - **Desventajas**:
    - Requiere definir un valor por defecto que tenga sentido, lo cual puede ser difícil en algunos casos.

- Puede crear datos ambiguos si el valor por defecto no es claro o no tiene sentido para la relación.

- `RESTRICT`:
  - **Ventajas**:
    - Garantiza la integridad referencial al impedir la eliminación o actualización de filas primarias si hay filas relacionadas en la tabla secundaria.
    - Proporciona seguridad adicional, evitando la eliminación accidental de datos críticos para otras tablas.
  - **Desventajas**:
    - Puede ser restrictivo, requiriendo pasos adicionales para eliminar datos.
    - Puede dificultar la eliminación o actualización de datos en flujos de trabajo que esperan cierta flexibilidad.

- `NO ACTION`:
  - **Ventajas**:
    - Permite mayor flexibilidad, ya que la verificación de restricciones ocurre al final de la transacción.
    - Similar a RESTRICT, pero puede ser más tolerante, permitiendo manipulación condicional de datos.
  - **Desventajas**:
    - Puede ser confuso, ya que los errores de restricción solo se detectan al final de la transacción.
    - Puede llevar a operaciones fallidas si se detectan problemas de integridad al final de la transacción.

4. Con base a lo anterior, ¿cuál política utilizarán para su esquema, y por qué motivo?

   Se decidió utilizar `CASCADE` puesto que garantiza que las relaciones entre tablas se mantengan consistentes automáticamente, minimizando errores humanos que podrían surgir al tratar de mantener manualmente estas relaciones, esto ocurre debido a que cuando se actualiza o elimina un valor en la tabla principal, los cambios se propagan automáticamente a las tablas relacionadas, evitando la necesidad de ejecutar consultas adicionales para mantener la consistencia de los datos. Esto reduce la sobrecarga en las operaciones de mantenimiento.

   Para el caso de uso correspondiente, se utilizó: `ON DELETE CASCADE` y `ON UPDATE CASCADE` en todas las llaves foráneas de la base de datos, con esto se asegura que cualquier eliminación o actualización en una tabla principal se propagará automáticamente a las tablas relacionadas.

II. CORRECCIONES

Modelo ER:

- Se borró el atributo id_empleado de la tabla empleado, puesto que ya se cuenta con la llave id_Persona.

Modelo relacional:

- Se añadió la conexión entre personalDeApoyo y las habitaciones (individual, doble, cuadruple y penthouse).
- Se actualizó el identificador de la tabla huesped, así como la inclusión del atributo RFC como llave, esto porque ya se contaba con el id_Persona.
- Se añadió la representación correspondiente al personal: telefonoLimpiezaInterna, telefonoPersonalApoyo, telefonoRecepcionista, telefonoServicioComida, correoLimpiezaInterna, correoPersonalApoyo, correoRecepcionista y correoServicioComida.
- Se actualizó DISPONER a: disponerIndividual, disponerDoble, disponerCuadruple y disponerPenthouse.
- De manera análoga con RENTAR: rentarIndividual, rentarDoble, rentarCuadruple y rentarPenthouse.
- Se retiró el RFC como llave de la relación ATENDER.
- Se añadieron las llaves idFormaEfectivo e idFormaTarjeta para distinguir el tipo de pago.

DDL:

- Se renombró al identificador de las tablas disponerIndividual, disponerDoble, disponerCuadruple y disponerPenthouse a idHabitacion para que correspondieran con la referencia.
- Se realizó la corrección de llave primaria a foránea del idHabitacion para las tablas de la relación disponer.
- Se borró la columna idEmpleado de las tablas del personal.
- Se modificaron los CHECK de la tabla pago para algunos campos. Esto para que el número de niños, adultos, mascotas y noches pueda tomar el valor de cero.
- Se borró la columna idPago de la tabla huesped, ya que no correspondía con el modelo relacional.