|  |  |
| --- | --- |
|  | **Universidad Nacional Autónoma de México** |
|  | **Facultad de Ingeniería** |
|  | **División de Ingeniería Eléctrica** |
|  | **Laboratorio de**  **Bases de Datos** |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Practica 9 | Lenguaje de Definición de Datos | |
|  |  |  |  | |
|  | Profesor: | Lugowski Rivero Czeslaw Kristofer |  | |
|  | Grupo: | 06 |  | |
|  | Alumnos: | Guerrero López Uriel Ivan 320046045  Muñoz San Agustin Victoria Monserrat 320094187 | | |
|  |  |  | Calificación | |
|  |  | Fecha de entrega | Jueves 17 de abril de 2025 | |
|  |  |  | Semestre 2025-2 | |

**Objetivo:**

* El alumno aprenderá el uso del lenguaje SQL para la creación de tablas e índices en una base de datos. Comprenderá y confirmará el uso del lenguaje SQL para realizar la modificación y/o eliminación de la estructura de los principales objetos de una base de datos empleando las instrucciones ALTER y DROP.

**Introducción**

Lenguaje de definición de datos

El Lenguaje de Definición de Datos (DDL Data Definition Language) nos permite realizar las tareas de creación, eliminación, modificación de un objeto de base de datos, además de la concesión y revocación de privilegios sobre estos objetos. Los ejemplos más comunes de comandos de este tipo son: CREATE, ALTER y DROP.

La primera fase de cualquier base de datos siempre comienza con órdenes DDL, para diseñar los objetos de la base de datos.

Diseño de tablas

Cuando diseñe una base de datos, deberá decidir qué tablas necesita, qué tipo de datos van destinados a cada tabla, quién puede tener acceso a cada tabla, etc. Cuando cree tablas y trabaje con ellas, seguirá tomando decisiones más específicas acerca de las mismas.

El método más eficiente para crear una tabla consiste en definir todo lo que se necesita en la tabla al mismo tiempo, incluidas las restricciones para los datos y los componentes adicionales. Entre las decisiones que deben tomarse, se incluyen:

* Los tipos de datos que debe contener la tabla.
* Las columnas de la tabla y los tipos de datos para cada columna (así como su longitud, si es preciso).
* Qué columnas aceptan valores NULL.
* Si deben utiliz[[1]](#footnote-2)arse (y cuándo) restricciones o valores predeterminados y reglas.
* Los tipos de índices necesarios, dónde se necesitan y qué columnas son claves principales y claves externas.

Creación de una tabla

La sintaxis para crear tablas en su forma más común es la siguiente1:

CREATE TABLE <nombre tabla>

(<nombre columna> <tipo de dato>

[NOT NULL] [UNIQUE] [CONSTRAINT <nombre restricción>][PRIMARY

KEY] [DEFAULT][CHECK]

| [PRIMARY KEY (<lista columnas>)]

| [FOREIGN KEY (<lista columnas>) REFERENCES <nombretabla>]

| [UNIQUE (<lista columnas>)] [CONSTRAINT <nombre

restricción>],[,..[[2]](#footnote-3).] )

| [CHECK (condición de búsqueda)])

;

Se explican a continuación cada uno de los comandos:

* La sentencia CREATE TABLE se utiliza para crear una tabla.
* Dentro de la tabla podemos especificar una serie de columnas que contienen datos y restricciones que verifican datos y especificar otras características de la tabla.
* Cada columna debe tener asignado un tipo de dato válido.
* Puede especificar NOT NULL para asegurar que la columna siempre tenga datos.
* También podemos especificar un valor por omisión para la columna que va a utilizarse cuando se inserte un nuevo registro en la tabla y no se especifique un valor, o se especifique como DEFAULT.
* Podemos definir una llave primaria (PRIMARY KEY) para la tabla o definir una columna particular o un conjunto de columnas que sean UNIQUE (que sus valores no se puedan repetir en la tabla). Además, podemos definir una llave foránea (FOREIGN KEY) tomando en cuenta, que el valor de la <lista de columnas> debe existir en otra tabla.

Eliminar una tabla

La sentencia DROP suprime todas las restricciones de integridad referencial que hagan referencia a llaves de la tabla eliminada.

DROP TABLE <nombre tabla> [CASCADE CONSTRAINTS]

Modificar una tabla

La sentencia ALTER TABLE se utiliza para modificar la estructura de una tabla existente.

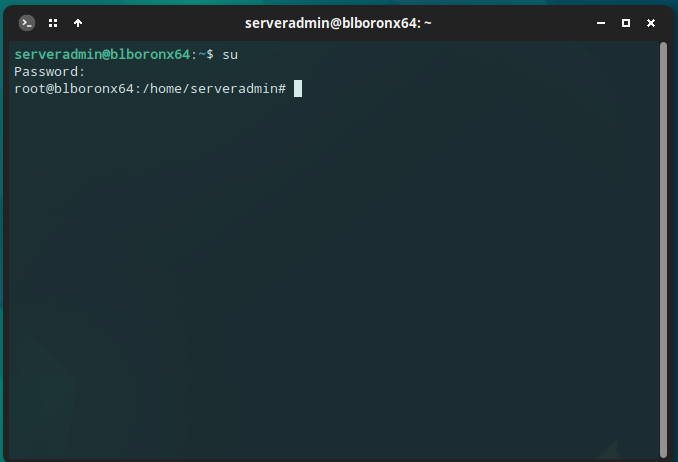
Dentro de la tabla podemos añadir (ADD) o borrar (DROP) columnas, restricciones, condiciones de comprobación, llaves primarias y foráneas.

**Actividades propuestas por la academia**

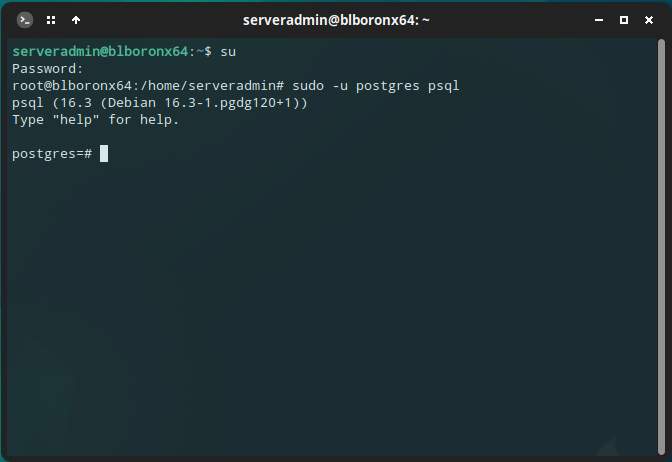
Actividad 1.

Crear las tablas, considerando las diferentes cardinalidades (1:1,1:m, m:m)

1. Ingresar en la máquina virtual como usuario root

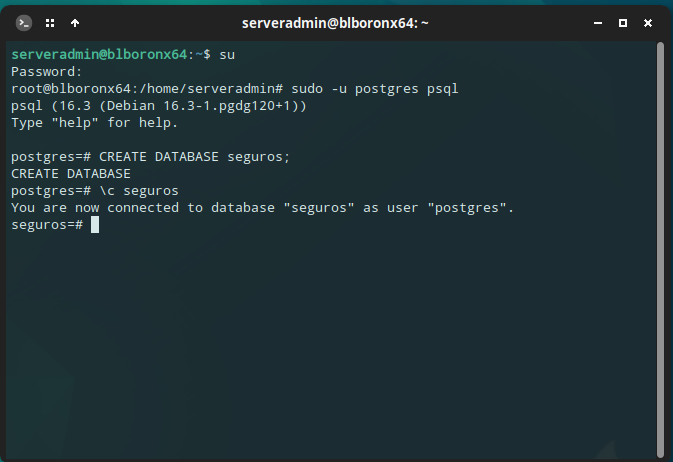


1. Ingresar mediante la terminal a postgres con la utileria psql y con el usuario postgres



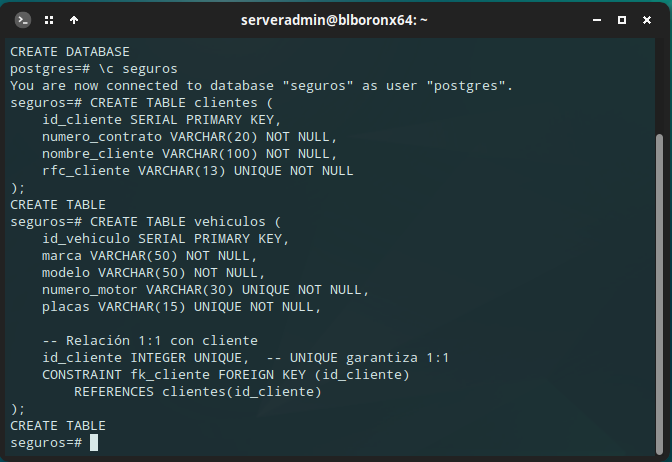
1. Realizar la creación de tablas físicas y columnas mediante ejecución de “query’s” en psql del sig. ejemplo: Una empresa de seguros vehiculares requiere de almacenar en una base de datos la relación del préstamo de un único vehículo a sus clientes que sufrieron algún percance y que tienen una póliza con ese extra. Se requiere conocer los datos mínimos del cliente como id del cliente, número de contrato, el nombre y el RFC del cliente, del vehículo se requiere saber su identificador interno, su marca, modelo, número de motor y placas del vehículo.

Primero creamos la Base de Datos y accedemos a ella para así trabajar las tablas



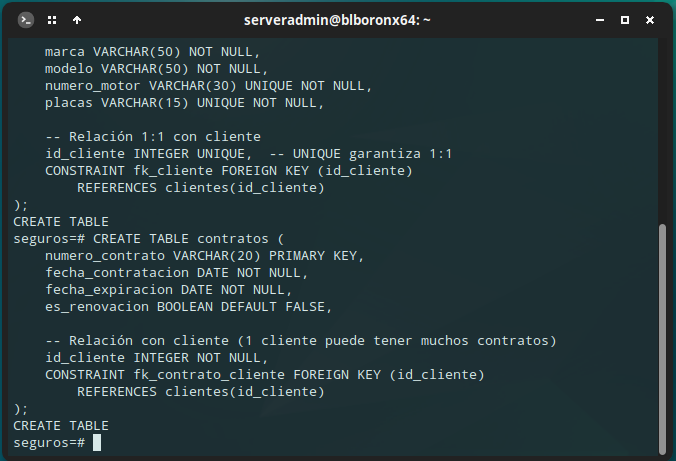
Creamos las dos tables que harán alusión a las necesidades de la empresa, donde:

* Usamos SERIAL para generar un ID automático.
* La columna id\_cliente en vehiculos tiene UNIQUE, lo que hace que ningún otro vehículo pueda compartir el mismo cliente. Así se mantiene una relación 1:1.

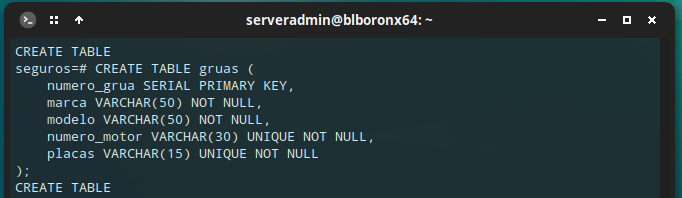


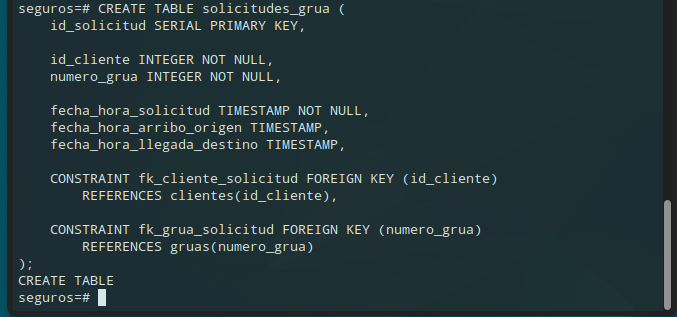
1. Agregar al ejemplo anterior los siguientes extras. La empresa requiere almacenar en la base de datos el número de contratos contradados con la compañía de seguros para lo cual se requiere tener el número de contrato, la fecha de contratación, la fecha de expiración y si ese contrato en particular es una renovación.

Ahora creamos una tercera tabla para las fechas de los contratos



1. Agregar al ejemplo anterior el extra de solicitud de grúa. En el cual se quiere registrar la cantidad de veces que el cliente solicita una grúa, esa grúa puede otorgar el servicio a muchos clientes, para ello la marca, modelo, número de motor, placas y número de la grúa utilizada, la fecha y hora de la solicitud, fecha y hora de arribo al lugar de la solicitud “origen”, fecha y hora de llegada al “destino”.





Actividad 2.

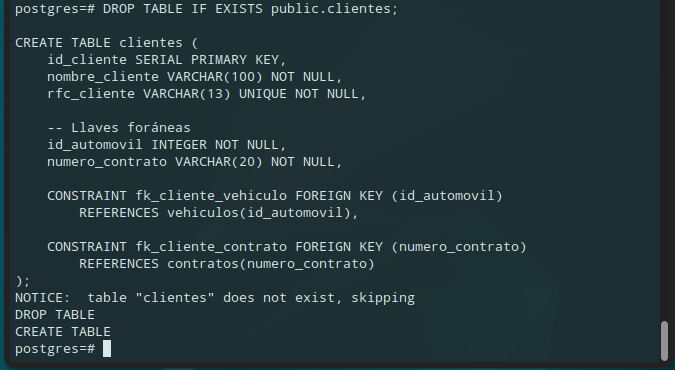
Restricciones de llave primaria, foránea, unicidad, valores obligatorios y de dominio. Contemplar el uso de valores por omisión y nulos. Considerar el uso de creación de restricciones tanto a nivel columna como a nivel tabla.

https://www.postgresql.org/docs/9.6/static/ddl-constraints.html

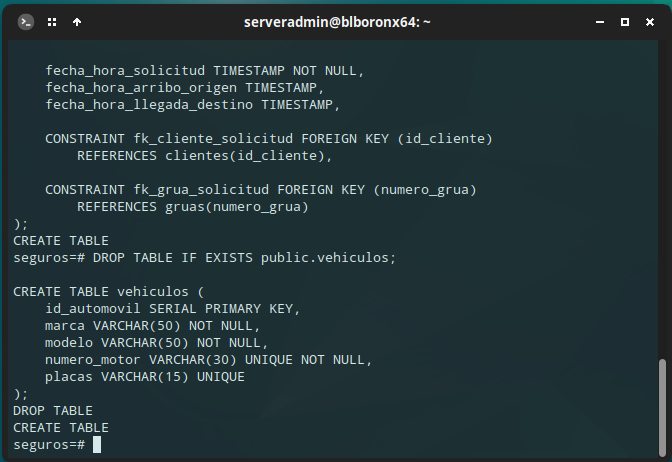
https://www.postgresql.org/docs/9.5/static/sql-createdomain.html

1. Para el ejemplo de la Actividad 1 verificar los siguientes puntos, en caso de que hagan falta algún punto eliminar por completo la tabla afectada con el comando (DROP TABLE public.Nombre\_de\_la\_tabla;) y crear nuevamente cumpliendo con todos los puntos:
   * Para la tabla (clientes/usuarios/personas) Es llave primaria id\_cliente, es llave candidato RFC, son valores necesarios el id cliente, el nombre, el RFC y el automóvil, es llave foránea el id automovil y número de contrato.

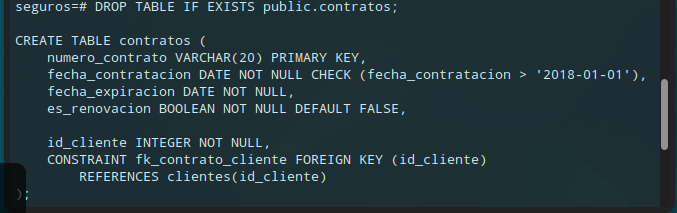
En este caso es nuestro primer intento usando el DROP TABLE public.Nombre\_de\_la\_tabla, para checar si tiene un funcionamiento correcto, además de la creación de las nuevas tablas con sus respectivas características. En este caso al estar utilizando para práctica nos aparece que no existe y se skipes, pero en las siguientes podemos ver su función.



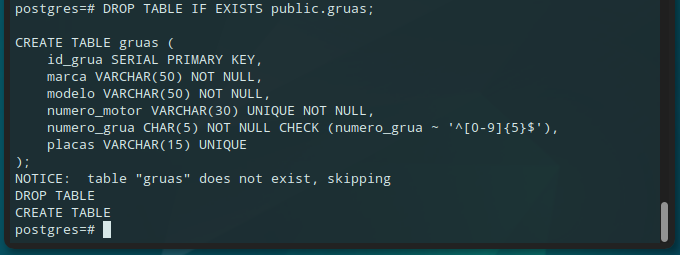
* + Para la tabla (automoviles/vehiculos/coches) Es llave primaria id\_automovil, es llave candidato número de motor, son valores necesarios marca, modelo, número de motor.



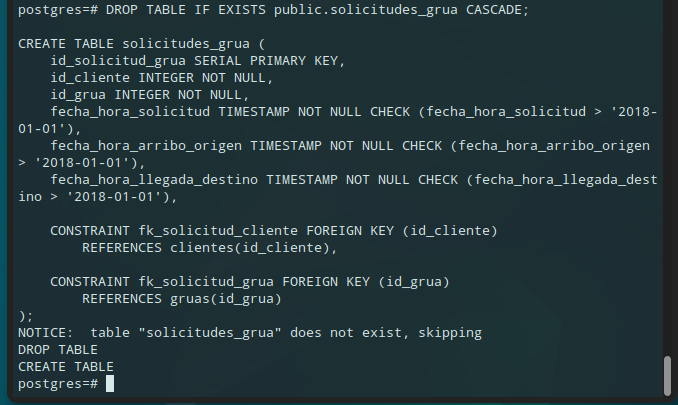
* + Para la tabla (contratos/seguros/pólizas) Es1o, son valores necesarios fecha de contratación, y la fecha de contratación tiene que ser mayor a enero de 2018, el valor por default de la renovación debe de ser cero.



* + Para la tabla (gruas) Es llave primaria id\_grua, es llave candidato número de motor, son valores necesarios marca, modelo y el número de grúa debe de ser únicamente de 5 dígitos númericos del 0 al 9.



* + Para la tabla (uso\_gruas/llamado\_grua/solicitud\_grua) Es llave primaria el id\_solicitud\_grua, es llave foránea id\_cliente y id\_grua, son valores necesarios la fecha y hora de la solicitud, fecha y hora de arribo al lugar de la solicitud “origen”, fecha y hora de llegada al “destino”. Las fechas deben de ser mayores a enero 2018.



Actividad 3

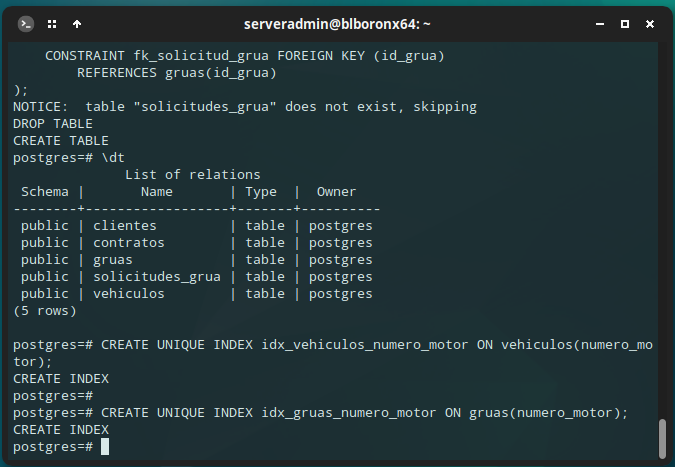
Generar índices (non unique, unique)

Explicar y analizar los casos donde es recomendable crear índices, por ejemplo, en llaves foráneas, atributos que se emplean con alta frecuencia en búsquedas, etc.

https://www.postgresql.org/docs/9.1/static/sql-createindex.html

1. Generar un indice único para los números de motor de vehículos y de las grúas.

En la siguiente imagen Podemos ver las 5 tablas anteriormente creadas y los dos índices únicos creados para el motor de vehículos y motor de grúas.



Explicación de los índices

* + Índice único: En estas consultas creamos índices únicos en las columnas numero\_motor tanto para vehiculos como para gruas. Esto asegura que no se repitan los números de motor dentro de estas tablas, y a la vez, mejora la velocidad al buscar o actualizar un registro basado en el número de motor.

Beneficios de crear índices:

* + Mejora en las consultas: Los índices mejoran la velocidad de las búsquedas en columnas con muchos registros.
  + Evitar duplicados: El índice único garantiza que no se repitan valores en las columnas relevantes.

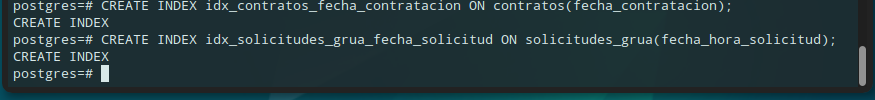
Recomendación:

* Los índices son muy útiles para columnas que son frecuentemente consultadas o usadas en filtros (como numero\_motor). En este caso, dado que es probable que busques vehículos o grúas por su número de motor, un índice único mejora la velocidad de esas búsquedas.
* También es recomendable crear índices en llaves foráneas si esperas hacer consultas de unión (JOIN) frecuentes. Por ejemplo, id\_cliente en clientes o id\_grua en solicitudes\_grua.

¿Cuándo NO usar índices?

Aunque los índices mejoran la velocidad de las consultas, también pueden ralentizar las inserciones, actualizaciones y eliminaciones de datos, ya que el índice debe actualizarse cada vez que se modifican los datos en la tabla. Por lo tanto, es importante crear índices solo en las columnas que se usan frecuentemente en búsquedas, filtros u ordenamientos.

1. Generar un indice no único para las fechas de contratación y para la fecha de solicitud de grúa.



Explicación de los índices:

Índice no único: El índice no único se crea en las columnas fecha\_contratacion y fecha\_hora\_solicitud. Esto mejorará la velocidad de las consultas que filtren por fecha, pero no garantizará que los valores en estas columnas sean únicos, ya que pueden existir múltiples registros con la misma fecha.

Beneficios de los índices no únicos:

* Mejora la consulta: Si realizas frecuentemente consultas para obtener contratos o solicitudes de grúa dentro de un cierto rango de fechas, este índice ayudará a acelerar esas búsquedas.
* No afecta la unicidad: A diferencia de los índices únicos, este índice no impedirá que haya registros con fechas duplicadas.

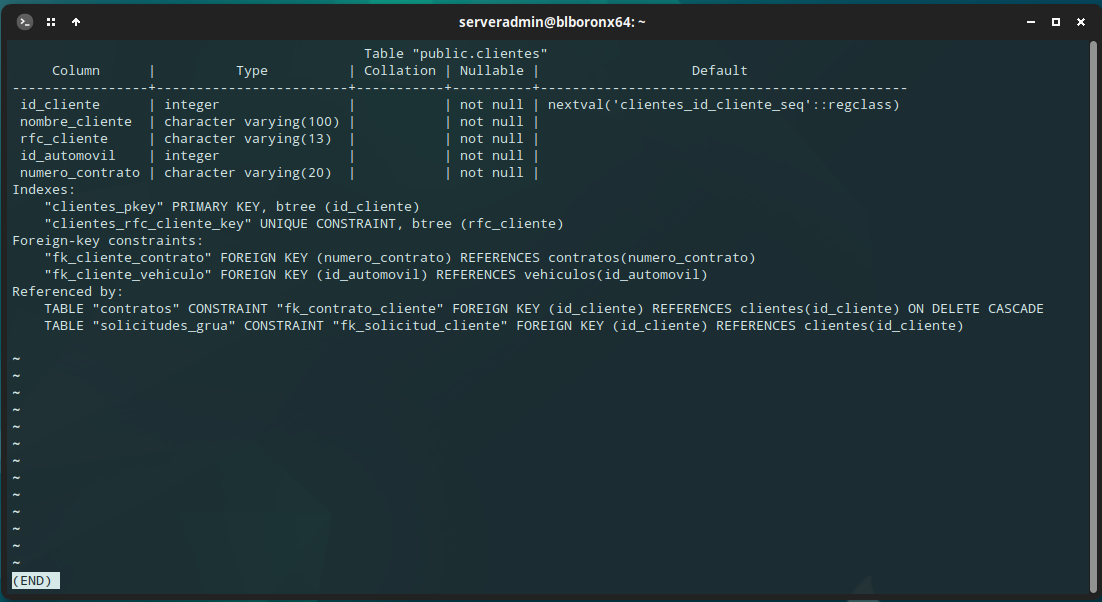
¿Cuándo usar índices no únicos?

* Son útiles cuando hay muchas consultas con filtros que usan estas columnas, como buscar todos los contratos de un mes o todas las solicitudes de grúa realizadas en un día específico.
* Si estas columnas tienen un alto volumen de registros con fechas repetidas, el índice no único es adecuado para mejorar el rendimiento sin afectar la unicidad de los datos.

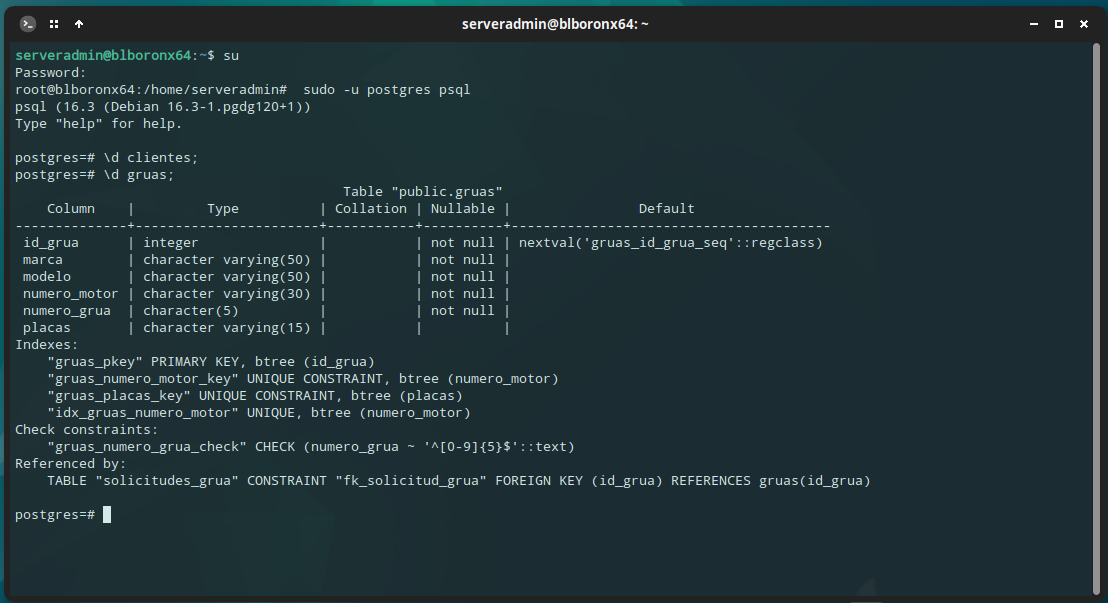
Actividad 4

Mostrar la estructura de las tablas.

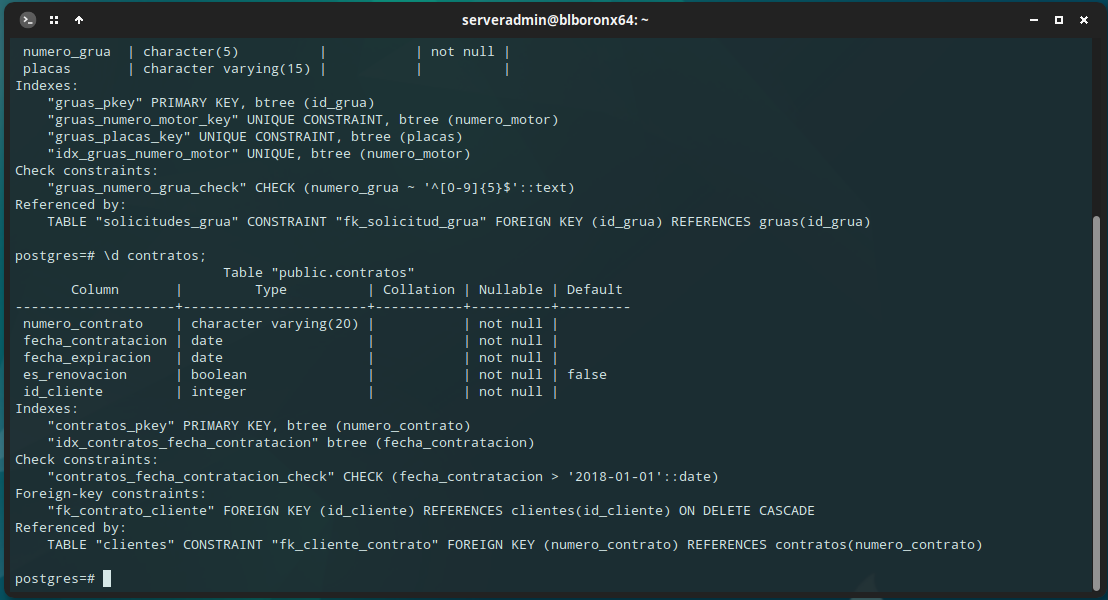
Clientes



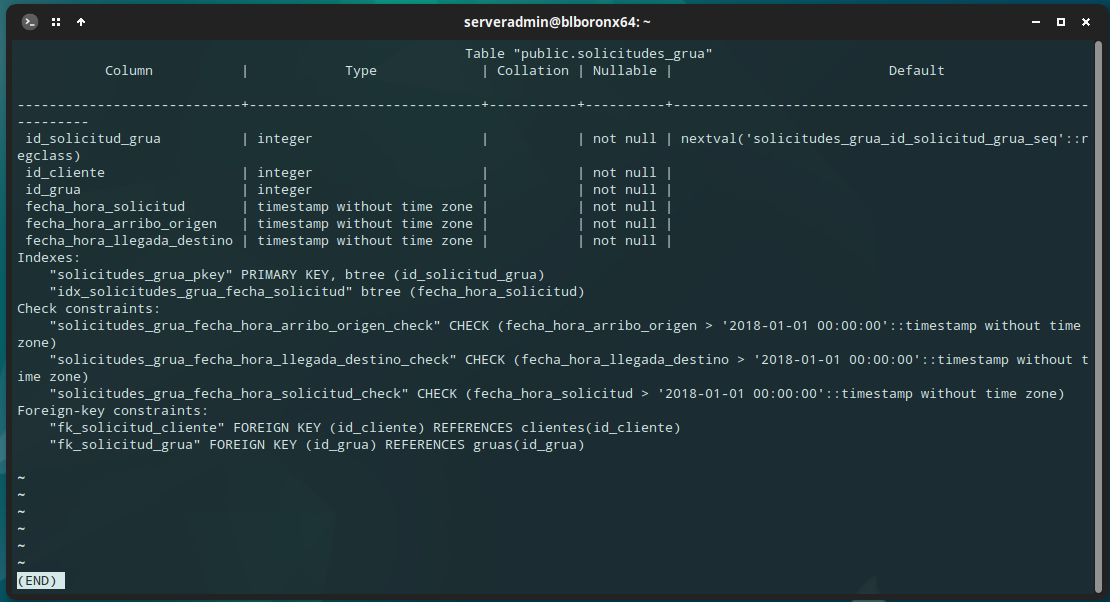
Grúas



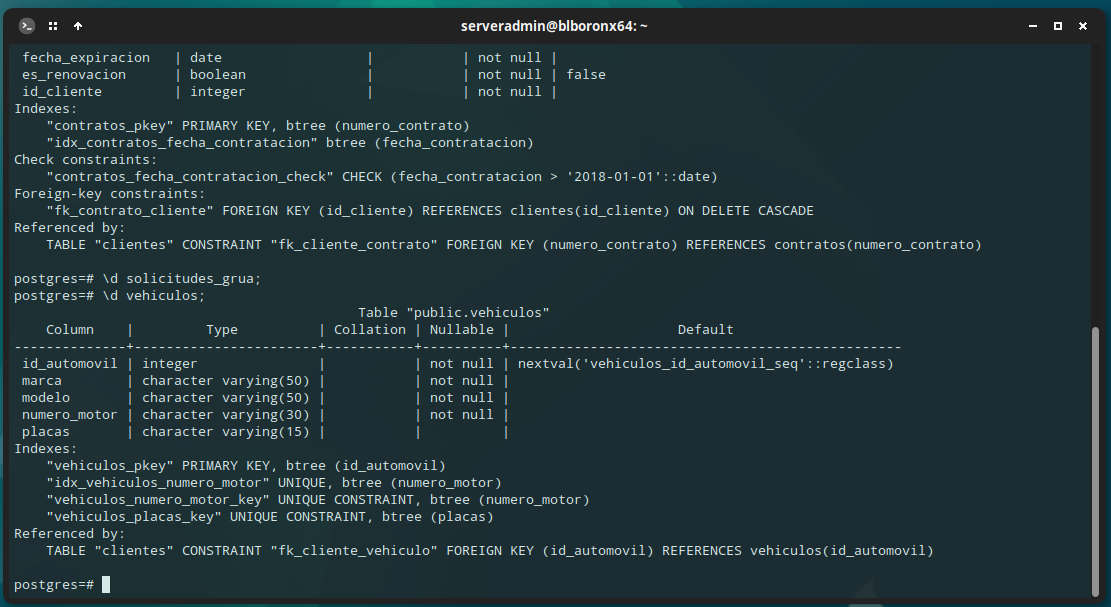
Contratos



Solicitudes



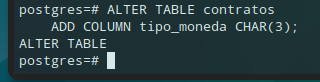
Vehículos



Actividad 5.

https://www.postgresql.org/docs/9.1/static/sql-altertable.html

Agregar a la tabla de (contratos/seguros/pólizas) una columna de tipo char de 3 caracteres para indicar si el tipo de moneda utilizado por ejemplo MXN, USD.



Cambiar el tipo de dato de renovación de contrato por un char de 5 caracteres.



Es necesario que la fecha de expiración sea campo obligado.

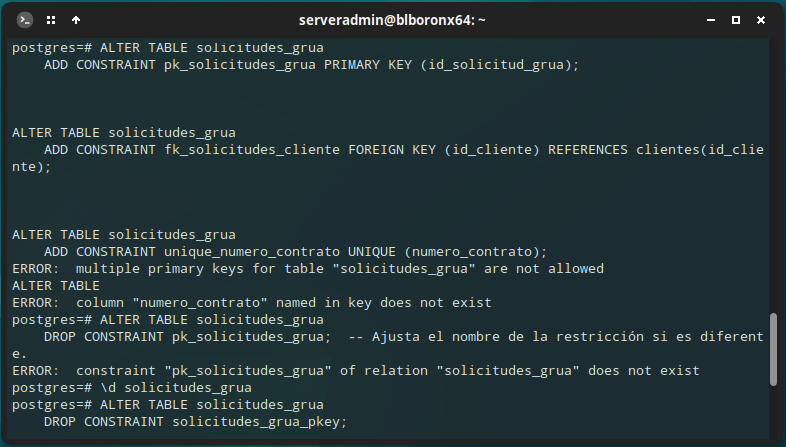


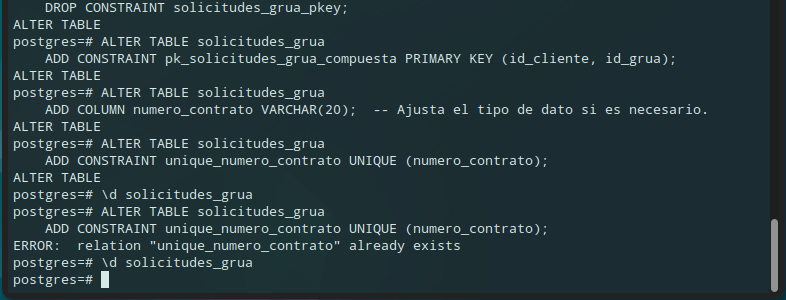
Actividad 6.

https://www.postgresql.org/docs/9.1/static/sql-altertable.html

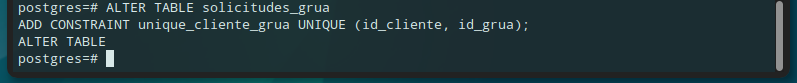
Utilizando el comando ALTER

* Adicionar tres constraints a una tabla (Primary Key, Foreign key y unique).

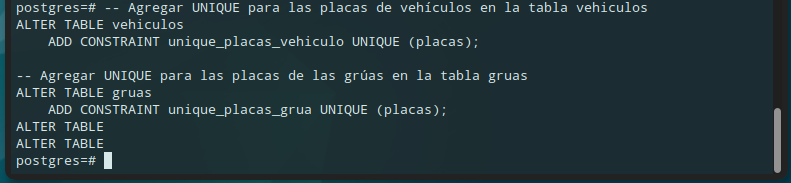




* Agregar mediante ALTER una llave primaria compuesta (id\_cliente, id\_grua) a la tabla (uso\_gruas/llamado\_grua/solicitud\_grua).



* Agregar mediante ALTER llaves candidato con UNIQUE las placas de los vehículos tanto para automóviles como para grúas.



* Inhabilitar el constraint foreign key ALTER TABLE nombre\_de\_la\_tabla DISABLE TRIGGER ALL;

Modificar objetos autoincrementables

https://www.postgresql.org/docs/9.6/static/sql-altersequence.html



* De forma gráfica cambiar el objeto secuanciador mediante la sección de sequences ó SELECT setval('public.automoviles\_id\_automovil\_seq', 1, true);

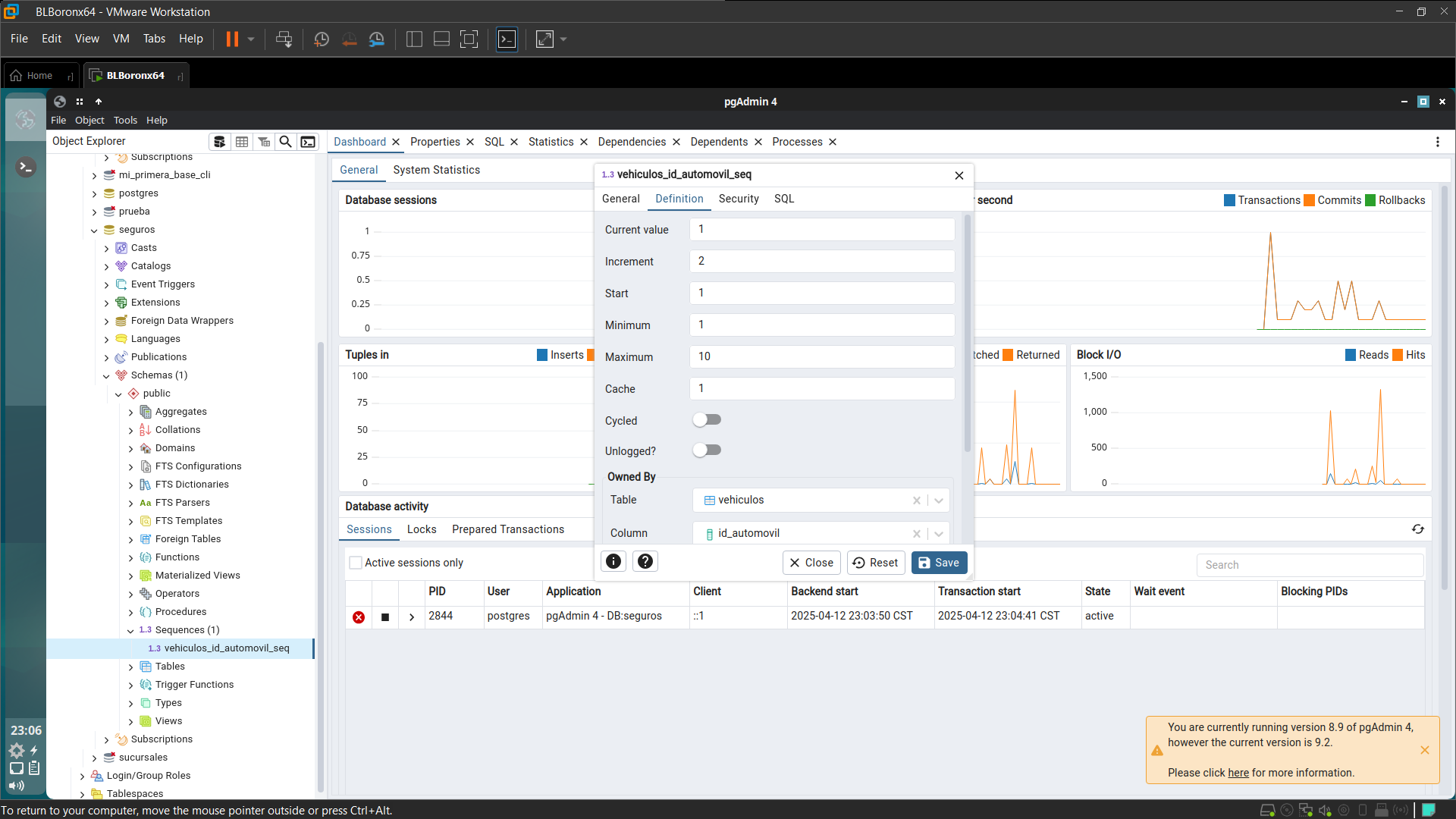
ALTER SEQUENCE public.automoviles\_id\_automovil\_seq

INCREMENT 2

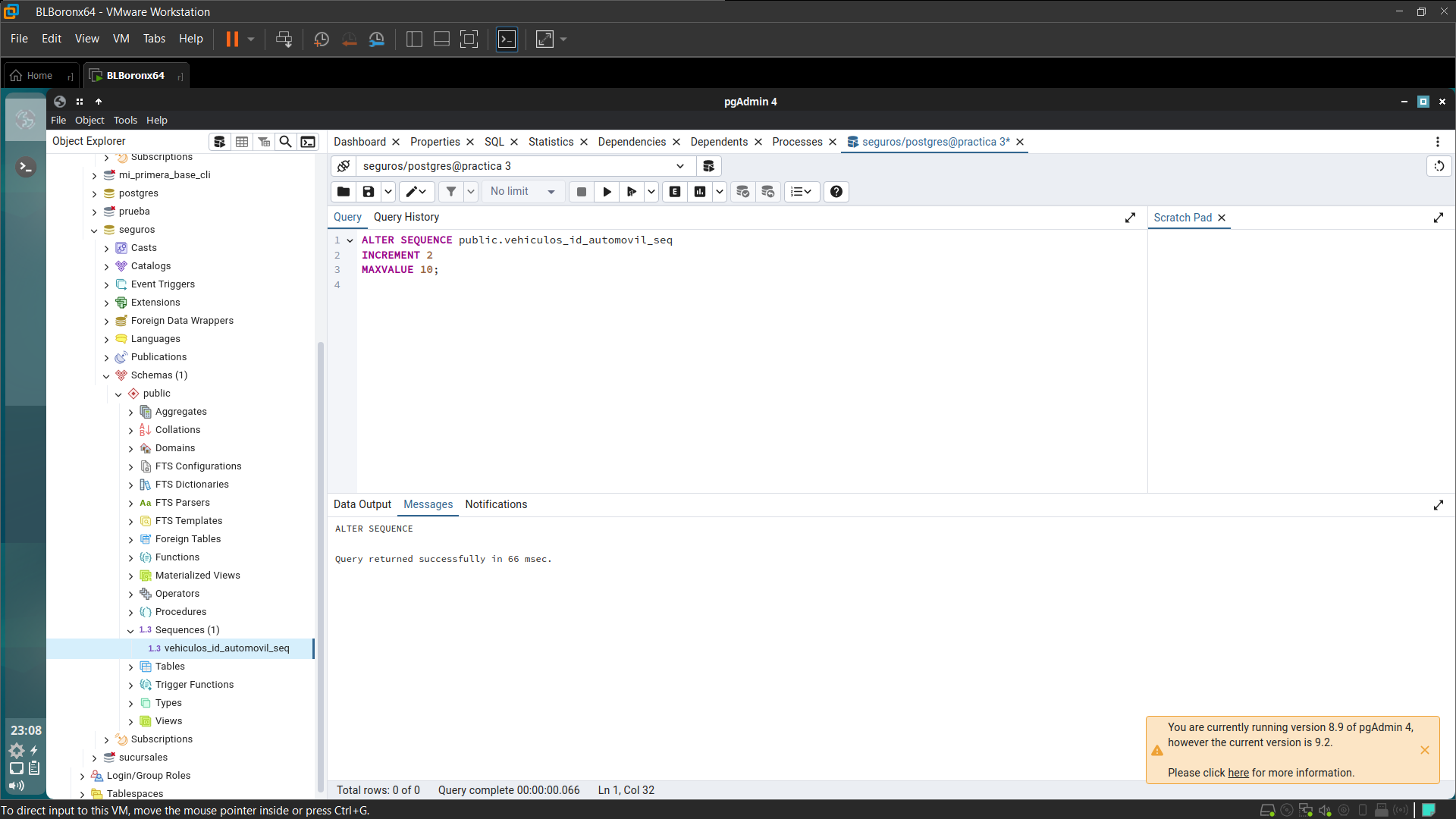
MAXVALUE 10;

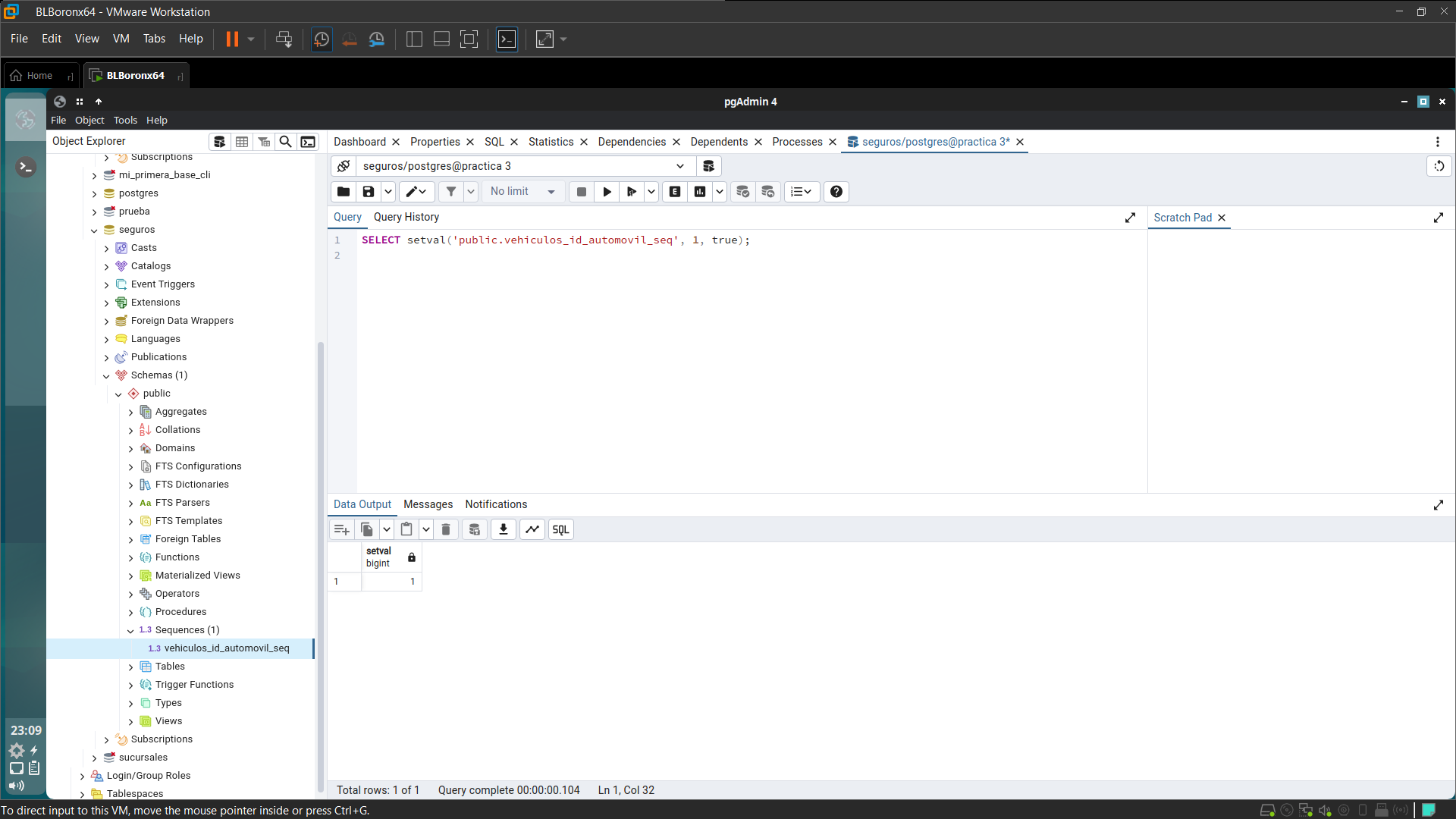
Existen dos fomas de hacerlo:

* Mediante las propiedades



* Mediante el query

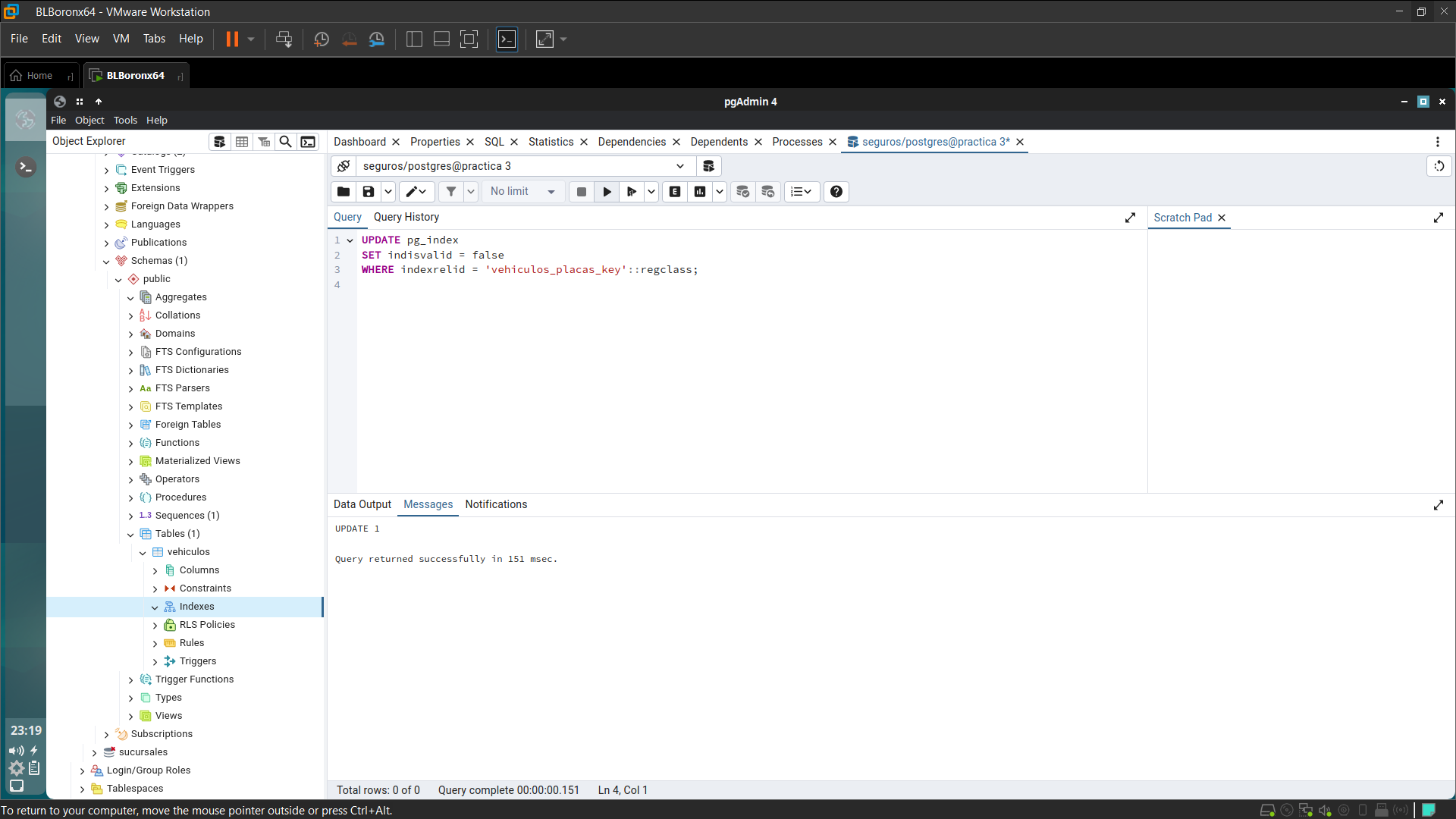




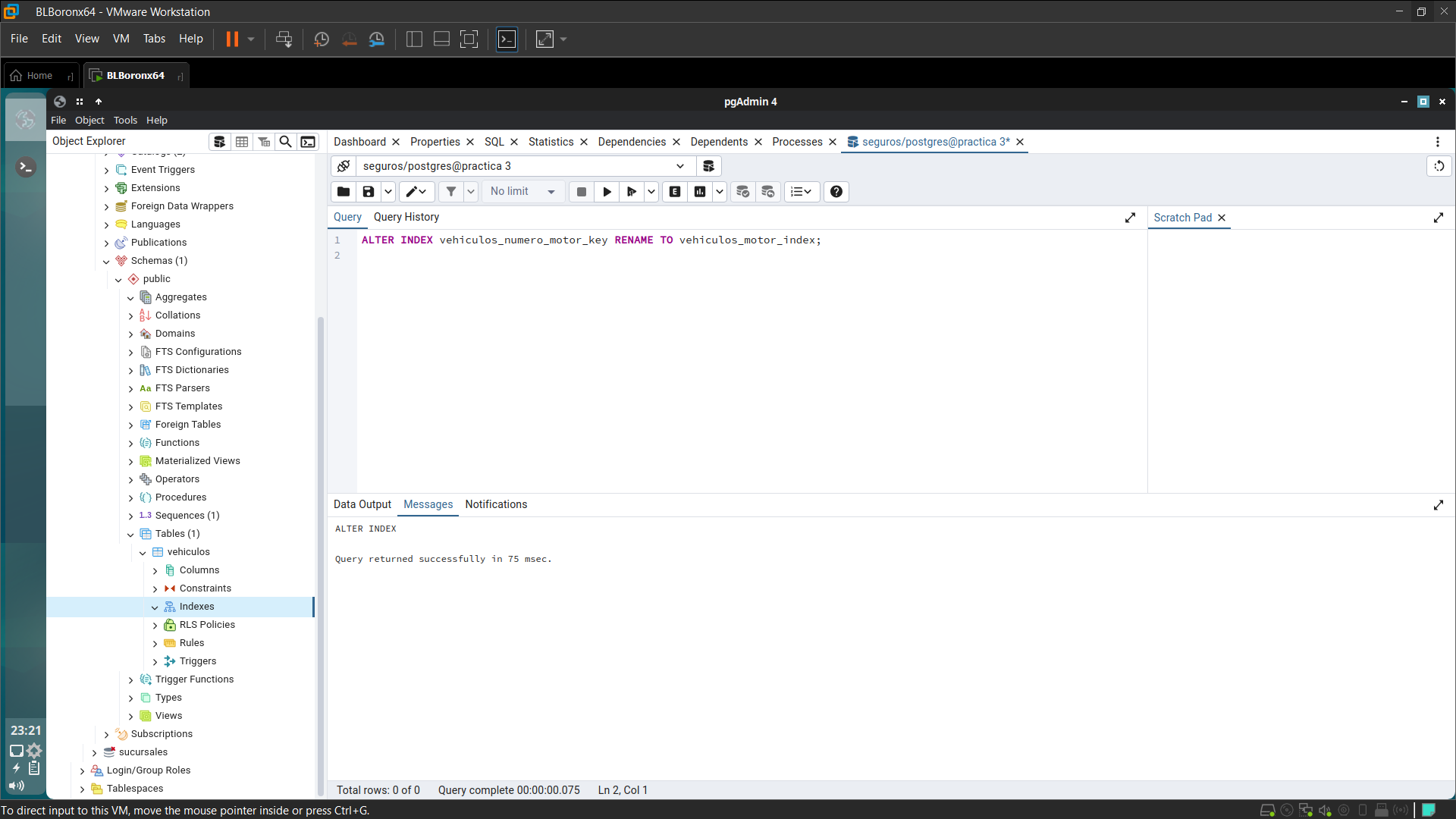
* Modificar un índice para (1) deshabilitarlo, (2) renombrarlo y (3) reconstruirlo

En este caso seleccionamos el índice referente a las placas de los vehículos

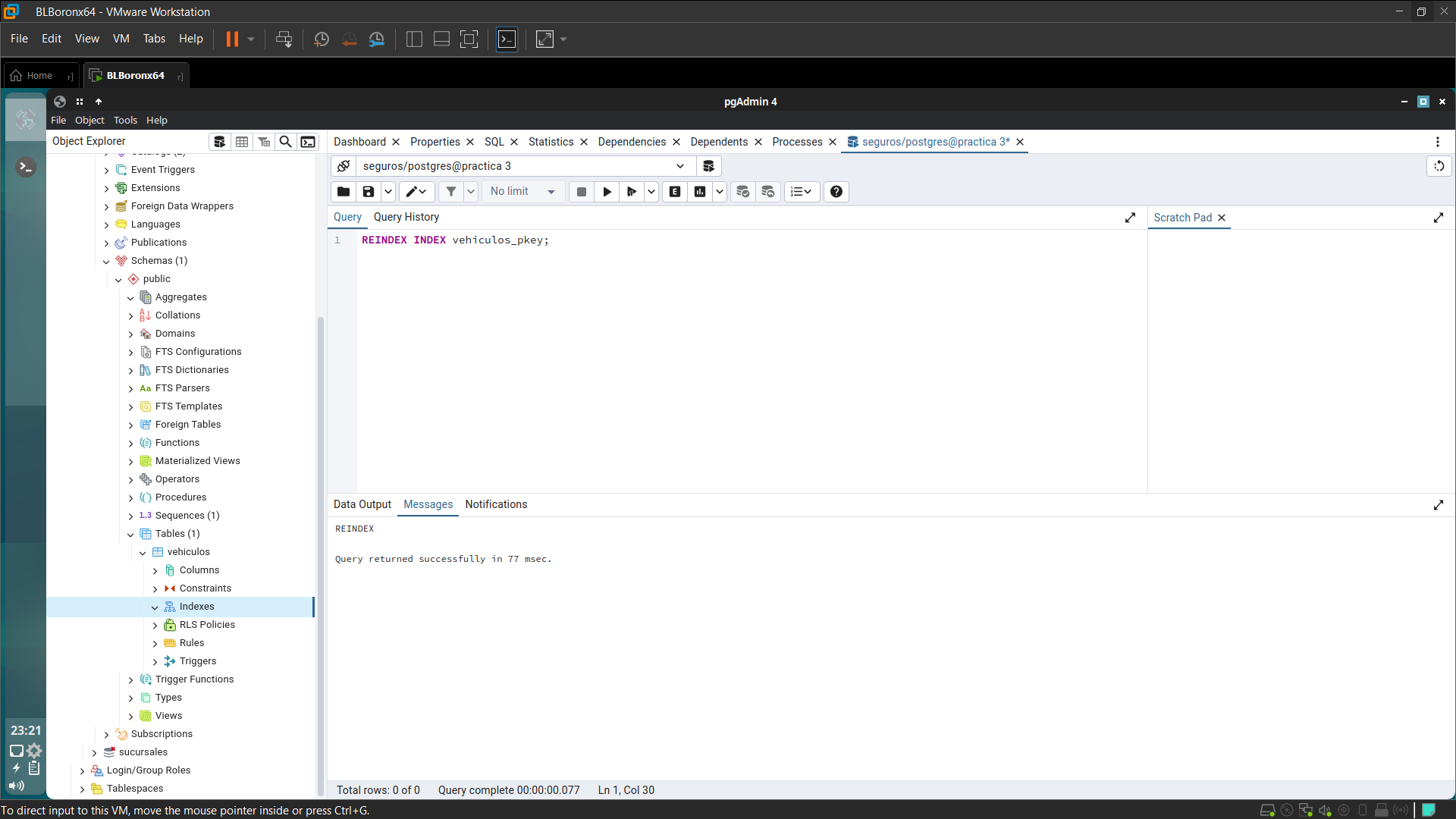
(1). update pg\_index set indisvalid = false where indexrelid = ‘nombre\_indice'::regclass



(2) https://www.postgresql.org/docs/9.6/static/sql-alterindex.html



(3) https://www.postgresql.org/docs/9.6/static/sql-reindex.html



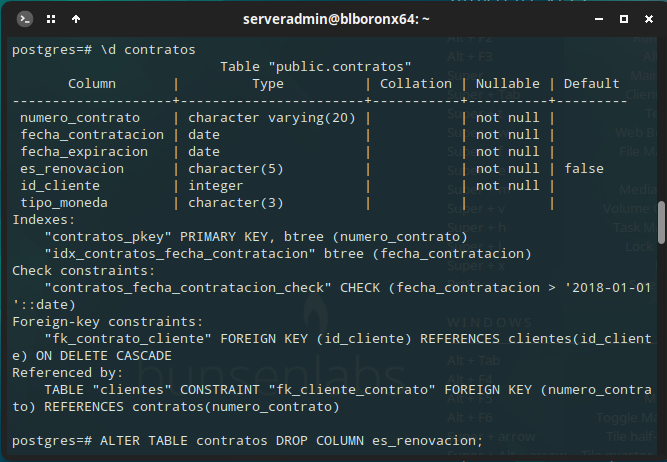
Actividad 7.

Realizar operaciones de borrado como:

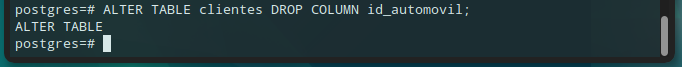
* Eliminar una columna

ALTER TABLE nombre\_de\_la\_tabla DROP COLUMN nombre\_de\_la\_columna.

Elimine la columna de renovación de contrato.



* Elimine en la tabla (clientes/usuarios/personas) la columna de id\_automovil. En caso de error explique por qué sucedió.



Esto no genera error porque PostgreSQL automáticamente elimina el constraint cuando eliminas una columna que es parte de una sola restricción, como en este caso. PostgreSQL te permite eliminar una columna con una clave foránea sin error, si esa columna es el único campo usado por la restricción. La restricción se elimina implícitamente.

* Eliminar el constraint unique de RFC de (clientes/usuarios/personas)

ALTER TABLE nombre\_de\_la\_tabla DROP nombre constraint.



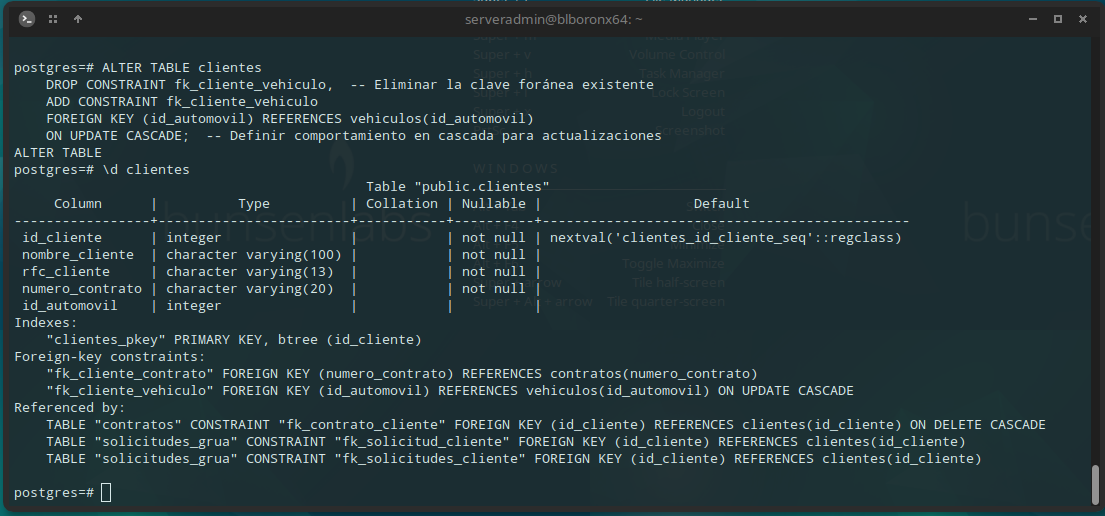
Actividad 8.

* Realizar operaciones restringidas, en cascada y puesta en nulo

UPDATE

DELETE

Intente modificar la tabla de (clientes/usuarios/personas) en la llave foránea id\_automovil para que en caso de una actualización el valor se actualice en cascada.



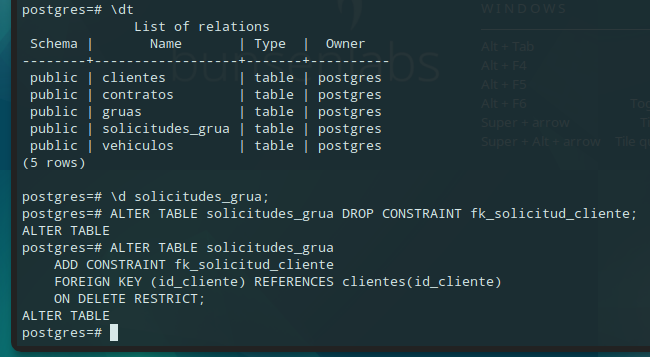
* Explique qué pasos son necesarios para lograrlo en postgres.

Para la alteración de una tabla a la que se desea agregar algún ON DELETE u ON UPDATE (sea el caso), ocupamos de lo siguiente:

1. Identificar la relación de clave foránea: Antes de modificar la tabla, necesitamos saber cuál es la columna de llave foránea y cuál es la tabla a la que se refiere.
2. Eliminar la clave foránea existente: Si la clave foránea ya está definida, deberás eliminarla antes de volver a agregarla con las nuevas reglas. Esto se puede hacer usando el comando DROP CONSTRAINT.
3. Agregar la clave foránea con CASCADE: Una vez eliminada la clave foránea anterior, puedes agregar la nueva definición de la clave foránea con la opción ON UPDATE CASCADE.
4. ON UPDATE CASCADE: Esta opción asegura que, si el valor de la clave primaria cambia en la tabla referenciada, los registros correspondientes en la tabla referenciadora también se actualizarán automáticamente.

Estos 4 pasos pueden verse reflejados dentro de la anterior imagen, donde justo la comentamos para que sea visible lo que en esta pregunta hemos resaltado.

* Intente modificar la tabla de (uso\_gruas/llamado\_grua/solicitud\_grua) para que en caso de eliminación se restrinja.



* Explique la importancia de definir estos valores al crear la tabla.

Definir los valores de restricción como ON DELETE RESTRICT, ON DELETE CASCADE, ON UPDATE CASCADE, y ON DELETE SET NULL al crear una tabla es crucial para garantizar la integridad referencial de los datos en una base de datos relacional.

* ON DELETE RESTRICT: Impide que se elimine un registro en la tabla principal (por ejemplo, clientes) si hay registros relacionados en la tabla secundaria. Previene la pérdida de datos relacionados y asegura que no se pierdan datos críticos.
* ON DELETE CASCADE: Permite que cuando se elimina un registro en la tabla principal, todos los registros relacionados en la tabla secundaria también se eliminen automáticamente. Mantiene la consistencia de los datos y facilita el mantenimiento.
* ON DELETE SET NULL: Cuando se elimina un registro en la tabla principal, los valores correspondientes en la tabla secundaria se establecen como NULL. Preserva la estructura de la tabla secundaria y evita la pérdida total de información.
* ON UPDATE CASCADE: Si el valor de una clave primaria o clave única en la tabla principal cambia, los valores correspondientes en la tabla secundaria también se actualizan automáticamente. Mantiene la coherencia de los datos y evita datos desactualizados
* ON UPDATE RESTRICT: Impide que se actualice un registro en la tabla principal si hay registros relacionados en la tabla secundaria. Mantiene la integridad referencial

**Conclusiones**

* Guerrero López Uriel Ivan

A lo largo de este proceso de aprendizaje, he desarrollado una comprensión sólida sobre la creación y modificación de tablas e índices utilizando el lenguaje SQL, una habilidad esencial para cualquier trabajo relacionado con bases de datos. La creación de tablas ha sido una de las primeras actividades en las que me enfoqué, donde aprendí a definir columnas con tipos de datos adecuados, establecer claves primarias y foráneas, y asegurarme de que las relaciones entre las tablas fueran claras y eficientes. De manera paralela, comprendí cómo los índices pueden mejorar la velocidad de las consultas, lo que resulta crucial cuando se trabaja con grandes volúmenes de datos.

Uno de los aspectos más interesantes que descubrí fue el uso de las instrucciones ALTER y DROP para modificar y eliminar objetos dentro de una base de datos. Estas instrucciones permiten hacer cambios en la estructura de la base de datos de manera eficiente sin perder datos, algo que inicialmente me parecía arriesgado. Sin embargo, entendí que, con una planificación adecuada y una ejecución precisa, estas herramientas son extremadamente útiles. A pesar de esto, la curva de aprendizaje fue algo empinada, sobre todo al principio, cuando la sintaxis exacta de los comandos me resultaba difícil de recordar y aplicar correctamente. Afortunadamente, con la práctica constante y la repetición de ejercicios, logré superar este obstáculo, lo que me permitió entender a fondo cómo hacer modificaciones estructurales en bases de datos de forma efectiva y sin comprometer la integridad de los datos.

* Muñoz San Agustin Victoria Monserrat

Este tema de SQL ha sido todo un reto, pero también muy enriquecedor. Aprender a crear tablas fue uno de los primeros pasos, y me di cuenta de lo importante que es definir bien las columnas, los tipos de datos y las relaciones entre las tablas desde el principio. Al principio me costaba trabajo asegurarme de que todo estuviera bien estructurado, ya que hay muchos detalles a tener en cuenta. También descubrí la importancia de los índices para que las búsquedas sean más rápidas, algo que no había pensado tanto antes.

Lo que más me sorprendió fue lo útil que es la instrucción ALTER. Aprender a cambiar la estructura de una tabla sin perder información me pareció una herramienta increíblemente poderosa, especialmente cuando tienes que agregar o quitar columnas en tablas ya existentes. De igual forma, el uso de DROP fue algo que al principio me dio un poco de miedo, ya que borrar algo de la base de datos puede parecer irreversible. Sin embargo, entendí que, si se usa con cuidado, es un comando indispensable cuando ya no se necesita un objeto en la base de datos. Al principio cometí varios errores con la sintaxis y me confundí con la secuencia de los comandos, pero con la práctica y haciendo más ejercicios, logré dominarlo. Hoy me siento mucho más confiado para trabajar con SQL y hacer cambios en la estructura de una base de datos sin miedo a equivocarme.

**Referencias**

Amazon (2024). ¿Qué es SQL (lenguaje de consulta estructurada)? Recuperado 10 de abril de 2025 de https://aws.amazon.com/es/what-is/sql/

IBM. (2025). Lenguaje de definición de datos. Recuperado el 10 de abril de 2025 de https://www.ibm.com/docs/es/i/7.5.0?topic=programming-data-definition-language

Pacheco, M. (2017) Lenguaje SQL Básico. Unidades de Apoyo para el Aprendizaje. CUAED/Facultad de Contaduría y Administración-UNAM. Recuperado el 10 de abril de 2025 de https://repositorio-uapa.cuaed.unam.mx/repositorio/moodle/pluginfile.php/1398/mod\_resource/content/5/contenido/index.html

1. < > Los corchetes triangulares deben ser reemplazados por el nombre del objeto correspondiente.

   ( ) Los paréntesis son elementos propios de la sintaxis.

   [ ] Los corchetes cuadrados indican argumentos opcionales.

   | La barra vertical separa distintas alternativas. [↑](#footnote-ref-2)
2. … Los puntos suspensivos indican repetición del elemento sintáctico precedente. [↑](#footnote-ref-3)