

Administración de base de datos con PostgreSQL



Ing. Mariuxi Paola Zea Ordóñez, Mg.
Ing. Jimmy Rolando Molina Ríos, Mg.
Ing. Fausto Fabián Redrován Castillo, Mg.

Ingeniería y Tecnología



ADMINISTRACIÓN DE BASES DE DATOS CON POSTGRESQL

Ing Mariuxi Paola Zea Ordóñez, Mg.

Ing. Jimmy Rolando Molina Ríos, Mg.

Ing. Fausto Fabían Redrován Castillo, Mg.



Editorial Área de Innovación y Desarrollo, S.L

Quedan todos los derechos reservados. Esta publicación no puede ser reproducida, distribuida, comunicada públicamente o utilizada, total o parcialmente, sin previa autorización.

ÁREA DE INNOVACIÓN Y DESARROLLO, S.L.

© del texto: **Los autores**

C/ Els Alzamora, 17 - 03802 - ALCOY (ALICANTE) info@3ciencias.com

Primera edición: **abril 2017**

ISBN: **978-84-946684-6-3**

DOI: <http://dx.doi.org/10.17993/IngyTec.2017.18>

ÍNDICE

0. INTRODUCCIÓN	9
1. BASE DE DATOS	11
1.1. IMPLEMENTACIÓN DE UNA BASE DE DATOS	12
2. POSTGRESQL	12
3. CASO DE ESTUDIO	13
4. MODELO ENTIDAD RELACIÓN	17
5. MODELO RELACIONAL	21
6. CREACIÓN DE LA BASE DE DATOS	25
7. CREACION DE TABLAS	25
8. INTEGRIDAD DE ENTIDAD	26
9. CLAVE PRINCIPAL	26
10. INTEGRIDAD REFERENCIAL	27
11. CLAVE FORÁNEA	27
12. CHECK	28
13. CONSULTAS CON LA CLÁUSULA ORDER BY	28
15. CONSULTAS COMPLEJAS	31
15.1 CONSULTA DE SELECCIÓN	32
15.2 PROYECCIÓN	33
15.3 UNIÓN	34
15.4 INTERSECCIÓN	35
15.5 DIFERENCIA	36
15.6 COMBINACIÓN	37
15.7 PRODUCTO CARTESIANO	37
15.8 COMBINACIONES EXTERNAS	39
16. FUNCIONES PROCEDURALES	42
16.1 FUNCIÓN SQL	43
16.2 FUNCIÓN PLPGSQL	43
17. TRIGGERS	46
18. ENCRIPCIÓN	47
19. PRIVILEGIOS Y USUARIOS	50
20. AUDITORIAS A LAS BASES DE DATOS	53

21.	TRANSACCIONES	56
21.1	PROPIEDADES DE LA TRANSACCIÓN	57
22.	HERRAMIENTA DE RESGUARDO.....	62
23.	HERRAMIENTA DE RESTAURACIÓN.....	70
24.	HERRAMIENTA DE MANTENIMIENTO	75
25.	BIBLIOGRAFÍA.....	79

ÍNDICE DE ILUSTRACIONES Y TABLAS

Ilustración 1. Ficha Caso de Estudio.....	14
Ilustración 2. Ejemplo.	18
Ilustración 3. Ejemplo.	19
Ilustración 4: Ejemplo Modelo Entidad Relación.	20
Ilustración 5. Modelo Relacional.....	22
Ilustración 6. Transformación del diagrama entidad relación al modelo relacional.....	22
Ilustración 7. Transformación del diagrama entidad relación al modelo relacional.....	23
Ilustración 8. Ejemplo Modelo Relacional.....	24
Ilustración 9. Creación de la Base de Datos.	25
Ilustración 10. Creación de Tablas.	25
Ilustración 11. Ejemplo Restricción Unique.	26
Ilustración 12. Ejemplo Clave Foránea.....	27
Ilustración 13. Ejemplo Check.....	28
Ilustración 14. Ejemplo Cláusula Order By.....	29
Ilustración 15. Registros Tabla Empleados.	29
Ilustración 16. Ejemplo Consultas con Predicado.	30
Ilustración 17. Ejemplo Consultas con Predicado.	31
Ilustración 18. Ejemplo Consulta de Selección.	32
Ilustración 19. Consulta a Estructurar.....	32
Ilustración 20. Secuencia ejecutada.....	32
Ilustración 21. Ejemplo Proyección.....	33
Ilustración 22. Sentencia Proyección.	33
Ilustración 23. Resultado Proyección.....	34
Ilustración 24. Ejemplo Unión.....	34
Ilustración 25. Resultado Unión.....	35
Ilustración 26. Resultado Intersección.....	35
Ilustración 27. Cláusula Intersect.....	36
Ilustración 28. Estructura Intersect.....	36
Ilustración 29. Ejemplo Diferencia.....	37
Ilustración 30. Ejemplo Producto Cartesiano.....	37
Ilustración 31. Ejemplo Producto Cartesiano.....	38
Ilustración 32. Sentencia Producto Cartesiano.....	38
Ilustración 33. Ejemplo Producto Cartesiano.....	38
Ilustración 34. Sentencia Combinaciones Internas.....	39
Ilustración 35. Resultados Combinaciones Internas.....	39
Ilustración 36. Cláusula Left Outer Join.....	40
Ilustración 37. Resultados Cláusula Left Outer Join.....	40
Ilustración 38. Ejemplo Right Outer Join.....	40
Ilustración 39. Resultados Right Outer Join.....	41
Ilustración 40. Ejemplo Full Outer Join.....	41
Ilustración 41. Resultado Full Outer Join.....	41
Ilustración 42. Funciones Procedurales.....	42
Ilustración 43. Ejemplo Función SQL.....	43
Ilustración 44. Función SQL.....	43
Ilustración 45. Ejemplo Función PLPGSQL.....	44
Ilustración 46. Función PLPGSQL.....	44
Ilustración 47. Ejemplo Función PLPGSQL.....	45
Ilustración 48. Resultado Función PLPGSQL.....	45
Ilustración 49. Creación de la Función.....	46
Ilustración 50. Creación del Trigger.....	47
Ilustración 51. Comprobación de Trigger.....	47
Ilustración 52. Ejemplo Encriptación.....	48
Ilustración 53. Resultados Consulta.....	49
Ilustración 54. Visualización de datos.....	49
Ilustración 55. Registro de Dato Encriptado.....	49

Ilustración 56. Vista con datos Encriptados.....	50
Ilustración 57. Creación de Usuarios.....	51
Ilustración 58. Creación de Permisos a Usuarios.....	52
Ilustración 59. Creación de Usuarios por Grupos.....	52
Ilustración 60. Sentencia para para visualizar Usuarios.....	52
Ilustración 61. Pasos para crear una Auditoría.....	53
Ilustración 62. Explicación Funcionamiento de la Función.....	55
Ilustración 63. Resultados Tabla Auditoría.....	56
Ilustración 64. Tabla Contratos.....	56
Ilustración 65. Ejemplo Transacción.....	58
Ilustración 66. Resultado Consulta.....	58
Ilustración 67. Propiedad de Atomicidad.....	58
Ilustración 68. Resultado Consulta.....	59
Ilustración 69. Comprobación de Atomicidad.....	59
Ilustración 70. Resultado Consulta.....	60
Ilustración 71. Propiedad de Consistencia.....	60
Ilustración 72. Ejemplo Transacción.....	61
Ilustración 73. Visualización de información.....	61
Ilustración 74. Uso de Comando COMMIT.....	62
Ilustración 75. Realización de Consulta.....	62
Ilustración 76. Ejemplo Herramienta de Resguardo.....	63
Ilustración 77. Ejemplo Herramienta de Resguardo.....	63
Ilustración 78. Campo de entrada "Format".....	64
Ilustración 79. Campo de entrada "Encoding".....	64
Ilustración 80. Campo de entrada "Rolename".....	65
Ilustración 81. Pestaña "Dump Options #1".....	65
Ilustración 82. Pestaña "Dump Options #2".....	66
Ilustración 83. Pestaña "Objects".....	66
Ilustración 84. Antes del Backup.....	70
Ilustración 85. Después del Backup.....	67
Ilustración 86. Dirección de la carpeta bin en cmd.....	67
Ilustración 87. Comando pg_dump.....	68
Ilustración 88. Verificación de Respaldo.....	68
Ilustración 89. Información comando pg_dump.....	69
Ilustración 90. Ejemplo Restauración.....	70
Ilustración 91. Pestañas de Restauración.....	71
Ilustración 92. Pestaña "Restore Options #1".....	71
Ilustración 93. Pestaña "Restore Options #2".....	72
Ilustración 94. Pestaña "Objects".....	72
Ilustración 95. Pestaña "Messages".....	73
Ilustración 96. Restauración por cmd.....	73
Ilustración 97. Comando createdb.....	74
Ilustración 98. Comandos psql.....	74
Ilustración 99. Explorador de objetos en pgAdmin III.....	74
Ilustración 100. Opción Mantenimiento.....	76
Ilustración 101. Opciones para el mantenimiento.....	76
Ilustración 102. Operación realizada en el mantenimiento.....	76
Ilustración 103. Editor de consultas para el mantenimiento.....	77
Ilustración 104. Acceso a la base datos a través del SQL Shell.....	77
Ilustración 105. Informe del Vacuum.....	78
Tabla 1. Componentes del modelo entidad relación.....	17
Tabla 2. Transformación del modelo entidad relación al modelo relacional.....	21
Tabla 3. Consultas con Predicado.....	30
Tabla 4. Encriptación.....	48
Tabla 5. Matriz de Trazabilidad de los Usuarios.....	51
Tabla 6. Formatos.....	64

o. INTRODUCCIÓN

En la actualidad, las bases de datos cumplen una función muy importante en los sistemas de información, la mayoría de las empresas sean públicas o privadas tiene sus procesos automatizados y esto hace que los sistemas manuales queden obsoletos a la hora de realizar búsquedas, modificaciones y cualquier obtención de información de dicha empresa, es por ello que la utilización de una base de datos se hace indispensable al momento de almacenar grandes volúmenes de información con la que cuenta la organización. Una base de datos es un banco de información, el cual contiene datos relacionados entre sí y se encuentran agrupados o estructurados; además son manipulados por programas conocidos actualmente como Sistema de Gestión de Base de Datos (SGBD). En este caso se ha utilizado PostgreSQL como SGBD para la realización y ejecución del proyecto.

Cabe mencionar que para poder desarrollar una correcta base de datos es necesario realizar el Modelo Entidad Relación y un buen modelado relacional englobando todos los requerimientos o acciones que cumple la empresa, una vez que esta fase se haya ejecutado con éxito se podría proceder a la creación y a la implementación de los conceptos para la buena administración de una base de datos.

Es por ello que el presente libro presenta un caso de estudio acerca de la empresa inmobiliaria “Tierra Prometida” que servirá de plataforma para realizar la administración de la estructura de la base de datos.

El documento se compone de cuatro capítulos que corresponden a: El primer Capítulo se trata acerca de la Programación Back-End, incluye scripts para crear la estructura de la base de datos, para consultas avanzadas, consultas con funciones, subqueries, store procedure, y triggers. En el Capítulo II, se aborda el tema de seguridad e integridad de la base de datos. A continuación, en el Capítulo III, se explica sobre Sistemas Transaccionales. Finalmente, en el Capítulo IV se trata acerca de las técnicas de recuperación.

1. BASE DE DATOS

Todas las bases de datos, desde las más sencillas hasta las más complejas, están compuestas por listas de información.

“Una base de datos es una colección de información organizada de tal modo que sea fácilmente accesible, gestionada y actualizada” (Rouse.).

Una base de datos permite almacenar diferentes tipos de información. Las bases de datos permiten a sus usuarios acceder, registrar y analizar datos de una manera rápida y sencilla.

“En informática, las bases de datos a veces se clasifican de acuerdo a su enfoque organizativo. El enfoque más frecuente es la base de datos relacional, una base de datos tabular en la que los datos se definen de manera que puede ser reorganizada y se accede en un número de maneras diferentes. Una base de datos distribuida es una que puede ser dispersada o replicada entre diferentes puntos de una red” (Rouse.).

“Los sistemas gestores de bases de datos son la herramienta más adecuada para almacenar los datos en un sistema de información debido a sus características de seguridad, recuperación ante fallos, gestión centralizada, estandarización del lenguaje de consulta y funcionalidad avanzada” (Rouse.).

Las ventajas de utilizar un almacenamiento estructurado se aprecian en diversos puntos, ya que afectan no solo a los datos sino también al propio uso.

Según, (Bases de datos) indica algunas de las ventajas más características como lo son las siguientes:

- **Mayor independencia.** Los datos son independientes de las aplicaciones que los usan, así como de los usuarios.
- **Mayor disponibilidad.** Se facilita el acceso a los datos desde contextos, aplicaciones y medios distintos, haciéndolos útiles para un mayor número de usuarios.
- **Mayor seguridad (protección de los datos).** Por ejemplo, resulta más fácil replicar una base de datos para mantener una copia de seguridad que hacerlo con un conjunto de ficheros almacenados de forma no estructurada. Además, al estar centralizado el acceso a los datos, existe una verdadera sincronización de todo el trabajo que se haya podido hacer sobre estos, con lo que esa copia de seguridad servirá a todos los usuarios.
- **Menor redundancia.** Un mismo dato no se encuentra almacenado en múltiples ficheros o con múltiples esquemas distintos, sino en una única instancia en la base de datos. Esto redundancia en menor volumen de datos y mayor rapidez de acceso.
- **Mayor eficiencia en la captura, codificación y entrada de datos.**

Siendo así ventajas importantes en el tratamiento de la información que se aloja dentro de una base de datos en cualquier organización.

1.1. IMPLEMENTACIÓN DE UNA BASE DE DATOS

Para la implementación de una base de datos, esta implica la definición de la estructura, más concretamente según **Fuente especificada no válida**. Se puede distinguir las siguientes fases en el proceso global de desarrollo de una base de datos:

- **Diseño lógico.** Independiente del SGBD empleado, es un diseño conceptual que pretende modelizar el contenido de la base de datos.
- **Diseño físico.** Es la adaptación del diseño conceptual a las particularidades del SGBD escogido.
- **Implementación.** Introducción de los datos en la base de datos.
- **Mantenimiento.** Monitorización de la actividad sobre la base de datos.

2. POSTGRESQL

“PostgreSQL es un sistema de gestión de bases de datos objeto-relacional, distribuido bajo licencia BSD y con su código fuente disponible libremente. Es el sistema de gestión de bases de datos de código abierto más potente del mercado” (Sobre PostgreSQL).

“PostgreSQL utiliza un modelo cliente/servidor y usa multiprocesos en vez de multihilos para garantizar la estabilidad del sistema. Un fallo en uno de los procesos no afectará el resto y el sistema continuará funcionando” (Sobre PostgreSQL).

Según el sitio oficial de PostgreSQL son varias las características de este software, las cuales se detallan a continuación:

- Es una base de datos 100% ACID
- Integridad referencial
- Tablespaces
- Nested transactions (savepoints)
- Replicación asincrónica/sincrónica / Streaming replication - Hot Standby
- Two-phase commit
- PITR - point in time recovery
- Copias de seguridad en caliente (Online/hot backups)
- Unicode
- Juegos de caracteres internacionales
- Regionalización por columna
- Multi-Version Concurrency Control (MVCC)
- Múltiples métodos de autenticación
- Acceso encriptado via SSL
- Actualización in-situ integrada (pg_upgrade)

- SE-postgres
- Completa documentación
- Licencia BSD
- Disponible para Linux y UNIX en todas sus variantes (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64) y Windows 32/64bit.

Otra definición sobre PostgreSQL indica que es “Un sistema de base de datos relacionales es un sistema que permite la manipulación de acuerdo con las reglas del álgebra relacional. Los datos se almacenan en tablas de columnas y renglones. Con el uso de llaves, esas tablas se pueden relacionar unas con otras.”

3. CASO DE ESTUDIO

Para comenzar iniciaremos con el siguiente caso de estudio que será automatizado en una base de datos.



Tierra Prometida S. A.

En este apartado se describe una empresa inmobiliaria, Tierra Prometida, que está especializada en el alquiler de pisos y casas amuebladas.

Esta empresa se encarga de dar publicidad a los inmuebles que ofrece en alquiler, tanto en prensa local como nacional, entrevista a los posibles inquilinos, organiza las visitas a los inmuebles y negocia los contratos de alquiler. Una vez firmado el alquiler, la empresa asume la responsabilidad del inmueble, realizando inspecciones periódicas para comprobar su correcto mantenimiento.

Esta información se encuentra actualmente en fichas:

Ilustración 1. Ficha Caso de Estudio.

Ref. 1234 Tipo: Piso de ocasión C/Lérida 24, zona Capuchinos. 90 m2. 3 hab., 1 baño, cocina, 5 armarios empotrados, puerta blindada, parquet, totalmente reformado, todo exterior. Precio venta: 14.000.000 Precio alquiler: Propietario: Luis Herranz. Telf. 964 223344. Visitas:	Ref. 9876 Tipo: Villa Urb. Las Palmas 8A, Benicasim. Villa de 140 m2, parcela de 820 m2. 7 hab., 3 baños, 5 armarios empotrados, terraza de 40 m2, gas ciudad, calefacción central. Precio venta: a convenir Precio alquiler: Propietario: Carmela Aparicio. Telf. 964 221144 Visitas:																								
<table border="1"> <tbody> <tr> <td>12/2/01</td> <td>19:00</td> <td>Marcela Torres</td> <td>baño peq.; piso alto</td> </tr> <tr> <td>21/3/01</td> <td>12:00</td> <td>Carlos Jara</td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	12/2/01	19:00	Marcela Torres	baño peq.; piso alto	21/3/01	12:00	Carlos Jara						<table border="1"> <tbody> <tr> <td>22/5/01</td> <td>9:30</td> <td>Antonio Ruiz</td> <td>interesado</td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>	22/5/01	9:30	Antonio Ruiz	interesado								
12/2/01	19:00	Marcela Torres	baño peq.; piso alto																						
21/3/01	12:00	Carlos Jara																							
22/5/01	9:30	Antonio Ruiz	interesado																						
Ref. 5678 Tipo: Local C/Ricardo Catalá 12, zona Avd. Valencia. 40 m2. Diáfano, con altillo de 10 m2, vado, agua y luz, chaflán. Precio venta: 6.000.000 Precio alquiler: 50.000 Propietario: Luis Herranz. Telf. 964 223344. Visitas:	Ref. 3456 Tipo: Casa Ctra. Alcora 43, zona El Pantano. 104 m2. 2 alturas, 3 hab., cocina, baño, salón, solarium, plaza de garaje. Precio venta: 8.000.000 Precio alquiler: Propietario: Gemma López. Telf. 96 3456789 Visitas:																								
<table border="1"> <tbody> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table>													<table border="1"> <tbody> <tr> <td>13/1/01</td> <td>17:00</td> <td>Felipe Ulloa</td> <td>no es lo que busca</td> </tr> <tr> <td>5/5/01</td> <td>17:30</td> <td>Carmen Fez</td> <td>le gusta</td> </tr> <tr> <td>23/5/01</td> <td>19:45</td> <td>Carmen Fez</td> <td>buscará financiación</td> </tr> </tbody> </table>	13/1/01	17:00	Felipe Ulloa	no es lo que busca	5/5/01	17:30	Carmen Fez	le gusta	23/5/01	19:45	Carmen Fez	buscará financiación
13/1/01	17:00	Felipe Ulloa	no es lo que busca																						
5/5/01	17:30	Carmen Fez	le gusta																						
23/5/01	19:45	Carmen Fez	buscará financiación																						

Fuente: autores.

La agencia posee varias oficinas. Ya que la ficha de cada inmueble se encuentra en la oficina a la que se ha dirigido el propietario para ponerlo en venta o alquiler, la forma de compartir esta información actualmente es consultándola telefónicamente entre oficinas.

A continuación, se describen los datos que se manejan en las oficinas de la empresa para llevar a cabo el trabajo diario.

- **Oficinas:**

La empresa tiene varias oficinas en todo el país. Cada oficina tiene un código de identificación que es único, tiene una dirección (calle, número y ciudad), un número de teléfono y un número de fax. Cada oficina tiene su propia plantilla.

- **Plantilla:**

Cada oficina tiene un director que se encarga de supervisar todas sus gestiones. La empresa sigue muy de cerca el trabajo de los directores y tiene registrada la fecha en que cada director empezó en el cargo en su oficina. Cada director tiene un pago anual por gastos de vehículo y una bonificación mensual que depende de los contratos de alquiler que haya realizado su oficina.

En cada oficina hay varios supervisores. Cada uno es responsable del trabajo diario de un grupo de entre cinco y diez empleados que realizan las gestiones de los alquileres. El trabajo administrativo de cada grupo lo lleva un administrativo.

Cada miembro de la plantilla tiene un código único que lo identifica en la empresa. De cada uno de ellos se quiere conocer el nombre, la dirección, el número de teléfono, la fecha de nacimiento, el número del DNI, su puesto en la empresa, el salario anual y la

fecha en que entró en la empresa. De los administrativos se desea conocer también la velocidad con que escriben a máquina (en pulsaciones por minuto).

Además, de cada empleado se debe guardar información sobre uno de sus parientes más próximos: nombre, relación con el empleado, dirección y número de teléfono.

- *Inmuebles para alquilar:*

Cada oficina de la empresa tiene una serie de inmuebles para alquilar. Estos inmuebles se identifican por un código que es único dentro de la empresa. Los datos que se guardan de cada inmueble son los siguientes: dirección completa (calle, número y ciudad), tipo de inmueble, número de habitaciones y precio del alquiler en euros (este precio es mensual), precio de venta, galería de imágenes. El precio del alquiler se revisa de forma anual.

Cada inmueble se asigna a un empleado que es el responsable de su gestión. Cada miembro de la plantilla puede tener asignados hasta veinte inmuebles para alquilar.

Si un propietario elimina su oferta de alquiler de la empresa, su información se mantiene durante al menos tres años.

- *Propietarios:*

Los propietarios de los inmuebles pueden ser particulares o empresas. A cada propietario se le asigna un código que es único en la empresa. De los particulares se guarda el nombre, la dirección y el número de teléfono. De las empresas se guarda el nombre comercial, tipo de empresa, la dirección, el número de teléfono y el nombre de la persona de contacto.

- *Inquilinos (clientes):*

Cuando un cliente contacta con la empresa por primera vez, se toman sus datos: nombre, dirección, número de teléfono, tipo de inmueble que prefiere e importe máximo que está dispuesto a pagar al mes por el alquiler. Ya que es un posible inquilino, se le asigna un código que es único en toda la empresa. De la entrevista inicial que se realiza con cada cliente se guarda la fecha, el empleado que la realizó y unos comentarios generales sobre el posible inquilino.

- *Visitas a los inmuebles:*

En la mayoría de los casos, los posibles inquilinos desean ver varios inmuebles antes de alquilar uno. De cada visita que se realiza se guarda la fecha y los comentarios realizados por el cliente respecto al inmueble.

- *Publicidad de los inmuebles:*

Cuando algún inmueble es difícil de alquilar, la empresa lo anuncia en la prensa local y nacional. De cada anuncio se guarda la fecha de publicación y el coste económico del anuncio. De los periódicos se guarda el nombre, la dirección, el número de teléfono, el número de fax y el nombre de la persona de contacto.

- *Contratos de alquiler:*

La empresa se encarga de redactar los términos de cada contrato de alquiler. Cada contrato tiene un número, un importe mensual, un método de pago, el importe del depósito, si se ha realizado el depósito, las fechas de inicio y finalización del contrato, la

duración del contrato en meses y el miembro de la plantilla que lo formalizó. La duración mínima de un contrato es de tres meses y la duración máxima es de un año. Cada cliente puede tener alquilados uno o varios inmuebles al mismo tiempo.

- *Venta:*

La empresa se encarga de redactar los términos de cada contrato de venta (o factura de venta). Cada contrato tiene al menos un número, un importe, un método de pago, el importe del depósito (en el caso de que el pago sea a crédito), plazo de pago, observaciones. Cada cliente puede comprar uno o varios inmuebles al mismo tiempo.

- *Pagos:*

La empresa se encarga de registrar los cobros (o pagos) realizados por la venta de inmuebles. Cada registro de pago (recibo) tiene al menos un número, un importe, fecha de pago.

- *Inspecciones:*

Como parte del servicio que presta la empresa, ésta se encarga de realizar inspecciones periódicas a los inmuebles para asegurarse de que se mantienen en buen estado. Cada inmueble se inspecciona al menos una vez cada seis meses. Se inspeccionan tanto los inmuebles alquilados, como los que están disponibles para alquilar. De cada inspección se anota la fecha y los comentarios sobre su estado que quiera incluir el empleado que la ha llevado a cabo.

- *Actividades de cada oficina:*

En cada oficina se llevan a cabo las siguientes actividades para garantizar que cada empleado tenga acceso a la información necesaria para desempeñar su tarea de modo efectivo y eficiente. Cada actividad está relacionada con una función específica de la empresa. Cada una de estas funciones corresponde a uno o varios puestos de los que ocupan los empleados, por lo que éstos se indican entre paréntesis.

1. Crear y mantener las fichas con los datos de los empleados y su familiar más próximo (director).
2. Realizar listados de los empleados de cada oficina (director).
3. Realizar listados del grupo de empleados de un supervisor (director y supervisor).
4. Realizar listados de los supervisores de cada oficina (director y supervisor).
5. Crear y mantener las fichas con los datos de los inmuebles para alquilar (y de sus propietarios) de cada oficina (supervisor).
6. Realizar listados de los inmuebles para alquilar en cada oficina (toda la plantilla).
7. Realizar listados de los inmuebles para alquilar asignados a un determinado miembro de la plantilla (supervisor).
8. Crear y mantener las fichas con los datos de los posibles inquilinos de cada oficina (supervisor).

9. Realizar listados de los posibles inquilinos registrados en cada oficina (toda la plantilla).
10. Buscar inmuebles para alquilar que satisfacen las necesidades de un posible inquilino (toda la plantilla).
11. Crear y mantener las fichas de las visitas realizadas por los posibles inquilinos (toda la plantilla).
12. Realizar listados con los comentarios hechos por los posibles inquilinos respecto a un inmueble concreto (toda la plantilla).
13. Crear y mantener las fichas con los datos de los anuncios insertados en los periódicos (toda la plantilla).
14. Realizar listados de todos los anuncios que se han hecho sobre un determinado inmueble (supervisor).
15. Realizar listados de todos los anuncios realizados en un determinado periódico (supervisor).
16. Crear y mantener las fichas que contienen los datos sobre cada contrato de alquiler (director y supervisor).
17. Realizar listados de los contratos de alquiler de un determinado inmueble (director y supervisor).
18. Crear y mantener las fichas con los datos de cada inspección realizada a los inmuebles en alquiler (toda la plantilla).
19. Realizar listados de todas las inspecciones realizadas a un determinado inmueble (supervisor).

4. MODELO ENTIDAD RELACIÓN

“El modelo de datos entidad-relación (E-R) está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados entidades, y de relaciones entre estos objetos” (Silberschatz, 2002).

Tabla 1. Componentes del modelo entidad relación.

Componentes del modelo entidad relación	
Concepto	Descripción
Entidad	Es cualquier objeto o evento, que interviene en el proceso y acerca del cual se recolectan datos.
Relación	Es un vínculo entre dos o más entidades.
Atributo	Son los datos que van a ser guardados en las entidades y relaciones.

Fuente: autores.

La estructura lógica general de una base de datos se puede expresar gráficamente mediante un diagrama entidad relación, que consta de los siguientes componentes:

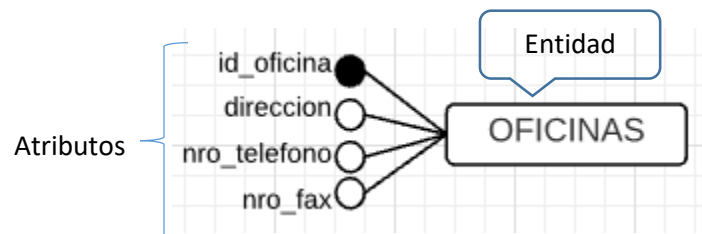
- Rectángulos: que representan conjuntos de entidades.
- Elipses: que representan atributos.
- Rombos: que representan relaciones entre conjuntos de entidades.
- Líneas: que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.



EJEMPLO

La empresa Tierra Prometida tiene varias oficinas en todo el país. Cada oficina tiene un código de identificación que es único, tiene una dirección un número de teléfono y un número de fax. Cada oficina tiene su propia plantilla.

Ilustración 2. Ejemplo.



Fuente: autores.

Cardinalidad: expresa cuántas del conjunto de entidades de un extremo de la relación están relacionadas con cuántas entidades del conjunto del otro extremo. Pudiendo ser de las siguientes maneras:

- Asociación uno a uno (1:1): “son aquellas en las cuales solo interviene un objeto de cada entidad” (Zea).
- Asociación uno a muchos (1:N): “son aquellas en las cuales interviene un objeto de una de las entidades asociadas a varios objetos de la otra entidad” (Zea).
- Asociación muchos a muchos (M:N): “son aquellas en las que intervienen varios objetos de una de las entidades asociadas a varios objetos de la otra entidad, es decir que describe la posibilidad de que las entidades puedan tener numerosas asociaciones en cualquier dirección” (Zea).



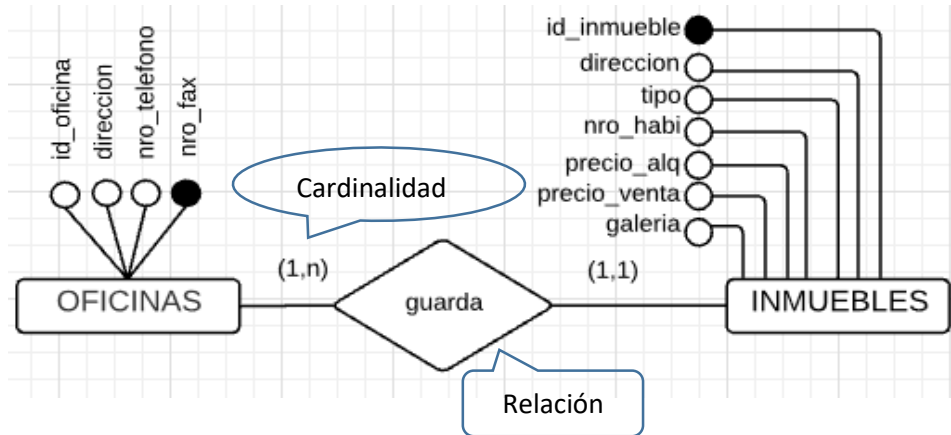
EJEMPLO

Cada oficina de la empresa tiene una serie de inmuebles para alquilar. Estos inmuebles se identifican por un código que es único dentro de la empresa. Los datos que se guardan de cada

inmueble son los siguientes: dirección, tipo de inmueble, número de habitaciones y precio del alquiler en euros (este precio es mensual), precio de venta, galería de imágenes. El precio del alquiler se revisa de forma anual.

Solución: Los sustantivos del enunciado, oficinas e inmuebles, son las entidades. La oficina guarda los datos de cada inmueble, con dicho verbo se evidencia la asociación entre las entidades.

Ilustración 3. Ejemplo.

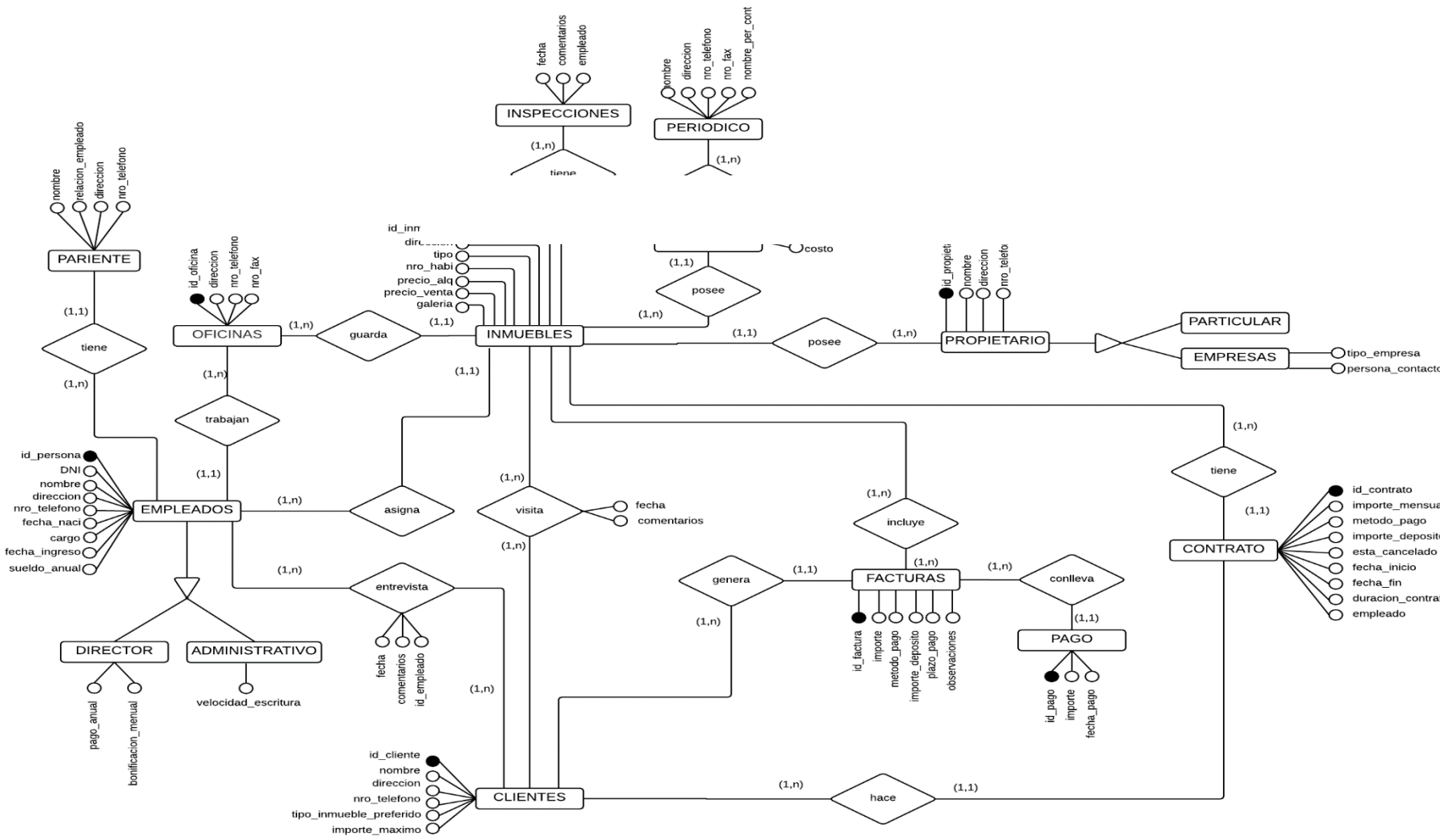


Fuente: autores.

La cardinalidad para este ejemplo es de 1:N, ya que cada oficina tiene una serie de inmuebles (varios), mientras que un inmueble sólo puede estar en una oficina.

MODELO ENTIDAD RELACIÓN

Ilustración 4: Ejemplo Modelo Entidad Relación.



5. MODELO RELACIONAL

En el modelo relacional se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. Cada tabla está compuesta por varias columnas, y cada columna tiene un nombre único (Silberschatz, 2002).

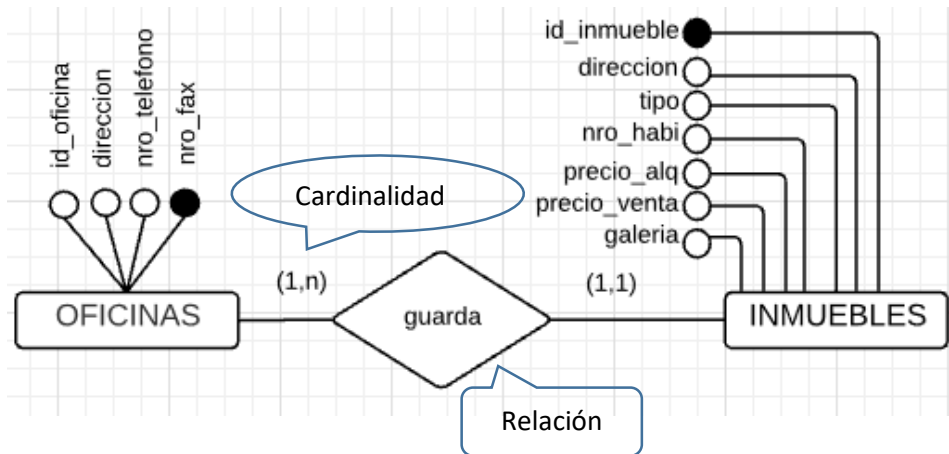
Transformación del modelo entidad relación al modelo relacional

Tabla 2. Transformación del modelo entidad relación al modelo relacional.

MODELO ENTIDAD/RELACIÓN	MODELO RELACIONAL
Entidad	Tabla
Atributo	Columna/Campo
Identificador Único	Clave Primaria
Relaciones N:M	Nueva tabla con clave primaria la concatenación de las claves de las entidades que la forman (la relación pasa a ser una tabla, y en esa tabla se pone como C.A. las entidades que une)
Relaciones 1:N	Propagando la de 1 en la de muchos (creando un campo en la de muchos que referencia a la de 1) si cada elemento de la entidad que participa con muchos aparece en la entidad de uno, es decir, si TODOS los elementos de la entidad de muchos tienen asociado uno de la entidad de uno.
Relaciones 1:1	Propagar la clave (igual que en la de 1:N) si cada elemento de la entidad que participa con muchos aparece en la entidad de uno, es decir, si TODOS los elementos de la entidad de muchos tienen asociado uno de la entidad de uno.

Fuente: autores.

Ilustración 5. Modelo Relacional.

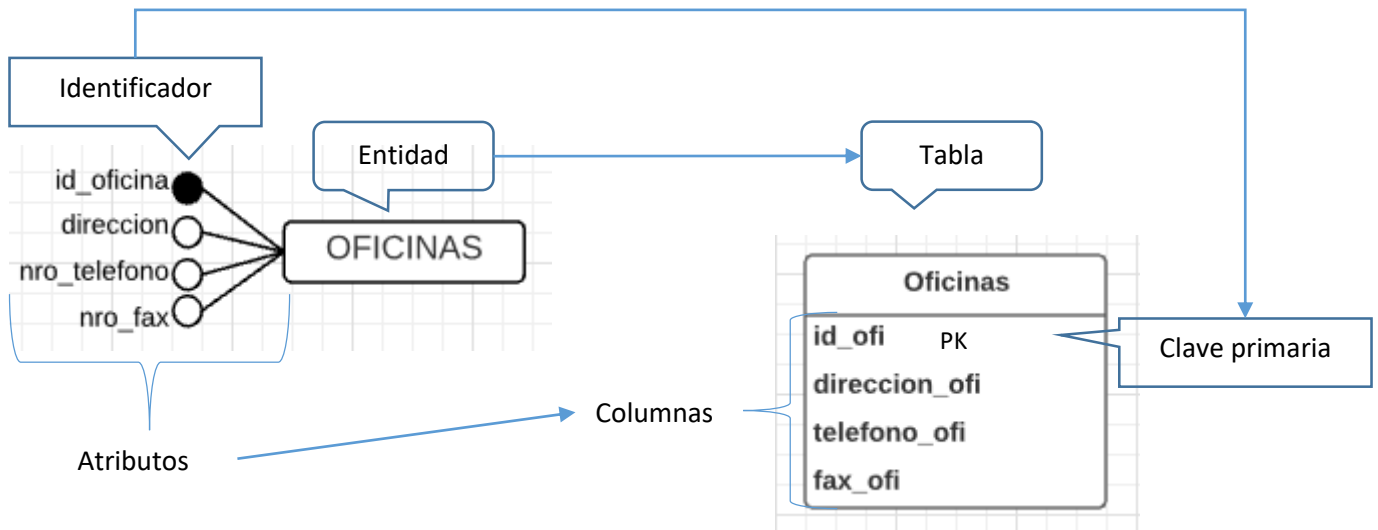


Fuente: autores.

Transformar la entidad Oficinas al Modelo relacional, cuyos atributos son id_oficina, dirección, nro_telefono, nro_fax.

Transformar el siguiente diagrama entidad relación al modelo relacional.

Ilustración 6. Transformación del diagrama entidad relación al modelo relacional.



Fuente: autores.


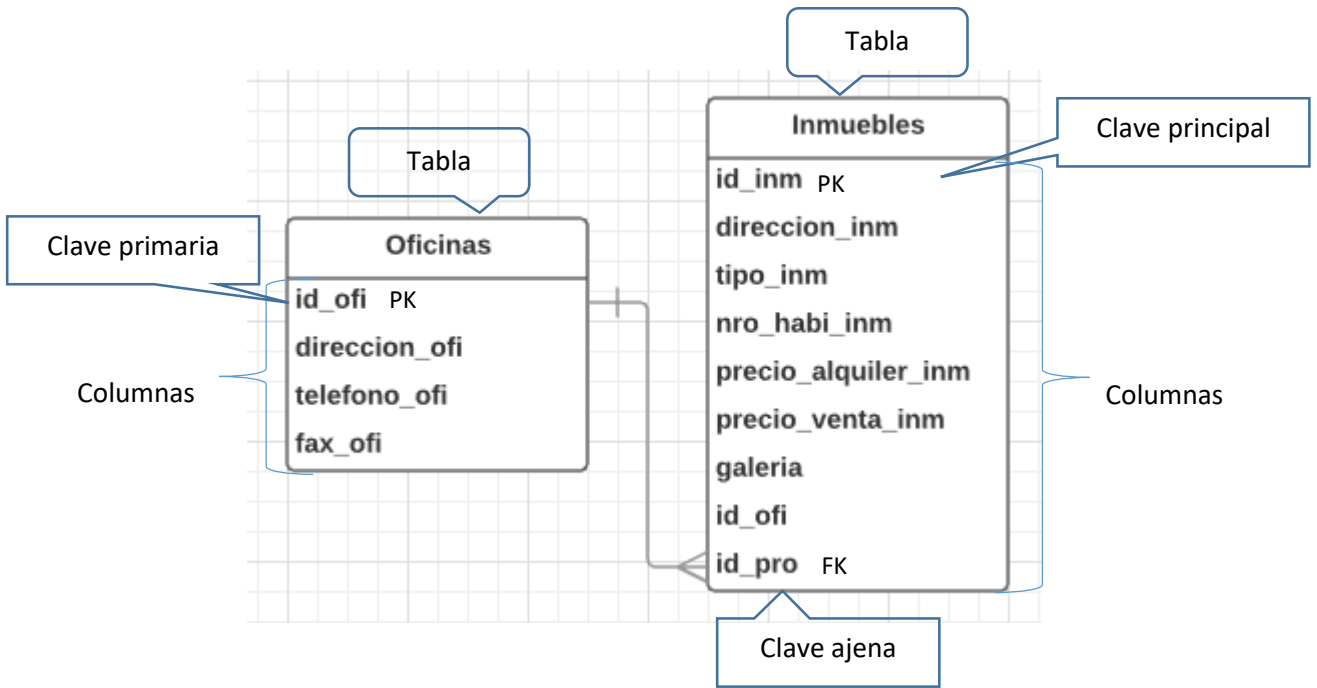
Como la cardinalidad es de uno a muchos, el atributo nro_fax de la tabla Oficinas se coloca como clave foránea (atributo) en la tabla Inmuebles, además se usa la siguiente línea  para indicar que una oficina tiene muchos inmuebles.

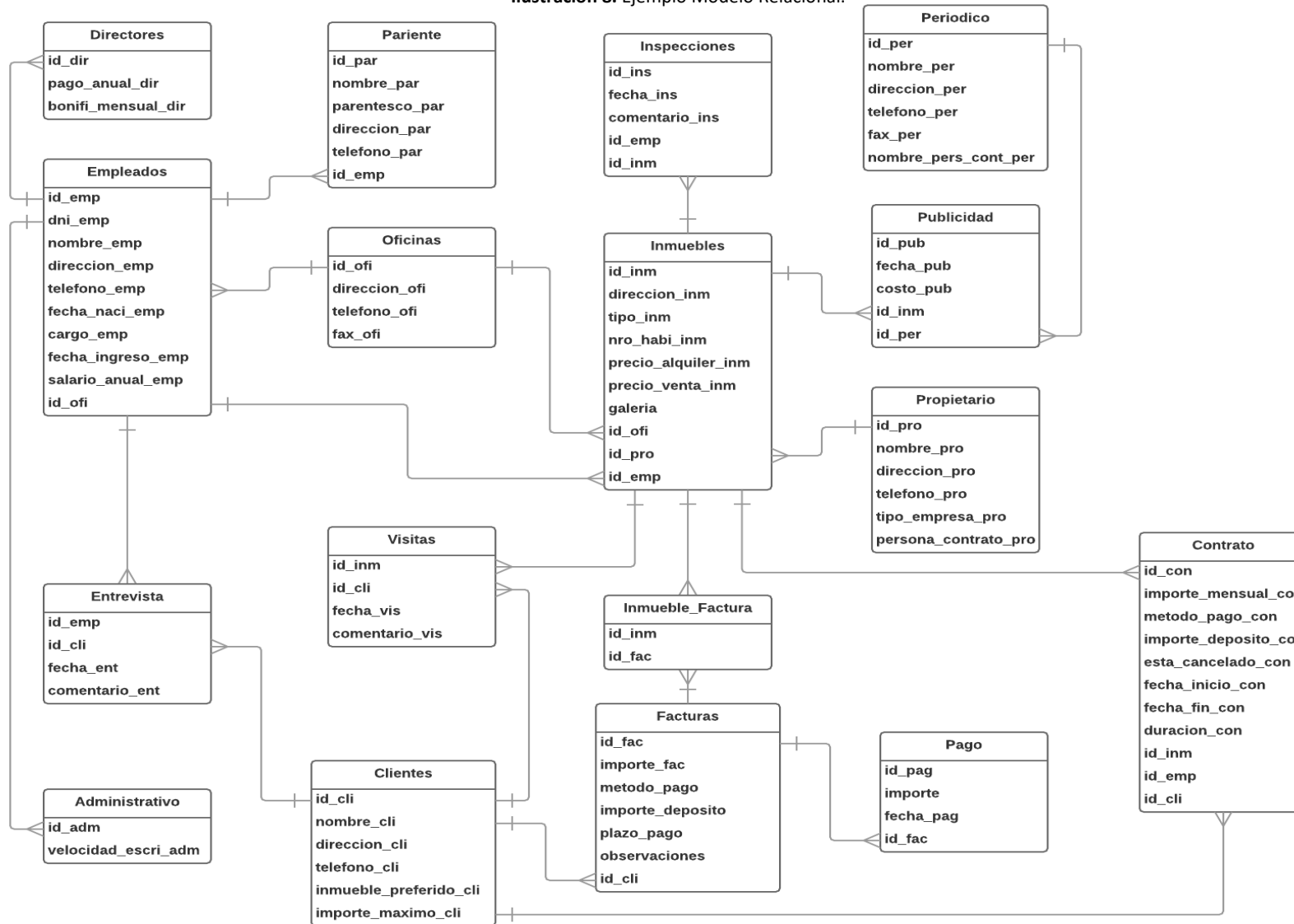
Ilustración 7. Transformación del diagrama entidad relación al modelo relacional.



Fuente: autores.

MODELO RELACIONAL

Ilustración 8. Ejemplo Modelo Relacional.

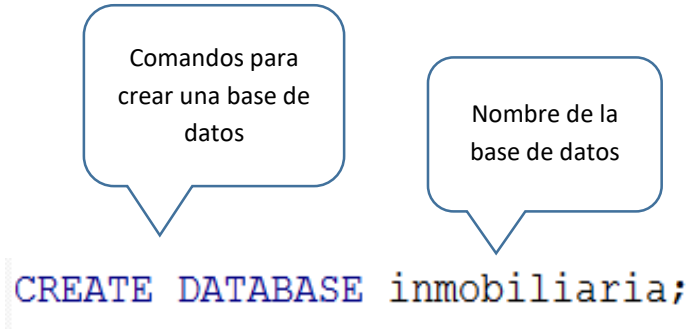


Fuente: autores.

6. CREACIÓN DE LA BASE DE DATOS

La sentencia para crear la base de datos inmobiliaria es: CREATE DATABASE inmobiliaria:

Ilustración 9. Creación de la Base de Datos.

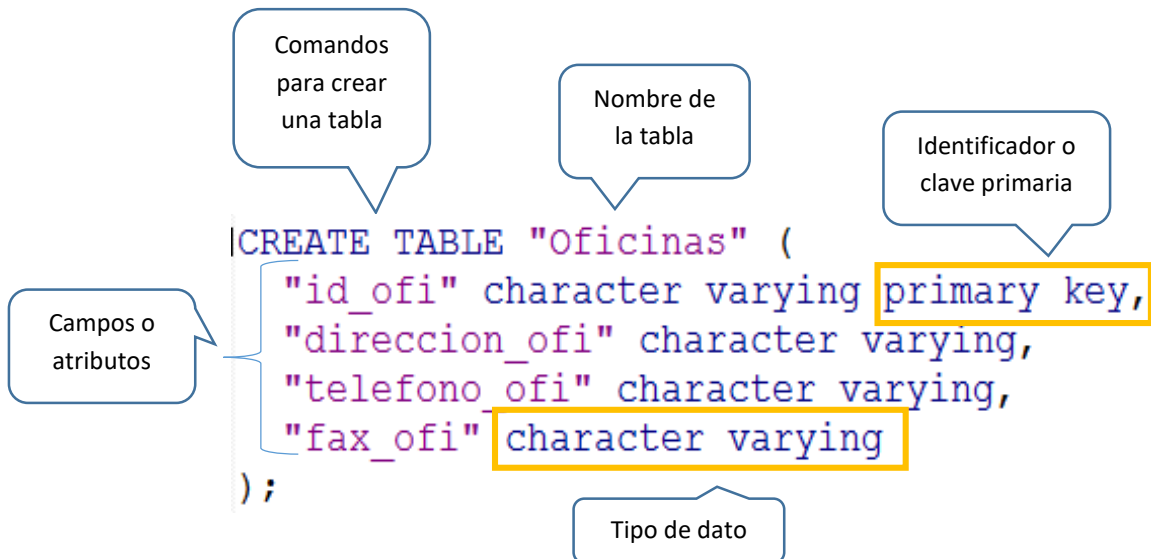


Fuente: autores.

7. CREACION DE TABLAS

Para crear una tabla se define el nombre, y los campos que contiene, a los cuales se les debe indicar el tipo de dato. Los comandos requeridos son:

Ilustración 10. Creación de Tablas.



Fuente: autores.

8. INTEGRIDAD DE ENTIDAD

La integridad de entidad “define una fila como entidad única para una tabla determinada. La integridad de entidad exige la integridad de las columnas de los identificadores o la clave principal de una tabla, mediante índices y restricciones UNIQUE, o restricciones PRIMARY KEY” (Zea).

9. CLAVE PRINCIPAL

La restricción de campo PRIMARY KEY “especifica que un campo de una tabla solamente puede contener valores únicos (no duplicados) y no nulos. La definición de la columna especificada no tiene que incluir una restricción explícita NOT NULL para ser incluida en una restricción PRIMARY KEY (Manual de usuario de postgresQL).

UNIQUE

La restricción UNIQUE “especifica una regla que obliga a un grupo de uno o más campos de una tabla a contener valores únicos” (Manual de usuario de postgresQL).

Tierra Prometida guarda algunos datos de sus empleados, entre estos datos el número de cédula, para controlar que este campo no se repita. Definir una restricción Unique.

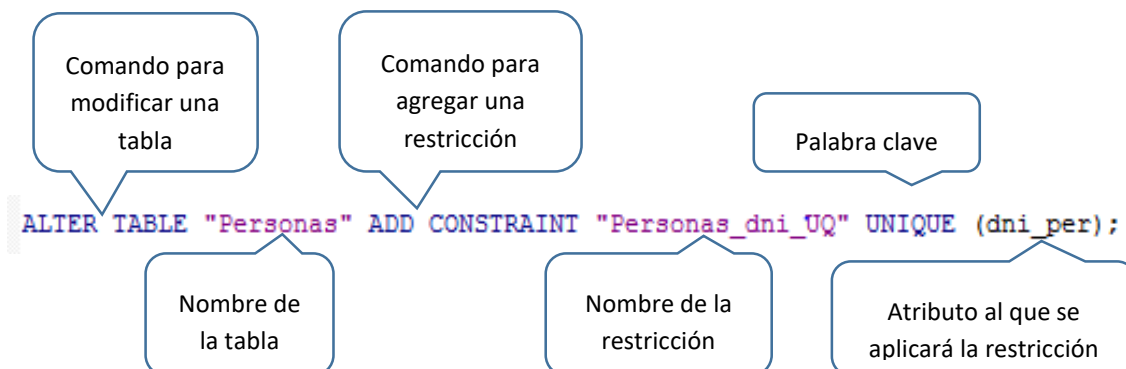
La sentencia que se ejecuta para conservar la integridad de los datos de acuerdo al problema propuesto es la siguiente:



EJEMPLO

```
ALTER TABLE "Personas" ADD CONSTRAINT "Personas_dni_UQ" UNIQUE (dni_per);
```

Ilustración 11. Ejemplo Restricción Unique.



Fuente: autores.

10. INTEGRIDAD REFERENCIAL

La integridad referencial “protege las relaciones definidas entre las tablas cuando se crean o se eliminan filas, se basa en las relaciones entre claves ajenas y claves principales, mediante restricciones FOREIGN KEY y CHECK” (Zea).

11. CLAVE FORÁNEA

Para crear una clave foránea o ajena se debe indicar a qué tabla se desea modificar, definir un nombre para la restricción, indicar al atributo que se aplicará la restricción, señalar la tabla de referencia con su clave principal.



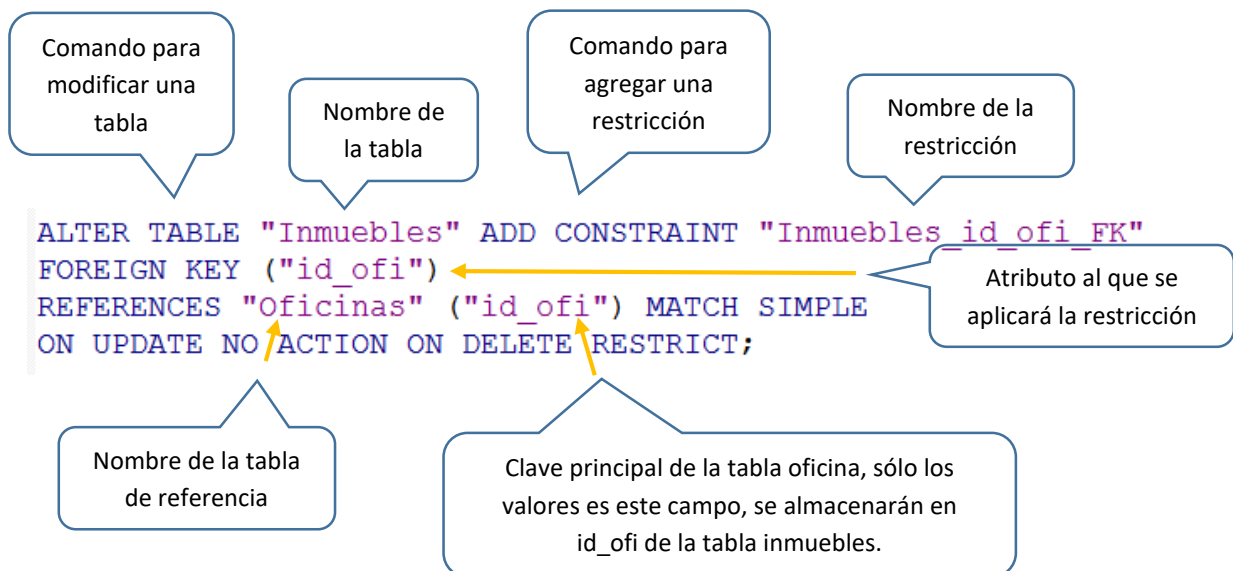
EJEMPLO

La tabla “Inmuebles” contiene el campo id_ofi, el cual sólo debe tomar los valores del campo id_ofi de la tabla “Oficinas”.

Los comandos para establecer una clave foránea son:

```
ALTER TABLE "Inmuebles" ADD CONSTRAINT "Inmuebles_id_ofi_FK" FOREIGN KEY ("id_ofi")  
REFERENCES "Oficinas" ("id_ofi") MATCH SIMPLE ON UPDATE NO ACTION ON DELETE RESTRICT;
```

Ilustración 12. Ejemplo Clave Foránea.



Fuente: autores.

12. CHECK

La restricción CHECK “especifica una restricción sobre los valores permitidos en un campo” (Manual de usuario de PostgreSQL).



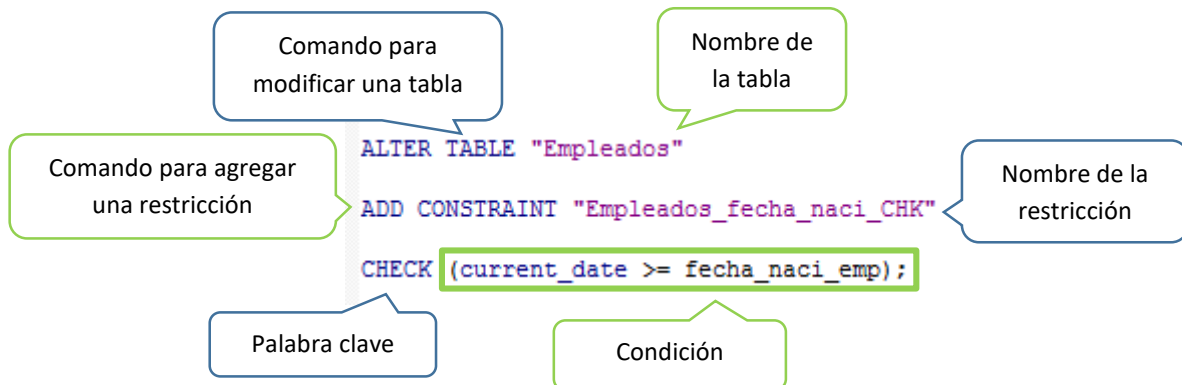
EJEMPLO

En la tabla “Personas” existe el campo fecha_naci_per, el cual no debe almacenar fechas superiores a la fecha actual.

La sentencia requerida es:

```
ALTER TABLE "Personas" ADD CONSTRAINT "Personas_fecha_naci_CHK" CHECK (current_date  
>= fecha_naci_per);
```

Ilustración 13. Ejemplo Check.



Fuente: autores.

Es importante mencionar que el comando `current_date` obtiene la fecha actual del sistema. En este ejemplo, si la condición es verdadera, es decir, la fecha actual es mayor o igual a la fecha de nacimiento, se realizará el ingreso de los valores a los diversos campos. Para de esta manera asegurar que los datos sean consistentes.

13. CONSULTAS CON LA CLÁUSULA ORDER BY

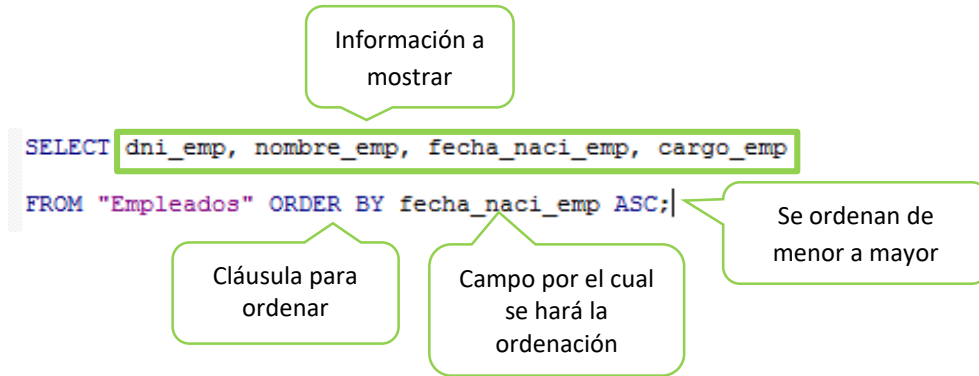
Se pueden efectuar consultas especificando “el orden en el que se desean recuperar los registros de las tablas mediante la cláusula ORDER BY” (SQL INNER JOIN). Donde la lista de campo serán los datos a ordenar.



EJEMPLO

Se necesita listar la información de los empleados de acuerdo a la fecha de nacimiento, de menor a mayor.

Ilustración 14. Ejemplo Cláusula Order By.



Fuente: autores.

De acuerdo a los registros que existan en la tabla empleados se podrá observar que los datos se ordenan por la fecha de nacimiento de menor a mayor.

Ilustración 15. Registros Tabla Empleados.

	dni_emp character varying	nombre_emp character varying	fecha_naci_emp date	cargo_emp character varying
1	0701011121	Luis	1970-01-17	DIRECTOR
2	0701011122	Marco	1970-01-19	DIRECTOR
3	0701011131	Felipe	1971-01-17	EMPLEADO
4	0701011142	Mateo	1972-01-19	EMPLEADO
5	0701011153	Rodrigo	1978-04-29	EMPLEADO
6	0701011123	Antonio	1980-04-19	DIRECTOR

Fuente: autores.

CONSULTAS CON PREDICADO

El predicado por lo general “se lo incluye entre la cláusula SELECT y el primer nombre del campo a recuperar” (SQL Básico). Los posibles predicados son:

Tabla 3. Consultas con Predicado.

Predicado	Descripción	Estructura de la consulta
ALL	Devuelve todos los campos de la tabla.	SELECT ALL FROM nombre de la tabla.
LIMIT	Devuelve un determinado número de registros de la tabla.	SELECT lista de atributos FROM nombre de la tabla LIMIT número de registros.
DISTINCT	Omite los registros cuyos campos seleccionados coincidan totalmente.	SELECT DISTINCT lista de atributos FROM nombre de la tabla.
OFFSET	Indica el número de filas que debe saltarse antes de comenzar a devolver las filas.	SELECT lista de atributos FROM nombre de la tabla OFFSET número de registros a saltar.

Fuente: autores.

Por lo general, se efectúan consultas sin especificar ningún predicado, sin embargo, el sistema gestor de base de datos asume el predicado ALL.

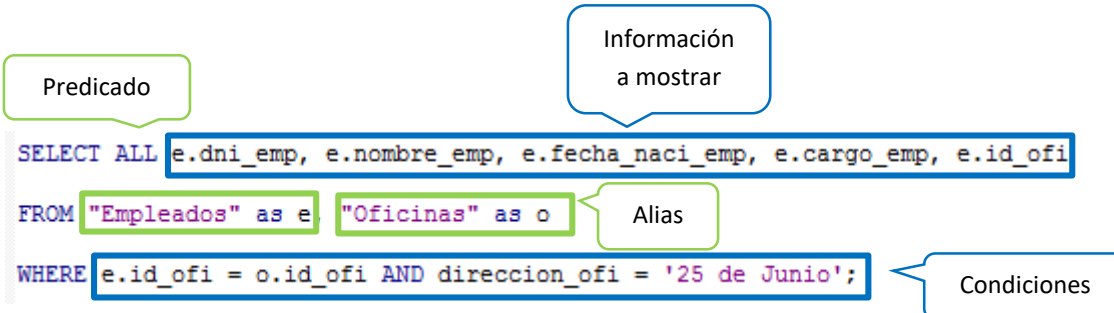
Al usar el predicado LIMIT es importante utilizar la cláusula ORDER BY para que el resultado no sea aleatorio, sino que se muestren los datos de acuerdo a un orden.



EJEMPLO

Mostrar todos los empleados que pertenecen a la oficina ubicada en la dirección 25 de junio.

Ilustración 16. Ejemplo Consultas con Predicado.



```

SELECT ALL e.dni_emp, e.nombre_emp, e.fecha_naci_emp, e.cargo_emp, e.id_ofi
FROM "Empleados" as e "Oficinas" as o
WHERE e.id_ofi = o.id_ofi AND direccion_ofi = '25 de Junio';
    
```

The diagram shows a SQL query with four callouts: 'Predicado' points to the WHERE clause; 'Información a mostrar' points to the SELECT clause; 'Alias' points to the table names 'Empleados' and 'Oficinas' in the FROM clause; and 'Condiciones' points to the WHERE clause.

Fuente: autores.

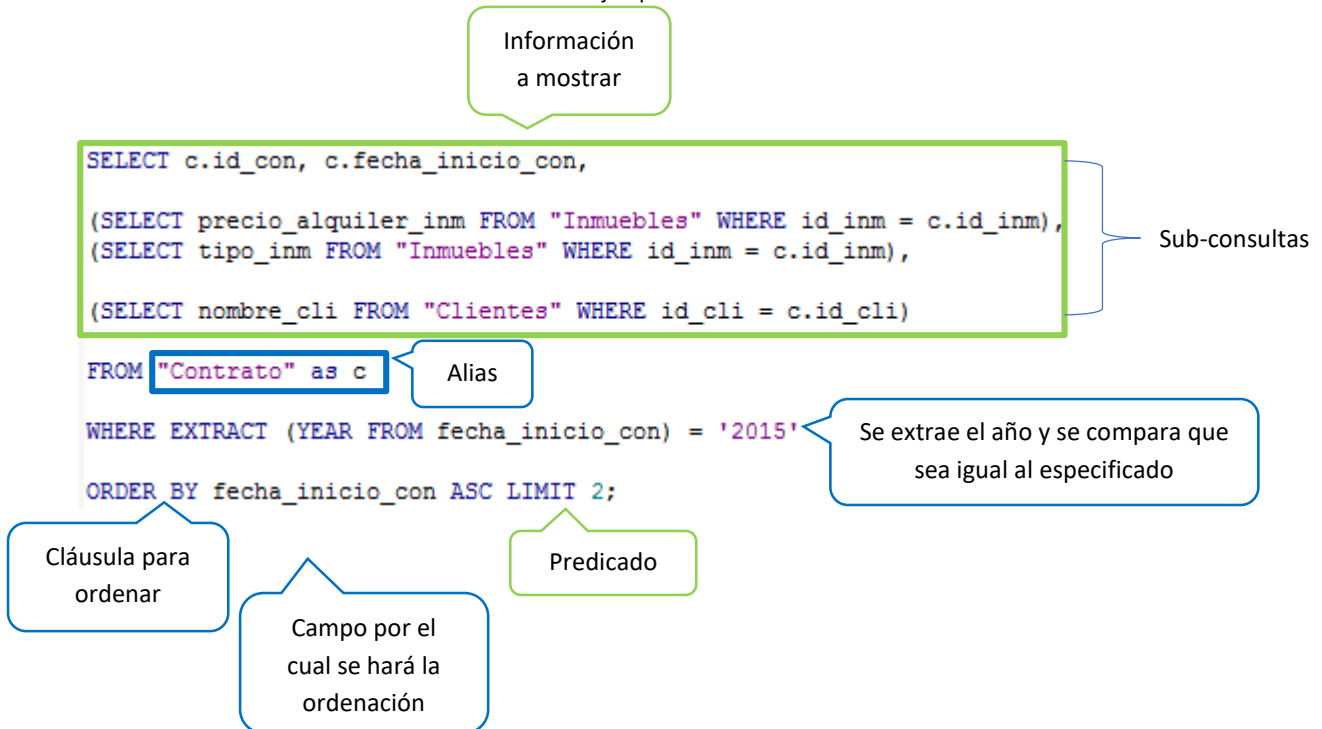
En la consulta se han utilizado alias para evitar repetir el nombre de la tabla en cada uno de los atributos y sobre todo para evitar ambigüedades, ya que algunos campos se llaman de igual manera en varias tablas (en las tablas oficinas y empleados existe el atributo id_ofi, si no se especifica a que tabla se hace referencia el sistema gestor de base de datos no sabría sobre que tabla mostrar la información).



EJEMPLO

Listar los dos primeros contratos efectuados en el año 2015, mostrando el precio de alquiler, el tipo de inmueble y el cliente que hecho el contrato.

Ilustración 17. Ejemplo Consultas con Predicado.



Fuente: autores.

15. CONSULTAS COMPLEJAS

“El álgebra relacional es un tipo de álgebra con una serie de operadores que trabajan sobre una o varias relaciones para obtener una relación resultado” (PostgreSQL).

Las operaciones más importantes disponibles en álgebra relacional son:

A continuación, se detalla el empleo de dichos operadores:

15.1 CONSULTA DE SELECCIÓN

Permiten visualizar determinados registros bajo condiciones que se establecen en la sentencia, extrayendo los resultados necesarios a consultar.



EJEMPLO

Mostrar los propietarios que no tienen datos vacíos en la información de la empresa y en la persona encargada del contrato.

Como se puede observar estos son los registros pertenecientes a la tabla de Propietario

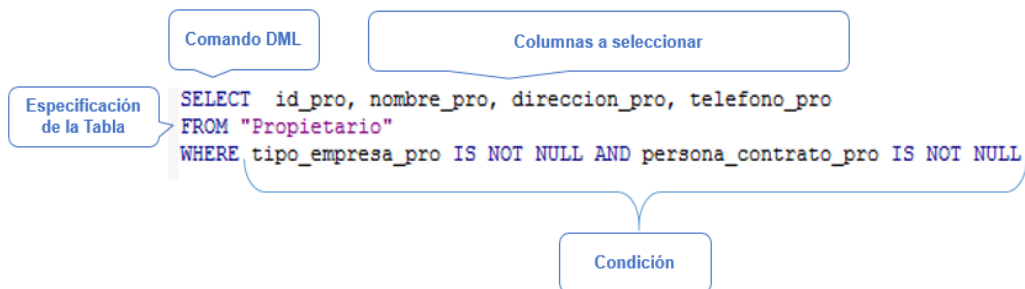
Ilustración 18. Ejemplo Consulta de Selección.

	id_pro [PK] character varying	nombre_pro character varying	direccion_pro character varying	telefono_pro character varying	tipo_empresa_pro character varying	persona_contrato_pro character varying
1	PRO-001	Jose Suarez	Machala	2976001		
2	PRO-002	Empresa ABC	Pasaje	2976002	Comercial	Carlos Fernandez
3	PRO-003	Diego Toro	Machala	2976003		
*						

Fuente: autores.

La consulta a estructurar es la siguiente:

Ilustración 19. Consulta a Estructurar.



Fuente: autores.

Al ejecutar dicha sentencia, se podrá visualizar únicamente los datos que cumplen con la condición establecida que corresponde a los propietarios tipo empresa:

Ilustración 20. Secuencia ejecutada.

	id_pro character varying	nombre_pro character varying	direccion_pro character varying	telefono_pro character varying
1	PRO-002	Empresa ABC	Pasaje	2976002

Fuente: autores.

15.2 PROYECCIÓN

“Esta operación aplicada a una relación que produce una nueva con solamente los atributos (columnas) especificados” (PostgreSQL).

Para ver cómo funciona este concepto se realiza el siguiente ejercicio práctico. Primero visualizaremos la tabla de Clientes, obteniendo los siguientes resultados:




EJEMPLO

Ilustración 21. Ejemplo Proyección.

	id_cli [PK] character varying	nombre_cli character varying	direccion_cli character varying	telefono_cli character varying	inmueble_preferido_cli character varying	importe_maximo_cli double precision
1	CLI001	Ufredo Romero	Santa Rosa	2890910	INM001	300
2	CLI002	Diego Romero	Santa Rosa	2890911	INM002	150
3	CLI003	Jazmin Quirola	Machala	2890912	INM002	190
*						

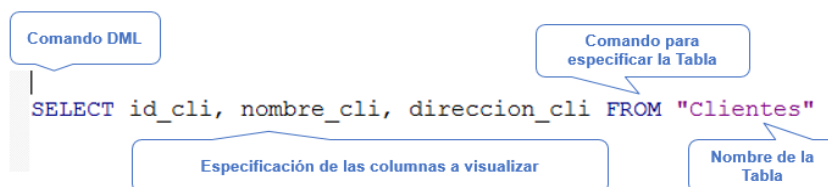
Fuente: autores.

Ahora, en la opción de  en la barra de menú de PostgreSQL se da clic para escribir la sentencia la cual especificaremos que campos deseamos ver de la tabla mencionada la cual cuenta con 6 columnas.

Mediante la siguiente sentencia se especifica que columnas se desea ver, en este caso de la tabla Clientes se desea visualizar únicamente los valores que corresponde al código, nombre y su dirección.

Para lo cual corresponde la siguiente sentencia que se encuentra estructurado de la siguiente forma:

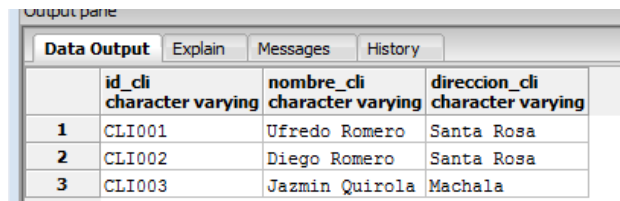
Ilustración 22. Sentencia Proyección.



Fuente: autores.

Lo cual se tiene el siguiente resultado:

Ilustración 23. Resultado Proyección.



	id_cli character varying	nombre_cli character varying	direccion_cli character varying
1	CLI001	Ufredo Romero	Santa Rosa
2	CLI002	Diego Romero	Santa Rosa
3	CLI003	Jazmin Quirola	Machala

Fuente: autores.

15.3 UNIÓN

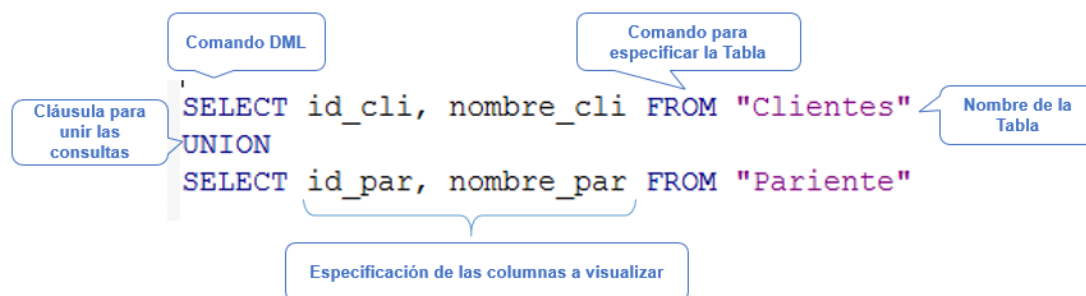
“La unión de R y S es el conjunto de elementos que existen en R, ó en S, ó en las dos. Un elemento que existe tanto en R como en S aparece solamente una vez en la unión” (PostgreSQL).

Mediante esta operación se puede realizar consultas de diversas tablas, presentando un solo conjunto de resultados que existen en las tablas establecidas. Hay que recordar que cada consulta con la cláusula UNION debe tener el mismo número de columnas y ser del mismo tipo de dato de las columnas establecidas secuencialmente.

EJEMPLO

Para visualizar el código y los nombres de la tabla Clientes y Pariente se realiza la siguiente sentencia, en donde especificamos las columnas a seleccionar en la consulta y la tabla a la que pertenece en donde se detalla lo siguiente:

Ilustración 24. Ejemplo Unión.



Fuente: autores.

Obteniendo los siguientes resultados, expresando la consulta de dos tablas en una sola.

Ilustración 25. Resultado Unión.

	id_cli character varying	nombre_cli character varying
1	CLI004	Alfredo Diaz
2	CLI005	Sandra Lavalle
3	PAR001	Jose
4	CLI001	Ufredo Romero
5	CLI002	Diego Romero
6	CLI006	Anguie Castro
7	CLI007	Anguie Cordova
8	PAR003	Maritza
9	CLI003	Jazmin Quirola
10	PAR002	Ramon

Fuente: autores.

15.4 INTERSECCIÓN

Esta operación une distintas consultas con la misma cantidad de parámetros en la Selección y retorna solo los registros duplicados. “La intersección de R y S es el conjunto de elementos que existen en R y en S.” (PostgreSQL) La cláusula para esta operación en SQL es INTERSECT.

Para visualizar el código de los directores de la compañía que se encuentran registrados en la tabla Persona, emplearemos la intersección.



EJEMPLO

Para conocer cómo funciona este operador hacemos la prueba de la siguiente sentencia con operador UNION ALL:

```
SELECT id_cli, nombre_cli FROM "Clientes"  
UNION ALL  
SELECT id_par, nombre_par FROM "Pariente"
```

El cual presenta el siguiente resultado:

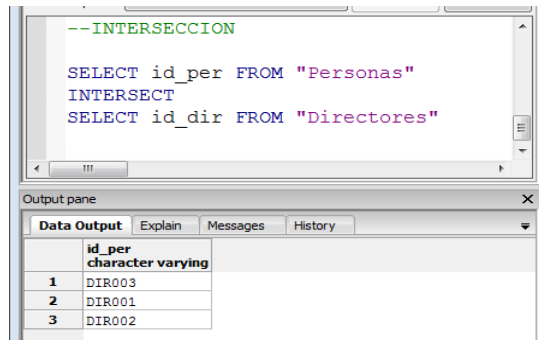
Ilustración 26. Resultado Intersección.

	id_per character varying
1	DIR001
2	DIR002
3	DIR003
4	EMP001
5	EMP002
6	EMP003
7	DIR001
8	DIR002
9	DIR003

Fuente: autores.

Por lo cual para obtener únicamente los códigos de los directores reemplazamos la sentencia de unión por la sentencia de inserción, correspondiente a la cláusula INTERSECT que permitirá que la sentencia muestre únicamente los valores repetidos.

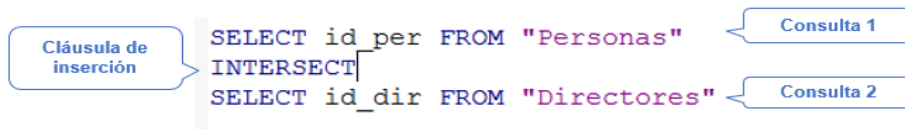
Ilustración 27. Cláusula Intersect.



Fuente: autores.

Teniendo su estructura de la siguiente manera:

Ilustración 28. Estructura Intersect.



Fuente: autores.

15.5 DIFERENCIA

“La diferencia de R y S es el conjunto de elementos que existen en R pero no en S. R-S es diferente a S-R, S-R sería el conjunto de elementos que existen en S pero no en R.” (PostgreSQL)

Mediante este operador se visualizan los datos de una consulta entre tablas en donde el resultado a presentar es de la primera consulta menos los valores de las siguientes.

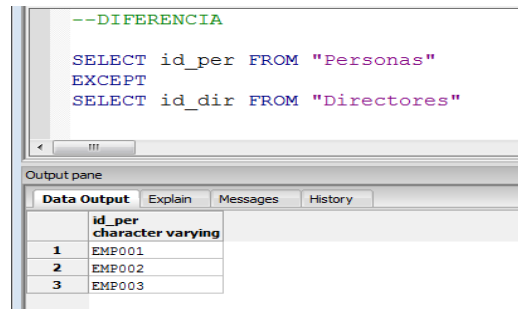
La cláusula en SQL es EXCEPT.



EJEMPLO

Para indicar el funcionamiento de esta cláusula, reemplazaremos a la consulta anterior la cláusula INTERSECT por EXCEPT, el cual nos mostrara el siguiente resultado:

Ilustración 29. Ejemplo Diferencia.



Fuente: autores.

Dado a que con esta expresión los resultados emitidos por la sentencia son los valores que resultan al restarles los resultados de la segunda consulta.

15.6 COMBINACIÓN

Esta acción se realiza por medio de la cláusula JOIN la cual “combinar dos o más relaciones según una condición para obtener tuplas compuestas por atributos de las dos relaciones combinadas” (PostgreSQL).

15.7 PRODUCTO CARTESIANO

Este término hace referencia a las combinaciones cruzadas en la cual “se emplea en lenguaje SQL el término de CROSS JOIN ó separando las relaciones usadas en el producto con comas, en el FROM de la sentencia SQL. para realizar dicha acción” (PostgreSQL).



Para visualizar el funcionamiento de esta cláusula visualizaremos el contenido de dos tablas e emplear.

```
SELECT * FROM "Personas"
```

Ilustración 30. Ejemplo Producto Cartesiano.

	id_per character varying	dni_per character varying	nombre_per character varying	direccion_per character varying	telefono_per character varying	fecha_naci_per date	cargo_per character varying	id_ofi character varying
1	DIR001	0701011121	Luis	10 de agosto	2514678	1970-01-17	DIRECTOR	OFI001
2	DIR002	0701011122	Marco	12 de noviembre	2514679	1970-01-19	DIRECTOR	OFI002
3	DIR003	0701011123	Antonio	Boyaca	2514680	1980-04-19	DIRECTOR	OFI003
4	EMP001	0701011131	Felipe	El Sol	2514699	1971-01-17	EMPLEADO	OFI001
5	EMP002	0701011142	Mateo	Ciudad El Sol	2514700	1972-01-19	EMPLEADO	OFI002
6	EMP003	0701011153	Rodrigo	Barcelona	2514701	1978-04-29	EMPLEADO	OFI003

Fuente: autores.

```
SELECT * FROM "Directores"
```

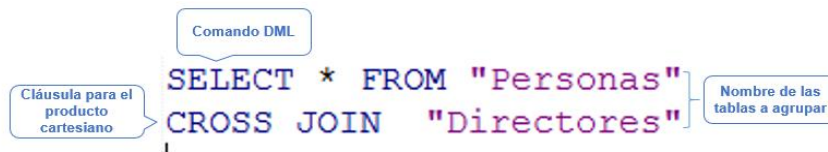
Ilustración 31. Ejemplo Producto Cartesiano.

Output pane				
Data Output				
	id_dir	salario_anual_dir	fecha_ingreso_dir	bonifi_mensual_dir
	character varying	double precision	date	double precision
1	DIR001	6000	2000-08-20	500
2	DIR002	6000	2001-03-21	500
3	DIR003	7200	2010-02-10	600

Fuente: autores.

Con la siguiente sentencia se multiplican los valores de las tablas en donde se visualizar el siguiente resultado:

Ilustración 32. Sentencia Producto Cartesiano.



Fuente: autores.

Ilustración 33. Ejemplo Producto Cartesiano.

Output pane												
Data Output												
	id_per	dni_per	nombre_per	direccion_per	telefono_per	fecha_naci_per	cargo_per	id_ofi	id_dir	salario_anual_dir	fecha_ingreso_dir	bonifi_mensual_dir
	character varying	character varying	character varying	character varying	character varying	date	character varying	character varying	character varying	double precision	date	double precision
1	DIR001	0701011121	Luis	10 de agosto	2514678	1970-01-17	DIRECTOR	OFI001	DIR001	6000	2000-08-20	500
2	DIR001	0701011121	Luis	10 de agosto	2514678	1970-01-17	DIRECTOR	OFI001	DIR002	6000	2001-03-21	500
3	DIR001	0701011121	Luis	10 de agosto	2514678	1970-01-17	DIRECTOR	OFI001	DIR003	7200	2010-02-10	600
4	DIR002	0701011122	Marco	12 de noviembre	2514679	1970-01-19	DIRECTOR	OFI002	DIR001	6000	2000-08-20	500
5	DIR002	0701011122	Marco	12 de noviembre	2514679	1970-01-19	DIRECTOR	OFI002	DIR002	6000	2001-03-21	500
6	DIR002	0701011122	Marco	12 de noviembre	2514679	1970-01-19	DIRECTOR	OFI002	DIR003	7200	2010-02-10	600
7	DIR003	0701011123	Antonio	Boyaca	2514680	1980-04-19	DIRECTOR	OFI003	DIR001	6000	2000-08-20	500
8	DIR003	0701011123	Antonio	Boyaca	2514680	1980-04-19	DIRECTOR	OFI003	DIR002	6000	2001-03-21	500
9	DIR003	0701011123	Antonio	Boyaca	2514680	1980-04-19	DIRECTOR	OFI003	DIR003	7200	2010-02-10	600
10	EMP001	0701011131	Felipe	El Sol	2514699	1971-01-17	EMPLEADO	OFI001	DIR001	6000	2000-08-20	500
11	EMP001	0701011131	Felipe	El Sol	2514699	1971-01-17	EMPLEADO	OFI001	DIR002	6000	2001-03-21	500
12	EMP001	0701011131	Felipe	El Sol	2514699	1971-01-17	EMPLEADO	OFI001	DIR003	7200	2010-02-10	600
13	EMP002	0701011142	Mateo	Ciudad El Sol	2514700	1972-01-19	EMPLEADO	OFI002	DIR001	6000	2000-08-20	500
14	EMP002	0701011142	Mateo	Ciudad El Sol	2514700	1972-01-19	EMPLEADO	OFI002	DIR002	6000	2001-03-21	500
15	EMP002	0701011142	Mateo	Ciudad El Sol	2514700	1972-01-19	EMPLEADO	OFI002	DIR003	7200	2010-02-10	600
16	EMP003	0701011153	Rodrigo	Barcelona	2514701	1978-04-29	EMPLEADO	OFI003	DIR001	6000	2000-08-20	500
17	EMP003	0701011153	Rodrigo	Barcelona	2514701	1978-04-29	EMPLEADO	OFI003	DIR002	6000	2001-03-21	500
18	EMP003	0701011153	Rodrigo	Barcelona	2514701	1978-04-29	EMPLEADO	OFI003	DIR003	7200	2010-02-10	600

Fuente: autores.

COMBINACIONES INTERNAS

“La sentencia INNER JOIN es la sentencia JOIN por defecto, y consiste en combinar cada fila de una tabla con cada fila de la otra tabla, seleccionando aquellas filas que cumplan una determinada condición” (SQL INNER JOIN).

La cláusula INNER JOIN interviene entre dos relaciones, para lo cual el resultado que se obtiene después de aplicar al producto cartesiano.



EJEMPLO

Como apreciamos en el ejercicio anterior la tabla de Personas consta de 8 columnas y la de Directores consta de 4 columnas, las cuales al agruparlas con la sentencia INNER JOIN se obtendrá 12 columnas con los valores correspondientes a la relación fijada, mediante la siguiente sentencia:

Ilustración 34. Sentencia Combinaciones Internas.

Comando DML

Cláusula para combinación interna

```
SELECT * FROM "Personas"
INNER JOIN "Directores" ON id_per=id_dir
```

Columnas de relación de las tablas

Fuente: autores.

Lo cual nos da los siguientes resultados:

Ilustración 35. Resultados Combinaciones Internas.

	id_per	dni_per	nombre_per	direccion_per	telefono_per	fecha_naci_per	cargo_per	id_ofi	id_dir	salario_anual_dir	fecha_ingreso_dir	bonifi_mensual_dir
	character varying	character varying	character varying	character varying	character varying	date	character varying	character varying	character varying	double precision	date	double precision
1	DIR001	0701011121	Luis	10 de agosto	2514678	1970-01-17	DIRECTOR	OFT001	DIR001	6000	2000-08-20	500
2	DIR002	0701011122	Marco	12 de noviembre	2514679	1970-01-19	DIRECTOR	OFT002	DIR002	6000	2001-03-21	500
3	DIR003	0701011123	Antonio	Boyaca	2514680	1980-04-19	DIRECTOR	OFT003	DIR003	7200	2010-02-10	600

Fuente: autores.

15.8 COMBINACIONES EXTERNAS

- LEFT OUTER JOIN

Retorna todas las tuplas de la combinación que tengan un atributo común, más todas las tuplas de la relación de la izquierda que no tengan un equivalente en la relación de la derecha. “El resultado es NULL en el lado derecho cuando no hay coincidencia. En algunas bases de datos LEFT JOIN se denomina LEFT OUTER JOIN” (w3schools.com).



EJEMPLO

Continuando con el mismo ejercicio, empleamos ahora la cláusula LEFT OUTER JOIN, teniendo la consulta de la siguiente forma:

Ilustración 36. Cláusula Left Outer Join.

Comando DML

Cláusula para combinación externa

```
SELECT * FROM "Personas"
LEFT OUTER JOIN "Directores" ON id_per=id_dir
```

Columnas de relación de las tablas

Fuente: autores.

Con lo cual obtendremos los siguientes resultados:

Ilustración 37. Resultados Cláusula Left Outer Join.

	id_per	dni_per	nombre_per	dirección_per	telefono_per	fecha_naci_per	cargo_per	id_ofi	id_dir	salario_anual_dir	fecha_ingreso_dir	bonifi_mensual_dir
	character varying	character varying	character varying	character varying	character varying	date	character varying	character varying	character varying	double precision	date	double precision
1	DIR001	0701011121	Luis	10 de agosto	2514678	1970-01-17	DIRECTOR	OFI001	DIR001	6000	2000-08-20	500
2	DIR002	0701011122	Marco	12 de noviembre	2514679	1970-01-19	DIRECTOR	OFI002	DIR002	6000	2001-03-21	500
3	DIR003	0701011123	Antonio	Boyaca	2514680	1980-04-19	DIRECTOR	OFI003	DIR003	7200	2010-02-10	600
4	EMP001	0701011131	Felipe	El Sol	2514699	1971-01-17	EMPLEADO	OFI001				
5	EMP002	0701011142	Mateo	Ciudad El Sol	2514700	1972-01-19	EMPLEADO	OFI002				
6	EMP003	0701011153	Rodrigo	Barcelona	2514701	1978-04-29	EMPLEADO	OFI003				

Fuente: autores.

- RIGHT OUTER JOIN

Con esta cláusula se retorna todas las tuplas de la combinación que tengan un atributo común, más todas las tuplas de la relación de la derecha que no tengan un equivalente en la relación de la izquierda.



Para ver el funcionamiento de esta cláusula aplicamos la siguiente sentencia a partir de la anterior:

Ilustración 38. Ejemplo Right Outer Join.

Comando DML

Cláusula para combinación externa

```
SELECT * FROM "Personas"
RIGHT OUTER JOIN "Directores" ON id_per=id_dir
```

Columnas de relación de las tablas

Fuente: autores.

Y con ello los resultados a visualizar son los siguientes:

Ilustración 39. Resultados Right Outer Join.

	id_per character varying	dni_per character varying	nombre_per character varying	direccion_per character varying	telefono_per character varying	fecha_naci_per date	cargo_per character varying	id_ofi character varying	id_dir character varying	salario_anual_dir double precision	fecha_ingreso_dir date	bonifi_mensual_dir double precision
1	DIR001	0701011121	Luis	10 de agosto	2514678	1970-01-17	DIRECTOR	OFI001	DIR001	6000	2000-08-20	500
2	DIR002	0701011122	Marco	12 de noviembre	2514679	1970-01-19	DIRECTOR	OFI002	DIR002	6000	2001-03-21	500
3	DIR003	0701011123	Antonio	Boyaca	2514680	1980-04-19	DIRECTOR	OFI003	DIR003	7200	2010-02-10	600

Fuente: autores.

- **FULL OUTER JOIN**

“El operador de combinación externa completa, FULL OUTER JOIN, que incluye todas las filas de ambas tablas, con independencia de que la otra tabla tenga o no un valor coincidente.” (Micrisoft) Mediante esta cláusula se retorna todas las tuplas de la combinación que tengan un atributo común y no común en ambas relaciones de izquierda y derecha



Aplicando para ello la cláusula FULL OUTER JOIN que permitirá la visualización de todos los datos según las tablas seleccionadas.

Ilustración 40. Ejemplo Full Outer Join.

Comando DML

Cláusula para combinación externa
SELECT * FROM "Personas"
FULL OUTER JOIN "Directores" ON id_per=id_dir

Columnas de relación de las tablas

Fuente: autores.

Ejecutando la sentencia anterior se obtiene el siguiente resultado:

Ilustración 41. Resultado Full Outer Join.

	id_per character varying	dni_per character varying	nombre_per character varying	direccion_per character varying	telefono_per character varying	fecha_naci_per date	cargo_per character varying	id_ofi character varying	id_dir character varying	salario_anual_dir double precision	fecha_ingreso_dir date	bonifi_mensual_dir double precision
1	DIR001	0701011121	Luis	10 de agosto	2514678	1970-01-17	DIRECTOR	OFI001	DIR001	6000	2000-08-20	500
2	DIR002	0701011122	Marco	12 de noviembre	2514679	1970-01-19	DIRECTOR	OFI002	DIR002	6000	2001-03-21	500
3	DIR003	0701011123	Antonio	Boyaca	2514680	1980-04-19	DIRECTOR	OFI003	DIR003	7200	2010-02-10	600
4	EMP001	0701011131	Felipe	El Sol	2514699	1971-01-17	EMPLEADO	OFI001				
5	EMP002	0701011142	Mateo	Ciudad El Sol	2514700	1972-01-19	EMPLEADO	OFI002				
6	EMP003	0701011153	Rodrigo	Barcelona	2514701	1978-04-29	EMPLEADO	OFI003				

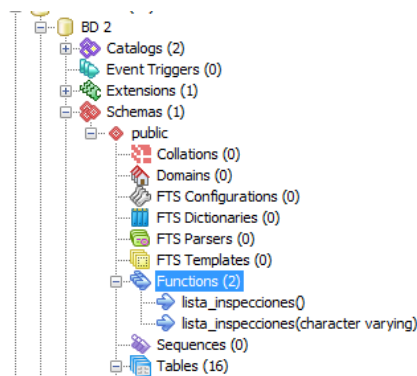
Fuente: autores.

16. FUNCIONES PROCEDURALES

“Postgres soporta la definición de lenguajes procedurales. En el caso de una función o procedimiento definido en un lenguaje procedural, la base de datos no tiene un conocimiento implícito sobre cómo interpretar el código fuente de las funciones” (Guía del Programador de PostgreSQL).

La función nos permite almacenar consultas en nuestra base de datos para ser solicitadas con tan solo el nombre de la función.

Ilustración 42. Funciones Procedurales.



Fuente: autores.

Estas se encuentran disponibles en Functions que esta desglosado de la opción Schemas → Public

SINTAXIS más comunes:

```
CREATE [OR REPLACE] FUNCIÓN nombre (argumentos)
RETURNS valor
AS ' Sentencia ' ;
Language ' sql ';
```

```
CREATE [OR REPLACE] FUNCIÓN nombre (argumentos)
RETURNS valor
AS $$
DECLARE
    Variable ALIAS FOR $número del argumento;
BEGIN
```

```
RETURN valor;  
END;  
$$Language 'plpgsql';
```

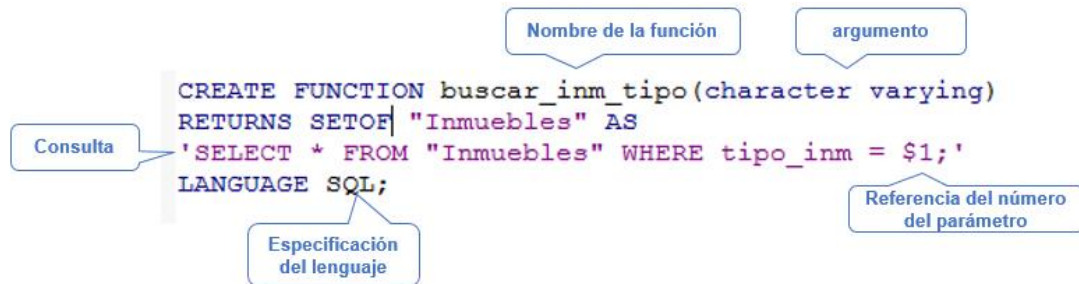
16.1 FUNCIÓN SQL

Función para buscar los datos de un inmueble que sea de algún tipo en específico.



La solución planteada está estructurada en lenguaje sql, en el cual se solicita un argumento que formara parte de la consulta.

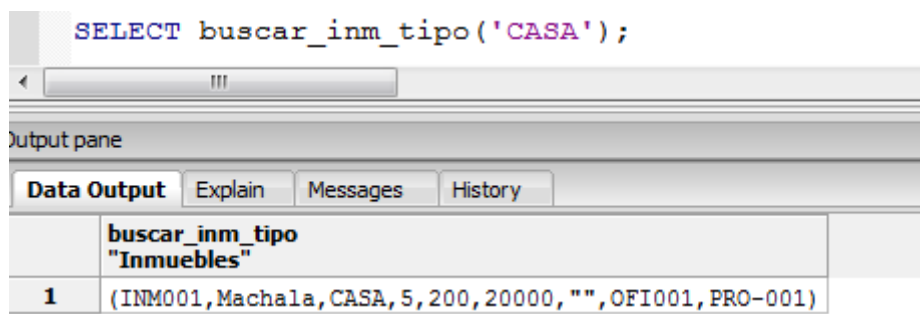
Ilustración 43. Ejemplo Función SQL.



Fuente: autores.

Para llamar la función realizada se debe de emplear la sentencia SELECT y el nombre de la función en donde se debe ingresar la clase de inmueble a buscar que corresponda al tipo de dato puesto en el argumento de la función realizada, como la siguiente ilustración indica:

Ilustración 44. Función SQL.



Fuente: autores.

16.2 FUNCIÓN PLPGSQL

Realizar una función que presente el listado de todas las inspecciones realizadas a un determinado inmueble.



EJEMPLO

Para ello se presenta la siguiente solución:

Ilustración 45. Ejemplo Función PLPGSQL.



Fuente: autores.

De lo cual se hace el llamado mediante la sentencia SELECT y el nombre establecido a la función. Como la función creada esta con parámetros se ingresa el código del inmueble que se registrado una inspección y se podrá visualizar el siguiente resultado:

Ilustración 46. Función PLPGSQL.

```
SELECT lista_inspecciones('INM002')
```

Output pane

	lista_inspecciones "Inspeccion"
1	(INS004,2016-10-10,"Buen estado",EMP002,INM002)
2	(INS007,2016-10-11,Excelente,EMP003,INM002)

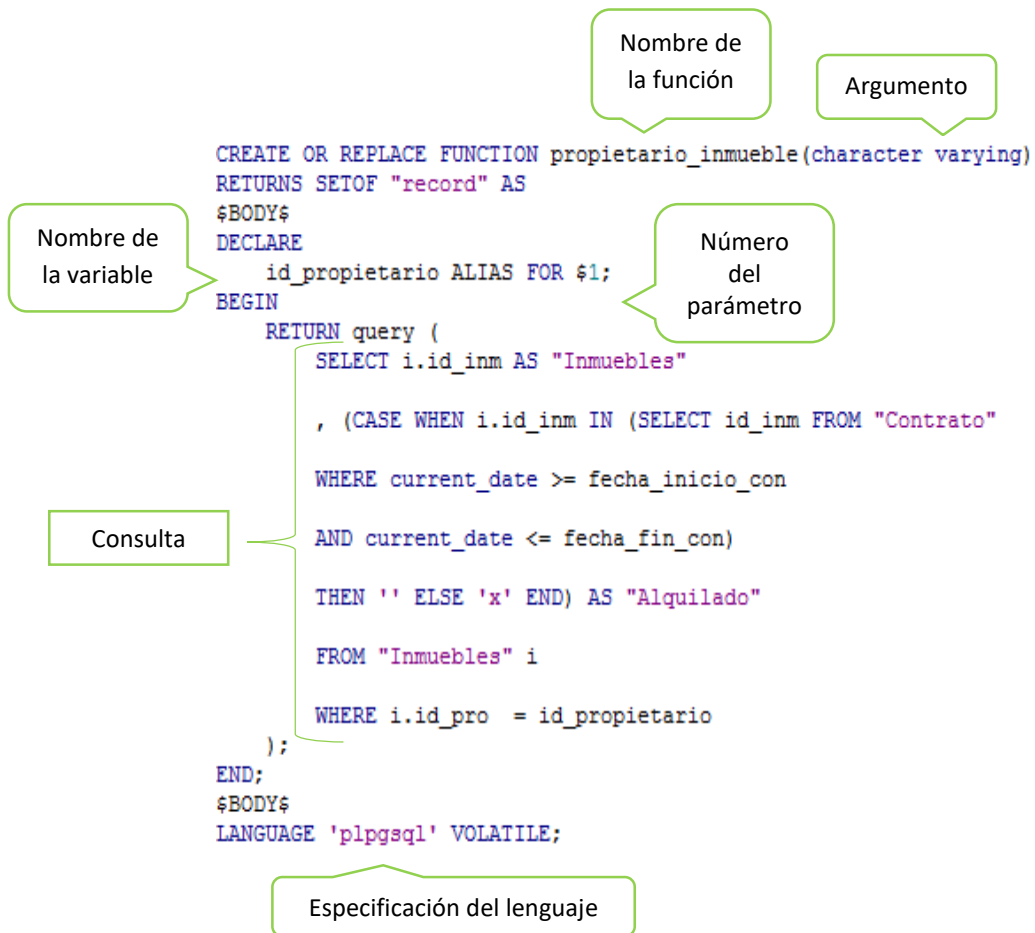
Fuente: autores.



EJEMPLO

Cree una función que reciba como parámetro el id de un propietario, muestre todos los inmuebles y una X en todos aquellos en la cual el inmueble no tiene un contrato vigente.

Ilustración 47. Ejemplo Función PLPGSQL.



Fuente: autores.

Para comprobar el correcto funcionamiento de la función se efectúa una consulta:

```

SELECT * FROM propietario_inmueble('PRO-001')
AS ("Inmuebles" character varying, "Alquilado" TEXT);
    
```

Con lo cual se obtiene como resultado todos los inmuebles que le pertenecen al propietario especificado en el parámetro, y una x en los inmuebles que no tienen un contrato vigente:

Ilustración 48. Resultado Función PLPGSQL.

	Inmuebles character varying	Alquilado text
1	INM001	
2	INM004	x
3	INM005	x

Fuente: autores.

17. TRIGGERS

“Un "trigger" (disparador o desencadenador) es un bloque de código que se ejecuta automáticamente cuando ocurre algún evento (como inserción, actualización o borrado) sobre una determinada tabla (o vista); es decir, cuando se intenta modificar los datos de una tabla (o vista) asociada al disparador” (Moisset, s.f.).

Se crean para conservar la integridad referencial y la coherencia entre los datos entre distintas tablas, para registrar los cambios que se efectúan sobre las tablas y la identidad de quien los realizó, para realizar cualquier acción cuando una tabla es modificada, etc.

Si se intenta modificar (agregar, actualizar o eliminar) datos de una tabla asociada a un disparador, el disparador se ejecuta (se dispara) en forma automática.

Sintaxis

```
CREATE OR REPLACE TRIGGER nombredisparador
MOMENTO -- before, after o instead of
EVENTO -- insert, update o delete
ON nombretabla
NIVEL --puede ser a nivel de sentencia (statement) o de fila (for each row)
WHEN <<CONDICION>> --opcional
BEGIN
-- CUERPO DEL DISPARADOR
END nombredisparador;
```



EJEMPLO

Para agilizar procesos se necesita que las personas registradas como empleados se almacenen al instante en la tabla “Empleados” con valores por defecto.

Para iniciar se procede a la creación de la función:

Ilustración 49. Creación de la Función.

```
CREATE OR REPLACE FUNCTION crea_empleado()
RETURNS TRIGGER AS $crea_empleado$
BEGIN
IF NEW.id_per LIKE 'EMP%' = TRUE THEN
INSERT INTO public."Empleados"(id_emp, salario_anual_emp, fecha_ingreso_emp, velocidad_escri_emp, clave_emp)
VALUES (NEW.id_per, 4800, CURRENT_DATE, 0, md5('usuario'));
END IF;
RETURN NULL;
END;
$crea_empleado$ LANGUAGE plpgsql;
```

← Nombre que tomara el trigger

← Condición

Fuente: autores.

Una vez creada esta función se culmina con la creación del respectivo trigger:

Ilustración 50. Creación del Trigger.

```
CREATE TRIGGER crea_empleado
AFTER INSERT ON "Personas"
FOR EACH ROW EXECUTE PROCEDURE crea_empleado();
```

Se establece que el trigger se activará luego de un INSERT en "Personas"

Fuente: autores.

Ahora se procede a la comprobación, realizando dos inserciones en la tabla Personas y luego se verifica que dicha información se guarde en la tabla Empleados:

Ilustración 51. Comprobación de Trigger.

```
--TABLA PERSONAS
--Registrar directores
insert into "Personas" values
('DIR001', '0701011121', 'Luis', '10 de agosto', '2514678', '17/01/1970', 'DIRECTOR', 'OFI001'),
('DIR002', '0701011122', 'Marco', '12 de noviembre', '2514679', '19/01/1970', 'DIRECTOR', 'OFI002'),
('DIR003', '0701011123', 'Antonio', 'Boyaca', '2514680', '19/04/1980', 'DIRECTOR', 'OFI003');
--Registrar empleados
insert into "Personas" values
('EMP001', '0701011131', 'Felipe', 'El Sol', '2514699', '17/01/1971', 'EMPLEADO', 'OFI001'),
('EMP002', '0701011142', 'Mateo', 'Ciudad El Sol', '2514700', '19/01/1972', 'EMPLEADO', 'OFI002'),
('EMP003', '0701011153', 'Rodrigo', 'Barcelona', '2514701', '29/04/1978', 'EMPLEADO', 'OFI003');

SELECT * FROM "Empleados";
```

	id_emp character varying	salario_anual_emp double precision	fecha_ingreso_emp date	velocidad_escri_emp double precision	clave_emp character varying
1	EMP001	4800	2010-02-10		0 fbc71ce36cc20790f2eed2197898e71
2	EMP002	4800	2010-02-10		0 fbc71ce36cc20790f2eed2197898e71
3	EMP003	4800	2010-02-10		0 fbc71ce36cc20790f2eed2197898e71

Fuente: autores.

18. ENCRIPCIÓN

“Encriptar es una manera de codificar la información para protegerla frente a terceros” (U.).

Por lo tanto, la encriptación informática es la codificación de la información de archivos, de un correo electrónico o cualquier dato para que no pueda ser descifrado en caso de ser interceptado por alguien mientras esta información viaja por la red.

Dicha información no puede ser des-encriptada sin un software de des-encriptación que únicamente conoce el propietario.

Dentro de los algoritmos de encriptación más usados encontramos: 3des, des, bf, aes y md5.

Tabla 4. Encriptación.

Algoritmo	Longitud máxima de contraseña	Bloque de Bits de salida	Recomendable
Des	8	64	No
3des	8	168	Sí
Bf (Blowfish)	72	64	Sí
Aes	Variable	128, 192, 256	Sí
Md5	Ilimitada	128	Sí

Fuente: autores.

Sintaxis de uso algoritmos:

```
insert into usuario (usuario, clave) values ('usuario', encrypt('pass', 'key','3des'))
```

Sintaxis de uso md5:

```
insert into usuario (usuario, clave) values ('usuario', md5('pass'))
```

Pass = dato que se desea encriptar.

Key = Llave identificadora que servirá para des-encriptar el dato.

3des = Algoritmo de encriptación, puede variar a des, bf, aes, etc.



EJEMPLO

Para empezar a usar cualquier método de encriptación ese necesario habilitar la extensión pgcrypto, la cual se lo hace de la siguiente manera:

```
CREATE EXTENSION pgcrypto;
```

Luego procedemos a realizar algún registro, en donde encriptaremos los campos que deseamos o creamos necesario:

Ilustración 52. Ejemplo Encriptación.

```
INSERT INTO public."Personas"( id_per, dni_per, nombre_per, direccion_per, telefono_per, fecha_naci_per, cargo_per, id_ofi)
VALUES ('EMP001', encrypt('0701011131', 'llave', '3des')::text, 'Felipe', encrypt('El Sol', 'llave', '3des')::text, '2514699', '1971-01-17', 'EMPLEADO', 'OFI001');
```

Fuente: autores.

Si aplicamos una consulta de la tabla los datos que obtendremos serán los siguientes:

Ilustración 53. Resultados Consulta.

```
SELECT * FROM "Personas";
```

	id_per character varying	dni_per character varying	nombre_per character varying	direccion_per character varying	telefono_per character varying	fecha_naci_per date	cargo_per character varying	id_ofi character varying
1	EMP001	\005\214o\014S\002By\375\010\212DbG\325g	Felipe	\366\253\020\235\270\2557e	2514699	1971-01-17	EMPLEADO	OFI001

Fuente: autores.

Y si se desea visualizar los datos se tiene que hacer un SELECT más elaborado:

Ilustración 54. Visualización de datos.

```
SELECT id_per, encode(decrypt(dni_per::bytea, 'llave', '3des'::text), 'escape'::text),
nombre_per, encode(decrypt(direccion_per::bytea, 'llave', '3des'::text), 'escape'::text),
telefono_per, fecha_naci_per, cargo_per, id_ofi
FROM "Personas";
```

	id_per character varying	encode text	nombre_per character varying	encode text	telefono_per character varying	fecha_naci_per date	cargo_per character varying	id_ofi character varying
	EMP001	0701011131	Felipe	El Sol	2514699	1971-01-17	EMPLEADO	OFI001

Fuente: autores.

Como se observa en la imagen, se aplicó el método “encode” para descifrar la información, además fue necesario aplicar la conversión de datos a tipo bytea pues con este es como trabaja el método decrypt.

Ahora en la tabla “Empleados” procederemos a realizar un registro, pero en la columna clave_emp realizaremos el registro encriptado con el método md5.

Ilustración 55. Registro de Dato Encriptado.

```
INSERT INTO public."Empleados"( id_emp, salario_anual_emp, fecha_ingreso_emp, velocidad_escri_emp, clave_emp)
VALUES ('EMP001', 48000, Current_date, 10, md5('clave'));
```

Fuente: autores.

Luego se realiza una vista hacia esta tabla y veremos como todos los campos están con su respectiva columna encriptada.

Ilustración 56. Vista con datos Encryptados.

```
SELECT * FROM "Empleados";
```

Output pane

	Data Output	Explain	Messages	History	
	id_emp character varying	salario_anual_emp double precision	fecha_ingreso_emp date	velocidad_escri_emp double precision	clave_emp character varying
1	EMP001	4800	2009-11-20	10	fb71ce36cc20790f2eed2197898e71

Fuente: autores.

Como md5 es un algoritmo el cual no se puede des-criptar la información entonces al momento de que algún empleado olvide la clave la única solución será cambiarla.

19. PRIVILEGIOS Y USUARIOS

Grupos de Usuarios

Esa propiedad nos permite agrupar a varios usuarios con el objetivo de asignar privilegios de manera general para optimizar tiempo. Luego podemos crear usuarios de manera independiente y enlazarlos con algún grupo.

Sintaxis grupo:

```
CREATE GROUP [nombregrupo]
```

Sintaxis usuario:

```
CREATE USER [nombreusuario] WITH PASSWORD 'password' IN GROUP [nombregrupo]
```

Privilegios

Con la asignación de privilegios a usuarios se da la autorización a que este o a un grupo de usuarios para que realice cualquier acción sobre una tabla específica. Dichas acciones pueden ser otorgadas con el comando "GRANT" o a su vez eliminadas con el comando "REVOKE".

Sintaxis:

```
GRANT [SELECT, INSERT UPDATE, DELETE, ALL] ON [nombretabla] TO [nombreusuario o nombreGrupo]
```

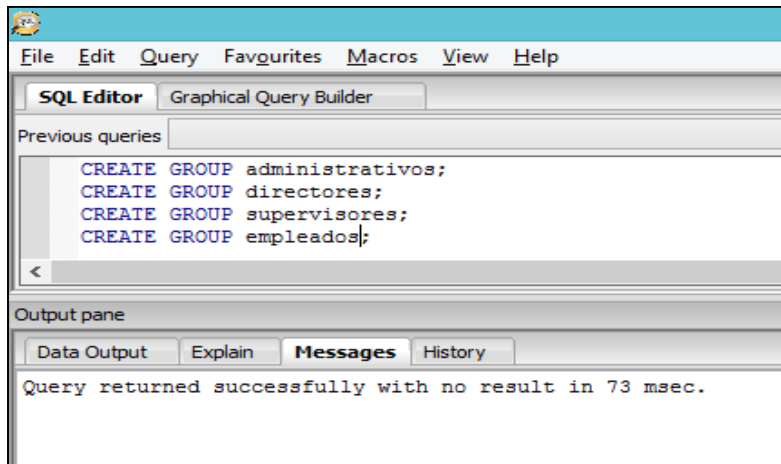


EJEMPLO

Bajo estos conceptos aplicaremos lo mismo a la base de datos de inmobiliaria.

Primero creamos los 4 grupos principales (Administrativo, Director, Supervisor, Empleado).

Ilustración 57. Creación de Usuarios.



Fuente: autores.

Luego nos guiaremos en la siguiente matriz de trazabilidad, en ella observamos que acciones se han asignado a los respectivos grupos en las diferentes tablas de nuestra base de datos.

Tabla 5. Matriz de Trazabilidad de los Usuarios.

	Administrativo	Director	Supervisor	Empleado	
Personas	ALL	ALL	ALL	ALL	
Directores	ALL	ALL	ALL	ALL	SELECT
Empleados	ALL	ALL	ALL	ALL	
Pariente	ALL	ALL	ALL	ALL	ALL
Oficinas	ALL	SELECT	ALL	ALL	
Visitas	ALL	ALL	SELECT	ALL	
Clientes	ALL	ALL	ALL	SELECT	
Inspección	ALL	ALL	ALL	ALL	
Inmuebles	ALL	ALL	ALL	SELECT	
Inmuebles - Factura	ALL	ALL	ALL	ALL	
Factura	ALL	ALL	ALL	ALL	
Periódico	ALL	ALL	SELECT	ALL	
Publicidad	ALL	ALL	SELECT	ALL	
Propietario	ALL	ALL	ALL	ALL	
Contrato	ALL	ALL	ALL	ALL	
Pago	ALL	ALL	ALL	ALL	

Fuente: autores.

Por lo tanto, quedaría así:

Ilustración 58. Creación de Permisos a Usuarios.

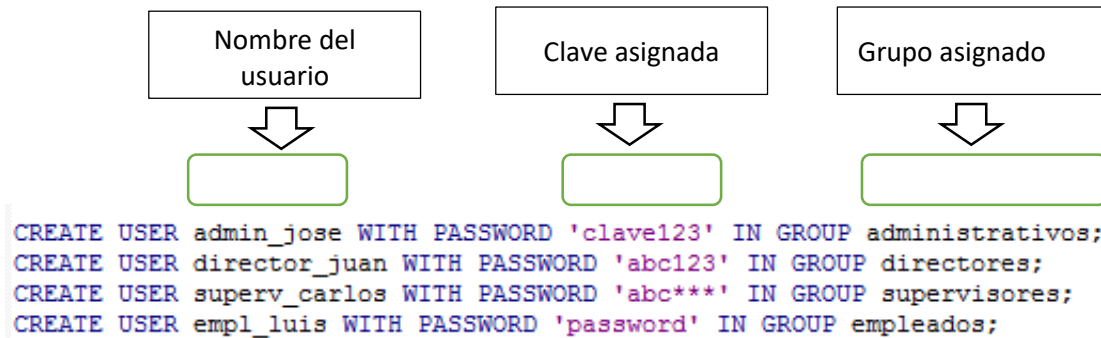
```
GRANT ALL ON "Personas" TO GROUP administrativos;
GRANT ALL ON "Directores" TO GROUP administrativos;
GRANT ALL ON "Empleados" TO GROUP administrativos;
GRANT ALL ON "Paciente" TO GROUP administrativos;
GRANT ALL ON "Oficinas" TO GROUP administrativos;
GRANT ALL ON "Visitas" TO GROUP administrativos;
GRANT ALL ON "Clientes" TO GROUP administrativos;
GRANT ALL ON "Inspeccion" TO GROUP administrativos;
GRANT ALL ON "Inmuebles" TO GROUP administrativos;
GRANT ALL ON "Inmueble_Factura" TO GROUP administrativos;
GRANT ALL ON "Facturas" TO GROUP administrativos;
GRANT ALL ON "Periodico" TO GROUP administrativos;
GRANT ALL ON "Publicidad" TO GROUP administrativos;
GRANT ALL ON "Propietario" TO GROUP administrativos;
GRANT ALL ON "Contrato" TO GROUP administrativos;
GRANT ALL ON "Pago" TO GROUP administrativos;
```

Output pane
Data Output Explain Messages History
Query returned successfully with no result in 23 msec.

Fuente: autores.

Una vez creado estos grupos de usuarios procedemos a crear a los respectivos usuarios, basándonos en la sintaxis presentada anteriormente. Se creará un usuario por grupo, pero se pueden añadir más dependiendo las necesidades de la empresa.

Ilustración 59. Creación de Usuarios por Grupos.



Fuente: autores.

Y si ejecutamos la sentencia: `SELECT * FROM pg_shadow` podremos ver como los usuarios han sido creados.

Ilustración 60. Sentencia para para visualizar Usuarios.

```
SELECT * from pg_shadow;
```

	username name	usesysid oid	usecreatedb boolean	usesuper boolean	userepl boolean	usebypassrls boolean	passwd text	valuntil abstime	useconfig text[]
7	admin_jose	25163	f	f	f	f	md5f88b89f6053da685eb10dc9987f93a0f		
8	director_juan	25164	f	f	f	f	md574a9f7b3165b009439ae89794a363bf5		
9	superv_carlos	25165	f	f	f	f	md53d90de7a08b80e7d61388bcd5c4a3424		
10	empl_luis	25166	f	f	f	f	md510bee7d2eddfe65b4c517f4c149570f		

Fuente: autores.

Y si queremos modificar por ejemplo la contraseña solo es necesario ejecutar la sentencia ALTER USER:

```
ALTER USER admin_jose WITH PASSWORD 'nueva_password';
```

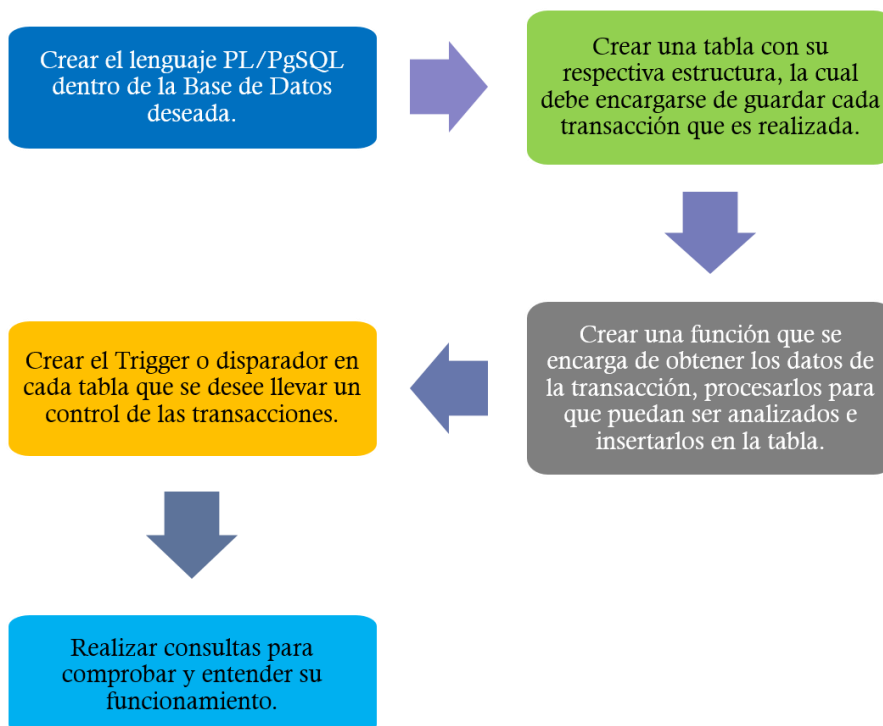
20. AUDITORIAS A LAS BASES DE DATOS

Es el proceso que permite medir, asegurar, demostrar, monitorear y registrar los accesos a la información en las bases de datos incluyendo la capacidad de determinar: quién accede a los datos, cuándo se accedió a los datos, desde qué tipo de dispositivo o aplicación, desde que ubicación en la red, cuál fue la sentencia SQL ejecutada y cuál fue el efecto del acceso a la base de datos (Mangones).

La auditoría de base de datos no es más que un proceso de control de la información que posee una organización, con fines estadísticos y sobre todo de seguridad, con respecto a las transacciones (INSERT, UPDATE, DELETE) que se realizan, para aquello se utiliza el lenguaje de manipulación de datos DML.

Los pasos para crear una auditoría son:

Ilustración 61. Pasos para crear una Auditoría.



Fuente: autores.



EJEMPLO

La empresa Tierra Prometida solicita que se controle las transacciones sobre los contratos, para mantener una mejor seguridad de la información.

Antes de comenzar con la auditoria que procede a crear una base de datos para almacenar ahí los cambios que se realicen en la tabla contratos de la base de datos inmobiliaria.

La sentencia requerida es: CREATE DATABASE auditoria;

```
CREATE DATABASE auditoria;
```

Luego hay que ubicarse en la base de datos creada y pulsar el botón de consultas 

En la ventana de consultas se escriben las siguientes instrucciones para crear una tabla, para que guarde la información de las transacciones de la empresa Tierra Prometida.

```
-- tabla para controlar los cambios sobre la tabla contratos
CREATE TABLE tabla_auditoria (
  id_auditoria serial NOT NULL,
  nombre_tabla character(45) NOT NULL,
  operacion char(1) NOT NULL,
  valor_viejo text,
  valor_nuevo text,
  fecha_actual timestamp without time zone NOT NULL,
  usuario character(45) NOT NULL,
  CONSTRAINT id_auditoria_pk PRIMARY KEY (id_auditoria));
```

Para poder enlazar las bases de datos se ejecuta la siguiente instrucción:

```
create extension dblink;
```

Luego se tiene que ubicar en la base de datos inmobiliaria y en el editor de consultas tiene que proceder a crear una función, la cual compara que tipo de operación se está efectuando, como puede ser eliminación, actualización o inserción de datos. Luego se hace una inserción dentro de la tabla denominada tabla_auditoria.

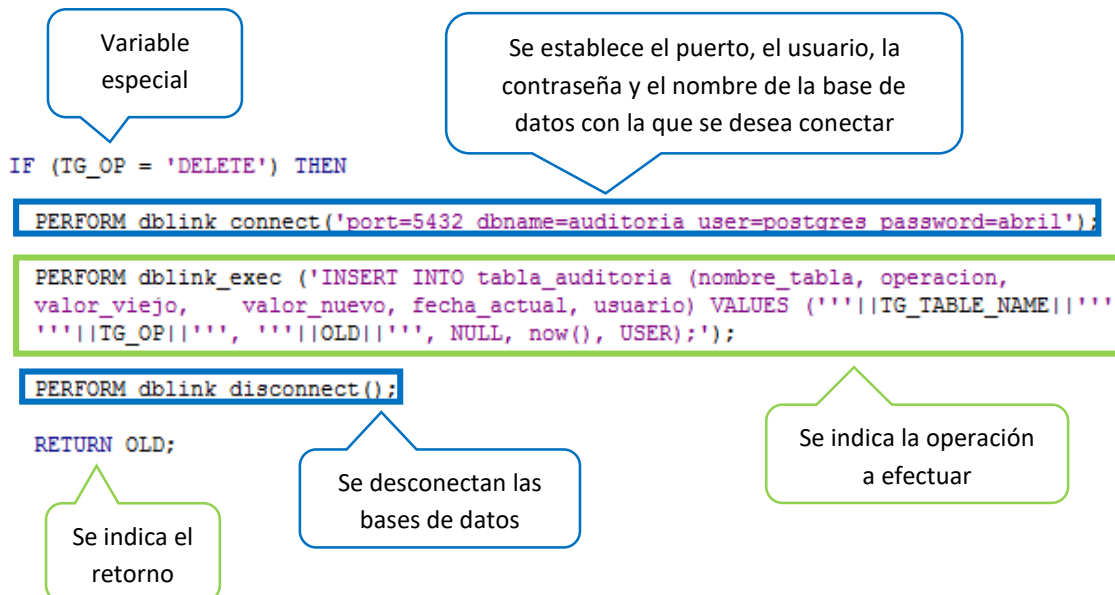
```

CREATE OR REPLACE FUNCTION funcion_auditoria() RETURNS trigger AS
$$
BEGIN
  IF (TG_OP = 'DELETE') THEN
    PERFORM dblink_connect('port=5432 dbname=auditoria user=postgres password=abril');
    PERFORM dblink_exec ('INSERT INTO tabla_auditoria (nombre_tabla, operacion,
    valor_viejo,      valor_nuevo, fecha_actual, usuario) VALUES (''||TG_TABLE_NAME||''',
    ''||TG_OP||''', ''||OLD||''', NULL, now(), USER);');
    PERFORM dblink_disconnect();
    RETURN OLD;
  ELSIF (TG_OP = 'UPDATE') THEN
    PERFORM dblink_connect('port=5432 dbname=auditoria user=postgres password=abril');
    PERFORM dblink_exec ('INSERT INTO tabla_auditoria (nombre_tabla, operacion,
    valor_viejo,      valor_nuevo, fecha_actual, usuario) VALUES (''||TG_TABLE_NAME||''',
    ''||TG_OP||''', ''||OLD||''', ''||NEW||''', now(), USER);');
    PERFORM dblink_disconnect();
    RETURN NEW;
  ELSIF (TG_OP = 'INSERT') THEN
    PERFORM dblink_connect('port=5432 dbname=auditoria user=postgres password=abril');
    PERFORM dblink_exec ('INSERT INTO tabla_auditoria (nombre_tabla, operacion, |
    valor_viejo, valor_nuevo, fecha_actual, usuario) VALUES (''||TG_TABLE_NAME||''',
    ''||TG_OP||''', NULL, ''||NEW||''', now(), USER);');
    PERFORM dblink_disconnect();
    RETURN NEW;
  END IF;
  RETURN NULL;
END;
$$
LANGUAGE 'plpgsql' VOLATILE;

```

A continuación, se explica el funcionamiento de la función:

Ilustración 62. Explicación Funcionamiento de la Función.



Fuente: autores.

TG_OP, TG_TABLE_NAME, OLD, NEW son variables especiales del lenguaje PL/PGSQL. TG_OP que contiene una cadena de texto con el valor de INSERT, UPDATE o DELETE de acuerdo a la operación que activó el disparador o trigger. TG_TABLE_NAME contiene el nombre de la tabla

que ha activado el disparador, OLD contiene la antigua fila para las operaciones UPDATE/DELETE, y NEW contiene la nueva fila para las operaciones INSERT/UPDATE.

Mientras que now() devuelve la fecha actual del sistema, y USER contiene el nombre del usuario que activó el disparador.

Para poder usar la función se crea un disparador o trigger, el cual se habilitará después de insertar, actualizar o eliminar registros en la tabla contrato.

```
CREATE TRIGGER tbl_atributos_tg_audit
AFTER INSERT OR UPDATE OR DELETE ON public."Contrato"
FOR EACH ROW EXECUTE PROCEDURE funcion_auditoria();
```

Para comprobar el funcionamiento de la función y trigger se ejecutan sentencias del lenguaje de manipulación de datos DML.

```
INSERT INTO "Contrato" (id_con, importe_mensual_con, metodo_pago_con,
importe_deposito_con, esta_cancelado_con, fecha_inicio_con,
fecha_fin_con, id_inm, id_emp, id_cli, duracion_con)
VALUES ('CON015', 300, 'CONTADO', 150, TRUE, '2016-12-12',
'2017-12-12', 'INM001', 'EMP002', 'CLI001', 4);
```

Tabla_auditoria de la base de datos auditoria:

Ilustración 63. Resultados Tabla Auditoría.

	pk_audit [PK] serial	nombre_tabla character(45)	operacion character(10)	valor_viejo text	valor_nuevo text	fecha_actual timestamp without time zone	usuario character(45)
1	20	Contrato	INSERT		(CON015,300,CONTADO,150,t,2016-12-12,2017-12-12,INM001,EMP002,CLI001,4)	2017-01-19 22:29:03.640872	postgres
*							

Fuente: autores.

Tabla contratos de la base de datos inmobiliaria:

Ilustración 64. Tabla Contratos.

	id_con [PK] char	importe double p	metodo_p character	importe double	esta_c boolea	fecha_inicio_ date	fecha_fin_con date	id_inm characte	id_emp characte	id_cli characte	durac intege
1	CON001	200	CONTADO	100	FALSE	2015-10-22	2016-08-22	INM001	EMP001	CLI001	10
2	CON002	140	CONTADO	140	TRUE	2015-10-25	2016-06-25	INM002	EMP001	CLI002	8
3	CON003	230	CONTADO	190	FALSE	2015-11-12	2016-09-26	INM003	EMP001	CLI003	10
4	CON004	300	CONTADO	150	TRUE	2016-12-12	2017-12-12	INM001	EMP002	CLI001	4
5	CON015	300	CONTADO	150	TRUE	2016-12-12	2017-12-12	INM001	EMP002	CLI001	4

Fuente: autores.

21. TRANSACCIONES

En la actualidad las grandes organizaciones que poseen sistemas automatizados no podrían funcionar de una manera adecuada sin un procesamiento de transacciones confiables y eficientes.

Una transacción de base de datos “comprende un grupo de operaciones que se deben procesar como una unidad de trabajo. Las transacciones se deben procesar de manera confiable, de modo

que no haya ninguna pérdida de datos debido a usuarios múltiples o fallas en el sistema” (Mannino).

Una transacción “se inicia por la ejecución de un programa de usuario escrito en un lenguaje de manipulación de datos de alto nivel o en un lenguaje de programación, y está delimitado por instrucciones de la forma inicio transacción y fin transacción. La transacción consiste en todas las operaciones que se ejecutan entre inicio transacción y el fin transacción” (Silberschatz, 2002).

21.1 PROPIEDADES DE LA TRANSACCIÓN

Los sistemas gestores de base de datos “DBMS” cuentan con propiedades para asegurar una correcta ejecución de las transacciones, una de las propiedades más importantes y conocidas es ACID (atómica, consistente, aislada, durable).

Atomicidad: “significa que una transacción no se puede subdividir. Ya sea que se realice todo el trabajo en la transacción o que no se haga nada”. (Mannino) Esta propiedad establece que se realizan todas las operaciones de la transacción de la base de datos o no se efectúa ninguna.

Consistencia: “significa que, si las limitaciones aplicables son ciertas antes de empezar la transacción, éstas también lo serán al terminarla”. (Mannino) Esta propiedad consiste en que los datos deben estar bien relacionados y no existan problemas de falta de información.

Aislamiento: “significa que las transacciones no interfieren entre sí, excepto en formas permitidas. Una transacción nunca debe sobrescribir los cambios realizados por otra. Además, una transacción no debe interferir en otros aspectos, como no ver los cambios temporales realizados por otras transacciones” (Mannino).

Durabilidad: “significa que cualquier cambio que resulte de una transacción es permanente. Ninguna falla va a borrar ningún cambio después de terminar la transacción”. (Mannino) Esta propiedad indica que, al finalizar una transacción con éxito, las modificaciones realizadas en la base de datos permanecen, incluso si se presentan fallos en el sistema.

Reglas ACID: comprobar la propiedad de atomicidad (lo que se ejecuta en una transacción se ejecuta todo o nada), usando la tabla “Contrato”.



EJEMPLO

Para comprender la propiedad de atomicidad se presentan algunos ejemplos, el primero de estos hace uso del comando COMMIT, el cual se utiliza para confirmar como permanentes las modificaciones realizadas en una transacción:

Ilustración 65. Ejemplo Transacción.

Indica el inicio de una secuencia de comandos

```
BEGIN;  
INSERT INTO public."Clientes" VALUES  
('CLI004', 'Nixon', 'El Cambio', '0981242156', 'INM003', 250);  
COMMIT;
```

Inserción de un registro en la tabla clientes

Confirma las modificaciones realizadas en una transacción

Fuente: autores.

Para comprobar que los comandos anteriores se ejecutaron y almacenaron en la base de datos se hace una consulta en la tabla clientes:

```
SELECT * FROM "Clientes";
```

Cuyo resultado es:

Ilustración 66. Resultado Consulta.

	id_cli character varying	nombre_cli character varying	direccion_cli character varying	telefono_cli character varying	inmueble_preferido_cli character varying	importe_maximo_cli double precision
1	CLI001	Ufredo Romero	Santa Rosa	2890910	INM001	300
2	CLI002	Diego Romero	Santa Rosa	2890911	INM002	150
3	CLI003	Jazmin Quirola	Machala	2890912	INM002	190
4	CLI004	Nixon	El Cambio	0981242156	INM003	250

Fuente: autores.

Con lo cual se puede evidenciar que el registro de la transacción se ha guardado en la base de datos inmobiliaria de una manera correcta.

Otra forma de comprobar la propiedad de atomicidad es la que se indica a continuación, en la cual se hace uso del comando ROLLBACK, el cual permite deshacer todas las modificaciones que se han realizado a la base de datos pero que no han sido escritas en el disco duro por la sentencia COMMIT.

Ilustración 67. Propiedad de Atomicidad.

Indica el inicio de una secuencia de comandos

```
BEGIN;  
INSERT INTO public."Clientes" VALUES  
('CLI005', 'Ezequiel', 'Atahualpa', '0981233158', 'INM004', 200);  
ROLLBACK;
```

Inserción de un registro en la tabla clientes

Deshace las modificaciones que no han sido escritas en el disco duro

Fuente: autores.

Para comprobar que los comandos anteriores se ejecutaron y no se almacenaron en la base de datos se hace una consulta en la tabla clientes:

```
SELECT * FROM "Clientes";
```

Cuyo resultado es:

Ilustración 68. Resultado Consulta.

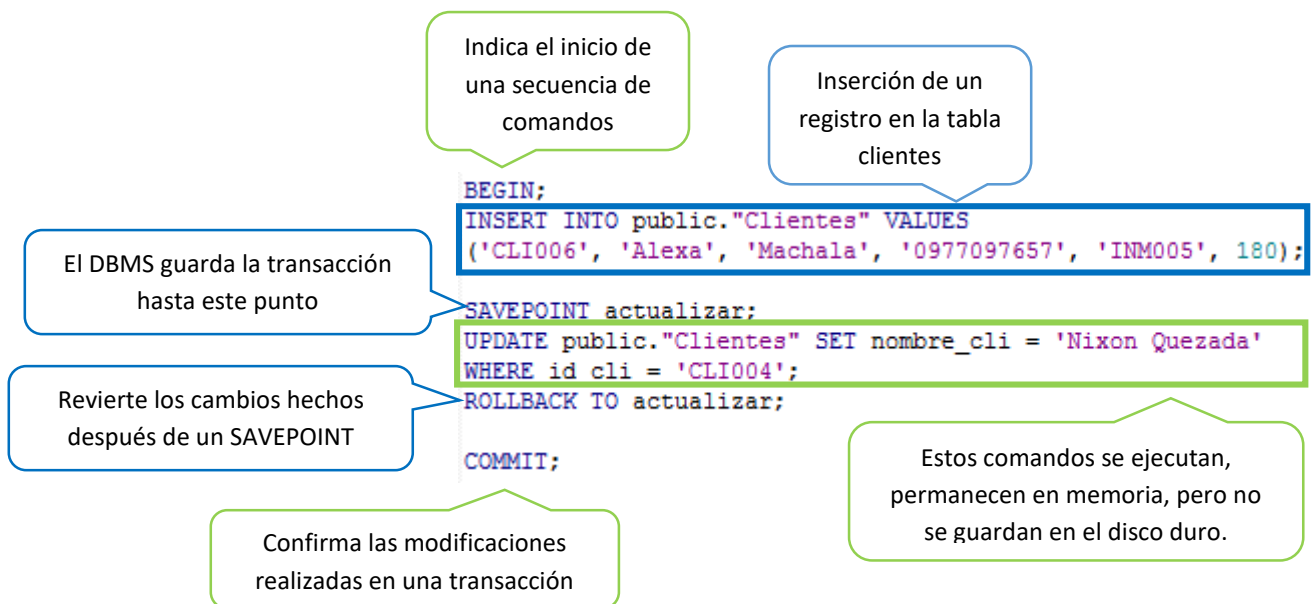
	id_cli character varying	nombre_cli character varying	direccion_cli character varying	telefono_cli character varying	inmueble_preferido_cli character varying	importe_maximo_cli double precision
1	CLI001	Ufredo Romero	Santa Rosa	2890910	INM001	300
2	CLI002	Diego Romero	Santa Rosa	2890911	INM002	150
3	CLI003	Jazmin Quirola	Machala	2890912	INM002	190
4	CLI004	Nixon	El Cambio	0981242156	INM003	250

Fuente: autores.

Con lo cual, se puede evidenciar que el registro de la transacción no se ha guardado en la base de datos inmobiliaria.

Otros de los comandos que se pueden usar para la comprobación de la propiedad de atomicidad son: SAVEPOINT y ROLLBACK TO. SAVEPOINT le indica al DBMS la ubicación de un punto de retorno en una transacción en caso de que la transacción sea cancelada. ROLLBACK TO revierte los cambios hechos después de un SAVEPOINT.

Ilustración 69. Comprobación de Atomicidad.



Fuente: autores.

Para comprobar que los comandos anteriores se ejecutaron y cumplen con las definiciones, se hace una consulta en la tabla clientes de la base de datos inmobiliaria:

```
SELECT * FROM "Clientes";
```

Cuyo resultado es:

Ilustración 70. Resultado Consulta.

	id_cli character varying	nombre_cli character varying	direccion_cli character varying	telefono_cli character varying	inmueble_preferido_cli character varying	importe_maximo_cli double precision
1	CLI001	Ufredo Romero	Santa Rosa	2890910	INM001	300
2	CLI002	Diego Romero	Santa Rosa	2890911	INM002	150
3	CLI003	Jazmin Quirola	Machala	2890912	INM002	190
4	CLI004	Nixon	El Cambio	0981242156	INM003	250
5	CLI006	Alexa	Machala	0977097657	INM005	180

Fuente: autores.

Con lo cual se evidencia que las instrucciones que se encuentran dentro del bloque SAVEPOINT <actualizar> y ROLLBACK TO <actualizar> no han efectuado ningún cambio, porque no han sido escritas en el disco duro, y en cambio la información que se ubica fuera del bloque SAVEPOINT <actualizar> y ROLLBACK TO <actualizar> si ha sido almacenada en la base de datos.

Reglas ACID: comprobar la propiedad de consistencia usando la tabla "Contrato".



EJEMPLO

Para comprender la propiedad de consistencia se presentan algunos ejemplos, los cuales enfatizan sobre la integridad de los datos que lleva a cabo postgresQL.

Ilustración 71. Propiedad de Consistencia.

Indica el inicio de una secuencia de comandos

Inserción de un registro en la tabla clientes

```
BEGIN;  
INSERT INTO public."Clientes" VALUES |  
('CLI004', 'Nixon', 'El Cambio', '0981242156', 'INM003', 250);  
COMMIT;
```

Confirma las modificaciones realizadas en una transacción

Fuente: autores.

Al ejecutar la transacción anterior se produce el siguiente resultado:

```
ERROR: llave duplicada viola restricción de unicidad «Clientes_pkey»  
DETAIL: Ya existe la llave (id_cli)=(CLI004).  
***** Error *****
```

Esto debido a que ya existe un registro que contiene el valor de CLI004 como clave primaria. Con lo cual se evidencia que postgresQL es un DBMS que controla la integridad de los datos, por lo tanto, cumple con la propiedad de consistencia.

Reglas ACID: comprobar la propiedad de aislamiento (los cambios en una transacción no terminada no se ven en otra sesión), usando la tabla “Contrato”.

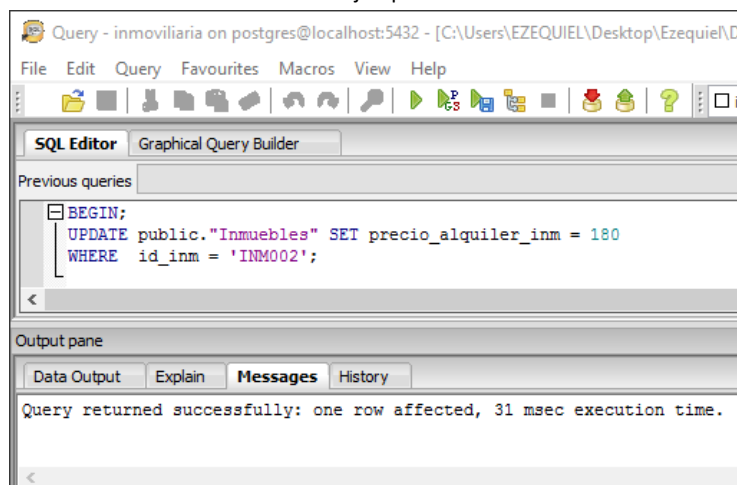
Para comprobar que PostgreSQL cumple con la propiedad de aislamiento se abren dos ventanas para ejecutar consultas SQL.



EJEMPLO

En la primera ventana se ejecutan los siguientes comandos, con lo cual se indica que la consulta fue exitosa:

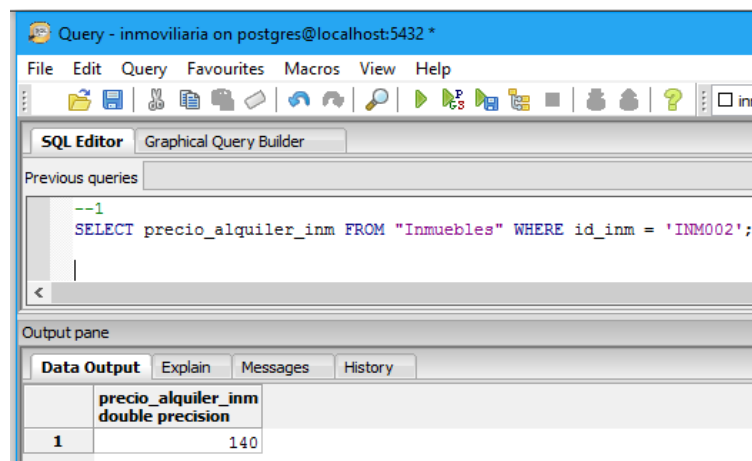
Ilustración 72. Ejemplo Transacción.



Fuente: autores.

Si otro usuario deseara visualizar cierta información de la base de datos inmobiliaria, específicamente el precio de alquiler del inmueble cuyo código es INM002, se muestra el valor que estaba registrado en la base y no el valor que se modificó en la ventana 1.

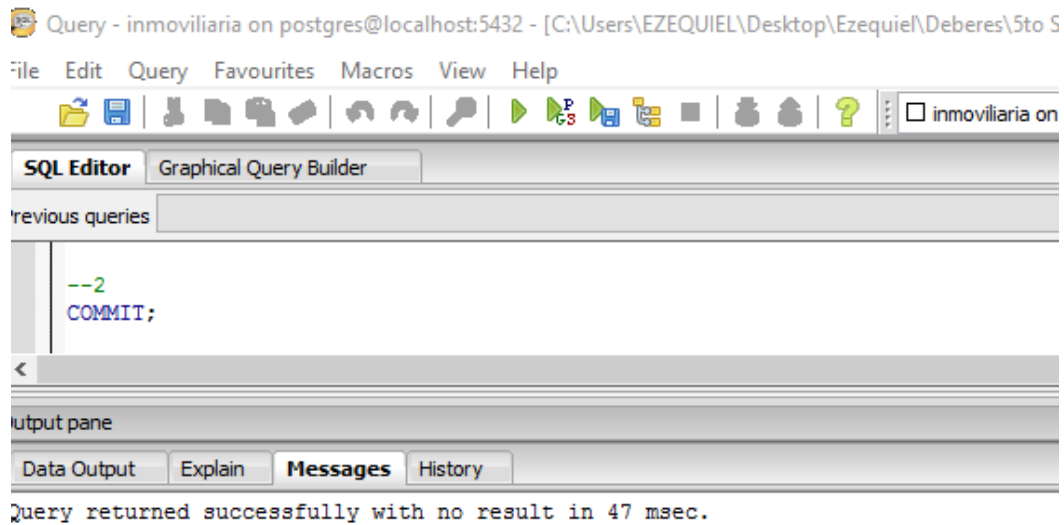
Ilustración 73. Visualización de información.



Fuente: autores.

Al escribir el comando COMMIT en la ventana 1 y ejecutarlo recién se estará guardando el registro en el disco duro.

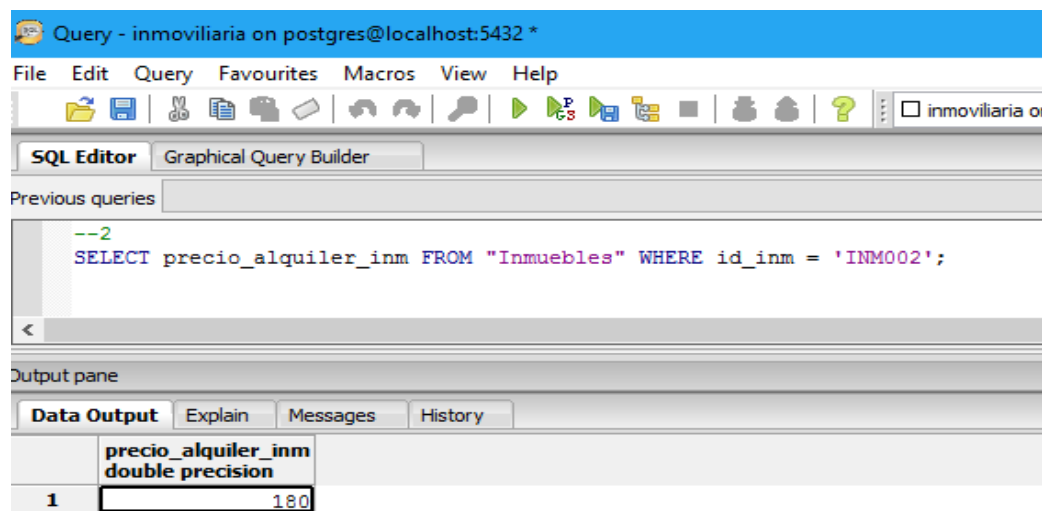
Ilustración 74. Uso de Comando COMMIT.



Fuente: autores.

Ahora, si se vuelve hacer la misma consulta en la ventana 2, se podrá visualizar que los cambios se han sido almacenados en el disco duro.

Ilustración 75. Realización de Consulta.



Fuente: autores.

22. HERRAMIENTA DE RESGUARDO

“La copia de seguridad, copia de respaldo o Backup es importante dentro de la administración de un sistema de base de datos. Es una copia de los datos originales que son realizados con la finalidad de disponer de un medio de recuperarlos en el caso de su pérdida” (Padrón, 2013).

Una copia de seguridad se la pueda hacer a través del pgAdmin III de forma gráfica, y otra manera es utilizando los comandos en el cmd de windows.

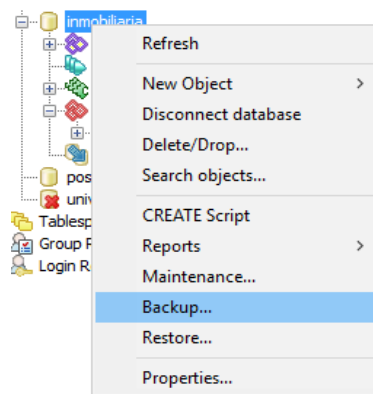


EJEMPLO

Hacer un respaldo de la base de datos inmobiliaria utilizando el pgAdmin III.

Abrir al pgAdmin III, dirigirse a la base de datos inmobiliaria y hacer clic derecho, escoger la opción que dice Backup.

Ilustración 76. Ejemplo Herramienta de Resguardo.

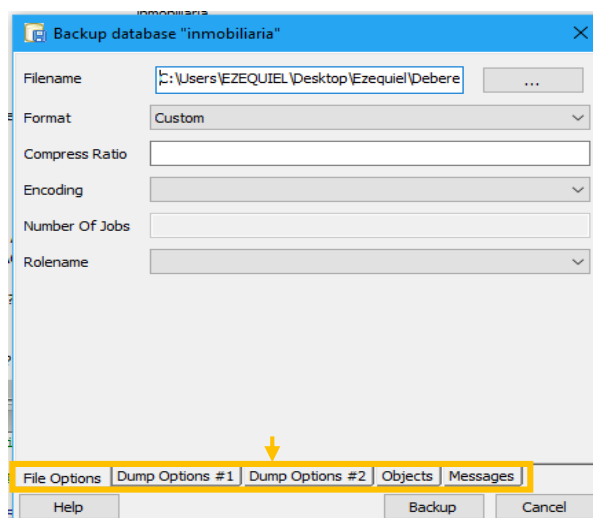


Fuente: autores.

En la parte inferior de la ventana que se abre se “puede observar una serie de pestañas que permiten acceder a las diferentes opciones que proporciona PostgreSQL para realizar el procedimiento de crear la copia de seguridad” (Copias de seguridad, restauración y recuperación de una bd).

En la pestaña “File Options” se ingresará en cada campo, los datos que se requieran, o se puede dejar los campos vacíos, a excepción del campo filename, y postgresQL se encargará de establecer los valores predeterminados.

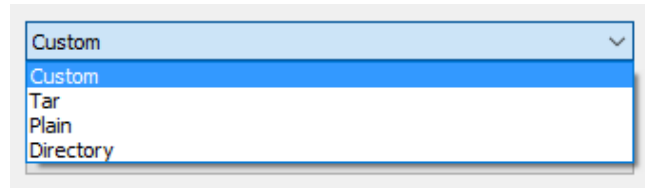
Ilustración 77. Ejemplo Herramienta de Resguardo.



Fuente: autores.

En el campo de entrada “Format” se puede escoger los siguientes formatos:

Ilustración 78. Campo de entrada “Format”.



Fuente: autores.

Acontinuación se describen cada uno de estos formatos:

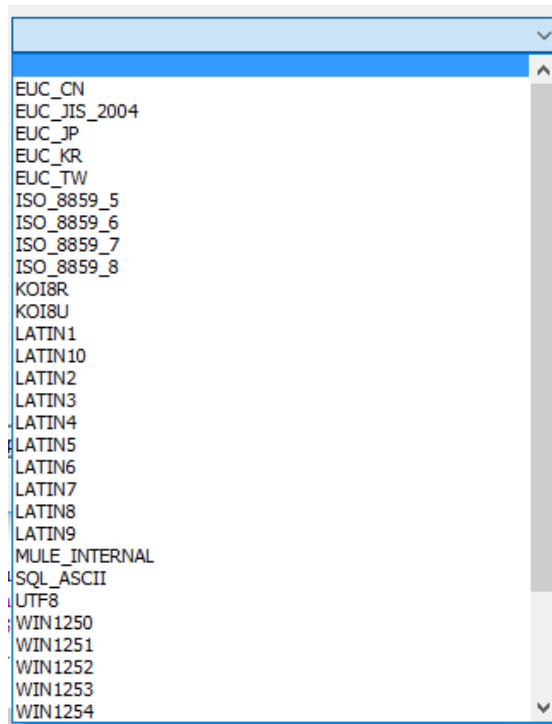
Tabla 6. Formatos.

FORMAT	DESCRIPCIÓN
Custom	Genera un archivo con extensión .backup
Tar	Genera un archivo con extensión .backup pero sin compresión
Plain	Genera un archivo con extensión .sql que se puede ejecutar desde psql.
Directorio	Genera una carpeta que contiene los archivos de respaldo de la base de datos.

Fuente: autores.

“El campo de entrada “Encoding” permite seleccionar los caracteres que se utilizará para exportar los datos al backup” (Copias de seguridad, restauración y recuperación de una bd).

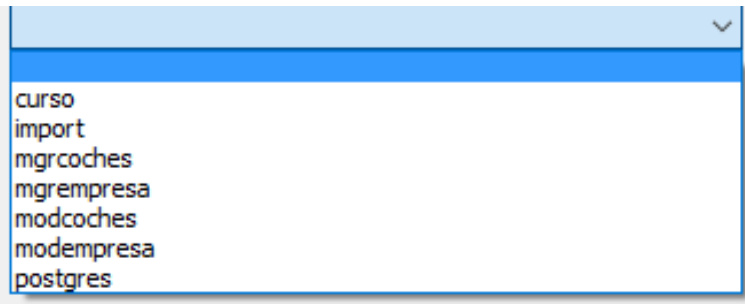
Ilustración 79. Campo de entrada “Encoding”.



Fuente: autores.

En el campo de entrada Rolename se puede escoger el usuario con que se creará el backup. Es importante seleccionar el usuario que tiene todos los privilegios para evitar problemas de permisos de usuarios cuando se utilice el archivo .backup en una restauración.

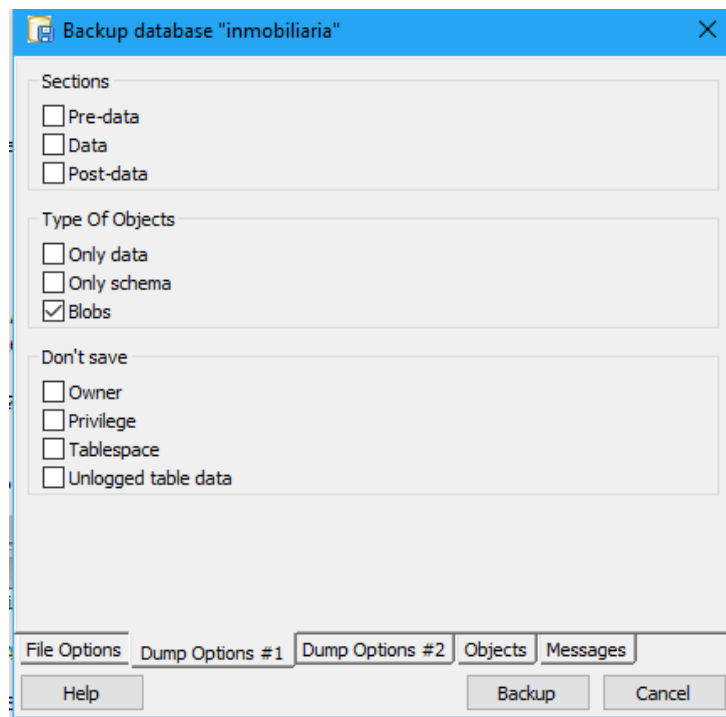
Ilustración 80. Campo de entrada "Rolename".



Fuente: autores.

En la pestaña "Dump Options #1" se visualizan tres secciones que son: sections, type of objects y don't save.

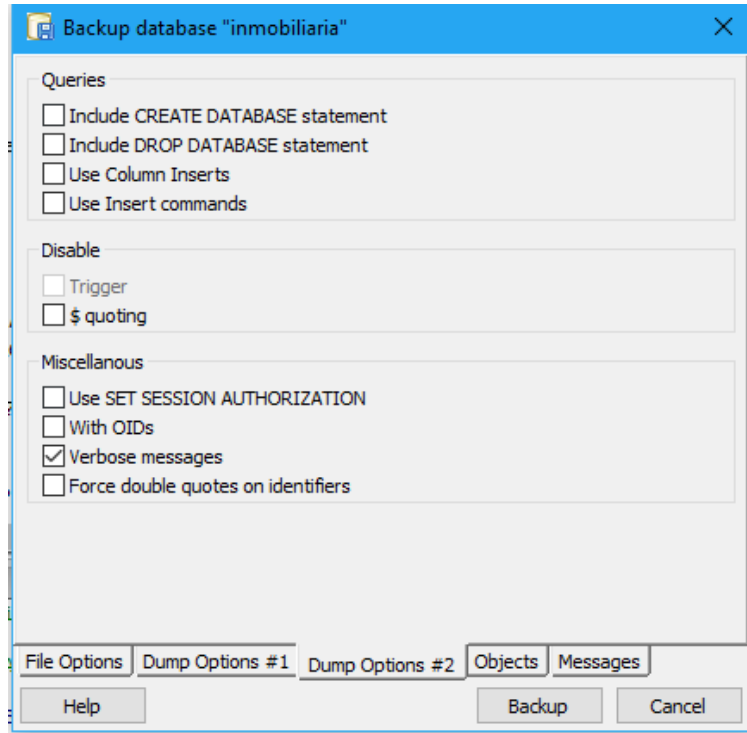
Ilustración 81. Pestaña "Dump Options #1".



Fuente: autores.

En la pestaña "Dump Options #2" se visualizan tres secciones que son: Queries, Disable y Miscellanous.

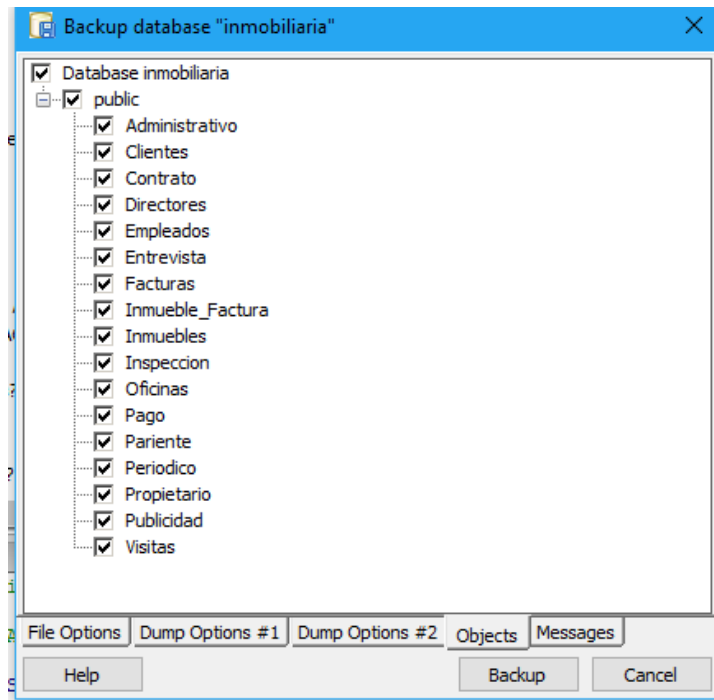
Ilustración 82. Pestaña "Dump Options #2".



Fuente: autores.

En la pestaña "Objects" se puede activar o desactivar las diversas tablas de la base de datos inmobiliaria.

Ilustración 83. Pestaña "Objects".



Fuente: autores.

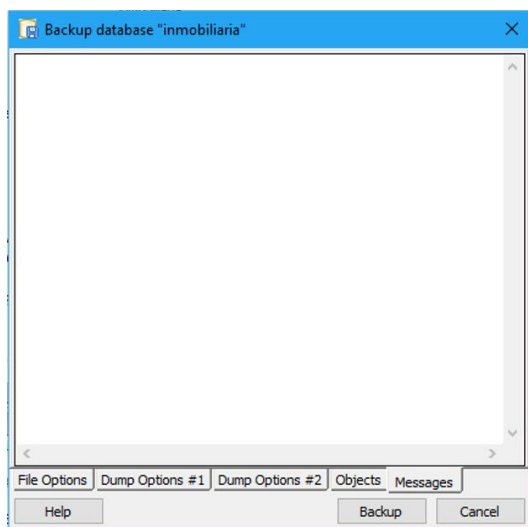
En la pestaña "Messages" "se puede visualizar cada uno de los procedimientos que se realizan para generar la copia de seguridad" (Copias de seguridad, restauración y recuperación de una bd).



EJEMPLO

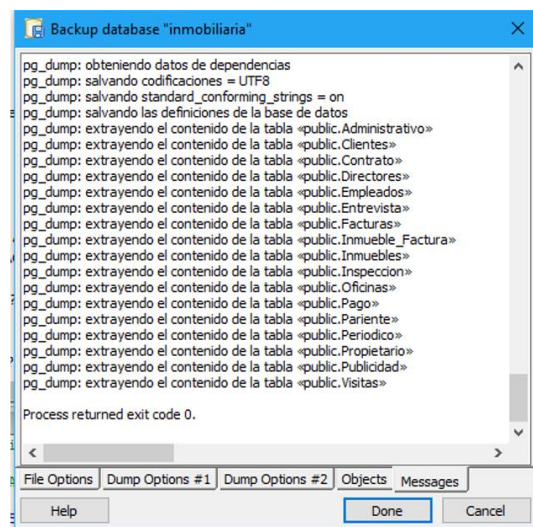
Si se hace clic en la pestaña de Messages cuando aún no se ha hecho clic en backup, no se muestra ninguna información, por lo tanto, se debe hacer clic en Backup y luego se observarán los mensajes, así como también el nombre del botón Backup cambiará por Done. Hacer un respaldo de la base de datos inmobiliaria utilizando el cmd de windows.

Ilustración 84. Antes del Backup.



Fuente: autores.

Ilustración 85. Después del Backup.



Fuente: autores.

Abrir el cmd y se ingresa la dirección de la carpeta bin, para poder ejecutar los comandos que se encuentran allí.

Ilustración 86. Dirección de la carpeta bin en cmd.

```
C:\> Símbolo del sistema
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\EZEQUIEL>cd C:\Program Files\PostgreSQL\9.5\bin
```

Fuente: autores.

Luego se ejecuta el siguiente comando: `pg_dump -U postgres -f D:\Backup\inmobiliaria.sql -F p -c -d inmobiliaria -E latin9`, para efectuar la copia de seguridad.

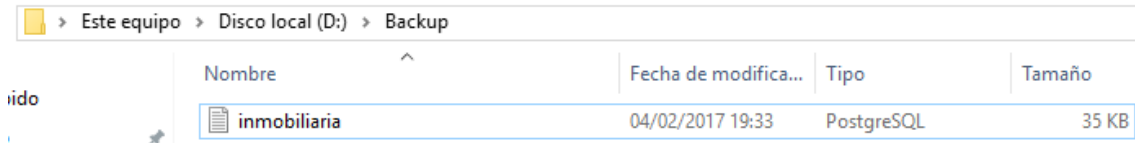
Ilustración 87. Comando pg_dump.

```
C:\Program Files\PostgreSQL\9.5\bin>pg_dump -U postgres -f D:\Backup\inmobiliaria.sql -F p -c -d inmobiliaria -E latin9
C:\Program Files\PostgreSQL\9.5\bin>
```

Fuente: autores.

Para verificar si los comandos funcionan se puede dirigir a la carpeta que se han indicado para el respaldo:

Ilustración 88. Verificación de Respaldo.



The screenshot shows a Windows Explorer window with the address bar set to 'Este equipo > Disco local (D:) > Backup'. The main area displays a table of files. The file 'inmobiliaria' is selected and highlighted. The table has columns for 'Nombre', 'Fecha de modifica...', 'Tipo', and 'Tamaño'.

Nombre	Fecha de modifica...	Tipo	Tamaño
inmobiliaria	04/02/2017 19:33	PostgreSQL	35 KB

Fuente: autores.

Para comprender la sintaxis con la que trabaja con el comando pg_dump de debe saber para qué sirve cada uno de sus parámetros, para ello escribir la siguiente instrucción: pg_dump –help y pulsar enter, con lo cual se observará la siguiente información:

Ilustración 89. Información comando pg_dump.

```
C:\Program Files\PostgreSQL\9.5\bin>pg_dump --help
pg_dump extrae una base de datos en formato de texto o en otros formatos.

Empleo:
  pg_dump [OPCIÓN]... [NOMBREDB]

Opciones generales:
  -f, --file=ARCHIVO      nombre del archivo o directorio de salida
  -F, --format=c|d|t|p   Formato del archivo de salida (c=personalizado,
                        d=directorio, t=tar, p=texto (por omisión))
  -j, --jobs=NUM         máximo de procesos paralelos para volcar
  -v, --verbose           modo verboso
  -V, --version          mostrar información de version y salir
  -Z, --compress=0-9     nivel de compresión para formatos comprimidos
  --lock-wait-timeout=SEGS espera a lo más SEGS segundos obtener un lock
  -?, --help            mostrar esta ayuda y salir

Opciones que controlan el contenido de la salida:
  -a, --data-only        extrae sólo los datos, no el esquema
  -b, --blobs            incluye objetos grandes en la extracción
  -c, --clean            tira (drop) la base de datos antes de crearla
  -C, --create           incluye órdenes para crear la base de datos
                        en la extracción
  -E, --encoding=CODIF  extrae los datos con la codificación CODIF
  -n, --schema=ESQUEMA  extrae sólo el esquema nombrado
  -N, --exclude-schema=ESQUEMA NO extrae el o los esquemas nombrados
  -o, --oids            incluye OIDs en la extracción
  -O, --no-owner        en formato de sólo texto, no reestablece
                        los dueños de los objetos
  -s, --schema-only     extrae sólo el esquema, no los datos
  -S, --superuser=NAME  superusuario a utilizar en el volcado de texto
  -t, --table=TABLE     extrae sólo la o las tablas nombradas
  -T, --exclude-table=TABLA NO extrae la(s) tabla(s) nombrada(s)
  -x, --no-privileges   no extrae los privilegios (grant/revoke)
  --binary-upgrade      sólo para uso de utilidades de upgrade
  --column-inserts      extrae los datos usando INSERT con nombres
                        de columnas
  --disable-dollar-quoting deshabilita el uso de «delimitadores de dólar»,
                        usa delimitadores de cadena estándares
  --disable-triggers    deshabilita los disparadores (triggers) durante el
                        restablecimiento de la extracción de sólo-datos
  --enable-row-security activa seguridad de filas (volcar sólo el
                        contenido al que el usuario tiene acceso)
  --exclude-table-data=TABLA NO extrae los datos de la(s) tabla(s) nombrada(s)
  --if-exists           usa IF EXISTS al eliminar objetos
  --inserts            extrae los datos usando INSERT, en vez de COPY
  --no-security-labels no volcar asignaciones de etiquetas de seguridad
  --no-synchronized-snapshots no usar snapshots sincronizados en trabajos
                        en paralelo
  --no-tablespaces     no volcar asignaciones de tablespace
  --no-unlogged-table-data no volcar datos de tablas unlogged
  --quote-all-identifiers entrecomilla todos los identificadores, incluso
                        si no son palabras clave
  --section=SECCIÓN    volcar la sección nombrada (pre-data, data,
                        post-data)
  --serializable-deferrable espera hasta que el respaldo pueda completarse
                        sin anomalías
  --snapshot=SNAPSHOT  use el snapshot dado para la extracción
  --use-set-session-authorization usa órdenes SESSION AUTHORIZATION en lugar de
                        ALTER OWNER para cambiar los dueño de los objetos

Opciones de conexión:
  -d, --dbname=NOMBRE   nombre de la base de datos que volcar
  -h, --host=ANFITRIÓN  anfitrión de la base de datos o
                        directorio del enchufe (socket)
  -p, --port=PUERTO     número del puerto de la base de datos
  -U, --username=USUARIO nombre de usuario con el cual conectarse
  -w, --no-password     nunca pedir una contraseña
  -W, --password        fuerza un prompt para la contraseña
                        (debería ser automático)
  --role=ROL            ejecuta SET ROLE antes del volcado
```

Fuente: autores.

23. HERRAMIENTA DE RESTAURACIÓN

Esta herramienta “llama a la herramienta homónima de PostgreSQL para restaurar los datos desde copias de seguridad (archivos de backup)”. (Padrón, 2013) Para utilizarla, se debe crear una base de datos.



EJEMPLO

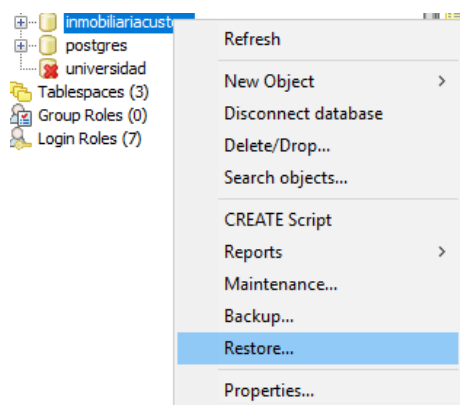
Hacer la restauración del archivo denominado inmobiliaria.backup, creado anteriormente con la herramienta resguardo, utilizando el pgAdmin III.

Ahora se creará la base de datos inmobiliariacustom:

```
CREATE DATABASE inmobiliariacustom
```

A continuación, se debe dirigir a la base de datos inmobiliariacustom y hacer clic derecho en la misma, y después seleccionar la opción de Restore.

Ilustración 90. Ejemplo Restauración.

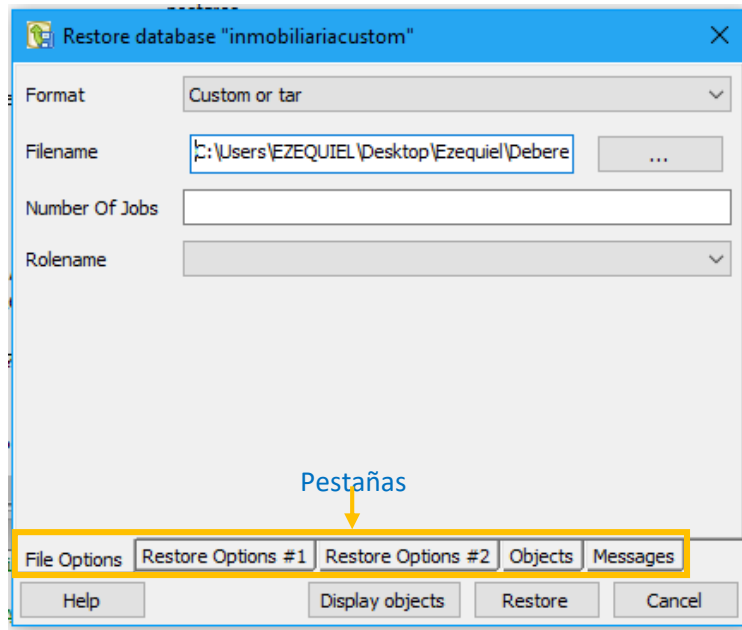


Fuente: autores.

En la parte inferior de la ventana que se abre se puede observar una serie de pestañas que permiten acceder a las diferentes opciones que proporciona PostgreSQL para realizar el procedimiento de restauración.

Observe que en el campo “Filename” se debe proporcionar la ruta “C:\Users\EZEQUIEL\Desktop\Ezequiel\Deberes\5to Semestre\Base de Datos II\Proyecto\inmobiliaria.backup” que es donde se guardó el archivo de la copia de seguridad denominado “inmobiliaria.backup”.

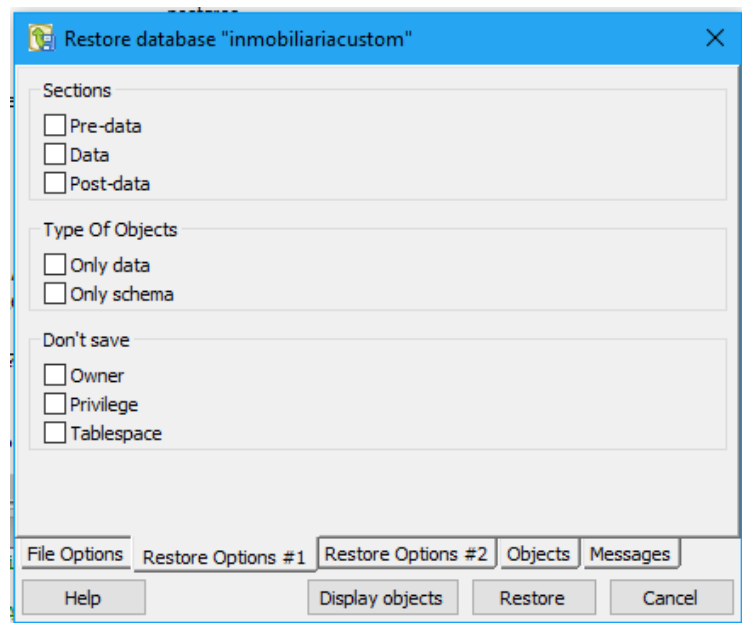
Ilustración 91. Pestañas de Restauración.



Fuente: autores.

En la pestaña “Restore Options #1” se tiene tres secciones, Sections, Tupe of Objects y Don’t sabe, con parámetros que resultan útiles cuando se realizan restauraciones parciales de la base de datos.

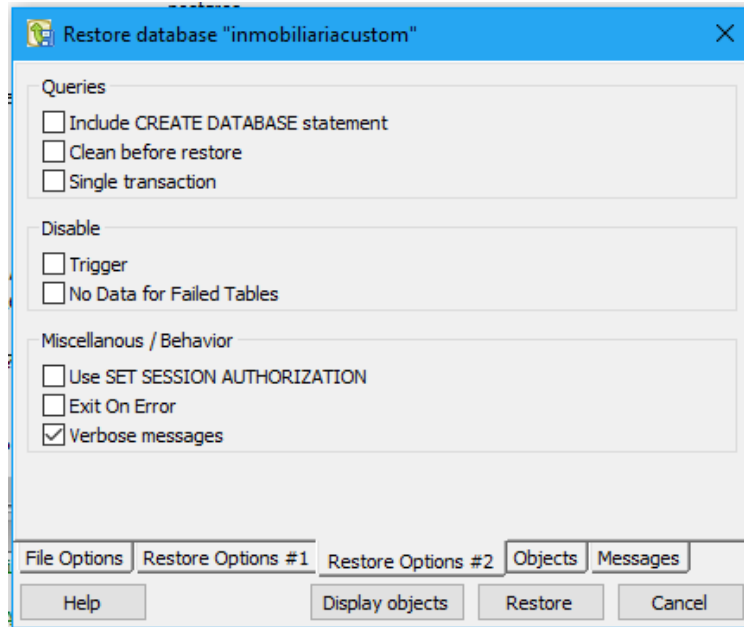
Ilustración 92. Pestaña “Restore Options #1”.



Fuente: autores.

En la pestaña “Restore Options #2” se tiene tres secciones “Queries, Disable” y “Miscellaneous” con parámetros que pueden ser activados y desactivados mediante el uso de los cuadros de chequeo, estos resultan útiles si se quiere desactivar los disparadores durante la restauración de la copia de seguridad.

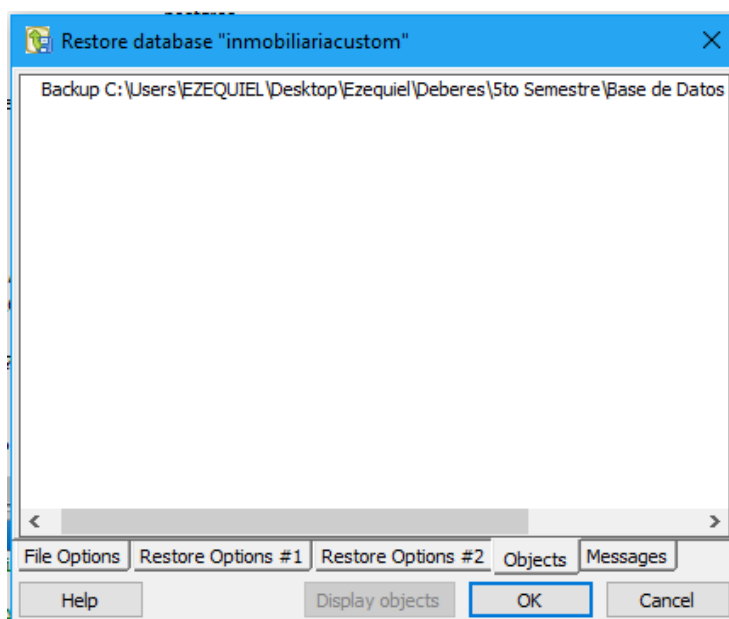
Ilustración 93. Pestaña “Restore Options #2”.



Fuente: autores.

En la pestaña “Objects” se visualiza la dirección del archivo .backup que se va a restaurar, para que estos se carguen en la pantalla se debe presionar el botón “Display objects”.

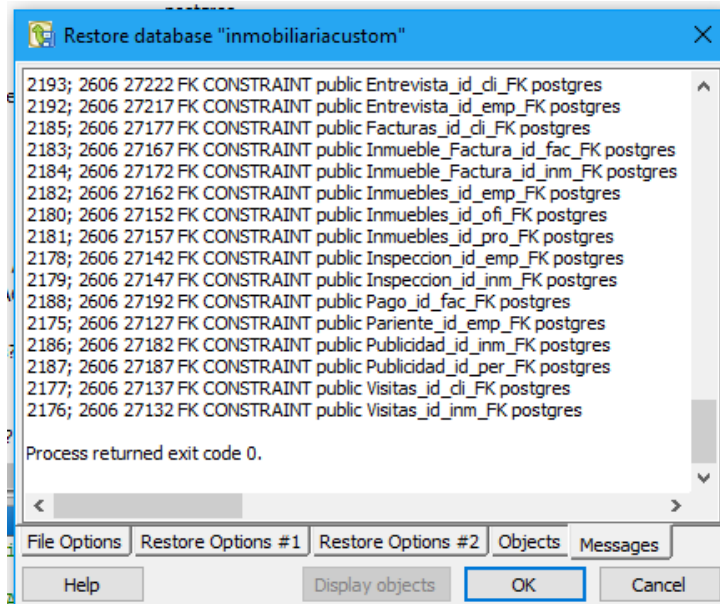
Ilustración 94. Pestaña “Objects”.



Fuente: autores.

En la pestaña de “Messages” se visualiza todos los procesos que se ejecutan mientras se realiza la restauración de la copia de seguridad. La información que suministra esta pestaña debe ser revisada para verificar que durante el proceso de restauración de la copia de seguridad no se generó ningún error. Un parámetro que ayuda a identificar si todo va bien, es que todos los procesos que se ejecutan durante la restauración de la copia de seguridad retornen el código “0”.

Ilustración 95. Pestaña “Messages”.



Fuente: autores.



EJEMPLO

Hacer la restauración del archivo inmobiliaria.sql utilizando el cmd de windows, el archivo mencionado se encuentra en el directorio D:\Backup.

Abrir el cmd y se ingresa la dirección de la carpeta bin, para poder ejecutar los comandos que se encuentran allí.

Ilustración 96. Restauración por cmd.

```
Microsoft Windows [Versión 10.0.14393]
(c) 2016 Microsoft Corporation. Todos los derechos reservados.

C:\Users\EZEQUIEL>cd C:\Program Files\PostgreSQL\9.5\bin

C:\Program Files\PostgreSQL\9.5\bin>
```

Fuente: autores.

Luego se crea la base de datos con el siguiente comando: `createdb -U postgres inmobiliariacopia`

Ilustración 97. Comando `createdb`.

```
C:\Program Files\PostgreSQL\9.5\bin>createdb -U postgres inmobiliariacopia
C:\Program Files\PostgreSQL\9.5\bin>
```

Fuente: autores.

Ahora se restaurará el archivo `inmobiliaria.sql` a la base de datos `inmobiliariacopia` con los siguientes comandos: `psql -U postgres inmobiliariacopia < D:\Backup\inmobiliaria.sql`

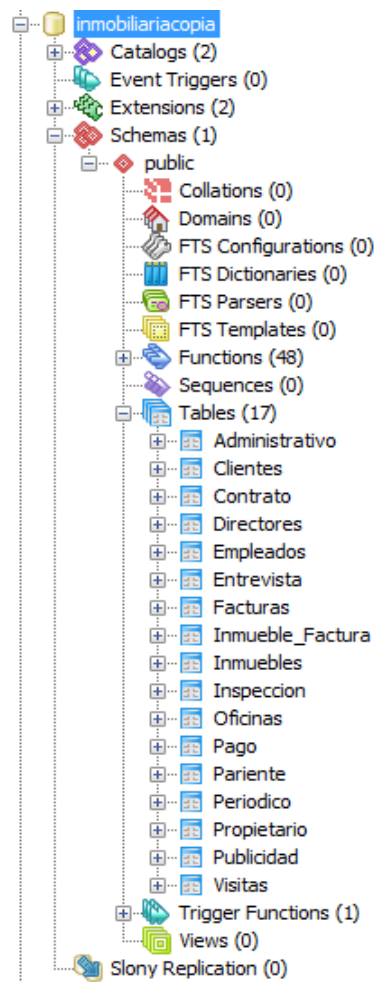
Ilustración 98. Comandos `psql`.

```
C:\Program Files\PostgreSQL\9.5\bin>psql -U postgres inmobiliariacopia < D:\Backup\inmobiliaria.sql
SET
```

Fuente: autores.

Al actualizar el explorador de objetos en el `pgAdmin III`, se podrá visualizar la base de datos `inmobiliariacopia`:

Ilustración 99. Explorador de objetos en `pgAdmin III`.



Fuente: autores.

24. HERRAMIENTA DE MANTENIMIENTO

Las herramientas de mantenimiento “ejecutan la tarea de reconstruir las estadísticas sobre la base de datos y tablas, limpiar los datos no utilizados y reorganizar los índices” (Padrón, 2013)

Las opciones que posee postgresQL para el mantenimiento de una base de datos son:

- Vacuum: “limpieza de las tablas muertas”. (Padrón, 2013) Es un proceso en el que se eliminan definitivamente las tuplas que han sido marcadas para borrar, y además se reorganizan los datos a nivel físico.

Se puede utilizar el comando vacuum con los siguientes parámetros:

- FULL: reclama el espacio en la base de datos que es ocupado por los registros marcados para ser eliminados.
- ANALYZE: actualiza las estadísticas para mejorar las consultas, sin bloquear las tablas.
- VERBOSE: imprime un informe detallado de la actividad vacuum para cada tabla
- TABLE: se puede especificar el nombre de la tabla para efectuar el vacuum, si no se indica a que tabla se hará el mantenimiento, de forma predeterminada se realiza en vacuum a todas las tablas de la base de datos con la que se está trabajando.
- COLUMN: el nombre de una columna para analizar, de forma predeterminada se toma en cuenta a todas las columnas.
- Analyze: “analiza los datos para calcular estadísticas” (Padrón, 2013).
- Reindex: “reorganiza los índices” (Padrón, 2013).

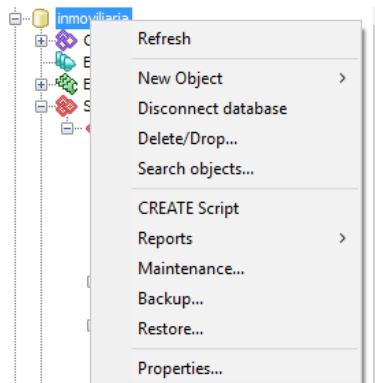


EJEMPLO

Utilizar el comando Vacuum en la interfaz de pgAdmin III para borrar los registros que han sido marcados para ser eliminados.

En el pgAdmin III de forma gráfica, dirigirse a la base de datos inmobiliario, hacer clic derecho y elegir la opción de mantenimiento:

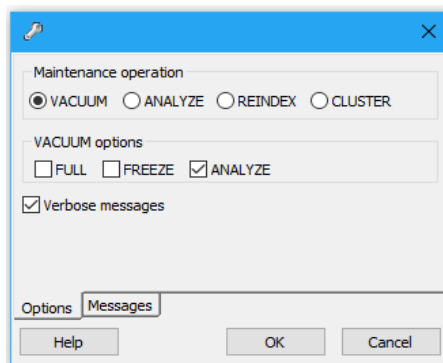
Ilustración 100. Opción Mantenimiento.



Fuente: autores.

Luego elegir las diferentes opciones para el mantenimiento:

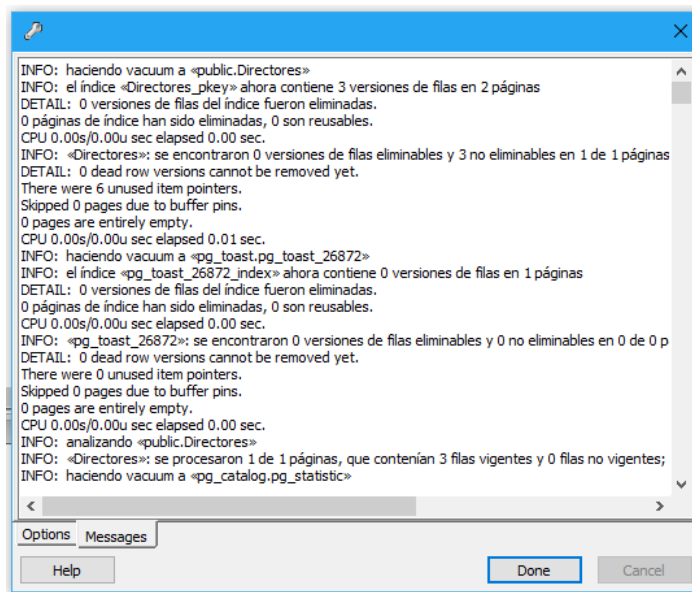
Ilustración 101. Opciones para el mantenimiento.



Fuente: autores.

Al pulsar en OK, se podrá visualizar los mensajes que la operación realizada:

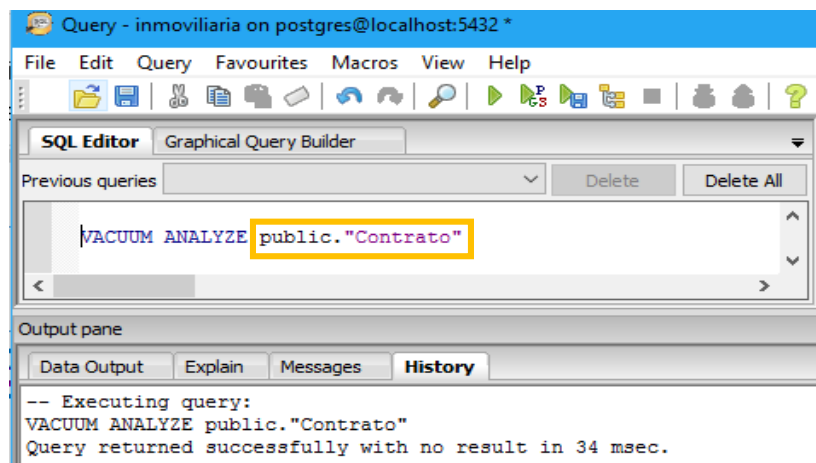
Ilustración 102. Operación realizada en el mantenimiento.



Fuente: autores.

En el pgAdmin III utilizando el editor de consultas, en la siguiente imagen se puede observar que se ha indica una tabla a la que se debe hacer el mantenimiento:

Ilustración 103. Editor de consultas para el mantenimiento.



Fuente: autores.

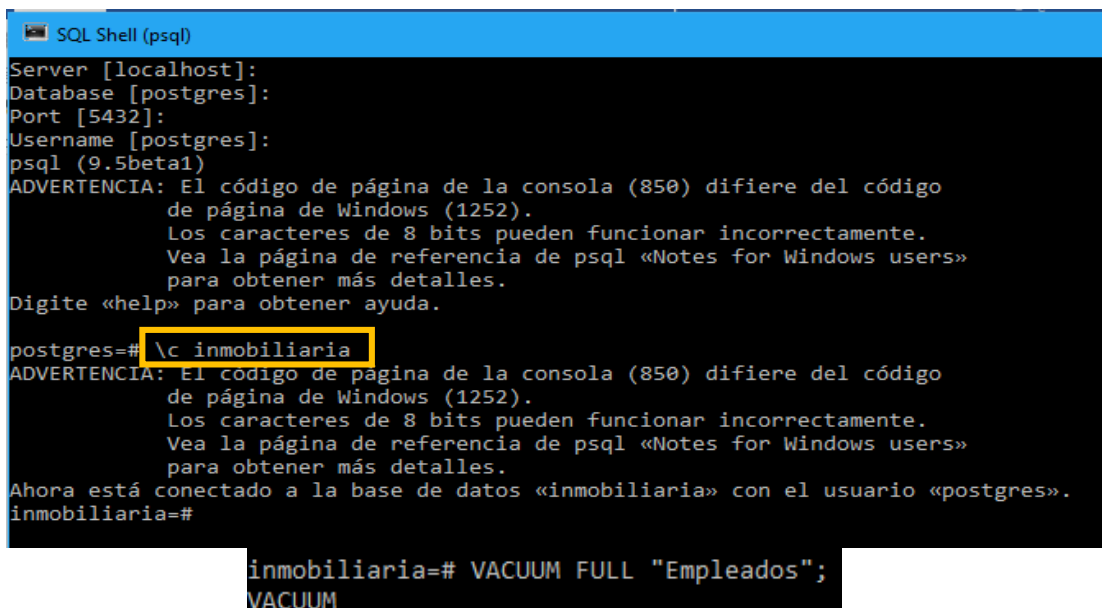


EJEMPLO

Utilizar el comando Vacuum en el SQL Shell (psql) para borrar los registros que han sido marcados para ser eliminados.

En el SQL Shell (psql), primero se debe acceder a la base de datos inmobiliaria, para esto se utiliza el comando `\c <nombre de la base de datos>`:

Ilustración 104. Acceso a la base datos a través del SQL Shell.



Fuente: autores.

Ejecutar el comando Vacuum con el parámetro FULL en la tabla empleados.

Si se desea observar el informe del Vacuum, se agrega el comando VERBOSE.

Ilustración 105. Informe del Vacuum.

```
inmobiliaria=# VACUUM FULL VERBOSE "Empleados";  
INFO: haciendo vacuum a «public.Empleados»  
INFO: «Empleados»: se encontraron 0 versiones eliminables de filas y 6 no eliminables en 1 páginas  
DETALLE: 0 versiones muertas de filas no pueden ser eliminadas aún.  
CPU 0.00s/0.00u sec elapsed 0.06 sec.  
VACUUM  
inmobiliaria=#
```

Fuente: autores.

25. BIBLIOGRAFÍA

- Bases de datos.* (s.f.). Obtenido de: http://volaya.github.io/librosig/chapters/Bases_datos.html.
- Copias de seguridad, restauración y recuperación de una bd.* (s.f.). Obtenido de: https://senaintro.blackboard.com/bbcswebdav/institution/semillas/217219_1_VIRTU-AL/OAAPs/OAAP4/aa6/lab_copiasseguridad/manuales/lab6-postgresql.pdf.
- García, A. (2007). *La web del programador*. Recuperado el 15 de 01 de 2017, de <http://www.lawebdelprogramador.com/cursos/archivos/ManualPracticoSQL.pdf>.
- Guía del Programador de PostgreSQL.* (s.f.). Obtenido de: <http://es.tldp.org/Postgresql-es/web/navegable/programmer/xplang.html>.
- Mangones, E. C. (s.f.). *Auditoría en un ambiente de base de datos*.
- Mannino, M. (s.f.). *Administración de Base de Datos*. México.
- Manual de usuario de postgresQL.* (s.f.). Obtenido de: <http://es.tldp.org/Postgresql-es/web/navegable/user/sql-createtable.html>.
- Micrisoft.* (s.f.). Obtenido de: [https://technet.microsoft.com/es-es/library/ms187518\(v=sql.105\).aspx](https://technet.microsoft.com/es-es/library/ms187518(v=sql.105).aspx).
- Moisset, D. (s.f.). *TutorialesProgramacionYa.* Obtenido de: <http://www.tutorialesprogramacionya.com/oracleya/temarios/descripcion.php?cod=261&punto=1&inicio>.
- Padrón, F. (2013). *PgAdmin III: administración de base de datos open source postgresql.* Obtenido de: <http://dspace.ucacue.edu.ec/bitstream/reducacue/5629/1/PGADMIN%20III%20Administrador%20de%20Base%20de%20Datos%20Open%20Source%20PostgreSQL.pdf>.
- PostgreSQL.* (s.f.). Obtenido de: <http://www.postgresql.org.es/node/352>.
- Rouse., M. (s.f.). Obtenido de: <http://searchdatacenter.techtarget.com/es/definicion/Base-de-datos>.
- Silberschatz, A. (2002). *Fundamento de bases de datos*. Madrid.
- Sobre PostgreSQL.* (s.f.). Obtenido de: http://www.postgresql.org.es/sobre_postgresql.
- SQL Básico.* (s.f.). Obtenido de: <http://biblio3.url.edu.gt/Libros/2011/SQL-b.pdf>.
- SQL INNER JOIN.* (s.f.). Obtenido de: <http://sql.11sql.com/sql-inner-join.htm>.
- U., F. (s.f.). *La revista informatica.* Obtenido de: <http://www.larevistainformatica.com/que-es-encryptacion-informatica.htm>.
- w3schools.com.* (s.f.). Obtenido de: http://www.w3schools.com/sql/sql_join_left.asp.
- Zea, M. (s.f.). *Diseño de bases de datos*. Machala.

Ingeniería y Tecnología

