Curso de Bases de Datos y PostgreSQL

Víctor Hugo Dorantes González[1]
Fernando Magariños Lamas[2]
José Neif Jury Fabre[3]

21 de abril de 2001

[1] vhdg@fciencias.unam.mx
[2] mancha@styx.net
[3] pepe@pepe.net.mx

Índice General

Capítulo 1

Qué es una Base de Datos

En rigor, una Base de Datos es el conjunto de datos almacenados con una estructura lógica. Es decir, tan importante como los datos, es la estructura conceptual con la que se relacionan entre ellos. En la práctica, podemos pensar esto como el conjunto de datos más los programas (o *software*) que hacen de ellos un conjunto consistente.

Si no tenemos los dos factores unidos, no podemos hablar de una base de datos, ya que ambos combinados dan la coherencia necesaria para poder trabajar con los datos de una manera sistemática.

1.1 Definiciones

Las siguientes son las definiciones que necesitamos manejar con claridad para comprender el resto de los conceptos en las bases de datos relacionales. Algunas son definiciones dadas a la ligera, que serán extendidas y formalizadas en las secciones correspondientes.

Tupla es una hilera o fila en una tabla.

Atributo es una columna en una tabla.

Dominio es el conjunto de valores de los cuales los atributos obtienen sus valores.

Llave es un atributo con una característica de relevancia para identificar la tupla.

Llave primaria es una llave con valores únicos, es decir, no ocurren más de una vez en el atributo.

Cardinalidad es el número de tuplas en una tabla.

Grado es el número de atributos en una tabla.

Relación una definición simple es que se corresponde con una tabla y en ocasiones es preferible pensarlo de esta manera. La definición canónica es que una relación es el producto cartesiano de dos o varios dominios.

Podemos también decir que un dominio es un conjunto de valores escalares del mismo tipo, dónde un valor escalar es la mínima unidad semántica de información en el sentido de que son valores atómicos.

Tabla base es una relación autónoma a diferencia de las *vistas* y las tablas intermedias construidas a partir de una consulta.

Vista es una relación *virtual*, que se construye a partir de tablas base o incluso otras vistas, formada por atributos de estas otras tablas de forma directa o como resultado de una consulta.

1.2 Estructura lógica *vs.* estructura física.

Es claro que la forma física como estén almacenados los datos es independiente del concepto que tengamos de ellos. Son el conjunto de programas que saben como traer, unir y mostrar los datos, así como aquellos encargados de almacenarlos, los que le dan coherencia al concepto Base de Datos.

Digamos que es como la diferencia entre harina, levadura, sal y agua por separado y una pieza de pan. Quién le dá coherencia a esa pieza de pan es el proceso que se sigue para elaborarlo.

Es importante conceptualizar esto, porque del diseño de la estructura lógica depende toda la funcionalidad del sistema. Almacenar datos en una base de datos aprovechando solamente la estructura física no ofrece, relativamente, ninguna ventaja. En cambio un buen diseño de acuerdo a la naturaleza de los datos y a la forma como serán explotados hace toda la diferencia.

Un gran problema es la inconsistencia de datos. Digamos que empleamos un archivo secuencial para almacenar la información de clientes. Supongamos que tenemos varios programas que utilizan esa información y que en un momento dado se pueden tener registros duplicados con atributos diferentes. Por ejemplo, una persona cambia de dirección y al no tener una estructura bien definida, no alteramos el registro, sino que lo damos de alta de nuevo con la nueva dirección. De esta manera, tendremos a la misma persona con dos datos diferentes y sin posibilidad de garantizar que todos los programas tendrán en cuenta que la dirección válida es la del segundo registro que aparece.

Puede ocurrir también que dos personas estén modificando simultáneamente atributos distintos del mismo registro. Sin un sistema de manejo de concurrencia, no podemos garantizar que ambos cambios permanezcan.

Bajo la misma suposición de uno o más archivos con la información y varios programas independientes que la explotan, es fácil ver que cualquier nueva explotación de la información implica un nuevo programa y que mantener un sistema como este conlleva toda la complejidad de mantener varios programas cuando se añade o elimina una columna a los registros.

Otro ejemplo, si tenemos un registro de personas donde almacenamos datos como: nombre, RFC, puesto, salario base, dirección, teléfono, fecha de nacimiento, gustos musicales, aficiones, nombre, fecha de nacimiento y ocupación del cónyugue, nombres y fechas de nacimiento de sus hijos, nombres de sus mascotas, autos que posee (con todas las características), etc; y frecuentemente sólo utilizaremos nombre, RFC, puesto y salario base para generar una nómina, esto implica que en cada corrida, recuperaremos información que no necesitamos, con el agravante de que tenemos que traer registros inmensos para utilizar sólo cuatro campos. El problema no es el almacenaje en disco, sino el tiempo desperdiciado en recuperar registros de tal magnitud.

Un diseño un poco más inteligente tendría dos tablas, una con los datos más frecuentemente empleados de cada persona y la otra con el resto de la información que se emplea quizá sólo para fines estadísticos o para enviar tarjetas de felicitación. Por supuesto, ambas tablas estarán relacionadas por una llave primaria común.

Si tenemos un sistema donde por un lado hacemos un abono y por otro un cargo en una operación que está relacionada, es de esperar que no ocurra el cargo si no puede ocurrir el abono, o viceversa. Es decir, debemos de tener transacciones atómicas. En este caso, la pareja de transacciones debe de ocurrir por completo o no debe de ocurrir en lo absoluto.

Finalmente, un terrible problema es la exposición de los datos. En muchos casos nos interesa que ciertas gentes tengan acceso sólo a parte de la información. Es de mal gusto que todos los empleados sepan cuál es el salario del director.

Capítulo 2

Qué es un Manejador de Bases de Datos.

De la lectura previa podemos deducir que el Manejador de Bases de Datos (DBM por sus siglas en inglés) facilita las funciones de:

- almacenar físicamente,
- garantizar consistencia,
- garantizar integridad,
- atomicidad transaccional,
- y manejar vistas a la información.

2.1 Sistemas de archivos

Algunos MBD implementan sus propios sistemas de archivos, manejando directamente particiones o discos completos. Esto se ha hecho principalmente para facilitar la portabilidad y hacer más eficiente esta parte. Sin embargo, PostgreSQL utiliza el sistema de archivos del sistema operativo huésped y en el caso de los derivados de Unix, esto ha mostrado ser eficiente a la vez que mantiene la portabilidad.

2.2 Índices

El empleo adecuado de índices en una relación acelera el acceso a la información, pero consume espacio considerable, es por esto que vale la pena hacer un análisis cuidadoso de cuáles atributos requieren ser indexados.

Capítulo 3

Álgebra y cálculo relacional

3.1 Concepto de relaciones

La definición dada en la sección 1.1, de relación es ambigüa y carente de sentido, pero es difícil hacerlo sin involucrar el formalismo matemático.

Se define una relación R sobre el conjunto de dominios $D_T = \{D_1, D_2, ..., D_n\}$ como la pareja $C = \{(A_1 : D_1), (A_2 : D_2), ..., (A_n : D_n) || D_j \in D_T, A_j \in D_j\}$ (llamado *cabecera*) y $\{(A_1 : v_{i1}), (A_2 : v_{i2}), \ldots, (A_n : v_{in}) || A_j \in D_j, v_{ij} \in D_j\}$ (llamado *cuerpo*).

3.2 Álgebra relacional

En este capítulo se presentan los fundamentos del modelo relacional, para más adelante mostrar los manejadores de bases de datos que se fundamentan en tal modelo. La idea es definir lo que es un modelo de datos, presentar el modelo relacional en términos de la definición anterior y posteriormente analizar a detalle los conceptos y definiciones del álgebra relacional.

3.2.1 Modelo de datos

De acuerdo a [Ullman1999]:

> *"Un modelo de datos es un sistema formal y abstracto que permite describir los datos de acuerdo con reglas y convenios predefinidos. Es formal pues los objetos del sistema se manipulan siguiendo reglas perfectamente definidas y utilizando exclusivamente los operadores definidos en el sistema, independientemente de lo que estos objetos y operadores puedan significar."*

Según Codd, en [Silberschatz]:
"Un modelo de datos es una combinación de tres componentes:

1) *una colección de estructuras de datos (los bloques constructores de cualquier base de datos que conforman el modelo);*

2) *una colección de operadores o reglas de inferencia, los cuales pueden ser aplicados a cualquier instancia de los tipos de datos listados en* (1)*, para consultar o derivar datos de cualquier parte de estas estructuras en cualquier combinación deseada;*

3) *una colección de reglas generales de integridad, las cuales explícita o implícitamente definen un conjunto de estados consistentes —estas reglas algunas veces son expresadas como reglas de insertar-actualizar-borrar."*

Un modelo de datos puede ser usado de las siguientes maneras:

i) como una herramienta para especificar los tipos de datos y la organización de los mismos que son permisibles en una base de datos específica;

ii) como una base para el desarrollo de una metodología general de diseño para las bases de datos;

iii) como una base para el desarrollo de familias de lenguajes de alto nivel para manipulación de consultas (*querys*) y datos;

iv) como el elemento clave en el diseño de la arquitectura de un manejador de bases de datos.

El primer modelo de datos desarrollado con toda la formalidad que esto implica fue el *modelo relacional*, en 1969, mucho antes incluso que los modelos jerárquicos y de red. A pesar de que los sistemas jerárquicos y de red como software para manejar bases de datos son previos al modelo relacional, no fue sino hasta 1973 que los modelos de tales sistemas fueron definidos, apenas unos cuantos años antes de que estos sistemas empezaran a caer en desuso.

3.3 El modelo relacional

Con base en las definiciones anterior sobre lo que es un modelo de datos, es posible analizar el modelo relacional en términos de los tres componentes ya mencionados: estructuras de datos, operadores y reglas de integridad.

3.3.1 Estructuras de datos

Las estructuras de datos que se manejan en el modelo relacional corresponden a los conceptos de relación, entidad, atributo y dominio, los cuales de introducen aquí intencionalmente:

Relación. Por una relación se entiende una colección o grupo de objetos que tienen en común un conjunto de características o atributos.

Entidad. Es una unidad de datos en una relación con un conjunto finito de atributos. Es también conocido como *n–ada*, a raíz de que consiste de *n–valores*, uno por cada atributo.

Atributo. También llamado *característica*, cada atributo de una relación tiene asociado un dominio en el cual toma sus valores.

Dominio. Es un conjunto de valores que puede tomar un atributo en una relación.

La notación más usual para denotar las relaciones en términos de estos conceptos es:

$$Relación = atributo_1, atributo_2, \ldots, atributo_n$$

por ejemplo

$$Alumnos = \left\{ \begin{array}{cccc} \mathbf{No\_Cta}, & \mathbf{Nombre}, & \mathbf{Ap\_Paterno}, & \mathbf{Ap\_Materno} \\ 8315637-7, & Víctor Hugo, & Dorantes, & González \\ 8105765-9, & Magnolia, & Avalos, & Vázquez \end{array} \right\}$$

Y para establecer la conexión entre un *atributo* y un *dominio* podemos usar la siguiente notación:

$$D_{atributo} = valor_1, valor_2, \ldots, valor_m$$

por ejemplo

$$D_{NoCta} = \{8315677 - 7, 8409695 - 9, \ldots, 8759452 - 6\}$$

Una forma de implementar las relaciones en una computadora, es a través de tablas de valores, de forma que se tienen las siguientes equivalencias: un atributo corresponde al encabezado de una columna, una *n–ada* equivale a un renglón de la tabla y por supuesto una relación equivale a la tabla misma. En capítulos posteriores, suponiendo natural esta equivalencia de términos, se utilizarán de forma indistinta.

3.3.2 Reglas de integridad

Los conceptos básicos de integridad en el modelo relacional son el de *llave primaria*, *llave foránea*, *valores nulos* y un par de *reglas de integridad*.

Una *llave primaria* es uno o un conjunto de atributos que permiten identificar a las *n–adas* de manera única en cualquier momento.

Una *llave foránea* de una relación es un atributo que hace *referencia* a una *llave primaria* de otra relación; esto da pie a que una relación pueda tener varias *llaves foráneas*.

Un *valor nulo* es un valor que está fuera de la definición de cualquier dominio el cual permite dejar el valor del atributo "*latente*", su uso es frecuente en las siguientes situaciones:

i) Cuando se crea una *n–ada* y no se conocen todos los valores de cada uno de los atributos.

ii) Cuando se agrega un atributo a una relación ya existente.

iii) Para no tomarse en cuenta al hacer cálculos numéricos.

Las dos *reglas de integridad* tienen que ver precisamente con los conceptos antes mencionados y son:

Integridad de Relaciones. Ningún atributo que forme parte de una *llave primaria* puede aceptar *valores nulos*.

Integridad Referencial. Al tener una relación Q con *llave primaria* A de dominio D y otra relación R con atributo A que no es llave primaria de R, entonces cualquier valor en el atributo A en R debe ser:

i) nulo, o

ii) un valor que esté en el atributo A de la llave primaria de una *n–ada* en la relación Q

3.3.3 Operadores

Los operadores del modelo relacional son de dos tipos: *operadores de actualización* y los *operadores del álgebra relacional*. Los operadores del *álgebra relacional* serán analizados en la sección 3.3.5, mientras tanto mostraremos los de *actualización*[1].

Las operaciones válidas para actualización de los valores de las *n–adas* son las de *borrar*, *agregar* o *modificar*. El manejo de las *llaves primarias* y *foráneas* incide directamente en procurar que no se violen las reglas de integridad, al determinar cómo han de manejarse los operadores de manera que al aplicar cualquiera de estas operaciones no se produzcan inconsistencias.
Las reglas son las siguientes:

Reglas para agregar. Al insertar una *n–ada* en una relación, el valor de un atributo que sea *llave foránea* puede ser *nulo*, o algún valor del atributo de la *llave primaria* en la relación correspondiente.

[1] Los operadores de actualización se basan en dos suposiciones: a) las relaciones se hayan contenidas en una "base de datos", dentro de la cual son modificadas, b) al cambiar un elemento de la relación sigue siendo "la misma", pero "cambiada"; es decir, pasó de un estado a otro, pero sigue siendo la misma relación.

Reglas para borrar. Si se tiene una *n–ada* en una relación R_1 con un atributo A_i como *llave primaria*, y otra relación R_2 que tiene ese mismo atributo A_i pero como *llave foránea*, tenemos 3 casos:

i) *Borrado restringido.* No se puede borrar la *n–ada* en la relación R_1 cuya *llave primaria* tenga un valor que en la relación R_2 exista como uno de los valores de la *llave foránea.*

ii) *Borrado en cascada.* Al borrar una *n–ada* en la relación R_1 con cierto valor en la *llave primaria*, se borrarán todas las *n–adas* en R_2 que tengan ese mismo valor en la *llave foránea.*

iii) *Borrado por nulificación.* Al borrar una *n–ada* en la relación R_1, a todas las *n–adas* con el mismo valor en la relación R_2 se les asigna un *valor nulo* en el atributo de la *llave foránea.*

Reglas para modificar. Tenemos dos opciones:

i) *Modificación en cascada.* Al modificar una *llave primaria* en R_1 se le cambian los valores correspondientes en la *llave foránea* de R_2.

ii) *Modificación por nulificación.* Al cambiar los valores de la *llave primaria* en R_1 a los correspondientes valores en la llave foránea de R_2 se les pone un *valor nulo.*

Los esquemas de nulificación, en cascada y restringido tienen una aplicación lógica en cuanto a cual de ellas utilizar, esto es, quien decide el esquema a utilizar es quien genera las relaciones y quien sabe cuales son las dependencias entre una relación o atributo con otros, ademas podemos pensar que por periodos o situaciones particulares podemos cambiar de uno a otro esquema.

Ademas la implementación del modelo relacional en un manejador de bases de datos no obedece al 100% con todo el modelo y en particular necesita uno ubicar cual de estos esquemas permite (si es que tiene alguno).

3.3.4 Relaciones.

Una *relación* R es una colección de dos partes, $R = \langle H, T \rangle$ con un *encabezado* (*Header*) H y un *cuerpo* T. El *encabezado* consiste de un conjunto de parejas ordenadas [2]

$$H = \{(A_1 : D_1), (A_2 : D_2), \ldots, (A_n : D_n)\}$$

de atributos y dominios tal que para cada atributo A_j se tiene una asociación con su correspondiente dominio D_j, con $j = 1, 2, \ldots, n$. Los atributos deben ser todos distintos; es decir, $A_i \neq A_j$ si $i \neq j$.

El *cuerpo* consiste de un conjunto de *n–adas*, donde cada *n–ada* se ve como un conjunto de parejas $(atributo : valor)$[3]

$$T_i = \{(A_{i1} : V_{i1}), (A_{i2} : V_{i2}), \ldots, (A_{in} : V_{in})\}$$

$(i = 1, 2, \ldots, m)$ donde m es el número de *n–adas* del conjunto, $A_{ij} = A_j$ y $V_{ij} \in D_j$; *i.e.* cada valor que toma un atributo A_j está contenido en el correspondiente dominio D_j.

A los valores m y n se les conoce como la *cardinalidad* y el *grado* de la relación respectivamente, la cardinalidad es de esperarse que varíe con el paso del tiempo, mientras que el grado no; en este sentido hay que remarcar que en contexto de una base de datos que contiene un conjunto de relaciones estos conceptos de cardinalidad y grado se aplican a cada estado de la relación con respecto a un cierto tiempo t. En estos términos, una relación de grado uno se le conoce como *unaria*, a una de grado dos se le conoce como *binaria*, etcétera.

[2] Se usan estos indices los cuales solo marcan una relación entre un atributo y su correspondiente dominio, pero no hay ninguna relación de orden entre las parejas.

[3] De nuevo es importante resaltar que solo existe una relación entre los valores que toma un atributo y el atributo mismo, pero ninguna relación de orden entre estos.

Es de resaltar el hecho de que estas definiciones se hacen en base a los conceptos matemáticos de la teoría de conjuntos y que las operaciones mismas de los conjuntos forman parte del álgebra relacional como se verá a continuación.

Propiedades de las relaciones

De la definición de relación se infieren las siguientes propiedades:

i) *No hay n–adas duplicadas*

Esta propiedad se obtiene del hecho de que el cuerpo de una relación es un conjunto (en el sentido matemático) y los conjuntos por definición no contienen elementos repetidos. De aquí podemos desprender un corolario importante: el conjunto total de atributos de la relación es siempre una *llave primaria.*

ii) *No hay orden en las n–adas*

De nuevo esta propiedad se desprende al observar que el cuerpo de la relación es un conjunto no ordenado; esto quiere decir que a ninguna *n–ada* se le puede asignar el título o nombre de la *primera n–ada*, segunda, o el número que sea de la relación y por supuesto tampoco existe el concepto de la *n–ada siguiente* en la relación misma[4].

iii) *No hay orden en los atributos*

Esta propiedad se sigue del hecho de que el encabezado de la relación está definido como un conjunto. Como es de esperarse, no existen los conceptos del *primer* ó *n–ésimo* atributo, ni del *siguiente* atributo. De manera analoga se puede manejar una relación que *controle* tal orden.

iv) *Todos los valores de los atributos son atómicos*

Esta propiedad debería decir: *todos los valores de los atributos sencillos son atómicos.* Esto es una consecuencia al hecho de que los dominios son sencillos —*i.e.* contienen valores atómicos solamente (si se tuviesen atributos compuestos, éstos siempre pueden verse como la concatenación de atributos sencillos).

Esta última propiedad implica que todas las relaciones están *normalizadas*[5] en lo que al *modelo relacional* concierne, de aquí que en el modelo relacional al hablar de una relación siempre se tiene en mente que es una relación normalizada. Hablaremos más al respecto en el capítulo 4.

3.3.5 Operadores del álgebra relacional

Para los ejemplos mostrados en los operadores utilizaremos las siguientes relaciones:

r:

A	*B*	*C*
a1	*b1*	*c1*
a1	*b2*	*c1*
a2	*b1*	*c2*

s:

A	*B*	*C*
a1	*b1*	*c1*
a2	*b2*	*c1*
a2	*b2*	*c2*

q:

D	*E*	*F*
d1	*e1*	*f1*
d2	*e2*	*f1*
d2	*e2*	*f2*

p:

D	*E*	*F*	*G*	*H*
d1	*e1*	*f1*	*g1*	*h2*
d2	*e2*	*f1*	*g2*	*h1*
d2	*e2*	*f2*	*g1*	*h1*

[4] Es importante señalar que el hecho de que se usen indices para enumerar los atributos o n–adas no preestablece un orden implícito entre estas, sino que es mera notación.

[5] El término "normalizadas" se refiere en este caso a la primera forma normal que definió Codd. Las formas normales son condiciones que se establecen sobre las relaciones, las cuales al diseñar un sistema y definir la estructura lógica de las relaciones se emplean para quitar de ellas problemas de redundancia y establecer de forma clara las dependencias funcionales entre los atributos en las relaciones. La definición original de Codd [Date1993] solo incluye tres formas normales las cuales pueden consultarse a detalle (junto con otras) en el capítulo 21 del mismo [Date1993].

- **Unión** Construye una relación consistente en todas las *n–adas* que forman parte de cada una de las dos relaciones especificadas.

  En términos abstractos podemos decir que este operador es una función que toma como argumentos un par de relaciones y da como resultado otra relación:

$$Unión: R \times R \longmapsto R$$

$$Unión(\langle H, R\rangle, \langle H, S\rangle) \longmapsto \langle H, R \cup S\rangle$$

  Las relaciones R y S deben tener el mismo encabezado, al igual que la relación resultado.

  Por ejemplo:

Unión (r, s):

A	B	C
$a1$	$b1$	$c1$
$a1$	$b2$	$c1$
$a2$	$b1$	$c2$
$a2$	$b2$	$c1$
$a2$	$b2$	$c2$

- **Intersección** Construye una relación consistente en aquellas *n–adas* que aparecen tanto en la primera como en la segunda relación dada.

$$Inter: R \times R \longmapsto R$$

$$Inter(\langle H, R\rangle, \langle H, S\rangle) \longmapsto \langle H, R \cap S\rangle$$

  Observamos que tanto las relaciones R y S como la relación resultante tienen el mismo encabezado.

  Por ejemplo:

Inter (r, s):

A	B	C
$a1$	$b1$	$c1$

- **Diferencia** Se construye una relación consistente en aquellas *n–adas* que pertenecen a la primera relación pero no a la segunda relación especificada.

$$Dif: R \times R \longmapsto R$$

$$Dif(\langle H, R\rangle, \langle H, S\rangle) \longmapsto \langle H, R - S\rangle$$

  De nuevo, el encabezado de las relaciones R y S, al igual que la relación resultante, tienen el mismo encabezado.

  Por ejemplo:

Dif (r, s):

A	B	C
$a1$	$b2$	$c1$
$a2$	$b1$	$c2$

- **Producto Cartesiano** Construye una relación consistente en todas las posibles combinaciones de *n–adas*.

  Este operador es una función que recibe una relación R con encabezado $\{(A_1 : D_1), (A_2 : D_2), \ldots, (A_n : D_n)\}$ y otra relación S con encabezado $\{(B_1 : D_1), (B_2 : D_2), \ldots, (B_m : D_m)\}$ para generar una relación que tenga encabezado $\{(A_1 : D_1), (A_2 : D_2), \ldots, (A_n : D_n), (B_1 : D_1), (B_2 : D_2), \ldots, (B_m : D_m)\}$[6]; *i.e.* la concatenación de los encabezados de R y S.

$$Cruz\colon R \times R \longmapsto R$$

$$Cruz(\langle H, R\rangle, \langle H', S\rangle) \longmapsto \langle (H, H'), R'\rangle$$

Por ejemplo:

Cruz (r, q):

A	B	C	D	E	F
$a1$	$b1$	$c1$	$d1$	$e1$	$f1$
$a1$	$b1$	$c1$	$d2$	$e2$	$f1$
$a1$	$b1$	$c1$	$d2$	$e2$	$f2$
$a1$	$b2$	$c1$	$d1$	$e1$	$f1$
$a1$	$b2$	$c1$	$d2$	$e2$	$f1$
$a1$	$b2$	$c1$	$d2$	$e2$	$f2$
$a2$	$b1$	$c2$	$d1$	$e1$	$f1$
$a2$	$b1$	$c2$	$d2$	$e2$	$f1$
$a2$	$b1$	$c2$	$d2$	$e2$	$f2$

- **Restricción** Extrae aquellas *n–adas* de una relación que satisfacen una condición especifica.

  Este operador es una función que toma una relación y una condición y devuelve una relación con todas aquellas *n–adas* que hayan cumplido con la condición.[7]:

$$Rest\colon R \times Condición \longmapsto R$$

$$Rest(\langle H, R\rangle, Condición) \longmapsto \langle H, R_{(Cond)}\rangle$$

Por ejemplo:

Rest (r, $B = b_1$):

A	B	C
$a1$	$b1$	$c1$
$a2$	$b1$	$c2$

[6] Es necesario que $A_i \neq B_j$, aunque siempre se puede renombrar los atributos identicos

[7] Por condición entenderemos una expresión formada por operadores lógicos y de comparación entre los atributos y valores dentro los dominios correspondientes.

- **Proyección** Extrae aquellos atributos especificados de una relación dada.

  Formalmente es una función que toma una relación que tiene un encabezado de n atributos y un conjunto de m atributos (que están dentro del encabezado de la relación dada), generando una relación con un encabezado con los m atributos dados.

$$Proy\text{: } R \times (atrib_1, atrib2, \ldots, atrib_m) \longmapsto R$$

$$Proy(\langle H, R\rangle, atrib_1, atrib2, \ldots, atrib_m) \longmapsto \langle H', R_{(atribs)}\rangle$$

  Por ejemplo:

Proy (r, {B, C}):

B	C
b1	*c1*
b2	*c1*
b1	*c2*

- **Theta-Unión (Join)** Construye una relación a partir de dos relaciones las cuales tienen conjuntos ajenos de atributos, a los cuales se les establece un condición lógica que permite unir las relaciones a través de estos atributos.

  Este operador es una función que toma dos relaciones y un conjunto de condiciones de comparación entre atributos de una y otra relación, tal condición sirve para establecer una conexión lógica entre las relaciones (de manera natural se supone que los atributos correspondientes estan sumergidos en los mismos dominios); de manera que la relación generada es una combinación de las *n–adas* de ambas relaciones pero que en los atributos que estan involucrados en la condición, cumplen con ésta.

$$\theta Unión\text{: } R_1 \times R_2 \times (Condición) \longmapsto R$$

$$\theta Unión(\langle H, R\rangle, \langle H', S\rangle, Condición)\text{:} \longmapsto \langle H'', R'S\rangle$$

  Por ejemplo, si renombramos como D, E y F los atributos A, B, C, de la relación s para cumplir con la condición de este operador. Es de observarse que si la relación R tiene n atributos y la relación S tiene m atributos, entonces la relación resultante tiene $n + m - j$ atributos, donde j es el número de atributos diferentes que estan involucrados en la condición.

  Ejemplo:

θ Unión (r, s, {A = D}):

A	B	C	E	F
a1	*b1*	*c1*	*b1*	*c1*
a1	*b2*	*c1*	*b1*	*c1*
a2	*b1*	*c2*	*b2*	*c1*
a2	*b1*	*c2*	*b2*	*c2*

- **División** Este operador toma dos relaciones y construye una relación consistente de todos los atributos de la primera relación que no estan en la segunda relación.

  Este operador es una función que toma dos relaciones, una R con encabezado $\{(A_1 : D_1), (A_2 : D_2), \ldots, (A_n : D_n), (B_{n+1} : D_{n+1}), \ldots, (B_m : D_m)\}$ y una S con encabezado $\{(A_1 : D_1), (A_2 :$

$D_2), \ldots, (A_n : D_n)\}$ y regresa una relación de grado $m-n$ con encabezado $\{(B_{n+1} : D_{n+1}), \ldots, (B_m : D_m)\}$.

$$Div\colon R \times R \longmapsto R$$

$$Div(\langle (HX, HY), R\rangle, \langle HX, S\rangle) \longmapsto \langle HY, R/S\rangle$$

Por ejemplo:

Div(p, q):

G	H
g1	h2
g2	h1
g1	h1

Como se puede apreciar, el resultado de cada operador es una relación. Esta es la llamada *propiedad de la cerradura* y resulta muy importante porque el resultado de un operador puede ser un argumento para otro operador; *i.e.* permite la composición de los operadores.

Capítulo 4

Normalización

En la sección 3.3.4 mencionamos la importancia de la normalización de las relaciones en el álgebra relacional. En este capítulo se presentarán las tres primeras formas normales, definidas por Codd, a profundidad y las formas cuarta y quinta de una manera sucinta.

Al modelar una base de datos, desearemos evitar puntos que crean confusión, duplicación de la información y por ende, un mal funcionamiento y exploración de la información. Entre las propiedades indeseables en un diseño de bases de datos tenemos:

- Redundancia en la información.
- Incapacidad de representar cierta información.
- Registrar información que no sea identificable.

4.1 Dependencia funcional

Dada una relación R, el atributo Y de R depende funcionalmente del atributo X de R si y sólo si cada valor de Y está asociado a cada valor de X. Ambos atributos pueden ser compuestos. Se denota Y depende funcionalmente de X, como $R.X \leftarrow R.Y$. Por ejemplo, en la relación de automóviles, las características del vehículo *dependen funcionalmente* de las placas o del número de registro, porque dadas unas placas, existe sólo un vehículo (con las características que lo describe) asociado a ellas.

Un contraejemplo, el atributo `color` no depende funcionalmente del atributo `numero_de_puertas`, o en todo caso sería una terrible coincidencia que únicamente los vehículos rojos tengan cinco puertas.

4.2 Primera, segunda y tercera formas normales

Utilizaremos como ejemplo las relaciones de *proveedores*, *mercancías* y *envíos*:

```
Proveedores(NumPro#, Nombre, Situacion, Ciudad)
Mercancias(NumPar#, Descripcion, Color, Peso, Ciudad)
Envio(NumPro#, NumPar#, Cantidad)
```

Las llaves primarias están marcadas por el carácter `#`.

4.2.1 Primera forma normal

Una relación está en primera forma normal (1FN) si y sólo si todos los dominios son atómicos. Un dominio es atómico si los elementos del dominio son indivisibles.

Es decir, no tenemos grupos de repetición o un conjunto de valores asociados repetidos asociados a una misma tupla.

4.2.2 Segunda forma normal

Una relación está en segunda forma normal (2FN) si y sólo si está en 1FN y todos los atributos que no sean llaves dependen por completo de llave primaria.

4.2.3 Tercera forma normal

Una relación están en tercera forma normal (3FN) si y sólo si están en 2FN y todos los atributos no llave dependen de manera no transitiva de la llave primaria. Se dice que existe una dependecia transitiva cuando tenemos el par de dependencias funcionales: $R.A \leftarrow R.B$ y $R.B \leftarrow R.C$, porque de ellas se sigue que $R.A \leftarrow R.C$.

4.3 Consideraciones acerca de velocidad de acceso, gasto de espacio y buen diseño

Digamos que tenemos una base de datos con los datos de ciudadanos para algún tipo de registro. A saber, los usuales: nombre(s), apellidos, dirección, teléfono, lugar y fecha de nacimiento, nivel socio económico estratificado, etc. y como dato adicional tenemos el RFC. Pero, en este caso, podríamos calcular la primera parte del RFC, es decir los primeros diez dígitos y almacenar sólo los últimos tres, que conforman la homoclave. Aún más, si contamos con el algoritmo empleado por la Secretaría de Hacienda y Crédito Público[1], podríamos incluso no almacenar los tres últimos carácteres.

Ahorraríamos mucho espacio, trece carácteres por persona, que si almacenamos un millón de personas, significa casi trece megabytes[2]. Pero debemos de tener en cuenta que esta ganancia nos pegaría en cuanto a tiempo de respuesta en caso de que el campo RFC sea empleado contínuamente.

Digamos que a la rapidísima función que *hackeamos* anoche y lo calcula, le toma tan sólo una centésima de segundo, es decir, cada segundo calculamos cien RFC. Cada consulta sobre toda la tabla que utilice el RFC tomaría del orden de 36 horas. El costo de los discos duros es realmente barato ¿no? Vale más tener tiempo libre para ir al cine que esperar tanto ¿no?.

Ahora veamos el caso contrario, el gasto en tiempo de acceso. Obviamente es más rápido tener juntos todos los atributos, es decir, tener en un sólo registro todos los campos que necesitaremos acerca del objeto. Entonces ¿por qué romper en varias tablas si podemos tenerlo todo en una sola?

Supongamos que tenemos la base de datos del Registro Estatal Vehicular de Cuévano (REVC) de, digamos, un millón de vehículos. Entre las restricciones del sistema, sabemos que una persona puede tener varios vehículos registrados a su nombre.

Como tenemos un número grande de combinaciones entre los factores de modelo de vehículo, marca, color, número de puertas, si está o no equipado, etc., probablemente esta supera al número de vehículos que tenemos registrados.

Entonces, ¿qué pasa si decidimos que el registro para cada auto también lleva el registro del dueño con todos los datos?

[1] Rechifla de fondo.

[2] En realidad, 13 MB son 13,631,488 bytes, algo mayor que trece millones.

Por el ejemplo anterior, extrapolando de tiempo de generación a tiempo de acceso, podemos llegar a la conclusión de que esto no es tan grave, porque a final de cuentas, lo más probable es que la mayoría de las personas sólo tengan un vehículo y poco probable que una persona tenga más de diez, por decir un número[3].

Sabemos que una vez al año haremos una corrida secuencial sobre el total de los registros para generar las ordenes de pago de impuestos de tenencia. Entonces, el tamaño de cada registro importa, porque no es lo mismo leer ocho registros de un kilobyte en cada paso que dos de cuatro kilobytes. Pero el resto del tiempo, las búsquedas sólo serán individuales. Por supuesto, tenemos indices, llegamos directo al registro buscado.

Ahora bien, al REVC se le encarga también llevar el registro de licencias de conducir y de infracciones. Comienzan los problemas. Obviamente la mayoría de los poseedores de una licencia, son también propietarios de vehículos y viceversa[4], de tal manera que entre la tabla de vehículos y la de licencias ya tenemos información duplicada. Y por supuesto, no tiene sentido almacenar en la tabla de infracciones el nombre y dirección del infractor, porque o bien el infractor tiene licencia o bien el dueño del vehículo es el responsable así que tenemos la manera de llegar hasta él por una de estas dos tablas.

Ahora bien, ¿qué pasa si pese a esto mantenemos la información de los ciudadanos en las dos tablas, de licencias y de control vehicular?

Un movimiento frecuente en vehículos es el de cambio de propietario. Por simplicidad diremos que la gran mayoría de los cambios son dentro del estado. Entonces si Perengano compra un auto nuevo y le vende el viejo a Sutano que da de baja el suyo, el REVC debe de copiar los datos de Perengano al nuevo vehículo, los de Sutano al nuevo y del vehículo viejo de Sutano, sólo examinamos que no adeude infracciones y lo pasamos a una tabla histórica. Pero estos cambios pueden llevar consigo errores de captura que nos dejan con una base de datos inconsistente, cuando puede ser que los datos anteriores fueran correctos.

Luego de esta pequeña disgresión esperamos que quede claro porque es importante evitar la redundancia y mantener la integridad de la información.

4.4 Gasto de espacio *vs.* velocidad de acceso

Supongamos que tenemos un archivo para un periódico. Estará compuesto de textos e imágenes (fotos y cartones). Tanto los textos como las imágenes serán de tamaño variable. Como veremos en la sección 14.1.1, PostgreSQL proveé de métodos para almacenar información grande de tamaño variable y lo hace bien. Pero de nuevo, tenemos que tomar en cuenta algunas consideraciones.

La primera es que el tamaño de bloque de PostgreSQL es de ocho kilobytes, lo cual quiere decir que si nuestro texto mide 8 193 bytes, usaremos dos bloques, uno para los primeros 8 192 bytes y el segundo para el byte restante. Esto puede sonar tremendista, pero las probabilidades de que entre miles de objetos (textos, fotos o cartones) sea significativa la incidencia de objetos de tamaño múltiplo de 8KB, es casi cero. Tendremos pues, un gran desperdicio de espacio. Por otro lado, con sistemas operativos modernos como Linux, podemos crear un sistema de archivos en un disco o partición aparte donde optimicemos el tamaño de bloque de tal manera que las pérdidas no sean tan grandes. Esto quiere decir que seguimos teniendo el mismo problema con el sistema de archivos, pero podemos modificarlo de acuerdo a nuestras necesidades particulares, mientras que con el manejador de bases de datos, una alteración similar le pega a las otras bases.

La segunda consideración es respecto a qué nos interesa de la información que almacenaremos en la base de datos. Por supuesto de las fotos y cartones sólo nos interesa hacer búsquedas sobre temas, involucrados, fechas de publicación, etc., es decir, meta información, no el contenido gráfico.

En el caso de los textos, necesitaremos hacer las mismas búsquedas metatemáticas y además sobre algunas palabras clave o incluso texto llano.

Además sabemos que las búsquedas serán discriminadas entre artículos, fotos y cartones. Es decir, la gente que busca las declaraciones hechas por el presidente en Mérida acerca de la política exterior de Madagascar, no le interesa la foto del momento en que lo dijo o el cartón alusivo a lo que dijo. De igual

[3] Olvidemos que existen las flotillas de empresas que pueden tener en el orden de los cientos de vehículos.

[4] Sin embargo podemos esperar relaciones de dos a uno en licencias contra vehículos.

manera, nos interesa la foto de un personaje, pero no simultáneamente el cartón particular del personaje en una situación.

Por otra parte, cada uno de los objetos tienen atributos diferentes, lo cual los hace disconexos en cuanto a información se refiere.

Entonces, el primer diseño que podemos mostrar es el siguiente:

1. cada uno de los objetos en una tabla independiente;

2. naturalmente fotos y cartones *no necesitamos* que estén en la base de datos, así que pueden quedar en archivos del sistema y con un atributo que nos indique cómo llegar hasta ellos;

3. en el caso de los textos, las búsquedas sobre campos textuales son brutalmente lentos, además de que no tiene sentido hacerlas si las vamos a guardar como BLOBs[5], así que para este caso, al momento de procesar los textos, extraemos una referencia a las palabras que ocurren en él —posiblemente eliminando las comunes— y que quedan almacenadas en una tabla anexa, con lo cuál también eliminamos la necesidad de tener los artículos en una tabla.

[5] *Binary Large Objects*, que se describen en la sección 14.1.1

Capítulo 5

Integridad relacional

En un momento dado, los valores de los datos en una base de datos son una representación de un fragmento de la realidad. Es decir, si tenemos una tabla con los atributos de personas y entre ellos el peso o la edad, estos no pueden ser negativos, porque en el mundo real, esto no es posible. Si añadimos una restricción de este tipo a una base de datos, estamos incluyéndole una *regla de integridad*. Por ejemplo, si tenemos una base de datos alumnos, profesores y cursos para una escuela o facultad, algunas reglas de integridad serían:

- Las claves de los alumnos son de la forma AL*aaaannnn* donde *aaaa* son los cuatro dígitos del año de ingreso y *nnnn* son cuatro dígitos que representan un número secuencial.
- Las claves de los profesores son de la forma AC*mmnn* donde *mm* es la clave del departamento al que está asociado y *nn* es un secuencial.
- Las claves de cursos son de la forma MA*mmnnaa* donde *mm* es la clave del departamento, *nn* es la clave de la materia y *aa* son los dos dígitos menos significativos del año.
- Un alumno no puede estar inscrito en más de cinco materias.
- Un maestro no puede dar más de tres materias.
- Un curso no puede tener menos de cinco alumnos ni más de doce.
- Un maestro no puede dar la misma materia más de dos semestres seguidos.
- Un alumno que no aprueba una materia en la segunda oportunidad será dado de baja.
- Los departamentos vienen de una determinada lista.
- Las materias tienen que existir en otra lista.
- Las calificaciones no pueden tomar valores fuera del rango [0..10].

Algunas de estas reglas son arbitrarias y para fines de ejemplificar el concepto y es inmediato notar que se aplican a tablas en específico.

Sin embargo, las bases de datos relacionales, tienen dos reglas *generales* de integridad que se aplican a las llaves primarias y a las llaves foráneas.

5.1 Llaves primarias

Definamos primero el concepto de *llave candidata*. Digamos que el atributo X de la relación T es llave candidata si cumple:

1. Unicidad. En cualquier momento no existen dos tuplas en R con el mismo valor X.
2. Minimalidad. Si X es una llave compuesta[1], al eliminar cualquiera de sus componentes perdemos la cualidad de unicidad.

Por supuesto, la combinación de todos los atributos es una llave candidata y pueden existir combinaciones de atributos que den llaves candidatas, e incluso con diferentes longitudes. Es decir, podemos tener una llave X_1 compuesta de tres atributos y otra llave X_2 de dos atributos. De esta manera, siempre es posible encontrar al menos una llave candidata.

De esto deducimos que como toda relación tiene al menos una llave candidata, entonces toda relación tiene una llave primaria. Como deducirla queda fuera de toda metodología no podemos dar una regla general. Habrá ocasiones en que incluso es necesario construir una llave primaria, como un número secuencial o el más conocido RFC. En el caso de Hacienda, es obvio que una llave primaria podría ser la combinación del nombre, domicilio y edad, pero esta sería una llave muy torpe. ¿Han pensado cuántos posibles RFC existen? ¿Cuál es la probabilidad de colisión en esta llave? Es un bonito ejercicio y queda al lector pensar porqué la SHCP se vió en la necesidad de añadir tres carácteres al RFC —los conocidos como homoclave— y las posibles maneras de generar dicha homoclave.

Otro ejemplo, para el hipotético Registro Estatal de Automóviles de Cuévano[2], el número de serie del vehículo en combinación con la marca y el modelo; la marca, el color y nombre del dueño[3]; y la matrícula, son llaves candidatas. Pero de entre estas tres llaves candidatas, nos basta con la matrícula porque cumple perfectamente con las dos condiciones y además no es compuesta, con el consecuente ahorro de espacio y tiempo de cómputo. Las otras dos pasan a ser *llaves alternas*.

La importancia de las llaves primarias se comprenderá mejor con la exposición de las llaves foráneas, pero además las llaves primarias constituyen el mecanismo de direccionamiento a nivel de tuplas en el modelo relacional. Es decir, es el único modo de acceder a una tupla específica. Por ejemplo, la consulta:

```
SELECT * FROM padron_vehicular WHERE placa = 'ABC123'
```

regresa una sola tupla mientras que las consultas

```
SELECT * FROM padron_vehicular WHERE nombre = 'José'
    AND apepat = 'Hernández' AND apemat = 'García'
```

y

```
SELECT * FROM padron_vehicular WHERE color_vehiculo = 'Rojo'
```

probablemente regresen una centena o más o de tuplas, aún en el caso de un estado pequeño como Cuévano.

En resúmen, de todas las llaves candidatas, la más adecuada para llave primaria es la llave candidata de menor grado.

[1] Es decir una llave que es la adición de dos o más campos.

[2] Ver *Las muertas* de Jorge Ibargüengoitia.

[3] En el caso de que una persona no decida comprar dos vehículos idénticos.

5.2 Reglas de integridad

Se definen dos reglas de integridad:

```
\item Integridad de las entidades. Ningún componente de la llave
primaria de una relación puede aceptar valores nulos.

\item Integridad referencial. La base de datos no debe contener
valores de llaves ajenas sin concordancia.
```

5.3 Llaves foráneas e integridad referencial

Una llave foránea es un atributo (puede ser compuesto) de una relación R_2 cuyos valores deben de concordar con los de una llave primaria de alguna relación R_1. Las relaciones R_1 y R_2 pueden incluso ser la misma.

La idea es que tengamos concordancia entre datos de dos relaciones. Por ejemplo, si en una tabla de facturación tenemos un atributo que es la llave de referencia a clientes, éstos necesariamente deben de existir, en la tabla de clientes. En este caso la llave primaria de la tabla de clientes, es la llave ajena en la tabla de facturación y no podemos emitir una factura a alguien que no es un cliente.

A esta correspondencia se le conoce como *integridad referencial*.

Capítulo 6

Diseño de Bases de Datos

Son muchas las consideraciones a tomar en cuenta al momento de hacer el diseño de la base de datos, quizá las más fuertes sean:

- la velocidad de acceso,
- el tamaño de la información,
- el tipo de la información,
- facilidad de acceso a la información,
- Facilidad para extraer la información requerida,
- el comportamiento del manejador de bases de datos con cada tipo de información.

6.0.1 Criterios para la creación de índices

Cuando tenemos búsquedas frecuentes sobre un atributo —lo mismo aplica para un conjunto de atributos— lo ideal es tener esa columna indexada para obtener más rápido la información.

PostgreSQL antes de crear el índice, hace un poco de heurística para determinar que tipo de estructura de datos utilizará y que sea óptima para esa columna —o conjunto de columnas—.

En realidad los criterios a seguir son muy simples, si la información de la base de datos se consultará con frecuencia sobre un atributo, ese es el candidato ideal para ser indexado.

En el caso de una llave primaria, PostgreSQL automáticamente crea el índice:

```
dbarc=> create table borrame (a text primary key, b int);
NOTICE: CREATE TABLE/PRIMARY KEY will create implicit
index 'borrame_pkey' for table 'borrame'
CREATE
```

Como mencionamos en la sección 4.4, almacenar grandes textos en una base de datos es algo completamente imbécil, y peor en el caso de crear índices sobre campos de texto. Si por alguna retorcida razón, se nos ocurre guardar una descripción textual de personas, indexar este campo no tendría ningún sentido, porque tendríamos que hacer la búsqueda exactamente con el mismo contenido del campo:

```
INSERT INTO persona (generales) VALUES ('Es una persona a todo dar, le
gustan los animalitos y los niños pequeños, come de todo.')
WHERE id = 10;
SELECT * FROM persona WHERE generales = 'Es una persona a todo dar, le
gustan los animalitos y los niños pequeños, come de todo.';
```

¿Queda claro el concepto? Pero bueno, puede darse el caso en que *realmente* sea necesario tener un campo de este estilo y para el cuál valga la pena crear un índice, pero en ese caso, con los fuentes de PostgreSQL se distribuye una función extra, `fulltextindex` como parte del directorio `contrib`, que implementa esto de una manera muy interesante. Lo que hace es extraer las palabras y las referencias del registro a una tabla auxiliar, donde las palabras están indexadas, de tal manera que cuando se quiere hacer una búsqueda sobre el campo de texto, se buscan las palabras en esa tabla y lo que regresa son las referencias a las tuplas donde aparecen. El ahorro de espacio y la ganancia en velocidad son evidentes.

Para el caso en particular de tener que mantener muchos archivos de texto, es más recomendable usar indexadores de texto que no sean directos[1], como *glimpse*, de Udi Manber. Glimpse utiliza algoritmos sofisticados para mantener los indices pequeños, de tal manera que los indices representan del 15 al 25% del tamaño del texto indexado. En el caso de los indices directos, el tamaño de estos con frecuencia es mayor que el tamaño del texto indexado.

Por último, la mejor manera de saber en que atributos conviene tener indices es correr un `vacuum verbose analyze` sobre la base de datos, para actualizar las estadísticas de las tablas y después utilizar la instrucción `explain` sobre las consultas que sabemos serán las más frecuentes:

```
discos=> VACUUM VERBOSE ANALYZE discos;
NOTICE:  --Relation discos--
NOTICE:  Pages 19: Changed 0, Reapped 19, Empty 0, New 0; Tup 1390:
Vac 0, Keep/VTL 0/0, Crash 0, UnUsed 144, MinLen 76, MaxLen 168;
Re-using: Free/Avail. Space 4772/288; EndEmpty/Avail. Pages 0/3.
Elapsed 0/0 sec.
NOTICE:  Index xtitulo: Pages 23; Tuples 1390: Deleted 0. Elapsed 0/0 sec.
NOTICE:  Index xmedio: Pages 9; Tuples 1390: Deleted 0. Elapsed 0/0 sec.
NOTICE:  Index xautor: Pages 18; Tuples 1390: Deleted 0. Elapsed 0/0 sec.
NOTICE:  Rel discos: Pages: 19 --> 19; Tuple(s) moved: 0. Elapsed 0/0 sec.
VACUUM
discos=> EXPLAIN SELECT autor FROM discos WHERE autor = 'Bob Dylan';
NOTICE:  QUERY PLAN:
Index Scan using xautor on discos  (cost=2.79 rows=16 width=12)
EXPLAIN
discos=> EXPLAIN SELECT * FROM discos WHERE categoria = 1;
NOTICE:  QUERY PLAN:
Seq Scan on discos  (cost=64.87 rows=1390 width=102)
EXPLAIN
```

De inmediato podemos ver que el costo cuando hacemos un `select` sobre un campo indexado es menor que cuando lo hacemos sobre un campo no indexado, pero es de notar también que el sistema nos reporta que en el primer caso la búsqueda es sobre 16 tuplas mientras que en el segundo caso la búsqueda es sobre el total de las tuplas. Es decir, como era de esperarse, en el segundo caso hace un barrido secuencial sobre toda la tabla, mientras que en el primer caso, en el índice "barre" 16 entradas hasta encontrar al autor buscado.

Por supuesto, si vamos a permitir búsquedas sobre cualquier atributo lo primero que se nos ocurrirá es indexar cada uno de ellos, pero esto consume espacio brutalmente:

```
-rw-------   1 postgres postgres   155648 Jul  2 03:27 discos
-rw-------   1 postgres postgres     8192 Dec 24  1999 sqcat
-rw-------   1 postgres postgres     8192 Dec 24  1999 sqlug
-rw-------   1 postgres postgres   147456 Jun  7 05:43 xautor
-rw-------   1 postgres postgres    73728 Jun  7 05:43 xmedio
-rw-------   1 postgres postgres   188416 Jun  7 05:43 xtitulo
```

[1] Se dice que un índice es directo cuando a cada entrada en el texto corresponde una entrada en el índice.

en efecto, el índice para autores es casi del mismo tamaño que la tabla que contiene ese campo y el índice por título es incluso mayor[2]. Si el espacio en disco no es importante, no hay problema, pero si el acceso a disco es lento, entonces sí es una consideración a tener en cuenta.

[2] `xmedio` es el índice sobre un atributo de dos carácteres.

Capítulo 7

El lenguaje SQL

7.1 CREATE

7.1.1 CREATE AGGREGATE

Sintáxis:

```
CREATE AGGREGATE agg_name [AS] (BASETYPE = data_type,
[SFUNC1 = sfunc_1, STYPE1 = sfunc1_return_type]
[SFUNC2 = sfunc_2, STYPE2 = sfunc2_return_type]
[,FINALFUNC = final-function]
[,INITCOND1 = initial-cond1][,INITCOND2 = initial-cond2]);
```

7.1.2 CREATE DATABASE

Sintáxis:

```
CREATE DATABASE dbname [WITH LOCATION = 'dbpath']
```

7.1.3 CREATE FUNCTION

Sintáxis:

```
CREATE FUNCTION function_name ([type1, ...typeN]) RETURNS return_type
AS 'object_filename'|'sql-queries'|'builtin_function_name'
LANGUAGE 'c'|'sql'|'internal';
```

7.1.4 CREATE INDEX

Sintáxis:

```
        CREATE [UNIQUE] INDEX indexname ON class_name [USING access_method]
( attr1 [type_class1], ...attrN | funcname(attr1, ...) [type_class] );
```

7.1.5 CREATE OPERATOR

Sintáxis:

```
        CREATE OPERATOR operator_name (
        [LEFTARG = type1][,RIGHTARG = type2]
        ,PROCEDURE = func_name,
        [,COMMUTATOR = com_op][,NEGATOR = neg_op]
        [,RESTRICT = res_proc][,JOIN = join_proc][,HASHES]
        [,SORT1 = left_sort_op][,SORT2 = right_sort_op]);
```

7.1.6 CREATE RULE

Sintáxis:

```
        CREATE RULE rule_name AS ON
        { SELECT | UPDATE | DELETE | INSERT }
        TO object [WHERE qual]
        DO [INSTEAD] [action|NOTHING|[actions]];
```

7.1.7 CREATE SEQUENCE

Sintáxis:

```
        CREATE SEQUENCE sequence_name
        [INCREMENT number]
        [START number]
        [MINVALUE number]
        [MAXVALUE number]
        [CACHE number]
        [CYCLE];
```

7.1.8 CREATE TABLE

Sintáxis:

```
        CREATE [TEMP] TABLE class_name
        (attr1 type1 [DEFAULT expression] [NOT NULL], ...attrN
        [[CONSTRAINT name] CHECK condition1, ...conditionN] )
        [INHERITS (class_name1, ...class_nameN)];
```

7.1.9 CREATE TRIGGER

Sintáxis:

```
CREATE TRIGGER trigger_name AFTER|BEFORE event1 [OR event2 [OR event3] ]
ON class_name FOR EACH ROW|STATEMENT
EXECUTE PROCEDURE func_name ([arguments])
```

Donde el acto puede ser `INSERT`, `DELETE` o `UPDATE`.

7.1.10 CREATE TYPE

Sintáxis:

```
CREATE TYPE typename (
INTERNALLENGTH = (number|VARIABLE),
[EXTERNALLENGTH = (number|VARIABLE),]
INPUT = input_function, OUTPUT = output_function
[,ELEMENT = typename][,DELIMITER = character][,DEFAULT='<string>']
[,SEND = send_function][,RECEIVE = receive_function][,PASSEDBYVALUE]);
```

7.1.11 CREATE USER

Sintáxis:

```
CREATE USER user_name
[WITH PASSWORD password]
[CREATEDB | NOCREATEDB]
[CREATEUSER | NOCREATEUSER]
[IN GROUP group1, ...groupN]
[VALID UNTIL 'abstime'];
```

7.1.12 CREATE VIEW

Sintáxis:

```
CREATE VIEW view_name AS
SELECT [DISTINCT [ON attrN]]
expr1 [AS attr1], ...exprN
[FROM from_list]
[WHERE qual]
[GROUP BY group_list];
```

Capítulo 8

Muchos comandos SQL mas...

8.1 Alterar...

8.1.1 ALTER GROUP

Descripción: Add users to a group, remove users from a group

```
ALTER GROUP name ADD USER username [, ... ]
ALTER GROUP name DROP USER username [, ... ]
```

8.1.2 ALTER TABLE

Descripción: Modifies table properties

```
ALTER TABLE table [ * ]
    ADD [ COLUMN ] column type
ALTER TABLE table [ * ]
    ALTER [ COLUMN ] column { SET DEFAULT value | DROP DEFAULT }
ALTER TABLE table [ * ]
    RENAME [ COLUMN ] column TO newcolumn
ALTER TABLE table
    RENAME TO newtable
ALTER TABLE table
    ADD table constraint definition
```

8.1.3 ALTER USER

Descripción: Modifies user account information

```
ALTER USER username
    [ WITH PASSWORD 'password' ]
    [ CREATEDB | NOCREATEDB ] [ CREATEUSER | NOCREATEUSER ]
    [ VALID UNTIL 'abstime' ]
```

8.1.4 CLOSE

Descripción: Close a cursor

```
CLOSE cursor
```

8.1.5 CLUSTER

Descripción: Gives storage clustering advice to the server

```
CLUSTER indexname ON table
```

8.1.6 COMMENT

Descripción: Add comment to an object

```
COMMENT ON
[
  [ DATABASE | INDEX | RULE | SEQUENCE | TABLE | TYPE | VIEW ]
  object_name |
  COLUMN table_name.column_name|
  AGGREGATE agg_name agg_type|
  FUNCTION func_name (arg1, arg2, ...)|
  OPERATOR op (leftoperand_type rightoperand_type) |
  TRIGGER trigger_name ON table_name
] IS 'text'
```

8.1.7 COPY

Descripción: Copies data between files and tables

```
COPY [ BINARY ] table [ WITH OIDS ]
    FROM { 'filename' | stdin }
    [ [USING] DELIMITERS 'delimiter' ]
    [ WITH NULL AS 'null string' ]
COPY [ BINARY ] table [ WITH OIDS ]
    TO { 'filename' | stdout }
    [ [USING] DELIMITERS 'delimiter' ]
    [ WITH NULL AS 'null string' ]
```

8.2 Crear

8.2.1 CREATE AGGREGATE

Descripción: Defines a new aggregate function

```
CREATE AGGREGATE name ( BASETYPE = input_data_type
    [ , SFUNC1 = sfunc1, STYPE1 = state1_type ]
    [ , SFUNC2 = sfunc2, STYPE2 = state2_type ]
    [ , FINALFUNC = ffunc ]
    [ , INITCOND1 = initial_condition1 ]
    [ , INITCOND2 = initial_condition2 ] )
```

8.2.2 CREATE CONSTRAINT TRIGGER

Descripción: Create a trigger to support a constraint

```
CREATE CONSTRAINT TRIGGER name
    AFTER events ON
    relation constraint attributes
    FOR EACH ROW EXECUTE PROCEDURE func '(' args ')'
```

8.2.3 CREATE DATABASE

Descripción: Creates a new database

```
CREATE DATABASE name [ WITH LOCATION = 'dbpath' ]
```

8.2.4 CREATE FUNCTION

Descripción: Defines a new function

```
CREATE FUNCTION name ( [ ftype [, ...] ] )
    RETURNS rtype
    AS definition
    LANGUAGE 'langname'
    [ WITH ( attribute [, ...] ) ]
CREATE FUNCTION name ( [ ftype [, ...] ] )
    RETURNS rtype
    AS obj_file , link_symbol
    LANGUAGE 'C'
    [ WITH ( attribute [, ...] ) ]
```

8.2.5 CREATE GROUP

Descripción: Creates a new group

```
CREATE GROUP name
    [ WITH
     [ SYSID gid ]
     [ USER  username [, ...] ] ]
```

8.2.6 CREATE INDEX

Descripción: Constructs a secondary index

```
CREATE [ UNIQUE ] INDEX index_name ON table
    [ USING acc_name ] ( column [ ops_name ] [, ...] )
CREATE [ UNIQUE ] INDEX index_name ON table
    [ USING acc_name ] ( func_name( column [, ... ]) [ ops_name ] )
```

8.2.7 CREATE LANGUAGE

Descripción: Defines a new language for functions

```
CREATE [ TRUSTED ] PROCEDURAL LANGUAGE 'langname'
    HANDLER call_handler
    LANCOMPILER 'comment'
```

8.2.8 CREATE OPERATOR

Descripción: Defines a new user operator

```
CREATE OPERATOR name ( PROCEDURE = func_name
     [, LEFTARG = type1 ] [, RIGHTARG = type2 ]
     [, COMMUTATOR = com_op ] [, NEGATOR = neg_op ]
     [, RESTRICT = res_proc ] [, JOIN = join_proc ]
     [, HASHES ] [, SORT1 = left_sort_op ] [, SORT2 = right_sort_op ] )
```

8.2.9 CREATE RULE

Descripción: Defines a new rule

```
CREATE RULE name AS ON event
    TO object [ WHERE condition ]
    DO [ INSTEAD ] [ action | NOTHING ]
```

8.2.10 CREATE SEQUENCE

Descripción: Creates a new sequence number generator

```
CREATE SEQUENCE seqname [ INCREMENT increment ]
    [ MINVALUE minvalue ] [ MAXVALUE maxvalue ]
    [ START start ] [ CACHE cache ] [ CYCLE ]
```

8.2.11 CREATE TABLE

Descripción: Creates a new table

```
CREATE [ TEMPORARY | TEMP ] TABLE table (
    column type
    [ NULL | NOT NULL ] [ UNIQUE ] [ DEFAULT value ]
    [column_constraint_clause | PRIMARY KEY } [ ... ] ]
    [, ... ]
    [, PRIMARY KEY ( column [, ...] ) ]
    [, CHECK ( condition ) ]
    [, table_constraint_clause ]
    ) [ INHERITS ( inherited_table [, ...] ) ]
```

8.2.12 CREATE TABLE AS

Descripción: Creates a new table

```
CREATE TABLE table [ (column [, ...] ) ]
     AS select_clause
```

8.2.13 CREATE TRIGGER

Descripción: Creates a new trigger

```
CREATE TRIGGER name { BEFORE | AFTER } { event [OR ...] }
    ON table FOR EACH { ROW | STATEMENT }
    EXECUTE PROCEDURE func ( arguments )
```

8.2.14 CREATE TYPE

Descripción: Defines a new base data type

```
CREATE TYPE typename ( INPUT = input_function, OUTPUT = output_function
      , INTERNALLENGTH = { internallength | VARIABLE }
    [ , EXTERNALLENGTH = { externallength | VARIABLE } ]
    [ , DEFAULT = \"default\" ]
    [ , ELEMENT = element ] [ , DELIMITER = delimiter ]
    [ , SEND = send_function ] [ , RECEIVE = receive_function ]
    [ , PASSEDBYVALUE ] )
```

8.2.15 CREATE USER

Descripción: Creates a new database user

```
CREATE USER username
    [ WITH
     [ SYSID uid ]
     [ PASSWORD 'password' ] ]
    [ CREATEDB   | NOCREATEDB ] [ CREATEUSER | NOCREATEUSER ]
    [ IN GROUP     groupname [, ...] ]
    [ VALID UNTIL  'abstime' ]
```

8.2.16 CREATE VIEW

Descripción: Constructs a virtual table

```
CREATE VIEW view AS SELECT query
```

8.2.17 DECLARE

Descripción: Defines a cursor for table access

```
DECLARE cursorname [ BINARY ] [ INSENSITIVE ] [ SCROLL ]
    CURSOR FOR query
    [ FOR { READ ONLY | UPDATE [ OF column [, ...] ] ]
```

8.2.18 DELETE

Descripción: Removes rows from a table

```
DELETE FROM table [ WHERE condition ]
```

8.2.19 DROP AGGREGATE

Descripción: Removes the definition of an aggregate function

```
DROP AGGREGATE name type
```

8.2.20 DROP DATABASE

Descripción: Removes an existing database

```
DROP DATABASE name
```

8.2.21 DROP FUNCTION

Descripción: Removes a user-defined C function

```
DROP FUNCTION name ( [ type [, ...] ] )
```

8.2.22 DROP GROUP

Descripción: Removes a group

```
DROP GROUP name
```

8.2.23 DROP INDEX

Descripción: Removes an index from a database

```
DROP INDEX index_name
```

8.2.24 DROP LANGUAGE

Descripción: Removes a user-defined procedural language

```
DROP PROCEDURAL LANGUAGE 'name'
```

8.2.25 DROP OPERATOR

Descripción: Removes an operator from the database

```
DROP OPERATOR id ( type | NONE [,...] )
```

8.2.26 DROP RULE

Descripción: Removes an existing rule from the database

```
DROP RULE name
```

8.2.27 DROP SEQUENCE

Descripción: Removes an existing sequence

```
DROP SEQUENCE name [, ...]
```

8.2.28 DROP TABLE

Descripción: Removes existing tables from a database

```
DROP TABLE name [, ...]
```

8.2.29 DROP TRIGGER

Descripción: Removes the definition of a trigger

```
DROP TRIGGER name ON table
```

8.2.30 DROP TYPE

Descripción: Removes a user-defined type from the system catalogs

```
DROP TYPE typename
```

8.2.31 DROP USER

Descripción: Removes a user

```
DROP USER name
```

8.2.32 DROP VIEW

Descripción: Removes an existing view from a database

```
DROP VIEW name
```

8.2.33 EXPLAIN

Descripción: Shows statement execution plan

```
EXPLAIN [ VERBOSE ] query
```

8.2.34 FETCH

Descripción: Gets rows using a cursor

```
FETCH [ selector ] [ count ] { IN | FROM } cursor
FETCH [ RELATIVE ] [ { [ # | ALL | NEXT | PRIOR ] } ] FROM ] cursor
```

8.2.35 INSERT

Descripción: Inserts new rows into a table

```
INSERT INTO table [ ( column [, ...] ) ]
    { VALUES ( expression [, ...] ) | SELECT query }
```

8.2.36 LISTEN

Descripción: Listen for a response on a notify condition

```
LISTEN name
```

8.2.37 LOAD

Descripción: Dynamically loads an object file

```
LOAD 'filename'
```

8.2.38 LOCK

Descripción: Explicitly lock a table inside a transaction

```
LOCK [ TABLE ] name
LOCK [ TABLE ] name IN [ ROW | ACCESS ] { SHARE | EXCLUSIVE } MODE
LOCK [ TABLE ] name IN SHARE ROW EXCLUSIVE MODE
```

8.2.39 MOVE

Descripción: Moves cursor position

```
MOVE [ selector ] [ count ]
    { IN | FROM } cursor
```

8.2.40 NOTIFY

Descripción: Signals all frontends and backends listening on a notify condition

```
NOTIFY name
```

8.2.41 REINDEX

Descripción: Recover corrupted system indexes under standalone Postgres

```
REINDEX { TABLE | DATABASE | INDEX } name [ FORCE ]
```

8.2.42 RESET

Descripción: Restores run-time parameters for session to default values

```
RESET variable
```

8.2.43 SELECT

Descripción: Retrieve rows from a table or view.

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    expression [ AS name ] [, ...]
    [ INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table ]
    [ FROM table [ alias ] [, ...] ]
    [ WHERE condition ]
    [ GROUP BY column [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION [ ALL ] | INTERSECT | EXCEPT } select ]
    [ ORDER BY column [ ASC | DESC | USING operator ] [, ...] ]
    [ FOR UPDATE [ OF class_name [, ...] ] ]
    LIMIT { count | ALL } [ { OFFSET | , } start ]
```

8.2.44 SELECT INTO

Descripción: Create a new table from an existing table or view

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    expression [ AS name ] [, ...]
    [ INTO [ TEMPORARY | TEMP ] [ TABLE ] new_table ]
    [ FROM table [ alias ] [, ...] ]
    [ WHERE condition ]
    [ GROUP BY column [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION [ ALL ] | INTERSECT | EXCEPT } select ]
    [ ORDER BY column [ ASC | DESC | USING operator ] [, ...] ]
    [ FOR UPDATE [ OF class_name [, ...] ] ]
    LIMIT { count | ALL } [ { OFFSET | , } start ]
```

8.2.45 SET

Descripción: Set run-time parameters for session

```
SET variable { TO | = } { value | 'value' | DEFAULT }
SET CONSTRAINTS {ALL | constraintlist} mode
SET TIME ZONE { 'timezone' | LOCAL | DEFAULT }
SET TRANSACTION ISOLATION LEVEL { READ COMMITTED | SERIALIZABLE }
```

8.2.46 SHOW

Descripción: Shows run-time parameters for session

```
SHOW keyword
```

8.2.47 TRUNCATE

Descripción: Empty a table

```
TRUNCATE [ TABLE ] name
```

8.2.48 UNLISTEN

Descripción: Stop listening for notification

```
UNLISTEN { notifyname | * }
```

8.2.49 UPDATE

Descripción: Replaces values of columns in a table

```
UPDATE table SET col = expression [, ...]
    [ FROM fromlist ]
    [ WHERE condition ]
```

Por ejemplo,

```
update lola set lola.dedonde = lupe.p from lupe where lola.proced = lupe.c;
```

8.2.50 VACUUM

Descripción: Clean and analyze a Postgres database

```
VACUUM [ VERBOSE ] [ ANALYZE ] [ table ]
VACUUM [ VERBOSE ] ANALYZE [ table [ (column [, ...] ) ] ]
```

8.2.51 Tipos de datos relevantes en PostgreSQL

Como todos los manejadores de bases de datos, PostgreSQL implementa los tipos de datos definidos para el estándar SQL3 y aumenta algunos otros. Los definidos por el estándar SQL3 se muestran en la tabla 8.1, mientras que las extensiones se encuentran en la tabla 8.2.

Tipos de datos del estándar SQL3 en PostgreSQL		
Tipo en Postgres	Correspondiente en SQL3	Descripción
bool	boolean	valor lógico o booleano (true/false)
char(n)	character(n)	cadena de carácteres de tamaño fijo
date	date	fecha (sin hora)
float4/8	float(p)	número de punto flotante con precisión p
float8	real, double precision	número de punto flotante de doble precisión
int2	smallint	entero de dos bytes con signo
int4	int, integer	entero de cuatro bytes con signo
int4	decimal(p, s)	número exacto con $p <= 9, s = 0$
int4	numeric(p, s)	número exacto con $p == 9, s = 0$
money	decimal(9,2)	cantidad monetaria
time	time	hora en horas, minutos, segundos y centésimas
timespan	interval	intervalo de tiempo
timestamp	timestamp with time zone	fecha y hora con zonificación
varchar(n)	character varying(n)	cadena de carácteres de tamaño variable

Tabla 8.1: Tipos de datos del estándar SQL3 en PostgreSQL

8.3 INSERT

Sintáxis:

```
INSERT INTO class_name [(attr1, ...attrN)]
VALUES (expr1,..exprN) |
SELECT [DISTINCT [ON attrN]]
expr1, ...exprN
[FROM from_clause]
[WHERE qual]
[GROUP BY group_list]
[HAVING having_clause]
[ { UNION [ALL] | INTERSECT | EXCEPT } SELECT ...];
```

8.4 SELECT

Sintáxis:

```
SELECT [DISTINCT [ON attrN]] expr1 [AS attr1], ...exprN
[INTO [TEMP] [TABLE] class_name]
[FROM from_list]
[WHERE qual]
[GROUP BY group_list]
[HAVING having_clause]
[ { UNION [ALL] | INTERSECT | EXCEPT } SELECT ...]
[ORDER BY attr1 [ASC|DESC] [USING op1], ...attrN ]
```

Tipos de datos extendidos en PostgreSQL	
Tipo	Descripción
box	caja rectangular en el plano
cidr	dirección de red o de *host* en IP versión 4
circle	círculo en el plano
inet	dirección de red o de *host* en IP versión 4
int8	entero de ocho bytes con signo
line	línea infinita en el plano
lseg	segmento de línea en el plano
path	trayectoria geométrica, abierta o cerrada, en el plano
point	punto geométrico en el plano
polygon	trayectoria geométrica cerrada en el plano
serial	identificador numerico único

Tabla 8.2: Tipos de datos extendidos por PostgreSQL

```
[FOR UPDATE [OF class_name...]]
[LIMIT count [OFFSET|, count]];
```

8.5 UPDATE

Sintáxis:

```
UPDATE class_name SET attr1 = expr1, ...attrN = exprN
[FROM from_clause]
[WHERE qual];
```

8.6 DELETE

Sintáxis:

```
DELETE FROM class_name [WHERE qual];
```

8.7 CREATE INDEX

Construye un índice.
Sintáxis:

```
CREATE [UNIQUE] INDEX indexname ON class_name [USING access_method]
( attr1 [type_class1], ...attrN | funcname(attr1, ...) [type_class] );
```

8.8 CREATE VIEW

Construye una vista.
Sintáxis:

```
CREATE VIEW view_name AS
SELECT [DISTINCT [ON attrN]]
expr1 [AS attr1], ...exprN
[FROM from_list]
[WHERE qual]
[GROUP BY group_list];
```

8.9 DROPs

8.9.1 DROP AGGREGATE

Destruye una función agregada
Sintáxis:

```
DROP AGGREGATE agg_name agg_type|*;
```

8.9.2 DROP DATABASE

Destruye una base de datos
Sintáxis:

```
DROP DATABASE dbname
```

8.9.3 DROP FUNCTION

Remueve una función definida por el usuario.
Sintáxis:

```
DROP FUNCTION funcname ([type1, ...typeN]);
```

8.9.4 DROP INDEX

Destruye un índice.
Sintáxis:

```
DROP INDEX indexname;
```

8.9.5 DROP OPERATOR

Remueve un operador definido por el usuario.
Sintáxis:

```
DROP OPERATOR operator_name ([ltype|NONE],[RTYPE|none]);
```

8.9.6 DROP RULE

Destruye una regla.
Sintáxis:

```
DROP RULE rulename;
```

8.9.7 DROP SEQUENCE

Destruye un generador de secuencias numéricas.
Sintáxis:

```
DROP SEQUENCE sequence_name[, ...sequence_nameN];
```

8.9.8 DROP TABLE

Destruye una tabla.
Sintáxis:

```
DROP TABLE class_name1, ...class_nameN;
```

8.9.9 DROP TRIGGER

Destruye un `trigger`.
Sintáxis:

```
DROP TRIGGER trigger_name ON class_name;
```

8.9.10 DROP TYPE

Elimina un tipo de datos definido por el usuario.
Sintáxis:

```
DROP TYPE typename;
```

8.9.11 DROP VIEW

Elimina una vista.
Sintáxis:

```
DROP VIEW view_name
```

8.10 Permisos de acceso

Para cada tabla de cada base de datos, es posible fijar permisos individuales de lectura, actualización, inserción, borrado y creación de reglas.

8.10.1 GRANT

Descripción: Grants access privilege to a user, a group or all users

```
GRANT privilege [, ...] ON object [, ...]
    TO { PUBLIC | GROUP group | username }
```

8.10.2 REVOKE

Descripción: Revokes access privilege from a user, a group or all users.

```
REVOKE privilege [, ...]
    ON object [, ...]
    FROM { PUBLIC | GROUP groupname | username }
```

8.11 Transacciones

Un bloque de transacciones comienza con un `BEGIN` y si la transacción fué válida se cierra con `COMMIT` y `END`. Si la transacción falla, se cierra con `ABORT` y `ROLLBACK`. Esta es una manera segura de garantizar integridad de la información. Digamos que estamos haciendo un sistema de inventarios y que, por definición, al retirar un artículo de la bodega, lo tenemos que dar de alta en el inventario de la tienda. No puede ocurrir una acción sin la correspondiente, es decir, si lo damos de baja de la bodega *necesariamente* tiene que ser cargado al inventario de la tienda y de igual manera, si lo damos de alta en el inventario de la tienda, tiene que haber sido descontado del inventario de la bodega. Si una de las dos acciones falla, nos deja un estado inconsistente. En metacódigo esto queda representado por:

Algorithm 1 Transacción segura.

```
1:  while queden registros do
2:    BEGIN
3:    trae el siguiente registro de la bodega
4:    insertalo en el inventario de la tienda
5:    elimínalo del inventario de la bodega
6:    if no hubo error then
7:      COMMIT
8:      END
9:    else
10:     ABORT
11:     ROLLBACK
12:   end if
13: end while
```

8.11.1 ABORT

Descripción: Aborts the current transaction

```
ABORT [ WORK | TRANSACTION ]
```

8.11.2 BEGIN

Descripción: Begins a transaction in chained mode

```
BEGIN [ WORK | TRANSACTION ]
```

8.11.3 COMMIT

Descripción: Commits the current transaction

```
COMMIT [ WORK | TRANSACTION ]
```

8.11.4 END

Descripción: Commits the current transaction

```
END [ WORK | TRANSACTION ]
```

8.11.5 ROLLBACK

Descripción: Aborts the current transaction

```
ROLLBACK [ WORK | TRANSACTION ]
```

Tipo	Descripción
SET	conjunto de tuplas
abstime	fecha y hora absoluta de rango limitado (Unix system time)
aclitem	lista de control de acceso
bool	booleano '*true*'/'*false*'
box	rectángulo geométrico '(izquierda abajo, derecha arriba)'
bpchar	carácteres rellenos con espacios, longitud especificada al momento de creación
bytea	arreglo de bytes de longitud variable
char	un sólo carácter
cid	*command identifier type*, identificador de secuencia en transacciones
cidr	dirección de red
circle	círculo geométrico '(centro, radio)'
date	fecha ANSI SQL 'aaaa-mm-dd'
datetime	fecha y hora 'aaaa-mm-dd hh:mm:ss'
filename	nombre de archivo usado en tablas del sistema
float4	número real de precisión simple de 4 bytes
float8	número real de precisión doble de 8 bytes
inet	dirección de red
int2	número entero de dos bytes, de -32k a 32k
int28	8 numeros enteros de 2 bytes, usado internamente
int4	número entero de 4 bytes, -2B to 2B
int8	número entero de 8 bytes, >18 dígitos
line	línea geométrica '(pt1, pt2)'
lseg	segmento de línea geométrica '(pt1, pt2)'
macaddr	dirección MAC
money	unidad monetaria '$d,ddd.cc'
name	tipo de 31 carácteres para guardar identificadores del sistema
numeric	número de precisión múltiple
oid	tipo de identificación de objetos
oid8	arreglo de 8 *oid*s, utilizado en tablas del sistema
path	trayectoria geométrica '(pt1, ...)'
point	punto geométrico '(x, y)'
polygon	polígono geométrico '(pt1, ...)'
regproc	procedimiento registrado
reltime	intervalo de tiempo de rango limitado y relativo (Unix delta time)
smgr	manejador de almacenamiento (*storage manager*)
text	cadena de carácteres nativa de longitud variable
tid	tipo de identificador de tupla, localización física de tupla
time	hora ANSI SQL 'hh:mm:ss'
timespan	intervalo de tiempo '@ ¡number¿ ¡units¿'
timestamp	fecha y hora en formato ISO de rango limitado
tinterval	intervalo de tiempo '(abstime, abstime)'
unknown	tipo desconocido
varchar	cadena de carácteres sin espacios al final, longitud especificada al momento de creación
xid	identificador de transacción

Tabla 8.3: Tipos de datos de PostgreSQL

Capítulo 9

Algunas características de PostgreSQL

En éste capítulo tocaremos algunos puntos sueltos particulares a PostgreSQL. En las instalaciones basadas en RedHat estándares, PostgreSQL almacena los datos en el directorio `/var/lib/pgsql/base/` y a partir de ahí un directorio para cada base.

Cada tabla es un archivo, así como los indices. Los nombres de las tablas pertenecientes al sistema llevan el prefijo `pg_`. El archivo `PG_VERSION` (presente en cada base) contiene la versión mayor con la que fué creada la base. Al cambiar de versión de PostgreSQL es importante verificar que no haya sido cambiado el formato de alguna de estas tablas, en cuyo caso será necesario respaldarlas (como se indica en la sección 13.3) antes de instalar la nueva versión y luego volver a cargarlas.

9.1 Tablas internas

En las tablas presentadas en esta sección mostramos el contenido de las tablas que PostgreSQL utiliza como catálogos para mantener el sistema. Es en base a la idea de mantener todo en tablas que PostgreSQL, a diferencia de otros manejadores de bases de datos, es extensible. La mayoría de la información presentada en esta sección ha sido extraída de examinar el código fuente de PostgreSQL y por esta razón se halla incompleta. Sin embargo, hemos procurado mostrar al menos la más importante para los desarrolladores.

Para saber que bases de datos hay en el sistema:

```
SELECT * FROM pg_database;
```

Para saber que tablas tengo en la base de datos actual:

```
SELECT * FROM pg_class;
```

Lo mismo, pero sólo las definidas por el usuario, excluyendo las del sistema:

```
SELECT * FROM pg_class WHERE relname !~~ 'pg%';
```

Si sólo queremos saber cuantos registros tiene una tabla, basta con preguntar:

```
SELECT relname,reltuples FROM pg_class WHERE relname=mitabla';
```

Nombre del catálogo	Descripción
pg_database	Bases de datos
pg_class	Clases o tablas
pg_attribute	Atributos o campos de la clase o tabla
pg_index	Índices secundarios
pg_proc	Procedimientos (en C y en SQL)
pg_type	Tipos de datos (del sistema y definidos por el usuario)
pg_operator	Operadores (del sistema y definidos por el usuario)
pg_aggregate	Agregados y funciones agregadas
pg_am	Métodos de acceso
pg_amop	Operadores de métodos de acceso
pg_amproc	Funciones de soporte para métodos de acceso
pg_opclass	Clases de operadores de métodos de acceso

Tabla 9.1: Catálogo del sistema PostgreSQL. Cada base de datos tiene estas mismas tablas, salvo por la primera que es única, que almacenan cada una de las partes que componen la base de datos.

Table = pg_database			
Field	**Type**	**Length**	**Description**
datname	name	32	Nombre de la base
datdba	int4	4	Uid del dababase admin
datpath	text	var	El path para llegar hasta la base

Tabla 9.2: Tabla que contiene todas las bases de datos que existen en el sistema. Vale la pena consultarla en aplicaciones que impliquen explorar diversos aspectos.

donde `mitabla` es el nombre de la tabla de la cual nos interesa saber el número de registros.

A continuación, presentamos una forma de extraer información de las tablas de la base de datos. `pg_database` tiene las bases de datos, `pg_class` tiene las tablas, y `pg_attribute` tiene los campos. Si lo que queremos saber es el número de registros en una tabla determinada, basta con preguntar lo siguiente, conociendo la tabla, adicionalmente se obtienen los campos. Para, además, saber los tipos de los campos, habría que hacer un *query* a `pg_type` con los `oids` de los campos. ¿Puede construir la consulta para hacerlo?

```
SELECT relname,reltuples,attname,attnum
    FROM pg_class,pg_attribute
    WHERE pg_class.relname='mitabla'
        AND pg_attribute.attrelid=pg_class.oid
        AND attnum > 0
    ORDER BY attnum;
```

Si lo que queremos es saber todo de todas las tablas de la base que no sean del sistema, sino del usuario, por supuesto ordenadas por `oid`, se hace lo siguiente, pero con el problema de que incluye los indices:

```
SELECT pg_class.oid,relname,reltuples,attname
    FROM pg_class,pg_attribute
    WHERE pg_class.relname !~~ '%pg%'
        AND pg_attribute.attrelid=pg_class.oid
```

Table = pg_class			
Field	**Type**	**Length**	**Description**
relname	name	32	Nombre de la tabla
reltype	oid	4	El Object Id de la tabla
relowner	oid	4	El UID (postgres) del dueño de la tabla
relam	oid	4	
relpages	int4	4	Número de páginas ocupadas por la tabla
reltuples	int4	4	Número de tuplas en la tabla
relhasindex	bool	1	Verdadero si tiene al menos un índice
relisshared	bool	1	
relkind	char	1	
relnatts	int2	2	Número de atributos
relchecks	int2	2	
reltriggers	int2	2	Número de *triggers* asociados
relhasrules	bool	1	Verdadero si tiene reglas asociadas
relacl	aclitem[]	var	

Tabla 9.3: Tabla que contiene todas las tablas en la base de datos actual.

```
    AND attnum > 0
ORDER BY pg_class.oid;
```

En el caso de que uno quiera saber cuales son los campos de todas las tablas que se tienen en una base de datos, esta consulta lo soluciona[1].

```
SELECT pg_class.relname,pg_attribute.attname,pg_class.reltuples,pg_type.typname
        FROM pg_attribute
        WHERE attrelid = pg_class.oid
            AND attnum > 0
            AND atttypid=pg_type.oid
            AND pg_class.oid IN
                (SELECT oid FROM pg_class WHERE relname !~~ 'pg%')
            AND pg_type.oid IN (SELECT oid FROM pg_type);
```

9.2 Funciones incluidas en PostgreSQL

Función	Argumentos	Valor de regreso	Descripción
_bpchar	_bpchar int4	_bpchar	truncate _char()
_varchar	_varchar int4	_varchar	truncate _varchar()
abs	numeric	numeric	absolute value
abstime	abstime	abstime	convert (noop)
abstime	datetime	abstime	convert datetime to abstime
abstime_date	abstime	date	convert abstime to date
abstime_datetime	abstime	datetime	convert abstime to datetime
abstime_finite	abstime	bool	
abstimeeq	abstime abstim	bool	equal
abstimege	abstime abstim	bool	greater-than-or-equal
abstimegt	abstime abstim	bool	greater-than
abstimele	abstime abstim	bool	less-than-or-equal
abstimelt	abstime abstim	bool	less-than

[1] Y además dice cuántos registros tiene cada tabla. Claro que lo repite para cada campo, pero bueno, SQL no fue hecho para formatear datos.

Table = pg_attribute			
Field	**Type**	**Length**	**Description**
attrelid	oid	4	OID del atributo
attname	name	32	Nombre del atributo
atttypid	oid	4	Oid del tipo definido para el atributo
attdisbursion	float4	4	
attlen	int2	2	Longitud en bytes del atributo
attnum	int2	2	
attnelems	int4	4	
attcacheoff	int4	4	
atttypmod	int2	2	
attbyval	bool	1	
attisset	bool	1	
attalign	char	1	
attnotnull	bool	1	
atthasdef	bool	1	

Tabla 9.4: Tabla que contiene los atributos de los atributos de todas las tablas en la base actual.

Función	Argumentos	Valor de regreso	Descripción
abstimene	abstime abstim	bool	not equal
aclcontains	_aclitem aclit	bool	matches regex., case-sensitive
aclinsert	_aclitem aclit	_aclitem	addition
aclremove	_aclitem aclit	_aclitem	subtract
age	datetime	timespan	difference between datetime and to
age	datetime datet	timespan	difference between datetimes but l
area	box	float8	box area
area	circle	float8	area of circle
areajoinsel	oid oid int2 o	float8	selectivity
areasel	oid oid int2 i	float8	selectivity
array_assgn	int4 int4 int4	int4	array
array_clip	int4 int4 int4	int4	array
array_in	int4	int4	array
array_ref	int4 int4 int4	int4	array
array_set	int4 int4 int4	int4	array
booleq	bool bool	bool	equal
boolgt	bool bool	bool	greater-than
boollt	bool bool	bool	less-than
boolne	bool bool	bool	not equal
box	box box	box	convert boxes to box (intersection
box	circle	box	convert circle to box
box	point point	box	convert points to box
box	polygon	box	convert polygon to box
box_above	box box	bool	is above
box_add	box point	box	add point to box (translate)
box_area	box	float8	box area
box_below	box box	bool	is below
box_center	box	point	center of
box_circle	box	circle	convert box to circle
box_contain	box box	bool	contains
box_contained	box box	bool	contained in
box_diagonal	box	lseg	box diagonal
box_distance	box box	float8	distance between

Función	Argumentos	Valor de regreso	Descripción
box_div	box point	box	divide box by point (scale)
box_eq	box box	bool	equal
box_ge	box box	bool	greater-than-or-equal
box_gt	box box	bool	greater-than
box_height	box	float8	box height
box_intersect	box box	box	intersects
box_le	box box	bool	less-than-or-equal
box_left	box box	bool	is left of
box_lt	box box	bool	less-than
box_mul	box point	box	multiply box by point (scale)
box_overlap	box box	bool	overlaps
box_overleft	box box	bool	overlaps, but does not extend to r
box_overright	box box	bool	overlaps, but does not extend to l
box_poly	box	polygon	convert box to polygon
box_right	box box	bool	is left of
box_same	box box	bool	same as
box_sub	box point	box	subtract point from box (translate
box_width	box	float8	box width
bpchar	bpchar int4	bpchar	truncate char()
bpchar	char	bpchar	convert char to char()
bpchar	name	bpchar	convert name to char()
bpchar_char	bpchar	char	convert char() to char
bpchar_name	bpchar	name	convert char() to name
bpcharcmp	bpchar bpchar	int4	less-equal-greater
bpchareq	bpchar bpchar	bool	equal
bpcharge	bpchar bpchar	bool	greater-than-or-equal
bpchargt	bpchar bpchar	bool	greater-than
bpcharin	int4	bpchar	(internal)
bpcharle	bpchar bpchar	bool	less-than-or-equal
bpcharlen	bpchar	int4	octet length
bpcharlt	bpchar bpchar	bool	less-than
bpcharne	bpchar bpchar	bool	not equal
bpcharoctetlen	bpchar	int4	octet length
broadcast	inet	text	broadcast address
btabstimecmp	abstime abstim	int4	btree less-equal-greater
btcharcmp	char char	int4	btree less-equal-greater
btfloat4cmp	float4 float4	int4	btree less-equal-greater
btfloat8cmp	float8 float8	int4	btree less-equal-greater
btint24cmp	int2 int4	int4	btree less-equal-greater
btint2cmp	int2 int2	int4	btree less-equal-greater
btint42cmp	int4 int2	int4	btree less-equal-greater
btint4cmp	int4 int4	int4	btree less-equal-greater
btint8cmp	int8 int8	int4	btree less-equal-greater
btnamecmp	name name	int4	btree less-equal-greater
btoid8cmp	oid8 oid8	int4	btree less-equal-greater
btoidcmp	oid oid	int4	btree less-equal-greater
btreenpage	oid oid int2 i	float8	btree
btreesel	oid oid int2 i	float8	btree selectivity
btrim	text	text	trim both ends of string
btrim	text text	text	trim both ends of string
bttextcmp	text text	int4	btree less-equal-greater
byteaGetBit	bytea int4	int4	
byteaGetByte	bytea int4	int4	
byteaGetSize	bytea	int4	
byteaSetBit	bytea int4 int	bytea	
byteaSetByte	bytea int4 int	bytea	
cash_div_flt4	money float4	money	divide

Función	Argumentos	Valor de regreso	Descripción
cash_div_flt8	money float8	money	divide
cash_div_int2	money int2	money	divide
cash_div_int4	money int4	money	divide
cash_eq	money money	bool	equal
cash_ge	money money	bool	greater-than-or-equal
cash_gt	money money	bool	greater-than
cash_le	money money	bool	less-than-or-equal
cash_lt	money money	bool	less-than
cash_mi	money money	money	subtract
cash_mul_flt4	money float4	money	multiply
cash_mul_flt8	money float8	money	multiply
cash_mul_int2	money int2	money	multiply
cash_mul_int4	money int4	money	multiply
cash_ne	money money	bool	not equal
cash_pl	money money	money	addition
cash_words_out	money	text	output amount as words
cashlarger	money money	money	larger of two
cashsmaller	money money	money	smaller of two
ceil	numeric	numeric	smallest integer >= value
center	box	point	box center
center	circle	point	center of circle
center	polygon	point	
char	bpchar	char	convert char() to char
char	text	char	convert text to char()
char_bpchar	char	bpchar	convert char to char()
char_text	char	text	convert char to text
chardiv	char char	char	divide
chareq	char char	bool	equal
charge	char char	bool	greater-than-or-equal
chargt	char char	bool	greater-than
charle	char char	bool	less-than-or-equal
charlt	char char	bool	less-than
charmi	char char	char	subtract
charmul	char char	char	multiply
charne	char char	bool	not equal
charpl	char char	char	addition
cideq	cid cid	bool	equal
circle	box	circle	convert box to circle
circle	point float8	circle	convert point and radius to circle
circle	polygon	circle	convert polygon to circle
circle_above	circle circle	bool	is above
circle_add_pt	circle point	circle	addition
circle_area	circle	float8	area
circle_below	circle circle	bool	is below
circle_box	circle	box	convert circle to box
circle_center	circle	point	center of
circle_contain	circle circle	bool	contains
circle_contain_pt	circle point	bool	
circle_contained	circle circle	bool	
circle_diameter	circle	float8	diameter
circle_distance	circle circle	float8	distance between
circle_div_pt	circle point	circle	divide
circle_eq	circle circle	bool	equal
circle_ge	circle circle	bool	greater-than-or-equal
circle_gt	circle circle	bool	greater-than
circle_le	circle circle	bool	less-than-or-equal
circle_left	circle circle	bool	is left of

Función	Argumentos	Valor de regreso	Descripción
circle_lt	circle circle	bool	less-than
circle_mul_pt	circle point	circle	multiply
circle_ne	circle circle	bool	not equal
circle_overlap	circle circle	bool	overlaps
circle_overleft	circle circle	bool	overlaps, but does not extend to r
circle_overright	circle circle	bool	
circle_poly	int4 circle	polygon	convert vertex count and circle to
circle_radius	circle	float8	radius
circle_right	circle circle	bool	is left of
circle_same	circle circle	bool	same as
circle_sub_pt	circle point	circle	subtract
close_lb	line box	point	closest point to line on box
close_ls	line lseg	point	closest point to line on line segm
close_lseg	lseg lseg	point	closest point to line segment on l
close_pb	point box	point	closest point on box
close_pl	point line	point	closest point on line
close_ps	point lseg	point	closest point on line segment
close_sb	lseg box	point	closest point to line segment on b
close_sl	lseg line	point	closest point to line segment on l
currval	text	int4	sequence current value
date	abstime	date	convert abstime to date
date	date	date	convert (noop)
date	datetime	date	convert datetime to date
date_cmp	date date	int4	less-equal-greater
date_datetime	date	datetime	convert date to datetime
date_eq	date date	bool	equal
date_ge	date date	bool	greater-than-or-equal
date_gt	date date	bool	greater-than
date_larger	date date	date	larger of two
date_le	date date	bool	less-than-or-equal
date_lt	date date	bool	less-than
date_mi	date date	int4	subtract
date_mii	date int4	date	subtract
date_ne	date date	bool	not equal
date_part	text abstime	float8	extract field from abstime
date_part	text date	float8	extract field from date
date_part	text datetime	float8	extract field from datetime
date_part	text reltime	float8	extract field from reltime
date_part	text time	float8	extract field from time
date_part	text timespan	float8	extract field from timespan
date_pli	date int4	date	addition
date_smaller	date date	date	smaller of two
date_trunc	text datetime	datetime	truncate datetime to field
date_trunc	text timespan	timespan	truncate timespan to field
date_zone	text datetime	text	
datetime	abstime	datetime	convert abstime to datetime
datetime	date	datetime	convert date to datetime
datetime	date time	datetime	convert date and time to datetime
datetime	datetime	datetime	convert (noop)
datetime	text	datetime	convert text to datetime
datetime	timestamp	datetime	convert timestamp to datetime
datetime_abstime	datetime	abstime	convert datetime to abstime
datetime_age	datetime datet	timespan	date difference preserving months
datetime_cmp	datetime datet	int4	less-equal-greater
datetime_date	datetime	date	convert datetime to date
datetime_datetime	date time	datetime	convert date and time to datetime
datetime_eq	datetime datet	bool	equal

Función	Argumentos	Valor de regreso	Descripción
datetime_finite	datetime	bool	
datetime_ge	datetime datet	bool	greater-than-or-equal
datetime_gt	datetime datet	bool	greater-than
datetime_larger	datetime datet	datetime	larger of two
datetime_le	datetime datet	bool	less-than-or-equal
datetime_lt	datetime datet	bool	less-than
datetime_mi	datetime datet	timespan	subtract
datetime_mi_span	datetime times	datetime	minus
datetime_ne	datetime datet	bool	not equal
datetime_part	text datetime	float8	extract field from datetime
datetime_pl_span	datetime times	datetime	plus
datetime_smaller	datetime datet	datetime	smaller of two
datetime_text	datetime	text	convert datetime to text
datetime_time	datetime	time	convert datetime to time
datetime_timestamp	datetime	timestamp	convert datetime to timestamp
datetime_trunc	text datetime	datetime	truncate datetime to specified uni
datetime_zone	text datetime	text	
dcbrt	float8	float8	cube root
dexp	float8	float8	exponential
diameter	circle	float8	diameter of circle
dist_cpoly	circle polygon	float8	distance between
dist_lb	line box	float8	distance between
dist_pb	point box	float8	distance between
dist_pc	point circle	float8	distance between
dist_pl	point line	float8	distance between
dist_ppath	point path	float8	distance between
dist_ps	point lseg	float8	distance between
dist_sb	lseg box	float8	distance between
dist_sl	lseg line	float8	distance between
dlog1	float8	float8	natural logarithm (in psql, protec
dpow	float8 float8	float8	exponentiation
dround	float8	float8	truncate to integer
dsqrt	float8	float8	square root
dtof	float8	float4	convert float8 to float4
dtoi2	float8	int2	convert float8 to int2
dtoi4	float8	int4	convert float8 to int4
dtoi8	float8	int8	convert float8 to int8
dtrunc	float8	float8	truncate to integer
eqjoinsel	oid oid int2 o	float8	selectivity
eqsel	oid oid int2 i	float8	general selectivity
exp	numeric	numeric	e raised to the power of n
float	float4	float8	convert float4 to float8
float	float8	float8	convert float8 to float8 (no-op)
float4	float4	float4	convert float4 to float4 (no-op)
float4	float8	float4	convert float8 to float4
float4	int2	float4	convert int2 to float4
float4	int4	float4	convert int4 to float4
float4	numeric	float4	(internal)
float4	text	float4	convert text to float4
float48div	float4 float8	float8	divide
float48eq	float4 float8	bool	equal
float48ge	float4 float8	bool	greater-than-or-equal
float48gt	float4 float8	bool	greater-than
float48le	float4 float8	bool	less-than-or-equal
float48lt	float4 float8	bool	less-than
float48mi	float4 float8	float8	subtract
float48mul	float4 float8	float8	multiply

Función	Argumentos	Valor de regreso	Descripción
float48ne	float4 float8	bool	not equal
float48pl	float4 float8	float8	addition
float4_numeric	float4	numeric	(internal)
float4_text	float4	text	convert float4 to text
float4abs	float4 float4	float4	absolute value
float4div	float4 float4	float4	divide
float4eq	float4 float4	bool	equal
float4ge	float4 float4	bool	greater-than-or-equal
float4gt	float4 float4	bool	greater-than
float4inc	float4	float4	increment
float4larger	float4 float4	float4	larger of two
float4le	float4 float4	bool	less-than-or-equal
float4lt	float4 float4	bool	less-than
float4mi	float4 float4	float4	subtract
float4mul	float4 float4	float4	multiply
float4ne	float4 float4	bool	not equal
float4pl	float4 float4	float4	addition
float4smaller	float4 float4	float4	smaller of two
float4um	float4	float4	subtract
float8	float4	float8	convert float4 to float8
float8	float8	float8	convert (no-op)
float8	int2	float8	convert int2 to float8
float8	int4	float8	convert int4 to float8
float8	int8	float8	convert int8 to float8
float8	numeric	float8	(internal)
float8	text	float8	convert text to float8
float84div	float8 float4	float8	divide
float84eq	float8 float4	bool	equal
float84ge	float8 float4	bool	greater-than-or-equal
float84gt	float8 float4	bool	greater-than
float84le	float8 float4	bool	less-than-or-equal
float84lt	float8 float4	bool	less-than
float84mi	float8 float4	float8	subtract
float84mul	float8 float4	float8	multiply
float84ne	float8 float4	bool	not equal
float84pl	float8 float4	float8	addition
float8_numeric	float8	numeric	(internal)
float8_text	float8	text	convert float8 to text
float8abs	float8	float8	absolute value
float8div	float8 float8	float8	divide
float8eq	float8 float8	bool	equal
float8ge	float8 float8	bool	greater-than-or-equal
float8gt	float8 float8	bool	greater-than
float8inc	float8	float8	increment
float8larger	float8 float8	float8	larger of two
float8le	float8 float8	bool	less-than-or-equal
float8lt	float8 float8	bool	less-than
float8mi	float8 float8	float8	subtract
float8mul	float8 float8	float8	multiply
float8ne	float8 float8	bool	not equal
float8pl	float8 float8	float8	addition
float8smaller	float8 float8	float8	smaller of two
float8um	float8	float8	subtract
floor	numeric	numeric	largest integer <= value
flt4_mul_cash	float4 money	money	multiply
flt8_mul_cash	float8 money	money	multiply
ftod	float4	float8	convert float4 to float8

Función	Argumentos	Valor de regreso	Descripción
ftoi2	float4	int2	convert float4 to int2
ftoi4	float4	int4	convert float4 to int4
getdatabaseencoding		name	
getpgusername		name	(internal)
gistnpage	oid oid int2 i	float8	gist
gistsel	oid oid int2 i	float8	gist selectivity
hashbpchar	bpchar	int4	hash
hashchar	char	int4	hash
hashfloat4	float4	int4	hash
hashfloat8	float8	int4	hash
hashint2	int2	int4	hash
hashint4	int4	int4	hash
hashint8	int8	int4	hash
hashname	name	int4	hash
hashnpage	oid oid int2 i	float8	hash
hashoid	oid	int4	hash
hashoid8	oid8	int4	hash
hashsel	oid oid int2 i	float8	selectivity
hashtext	text	int4	hash
hashvarchar	varchar	int4	hash
height	box	float8	box height
host	inet	text	host address
i2tod	int2	float8	convert int2 to float8
i2tof	int2	float4	convert int2 to float4
i2toi4	int2	int4	convert int2 to int4
i4tod	int4	float8	convert int4 to float8
i4tof	int4	float4	convert int4 to float4
i4toi2	int4	int2	convert int4 to int2
i8tod	int8	float8	convert int8 to float8
ininterval	abstime tinter	bool	abstime in tinterval
initcap	text	text	capitalize each word
int	int4	int4	convert (no-op)
int2	float4	int2	convert float4 to int2
int2	float8	int2	convert float8 to int2
int2	int2	int2	convert (no-op)
int2	int4	int2	convert int4 to int2
int2	text	int2	convert text to int2
int24div	int2 int4	int4	divide
int24eq	int2 int4	bool	equal
int24ge	int2 int4	bool	greater-than-or-equal
int24gt	int2 int4	bool	greater-than
int24le	int2 int4	bool	less-than-or-equal
int24lt	int2 int4	bool	less-than
int24mi	int2 int4	int4	subtract
int24mod	int2 int4	int4	modulus
int24mul	int2 int4	int4	multiply
int24ne	int2 int4	bool	not equal
int24pl	int2 int4	int4	addition
int2_mul_cash	int2 money	money	multiply
int2_text	int2	text	convert int2 to text
int2div	int2 int2	int2	divide
int2eq	int2 int2	bool	equal
int2fac	int2	int2	
int2ge	int2 int2	bool	greater-than-or-equal
int2gt	int2 int2	bool	greater-than
int2inc	int2	int2	increment
int2larger	int2 int2	int2	larger of two

Función	Argumentos	Valor de regreso	Descripción
int2le	int2 int2	bool	less-than-or-equal
int2lt	int2 int2	bool	less-than
int2mi	int2 int2	int2	subtract
int2mod	int2 int2	int2	modulus
int2mul	int2 int2	int2	multiply
int2ne	int2 int2	bool	not equal
int2pl	int2 int2	int2	addition
int2smaller	int2 int2	int2	smaller of two
int2um	int2	int2	subtract
int4	float4	int4	convert float4 to int4
int4	float8	int4	convert float8 to int4
int4	int2	int4	convert int2 to int4
int4	int4	int4	convert (no-op)
int4	int8	int4	convert int8 to int4
int4	numeric	int4	(internal)
int4	text	int4	convert text to int4
int4	varchar	int4	convert varchar to int4
int42div	int4 int2	int4	divide
int42eq	int4 int2	bool	equal
int42ge	int4 int2	bool	greater-than-or-equal
int42gt	int4 int2	bool	greater-than
int42le	int4 int2	bool	less-than-or-equal
int42lt	int4 int2	bool	less-than
int42mi	int4 int2	int4	subtract
int42mod	int4 int2	int4	modulus
int42mul	int4 int2	int4	multiply
int42ne	int4 int2	bool	not equal
int42pl	int4 int2	int4	addition
int48	int4	int8	convert int4 to int8
int48div	int4 int8	int8	divide
int48eq	int4 int8	bool	equal
int48ge	int4 int8	bool	greater-than-or-equal
int48gt	int4 int8	bool	greater-than
int48le	int4 int8	bool	less-than-or-equal
int48lt	int4 int8	bool	less-than
int48mi	int4 int8	int8	subtraction
int48mul	int4 int8	int8	multiply
int48ne	int4 int8	bool	not equal
int48pl	int4 int8	int8	addition
int4_mul_cash	int4 money	money	multiply
int4_numeric	int4	numeric	(internal)
int4_text	int4	text	convert int4 to text
int4div	int4 int4	int4	divide
int4eq	int4 int4	bool	equal
int4eqoid	int4 oid	bool	equal
int4fac	int4	int4	fraction
int4ge	int4 int4	bool	greater-than-or-equal
int4gt	int4 int4	bool	greater-than
int4inc	int4	int4	increment
int4larger	int4 int4	int4	larger of two
int4le	int4 int4	bool	less-than-or-equal
int4lt	int4 int4	bool	less-than
int4mi	int4 int4	int4	subtract
int4mod	int4 int4	int4	modulus
int4mul	int4 int4	int4	multiply
int4ne	int4 int4	bool	not equal
int4notin	int4	bool	not in

Función	Argumentos	Valor de regreso	Descripción
int4pl	int4 int4	int4	addition
int4reltime	int4	reltime	convert int4 to reltime
int4smaller	int4 int4	int4	smaller of two
int4um	int4	int4	subtract
int8	float8	int8	convert float8 to int8
int8	int4	int8	convert int4 to int8
int8	int8	int8	convert int8 to int8 (no-op)
int8	text	int8	convert text to int8
int8	varchar	int8	convert varchar to int8
int84	int8	int4	convert int8 to int4
int84div	int8 int4	int8	divide
int84eq	int8 int4	bool	equal
int84ge	int8 int4	bool	greater-than-or-equal
int84gt	int8 int4	bool	greater-than
int84le	int8 int4	bool	less-than-or-equal
int84lt	int8 int4	bool	less-than
int84mi	int8 int4	int8	subtraction
int84mul	int8 int4	int8	multiply
int84ne	int8 int4	bool	not equal
int84pl	int8 int4	int8	addition
int8_text	int8	text	convert int8 to text
int8div	int8 int8	int8	divide
int8eq	int8 int8	bool	equal
int8ge	int8 int8	bool	greater-than-or-equal
int8gt	int8 int8	bool	greater-than
int8larger	int8 int8	int8	larger of two
int8le	int8 int8	bool	less-than-or-equal
int8lt	int8 int8	bool	less-than
int8mi	int8 int8	int8	subtraction
int8mul	int8 int8	int8	multiply
int8ne	int8 int8	bool	not equal
int8pl	int8 int8	int8	addition
int8smaller	int8 int8	int8	smaller of two
int8um	int8	int8	unary minus
inter_lb	line box	bool	
inter_sb	lseg box	bool	
inter_sl	lseg line	bool	
intervalct	tinterval tint	bool	less-than
intervalend	tinterval	abstime	
intervaleq	tinterval tint	bool	equal
intervalge	tinterval tint	bool	greater-than-or-equal
intervalgt	tinterval tint	bool	greater-than
intervalle	tinterval tint	bool	less-than-or-equal
intervalleneq	tinterval relt	bool	length equal
intervallenge	tinterval relt	bool	length greater-than-or-equal
intervallengt	tinterval relt	bool	length greater-than
intervallenle	tinterval relt	bool	length less-than-or-equal
intervallenlt	tinterval relt	bool	length less-than
intervallenne	tinterval relt	bool	length not equal to
intervallt	tinterval tint	bool	less-than
intervalne	tinterval tint	bool	not equal
intervalov	tinterval tint	bool	overlaps
intervalrel	tinterval	reltime	
intervalsame	tinterval tint	bool	same as
intervalstart	tinterval	abstime	start of interval
intgtjoinsel	oid oid int2 o	float8	selectivity
intgtsel	oid oid int2 i	float8	selectivity

Función	Argumentos	Valor de regreso	Descripción
intltjoinsel	oid oid int2 o	float8	selectivity
intltsel	oid oid int2 i	float8	selectivity
isclosed	path	bool	
isfalse	bool	bool	
isfinite	abstime	bool	boolean test
isfinite	datetime	bool	boolean test
isfinite	timespan	bool	boolean test
ishorizontal	line	bool	is line horizontal?
ishorizontal	lseg	bool	
ishorizontal	point point	bool	
isoldpath	path	bool	
isopen	path	bool	
isparallel	line line	bool	are lines parallel?
isparallel	lseg lseg	bool	
isperpendicular	lseg lseg	bool	
istrue	bool	bool	
isvertical	line	bool	is line vertical?
isvertical	lseg	bool	
isvertical	point point	bool	
keyfirsteq	int2	bool	
length	bpchar	int4	character length
length	lseg	float8	distance between endpoints
length	path	float8	sum of lengths of path segments
length	text	int4	character length
length	varchar	int4	character length
line	point point	line	points to line
line_construct_pp	point point	line	line from points
line_distance	line line	float8	distance between
line_eq	line line	bool	lines equal?
line_horizontal	line	bool	lines horizontal?
line_interpt	line line	point	intersection point
line_intersect	line line	bool	lines intersect?
line_parallel	line line	bool	lines parallel?
line_perp	line line	bool	lines perpendicular?
line_vertical	line	bool	lines vertical?
ln	numeric	numeric	natural logarithm of n
lo_close	int4	int4	large object close
lo_creat	int4	oid	large object create
lo_export	oid text	int4	large object export
lo_import	text	oid	large object import
lo_lseek	int4 int4 int4	int4	large object seek
lo_open	oid int4	int4	large object open
lo_tell	int4	int4	large object position
lo_unlink	oid	int4	large object unlink(delete)
log	numeric numeri	numeric	logarithm base m of n
loread	int4 int4	bytea	large object read
lower	text	text	lowercase
lowrite	int4 bytea	int4	large object write
lpad	text int4	text	left-pad string to length
lpad	text int4 text	text	left-pad string to length
lseg	box	lseg	
lseg	point point	lseg	
lseg_center	lseg	point	center of
lseg_construct	point point	lseg	convert points to line segment
lseg_distance	lseg lseg	float8	distance between
lseg_eq	lseg lseg	bool	equal
lseg_ge	lseg lseg	bool	greater-than-or-equal

Función	Argumentos	Valor de regreso	Descripción
lseg_gt	lseg lseg	bool	greater-than
lseg_horizontal	lseg	bool	is horizontal
lseg_interpt	lseg lseg	point	
lseg_intersect	lseg lseg	bool	intersects
lseg_le	lseg lseg	bool	less-than-or-equal
lseg_length	lseg	float8	distance between endpoints
lseg_lt	lseg lseg	bool	less-than
lseg_ne	lseg lseg	bool	not equal
lseg_parallel	lseg lseg	bool	is parallel to
lseg_perp	lseg lseg	bool	is perpendicular to
lseg_vertical	lseg	bool	is vertical
ltrim	text	text	remove initial characters from str
ltrim	text text	text	left-pad string to length
macaddr_cmp	macaddr macadd	int4	less-equal-greater
macaddr_eq	macaddr macadd	bool	equal
macaddr_ge	macaddr macadd	bool	greater-than-or-equal
macaddr_gt	macaddr macadd	bool	greater-than
macaddr_le	macaddr macadd	bool	less-than-or-equal
macaddr_lt	macaddr macadd	bool	less-than
macaddr_manuf	macaddr	text	MAC manufacturer
macaddr_ne	macaddr macadd	bool	not equal
masklen	inet	int4	netmask length
mktinterval	abstime abstim	tinterval	convert to interval
mod	numeric numeri	numeric	modulus
name	bpchar	name	convert char() to name
name	text	name	convert text to name
name	varchar	name	convert varchar to name
name_bpchar	name	bpchar	convert name to char()
name_text	name	text	convert name to text
nameeq	name name	bool	equal
namege	name name	bool	greater-than-or-equal
namegt	name name	bool	greater-than
nameicregexeq	name text	bool	matches regex., case-insensitive
nameicregexne	name text	bool	does not match regex., case-insens
namele	name name	bool	less-than-or-equal
namelike	name text	bool	matches LIKE expression
namelt	name name	bool	less-than
namene	name name	bool	not equal
namenlike	name text	bool	does not match LIKE expression
nameregexeq	name text	bool	matches regex., case-sensitive
nameregexne	name text	bool	does not match regex., case-sensit
neqjoinsel	oid oid int2 o	float8	selectivity
neqsel	oid oid int2 i	float8	not-equal selectivity
netmask	inet	text	netmask of address
network	inet	text	network address
network_broadcast	inet	text	broadcast address
network_cmp	inet inet	int4	less-equal-greater
network_eq	inet inet	bool	equal
network_ge	inet inet	bool	greater-than-or-equal
network_gt	inet inet	bool	greater-than
network_host	inet	text	host address
network_le	inet inet	bool	less-than-or-equal
network_lt	inet inet	bool	less-than
network_masklen	inet	int4	netmask length
network_ne	inet inet	bool	not equal
network_netmask	inet	text	netmask of address
network_network	inet	text	network address

Función	Argumentos	Valor de regreso	Descripción
network_sub	inet inet	bool	is-subnet
network_subeq	inet inet	bool	is-subnet-or-equal
network_sup	inet inet	bool	is-supernet
network_supeq	inet inet	bool	is-supernet-or-equal
nextval	text	int4	sequence next value
now		timestamp	current transaction time
numeric	float4	numeric	(internal)
numeric	float8	numeric	(internal)
numeric	int4	numeric	(internal)
numeric	numeric int4	numeric	(internal)
numeric_abs	numeric	numeric	absolute value
numeric_add	numeric numeri	numeric	addition
numeric_ceil	numeric	numeric	smallest integer >= value
numeric_dec	numeric	numeric	decrement by one
numeric_div	numeric numeri	numeric	divide
numeric_eq	numeric numeri	bool	equal
numeric_exp	numeric	numeric	e raised to the power of n
numeric_float4	numeric	float4	(internal)
numeric_float8	numeric	float8	(internal)
numeric_floor	numeric	numeric	largest integer <= value
numeric_ge	numeric numeri	bool	greater-than-or-equal
numeric_gt	numeric numeri	bool	greater-than
numeric_in	int4	numeric	(internal)
numeric_inc	numeric	numeric	increment by one
numeric_int4	numeric	int4	(internal)
numeric_larger	numeric numeri	numeric	larger of two numbers
numeric_le	numeric numeri	bool	lower-than-or-equal
numeric_ln	numeric	numeric	natural logarithm of n
numeric_log	numeric numeri	numeric	logarithm base m of n
numeric_lt	numeric numeri	bool	lower-than
numeric_mod	numeric numeri	numeric	modulus
numeric_mul	numeric numeri	numeric	multiply
numeric_ne	numeric numeri	bool	not equal
numeric_power	numeric numeri	numeric	m raised to the power of n
numeric_round	numeric int4	numeric	value rounded to 'scale'
numeric_sign	numeric	numeric	sign of value
numeric_smaller	numeric numeri	numeric	smaller of two numbers
numeric_sqrt	numeric	numeric	square root
numeric_sub	numeric numeri	numeric	subtract
numeric_trunc	numeric int4	numeric	value truncated to 'scale'
obj_description	oid	text	get description for object id
octet_length	bpchar	int4	octet length
octet_length	text	int4	octet length
octet_length	varchar	int4	octet length
oid	text	oid	convert text to oid
oid8eq	oid8 oid8	bool	equal
oid8ge	oid8 oid8	bool	greater-than-or-equal
oid8gt	oid8 oid8	bool	greater-than
oid8le	oid8 oid8	bool	less-than-or-equal
oid8lt	oid8 oid8	bool	less-than
oid8ne	oid8 oid8	bool	less-than
oid8types	oid8	text	print type names of oid8 field
oid_text	oid	text	convert oid to text
oideq	oid oid	bool	equal
oideqint4	oid int4	bool	equal
oidne	oid oid	bool	not equal
oidnotin	oid	bool	not in

Función	Argumentos	Valor de regreso	Descripción
oidrand	oid int4	bool	random
oidsrand	int4	bool	seed random number generator
on_pb	point box	bool	point is inside
on_pl	point line	bool	contained in
on_ppath	point path	bool	contained in
on_ps	point lseg	bool	contained in
on_sb	lseg box	bool	contained in
on_sl	lseg line	bool	contained in
path	polygon	path	
path_add	path path	path	addition
path_add_pt	path point	path	addition
path_center	path	point	center of
path_close	path	path	
path_contain_pt	path point	bool	path contains point?
path_distance	path path	float8	distance between
path_div_pt	path point	path	divide
path_inter	path path	bool	
path_isclosed	path	bool	
path_isopen	path	bool	
path_length	path	float8	sum of path segments
path_mul_pt	path point	path	multiply
path_n_eq	path path	bool	equal
path_n_ge	path path	bool	greater-than-or-equal
path_n_gt	path path	bool	greater-than
path_n_le	path path	bool	less-than-or-equal
path_n_lt	path path	bool	less-than
path_npoints	path	int4	
path_open	path	path	
path_poly	path	polygon	convert path to polygon
path_sub_pt	path point	path	subtract
pclose	path	path	
pg_get_indexdef	oid	text	index description
pg_get_ruledef	name	text	source text of a rule
pg_get_userbyid	int4	name	user name by UID (with fallback)
pg_get_viewdef	name	text	select statement of a view
point	circle	point	convert circle to point (center)
point	float8 float8	point	convert x, y to point
point	lseg lseg	point	convert two line segments to point
point_above	point point	bool	is above
point_add	point point	point	add points (translate)
point_below	point point	bool	is below
point_distance	point point	float8	distance between
point_div	point point	point	divide points (scale/rotate)
point_eq	point point	bool	same as
point_horiz	point point	bool	is horizontal
point_left	point point	bool	is left of
point_mul	point point	point	multiply points (scale/rotate)
point_ne	point point	bool	not equal
point_right	point point	bool	is left of
point_slope	point point	float8	slope between points
point_sub	point point	point	subtract points (translate)
point_vert	point point	bool	is vertical
pointdist	point point	int4	
points	path	int4	
points	polygon	int4	
poly_box	polygon	box	convert polygon to bounding box
poly_center	polygon	point	center of

Función	Argumentos	Valor de regreso	Descripción
poly_circle	polygon	circle	convert polygon to circle
poly_contain	polygon polygo	bool	contains
poly_contain_pt	polygon point	bool	polygon contains point?
poly_contained	polygon polygo	bool	contained in
poly_distance	polygon polygo	float8	distance between
poly_left	polygon polygo	bool	is left of
poly_npoints	polygon	int4	number of points in polygon
poly_overlap	polygon polygo	bool	overlaps
poly_overleft	polygon polygo	bool	overlaps, but does not extend to r
poly_overright	polygon polygo	bool	overlaps, but does not extend to l
poly_path	polygon	path	convert polygon to path
poly_right	polygon polygo	bool	is left of
poly_same	polygon polygo	bool	same as
polygon	box	polygon	convert box to polygon
polygon	circle	polygon	convert circle to 12-vertex polygo
polygon	int4 circle	polygon	convert circle to polygon
polygon	path	polygon	convert path to polygon
popen	path	path	
power	numeric numeri	numeric	m raised to the power of n
pqtest	text	int4	
pt_contained_circle	point circle	bool	
pt_contained_path	point path	bool	point contained in path?
pt_contained_poly	point polygon	bool	point contained by polygon?
radius	circle	float8	radius of circle
regproctooid	regproc	oid	get oid for regproc
reltime	reltime	reltime	convert (noop)
reltime	timespan	reltime	convert timespan to reltime
reltime_timespan	reltime	timespan	convert reltime to timespan
reltimeeq	reltime reltim	bool	equal
reltimege	reltime reltim	bool	greater-than-or-equal
reltimegt	reltime reltim	bool	greater-than
reltimele	reltime reltim	bool	less-than-or-equal
reltimelt	reltime reltim	bool	less-than
reltimene	reltime reltim	bool	not equal
revertpoly	polygon	polygon	
round	numeric	numeric	value rounded to 'scale' of zero
round	numeric int4	numeric	value rounded to 'scale'
rpad	text int4	text	right-pad string to length
rpad	text int4 text	text	right-pad string to length
rt_bigbox_size	box float4	float4	r-tree
rt_box_inter	box box	box	r-tree
rt_box_size	box float4	float4	r-tree
rt_box_union	box box	box	r-tree
rt_poly_inter	polygon polygo	polygon	r-tree
rt_poly_size	polygon int4	int4	r-tree
rt_poly_union	polygon polygo	polygon	r-tree
rtnpage	oid oid int2 i	float8	r-tree
rtrim	text	text	remove trailing characters from st
rtrim	text text	text	right-pad string to length
rtsel	oid oid int2 i	float8	r-tree
seteval	oid	int4	
setval	text int4	int4	sequence set value
sign	numeric	numeric	sign of value
slope	point point	float8	
smgreq	smgr smgr	bool	storage manager
smgrne	smgr smgr	bool	storage manager
sqrt	numeric	numeric	square root

Función	Argumentos	Valor de regreso	Descripción
strpos	text text	int4	find position of substring
substr	text int4	text	return portion of string
substr	text int4 int4	text	return portion of string
text	char	text	convert char to text
text	datetime	text	convert datetime to text
text	float4	text	convert float4 to text
text	float8	text	convert float8 to text
text	int2	text	convert int2 to text
text	int4	text	convert int4 to text
text	int8	text	convert int8 to text
text	name	text	convert name to text
text	oid	text	convert oid to text
text	timespan	text	convert timespan to text
text_char	text	char	convert text to char
text_datetime	text	datetime	convert text to datetime
text_float4	text	float4	convert text to float4
text_float8	text	float8	convert text to float8
text_ge	text text	bool	greater-than-or-equal
text_gt	text text	bool	greater-than
text_int2	text	int2	convert text to int2
text_int4	text	int4	convert text to int4
text_int8	text	int8	convert text to int8
text_larger	text text	text	larger of two
text_le	text text	bool	less-than-or-equal
text_lt	text text	bool	less-than
text_name	text	name	convert text to name
text_oid	text	oid	convert text to oid
text_smaller	text text	text	smaller of two
text_substr	text int4 int4	text	return portion of string
text_timespan	text	timespan	convert text to timespan
textcat	text text	text	concatenate
texteq	text text	bool	equal
texticregexeq	text text	bool	matches regex., case-insensitive
texticregexne	text text	bool	does not match regex., case-insens
textlen	text	int4	length
textlike	text text	bool	matches LIKE expression
textne	text text	bool	not equal
textnlike	text text	bool	does not match LIKE expression
textoctetlen	text	int4	octet length
textpos	text text	int4	return position of substring
textregexeq	text text	bool	matches regex., case-sensitive
textregexne	text text	bool	does not match regex., case-sensit
time	abstime	time	convert abstime to time
time	datetime	time	convert datetime to time
time	time	time	convert (noop)
time_cmp	time time	int4	less-equal-greater
time_eq	time time	bool	equal
time_ge	time time	bool	greater-than-or-equal
time_gt	time time	bool	greater-than
time_le	time time	bool	less-than-or-equal
time_lt	time time	bool	less-than
time_ne	time time	bool	not equal
timemi	abstime reltim	abstime	subtract
timenow		abstime	(internal)
timeofday		text	(internal)
timepl	abstime reltim	abstime	addition
timespan	reltime	timespan	convert reltime to timespan

Función	Argumentos	Valor de regreso	Descripción
timespan	text	timespan	convert text to timespan
timespan	time	timespan	convert time to timespan
timespan	timespan	timespan	convert (noop)
timespan_cmp	timespan times	int4	less-equal-greater
timespan_div	timespan float	timespan	divide
timespan_eq	timespan times	bool	equal
timespan_finite	timespan	bool	boolean test
timespan_ge	timespan times	bool	greater-than-or-equal
timespan_gt	timespan times	bool	greater-than
timespan_larger	timespan times	timespan	larger of two
timespan_le	timespan times	bool	less-than-or-equal
timespan_lt	timespan times	bool	less-than
timespan_mi	timespan times	timespan	subtract
timespan_ne	timespan times	bool	not equal
timespan_part	text timespan	float8	extract field from timespan
timespan_pl	timespan times	timespan	addition
timespan_reltime	timespan	reltime	convert timespan to reltime
timespan_smaller	timespan times	timespan	smaller of two
timespan_text	timespan	text	convert timespan to text
timespan_trunc	text timespan	timespan	truncate timespan to specified uni
timespan_um	timespan	timespan	subtract
timestamp	datetime	timestamp	convert datetime to timestamp
timestamp	timestamp	timestamp	convert (noop)
timestamp_datetime	timestamp	datetime	convert timestamp to datetime
timestampeq	timestamp time	bool	equal
timestampge	timestamp time	bool	greater-than-or-equal
timestampgt	timestamp time	bool	greater-than
timestample	timestamp time	bool	less-than-or-equal
timestamplt	timestamp time	bool	less-than
timestampne	timestamp time	bool	not equal
translate	text char char	text	modify string by substring replace
trunc	numeric	numeric	value truncated to 'scale' of zero
trunc	numeric int4	numeric	value truncated to 'scale'
upgradepath	path	path	
upgradepoly	polygon	polygon	
upper	text	text	uppercase
userfntest	int4	int4	
varchar	int4	varchar	convert int4 to varchar
varchar	int8	varchar	convert int8 to varchar
varchar	name	varchar	convert convert name to varchar
varchar	varchar	text	convert char to text
varchar	varchar int4	varchar	truncate varchar()
varcharcmp	varchar varcha	int4	less-equal-greater
varchareq	varchar varcha	bool	equal
varcharge	varchar varcha	bool	greater-than-or-equal
varchargt	varchar varcha	bool	greater-than
varcharin	int4	varchar	(internal)
varcharle	varchar varcha	bool	less-than-or-equal
varcharlen	varchar	int4	character length
varcharlt	varchar varcha	bool	less-than
varcharne	varchar varcha	bool	not equal
varcharoctetlen	varchar	int4	octet length
version		text	PostgreSQL version string
width	box	float8	box width
xideq	xid xid	bool	equal

9.3 Operadores

	Argumentos			
Operador	Izquierdo	Derecho	Resultado	Descripción
!	int4		int4	fraction
!!		int4	int4	fraction
!!=	int4	name	bool	not in
!!=	oid	name	bool	not in
!~	bpchar	text	bool	does not match regex., case-sensitive
!~	name	text	bool	does not match regex., case-sensitive
!~	text	text	bool	does not match regex., case-sensitive
!~	varchar	text	bool	does not match regex., case-sensitive
!~*	bpchar	text	bool	does not match regex., case-insensitive
!~*	name	text	bool	does not match regex., case-insensitive
!~*	text	text	bool	does not match regex., case-insensitive
!~*	varchar	text	bool	does not match regex., case-insensitive
!~~	bpchar	text	bool	does not match LIKE expression
!~~	name	text	bool	does not match LIKE expression
!~~	text	text	bool	does not match LIKE expression
!~~	varchar	text	bool	does not match LIKE expression
#		path	int4	
#		polygon	int4	number of points in polygon
#	box	box	box	intersects
#	line	line	point	intersection point
#	lseg	lseg	point	
##	line	box	point	closest point to line on box
##	line	lseg	point	closest point to line on line segment
##	lseg	box	point	closest point to line segment on box
##	lseg	line	point	closest point to line segment on line
##	lseg	lseg	point	closest point to line segment on line seg
##	point	box	point	closest point on box
##	point	line	point	closest point on line
##	point	lseg	point	closest point on line segment
#<	tinterval	reltime	bool	length less-than
#<=	tinterval	reltime	bool	length less-than-or-equal
#<>	tinterval	reltime	bool	length not equal to
#=	tinterval	reltime	bool	length equal
#>	tinterval	reltime	bool	length greater-than
#>=	tinterval	reltime	bool	length greater-than-or-equal
%		float8	float8	truncate to integer
%	float8		float8	truncate to integer
%	int2	int2	int2	modulus
%	int2	int4	int4	modulus
%	int4	int2	int4	modulus
%	int4	int4	int4	modulus
%	numeric	numeric	numeric	modulus
&&	box	box	bool	overlaps
&&	circle	circle	bool	overlaps
&&	polygon	polygon	bool	overlaps
&&	tinterval	tinterval	bool	overlaps
&<	box	box	bool	overlaps, but does not extend to right of
&<	circle	circle	bool	overlaps, but does not extend to right of
&<	polygon	polygon	bool	overlaps, but does not extend to right of
&>	box	box	bool	overlaps, but does not extend to left of
&>	circle	circle	bool	
&>	polygon	polygon	bool	overlaps, but does not extend to left of
*	box	point	box	multiply box by point (scale)
*	char	char	char	multiply
*	circle	point	circle	multiply
*	float4	float4	float4	multiply
*	float4	float8	float8	multiply

	Argumentos			
Operador	Izquierdo	Derecho	Resultado	Descripción
*	float4	money	money	multiply
*	float8	float4	float8	multiply
*	float8	float8	float8	multiply
*	float8	money	money	multiply
*	int2	int2	int2	multiply
*	int2	int4	int4	multiply
*	int2	money	money	multiply
*	int4	int2	int4	multiply
*	int4	int4	int4	multiply
*	int4	int8	int8	multiply
*	int4	money	money	multiply
*	int8	int4	int8	multiply
*	int8	int8	int8	multiply
*	money	float4	money	multiply
*	money	float8	money	multiply
*	money	int2	money	multiply
*	money	int4	money	multiply
*	numeric	numeric	numeric	multiply
*	path	point	path	multiply
*	point	point	point	multiply points (scale/rotate)
+	_aclitem	aclitem	_aclitem	addition
+	abstime	reltime	abstime	addition
+	box	point	box	add point to box (translate)
+	char	char	char	addition
+	circle	point	circle	addition
+	date	int4	date	addition
+	datetime	timespan	datetime	plus
+	float4	float4	float4	addition
+	float4	float8	float8	addition
+	float8	float4	float8	addition
+	float8	float8	float8	addition
+	int2	int2	int2	addition
+	int2	int4	int4	addition
+	int4	int2	int4	addition
+	int4	int4	int4	addition
+	int4	int8	int8	addition
+	int8	int4	int8	addition
+	int8	int8	int8	addition
+	money	money	money	addition
+	numeric	numeric	numeric	addition
+	path	path	path	addition
+	path	point	path	addition
+	point	point	point	add points (translate)
+	timespan	timespan	timespan	addition
-		float4	float4	subtract
-		float8	float8	subtract
-		int2	int2	subtract
-		int4	int4	subtract
-		int8	int8	unary minus
-		timespan	timespan	subtract
-	_aclitem	aclitem	_aclitem	subtract
-	abstime	reltime	abstime	subtract
-	box	point	box	subtract point from box (translate)
-	char	char	char	subtract
-	circle	point	circle	subtract
-	date	date	int4	subtract
-	date	int4	date	subtract

	Argumentos			
Operador	Izquierdo	Derecho	Resultado	Descripción
-	datetime	datetime	timespan	subtract
-	datetime	timespan	datetime	minus
-	float4	float4	float4	subtract
-	float4	float8	float8	subtract
-	float8	float4	float8	subtract
-	float8	float8	float8	subtract
-	int2	int2	int2	subtract
-	int2	int4	int4	subtract
-	int4	int2	int4	subtract
-	int4	int4	int4	subtract
-	int4	int8	int8	subtraction
-	int8	int4	int8	subtraction
-	int8	int8	int8	subtraction
-	money	money	money	subtract
-	numeric	numeric	numeric	subtract
-	path	point	path	subtract
-	point	point	point	subtract points (translate)
-	timespan	timespan	timespan	subtract
/	box	point	box	divide box by point (scale)
/	char	char	char	divide
/	circle	point	circle	divide
/	float4	float4	float4	divide
/	float4	float8	float8	divide
/	float8	float4	float8	divide
/	float8	float8	float8	divide
/	int2	int2	int2	divide
/	int2	int4	int4	divide
/	int4	int2	int4	divide
/	int4	int4	int4	divide
/	int4	int8	int8	divide
/	int8	int4	int8	divide
/	int8	int8	int8	divide
/	money	float4	money	divide
/	money	float8	money	divide
/	money	int2	money	divide
/	money	int4	money	divide
/	numeric	numeric	numeric	divide
/	path	point	path	divide
/	point	point	point	divide points (scale/rotate)
/	timespan	float8	timespan	divide
:		float8	float8	exponential
;		float8	float8	natural logarithm (in psql, protect with
<	abstime	abstime	bool	less-than
<	bool	bool	bool	less-than
<	box	box	bool	less-than
<	bpchar	bpchar	bool	less-than
<	char	char	bool	less-than
<	cidr	cidr	bool	less-than
<	circle	circle	bool	less-than
<	date	date	bool	less-than
<	datetime	datetime	bool	less-than
<	float4	float4	bool	less-than
<	float4	float8	bool	less-than
<	float8	float4	bool	less-than
<	float8	float8	bool	less-than
<	inet	inet	bool	less-than
<	int2	int2	bool	less-than

	Argumentos			
Operador	Izquierdo	Derecho	Resultado	Descripción
<	int2	int4	bool	less-than
<	int4	int2	bool	less-than
<	int4	int4	bool	less-than
<	int4	int8	bool	less-than
<	int8	int4	bool	less-than
<	int8	int8	bool	less-than
<	lseg	lseg	bool	less-than
<	macaddr	macaddr	bool	less-than
<	money	money	bool	less-than
<	name	name	bool	less-than
<	numeric	numeric	bool	lower-than
<	oid	oid	bool	less-than
<	oid8	oid8	bool	less-than
<	path	path	bool	less-than
<	reltime	reltime	bool	less-than
<	text	text	bool	less-than
<	time	time	bool	less-than
<	timespan	timespan	bool	less-than
<	timestamp	timestamp	bool	less-than
<	tinterval	tinterval	bool	less-than
<	varchar	varchar	bool	less-than
<#>	abstime	abstime	tinterval	convert to interval
<->	box	box	float8	distance between
<->	circle	circle	float8	distance between
<->	circle	polygon	float8	distance between
<->	line	box	float8	distance between
<->	line	line	float8	distance between
<->	lseg	box	float8	distance between
<->	lseg	line	float8	distance between
<->	lseg	lseg	float8	distance between
<->	path	path	float8	distance between
<->	point	box	float8	distance between
<->	point	circle	float8	distance between
<->	point	line	float8	distance between
<->	point	lseg	float8	distance between
<->	point	path	float8	distance between
<->	point	point	float8	distance between
<->	polygon	polygon	float8	distance between
<<	box	box	bool	is left of
<<	cidr	cidr	bool	is-subnet
<<	circle	circle	bool	is left of
<<	inet	inet	bool	is-subnet
<<	point	point	bool	is left of
<<	polygon	polygon	bool	is left of
<<	tinterval	tinterval	bool	less-than
<<=	cidr	cidr	bool	is-subnet-or-equal
<<=	inet	inet	bool	is-subnet-or-equal
<=	abstime	abstime	bool	less-than-or-equal
<=	box	box	bool	less-than-or-equal
<=	bpchar	bpchar	bool	less-than-or-equal
<=	char	char	bool	less-than-or-equal
<=	cidr	cidr	bool	less-than-or-equal
<=	circle	circle	bool	less-than-or-equal
<=	date	date	bool	less-than-or-equal
<=	datetime	datetime	bool	less-than-or-equal
<=	float4	float4	bool	less-than-or-equal
<=	float4	float8	bool	less-than-or-equal

	Argumentos			
Operador	Izquierdo	Derecho	Resultado	Descripción
<=	float8	float4	bool	less-than-or-equal
<=	float8	float8	bool	less-than-or-equal
<=	inet	inet	bool	less-than-or-equal
<=	int2	int2	bool	less-than-or-equal
<=	int2	int4	bool	less-than-or-equal
<=	int4	int2	bool	less-than-or-equal
<=	int4	int4	bool	less-than-or-equal
<=	int4	int8	bool	less-than-or-equal
<=	int8	int4	bool	less-than-or-equal
<=	int8	int8	bool	less-than-or-equal
<=	lseg	lseg	bool	less-than-or-equal
<=	macaddr	macaddr	bool	less-than-or-equal
<=	money	money	bool	less-than-or-equal
<=	name	name	bool	less-than-or-equal
<=	numeric	numeric	bool	lower-than-or-equal
<=	oid	oid	bool	less-than-or-equal
<=	oid8	oid8	bool	less-than-or-equal
<=	path	path	bool	less-than-or-equal
<=	reltime	reltime	bool	less-than-or-equal
<=	text	text	bool	less-than-or-equal
<=	time	time	bool	less-than-or-equal
<=	timespan	timespan	bool	less-than-or-equal
<=	timestamp	timestamp	bool	less-than-or-equal
<=	tinterval	tinterval	bool	less-than-or-equal
<=	varchar	varchar	bool	less-than-or-equal
<>	abstime	abstime	bool	not equal
<>	bool	bool	bool	not equal
<>	bpchar	bpchar	bool	not equal
<>	char	char	bool	not equal
<>	cidr	cidr	bool	not equal
<>	circle	circle	bool	not equal
<>	date	date	bool	not equal
<>	datetime	datetime	bool	not equal
<>	float4	float4	bool	not equal
<>	float4	float8	bool	not equal
<>	float8	float4	bool	not equal
<>	float8	float8	bool	not equal
<>	inet	inet	bool	not equal
<>	int2	int2	bool	not equal
<>	int2	int4	bool	not equal
<>	int4	int2	bool	not equal
<>	int4	int4	bool	not equal
<>	int4	int8	bool	not equal
<>	int8	int4	bool	not equal
<>	int8	int8	bool	not equal
<>	lseg	lseg	bool	not equal
<>	macaddr	macaddr	bool	not equal
<>	money	money	bool	not equal
<>	name	name	bool	not equal
<>	numeric	numeric	bool	not equal
<>	oid	oid	bool	not equal
<>	oid8	oid8	bool	less-than
<>	reltime	reltime	bool	not equal
<>	text	text	bool	not equal
<>	time	time	bool	not equal
<>	timespan	timespan	bool	not equal
<>	timestamp	timestamp	bool	not equal

	Argumentos			
Operador	Izquierdo	Derecho	Resultado	Descripción
<>	tinterval	tinterval	bool	not equal
<>	varchar	varchar	bool	not equal
<?>	abstime	tinterval	bool	abstime in tinterval
<^	box	box	bool	is below
<^	circle	circle	bool	is below
<^	point	point	bool	is below
=	_abstime	_abstime	bool	equal
=	_aclitem	_aclitem	bool	equal
=	_bool	_bool	bool	equal
=	_box	_box	bool	equal
=	_bytea	_bytea	bool	equal
=	_char	_char	bool	equal
=	_cid	_cid	bool	equal
=	_filename	_filename	bool	equal
=	_float4	_float4	bool	equal
=	_float8	_float8	bool	equal
=	_int2	_int2	bool	equal
=	_int28	_int28	bool	equal
=	_int4	_int4	bool	equal
=	_lseg	_lseg	bool	equal
=	_name	_name	bool	equal
=	_oid	_oid	bool	equal
=	_oid8	_oid8	bool	equal
=	_path	_path	bool	equal
=	_point	_point	bool	equal
=	_polygon	_polygon	bool	equal
=	_regproc	_regproc	bool	equal
=	_reltime	_reltime	bool	equal
=	_text	_text	bool	equal
=	_tid	_tid	bool	equal
=	_tinterval	_tinterval	bool	equal
=	_xid	_xid	bool	equal
=	abstime	abstime	bool	equal
=	bool	bool	bool	equal
=	box	box	bool	equal
=	bpchar	bpchar	bool	equal
=	char	char	bool	equal
=	cidr	cidr	bool	equal
=	circle	circle	bool	equal
=	date	date	bool	equal
=	datetime	datetime	bool	equal
=	float4	float4	bool	equal
=	float4	float8	bool	equal
=	float8	float4	bool	equal
=	float8	float8	bool	equal
=	inet	inet	bool	equal
=	int2	int2	bool	equal
=	int2	int4	bool	equal
=	int4	int2	bool	equal
=	int4	int4	bool	equal
=	int4	int8	bool	equal
=	int4	oid	bool	equal
=	int8	int4	bool	equal
=	int8	int8	bool	equal
=	line	line	bool	lines equal?
=	lseg	lseg	bool	equal
=	macaddr	macaddr	bool	equal

	Argumentos			
Operador	Izquierdo	Derecho	Resultado	Descripción
=	money	money	bool	equal
=	name	name	bool	equal
=	numeric	numeric	bool	equal
=	oid	int4	bool	equal
=	oid	oid	bool	equal
=	oid8	oid8	bool	equal
=	path	path	bool	equal
=	reltime	reltime	bool	equal
=	text	text	bool	equal
=	time	time	bool	equal
=	timespan	timespan	bool	equal
=	timestamp	timestamp	bool	equal
=	tinterval	tinterval	bool	equal
=	varchar	varchar	bool	equal
>	abstime	abstime	bool	greater-than
>	bool	bool	bool	greater-than
>	box	box	bool	greater-than
>	bpchar	bpchar	bool	greater-than
>	char	char	bool	greater-than
>	cidr	cidr	bool	greater-than
>	circle	circle	bool	greater-than
>	date	date	bool	greater-than
>	datetime	datetime	bool	greater-than
>	float4	float4	bool	greater-than
>	float4	float8	bool	greater-than
>	float8	float4	bool	greater-than
>	float8	float8	bool	greater-than
>	inet	inet	bool	greater-than
>	int2	int2	bool	greater-than
>	int2	int4	bool	greater-than
>	int4	int2	bool	greater-than
>	int4	int4	bool	greater-than
>	int4	int8	bool	greater-than
>	int8	int4	bool	greater-than
>	int8	int8	bool	greater-than
>	lseg	lseg	bool	greater-than
>	macaddr	macaddr	bool	greater-than
>	money	money	bool	greater-than
>	name	name	bool	greater-than
>	numeric	numeric	bool	greater-than
>	oid	oid	bool	greater-than
>	oid8	oid8	bool	greater-than
>	path	path	bool	greater-than
>	reltime	reltime	bool	greater-than
>	text	text	bool	greater-than
>	time	time	bool	greater-than
>	timespan	timespan	bool	greater-than
>	timestamp	timestamp	bool	greater-than
>	tinterval	tinterval	bool	greater-than
>	varchar	varchar	bool	greater-than
>=	abstime	abstime	bool	greater-than-or-equal
>=	box	box	bool	greater-than-or-equal
>=	bpchar	bpchar	bool	greater-than-or-equal
>=	char	char	bool	greater-than-or-equal
>=	cidr	cidr	bool	greater-than-or-equal
>=	circle	circle	bool	greater-than-or-equal
>=	date	date	bool	greater-than-or-equal

	Argumentos			
Operador	Izquierdo	Derecho	Resultado	Descripción
>=	datetime	datetime	bool	greater-than-or-equal
>=	float4	float4	bool	greater-than-or-equal
>=	float4	float8	bool	greater-than-or-equal
>=	float8	float4	bool	greater-than-or-equal
>=	float8	float8	bool	greater-than-or-equal
>=	inet	inet	bool	greater-than-or-equal
>=	int2	int2	bool	greater-than-or-equal
>=	int2	int4	bool	greater-than-or-equal
>=	int4	int2	bool	greater-than-or-equal
>=	int4	int4	bool	greater-than-or-equal
>=	int4	int8	bool	greater-than-or-equal
>=	int8	int4	bool	greater-than-or-equal
>=	int8	int8	bool	greater-than-or-equal
>=	lseg	lseg	bool	greater-than-or-equal
>=	macaddr	macaddr	bool	greater-than-or-equal
>=	money	money	bool	greater-than-or-equal
>=	name	name	bool	greater-than-or-equal
>=	numeric	numeric	bool	greater-than-or-equal
>=	oid	oid	bool	greater-than-or-equal
>=	oid8	oid8	bool	greater-than-or-equal
>=	path	path	bool	greater-than-or-equal
>=	reltime	reltime	bool	greater-than-or-equal
>=	text	text	bool	greater-than-or-equal
>=	time	time	bool	greater-than-or-equal
>=	timespan	timespan	bool	greater-than-or-equal
>=	timestamp	timestamp	bool	greater-than-or-equal
>=	tinterval	tinterval	bool	greater-than-or-equal
>=	varchar	varchar	bool	greater-than-or-equal
>>	box	box	bool	is left of
>>	cidr	cidr	bool	is-supernet
>>	circle	circle	bool	is left of
>>	inet	inet	bool	is-supernet
>>	point	point	bool	is left of
>>	polygon	polygon	bool	is left of
>>=	cidr	cidr	bool	is-supernet-or-equal
>>=	inet	inet	bool	is-supernet-or-equal
>^	box	box	bool	is above
>^	circle	circle	bool	is above
>^	point	point	bool	is above
?#	box	box	bool	overlaps
?#	line	box	bool	
?#	line	line	bool	lines intersect?
?#	lseg	box	bool	
?#	lseg	line	bool	
?#	lseg	lseg	bool	intersects
?#	path	path	bool	
?-		line	bool	lines horizontal?
?-		lseg	bool	is horizontal
?-	point	point	bool	is horizontal
?-|	line	line	bool	lines perpendicular?
?-|	lseg	lseg	bool	is perpendicular to
?|		line	bool	lines vertical?
?|		lseg	bool	is vertical
?|	point	point	bool	is vertical
?||	line	line	bool	lines parallel?
?||	lseg	lseg	bool	is parallel to
@		float4	float4	absolute value

	Argumentos			
Operador	Izquierdo	Derecho	Resultado	Descripción
@		float8	float8	absolute value
@		numeric	numeric	absolute value
@	box	box	bool	contained in
@	circle	circle	bool	
@	lseg	box	bool	contained in
@	lseg	line	bool	contained in
@	point	box	bool	point is inside
@	point	circle	bool	
@	point	line	bool	contained in
@	point	lseg	bool	contained in
@	point	path	bool	contained in
@	point	polygon	bool	point contained by polygon?
@	polygon	polygon	bool	contained in
@-@		lseg	float8	distance between endpoints
@-@		path	float8	sum of path segments
@@		box	point	center of
@@		circle	point	center of
@@		lseg	point	center of
@@		path	point	center of
@@		polygon	point	center of
^	float8	float8	float8	exponentiation
|		tinterval	abstime	start of interval
|/		float8	float8	square root
||	bpchar	bpchar	text	concatenate
||	text	text	text	concatenate
||	varchar	varchar	text	concatenate
||/		float8	float8	cube root
~	_aclitem	aclitem	bool	matches regex., case-sensitive
~	box	box	bool	contains
~	bpchar	text	bool	matches regex., case-sensitive
~	circle	circle	bool	contains
~	circle	point	bool	
~	name	text	bool	matches regex., case-sensitive
~	path	point	bool	path contains point?
~	polygon	point	bool	polygon contains point?
~	polygon	polygon	bool	contains
~	text	text	bool	matches regex., case-sensitive
~	varchar	text	bool	matches regex., case-sensitive
~*	bpchar	text	bool	matches regex., case-insensitive
~*	name	text	bool	matches regex., case-insensitive
~*	text	text	bool	matches regex., case-insensitive
~*	varchar	text	bool	matches regex., case-insensitive
~=	box	box	bool	same as
~=	circle	circle	bool	same as
~=	point	point	bool	same as
~=	polygon	polygon	bool	same as
~=	tinterval	tinterval	bool	same as
~~	bpchar	text	bool	matches LIKE expression
~~	name	text	bool	matches LIKE expression
~~	text	text	bool	matches LIKE expression
~~	varchar	text	bool	matches LIKE expression

9.4 Cursores

El concepto de cursores es el de un índice que se desliza sobre el resultado de una consulta y que permite traer unas cuantas tuplas dentro del total de las resultantes. Se utiliza siempre dentro de una transacción para garantizar permanencia.

```
BEGIN;
DECLARE ccur CURSOR FOR SELECT * FROM mitabla;
FETCH 1 IN ccur;            # lee la siguiente tupla
FETCH 5 IN ccur;            # lee las siguientes cinco tuplas
FETCH FORWARD 5 IN ccur;    # lee las siguientes cinco tuplas
FETCH BACKWARD 1 IN ccur;   # lee la tupla anterior
FETCH ALL IN ccur;          # lee las tuplas restantes
END;
```

Capítulo 10

Lenguajes procedurales

A partir de la versión 6.3 de PostgreSQL se permite la inclusión de lenguajes procedurales. El DBMS no sabe interpretar las funciones o *triggers* escritos en estos lenguajes, sino que pasa el control al *parser/ejecutor*, llamado manejador, del lenguaje pertinente. El manejador es un módulo compartido que se carga bajo demanda.

Los dos lenguajes que son distribuidos junto con PostgreSQL, pero no integrados son PL/pgSQL y PL/Tcl. Al momento de hacer la instalación, sólo PL/pgSQL se construye y se incluye en el directorio de librerías. Para PL/Tcl es necesario reconfigurar el DBMS y recompilarlo.

10.1 Instalando lenguajes procedurales

Es necesario dar de alta el lenguaje procedural que se va a emplear en cada base de datos o, alternativamente, el usuario `postgres` puede darlos de alta en la base `template1` y desde ese momento todas las bases que sean creadas heredarán estos lenguajes.

En primer lugar es necesario crear la función que se encargará de procesar el lenguaje:

```
DROP FUNCTION plpgsql_call_handler ();
CREATE FUNCTION plpgsql_call_handler () RETURNS OPAQUE
    AS '/usr/lib/pgsql/plpgsql.so' LANGUAGE 'C';
```

En este caso el directorio es el que corresponde en caso de que se haya instalado PostgreSQL desde un `rpm` de RedHat. Por supuesto, si se instaló PostgreSQL compilándolo (muy sana opción), el directorio correspondiente será `/usr/local/pgsql/lib/...`.

El tipo `OPAQUE` le indica al DBMS que la función regresa un tipo que no es de los construidos o definidos por el DBMS y que el valor retornado no es usable directamente dentro de un comando SQL.

Después de crear la función `handler`, debemos de dar de alta el lenguaje en sí y, de nuevo, por sanidad primero borramos la referencia a un posible intento previo, aunque cabe aclarar que cualquier función en lenguaje procedural que se haya creado con anterioridad dejará de funcionar, porque en `pg_proc`, lugar donde se almacenan las funciones, quedará la referencia al `oid` anterior del lenguaje. Esto es muy importante, una vez que se haya dado de alta el lenguaje, todas las funciones creadas, harán referencia al `oid` con que fué creado y lo mismo se aplica al lenguaje con respecto a la función `handler` que lo invoca.

```
DELETE FROM pg_language WHERE lanname='plpgsql';
CREATE TRUSTED PROCEDURAL LANGUAGE 'plpgsql' HANDLER
    plpgsql_call_handler LANCOMPILER 'PL/pgSQL';
```

En adelante podremos utilizar el lenguaje procedural SQL. El parámetro `TRUSTED`, como su nombre lo indica, se refiere a que dicho lenguaje, es confiable para la base de datos. Esto permite que cualquier

usuario sin atribuciones de súper usuario puede crear funciones y *triggers*. Esto se debe a que el lenguaje es interpretado dentro del `backend` de la base de datos y podría tener acceso a funciones internas y al sistema de archivos, con gran riesgo. PL/pgSQL y PL/Tcl son confiables en el sentido de que no permiten accesar dichas partes del `backend`.

El mecanismo de ejecución de funciones definidas en lenguajes procedurales es a *grosso modo* el siguiente: uno de los argumentos dados al manejador es el `OID` de la función en la tabla `pg_proc`. El manejador examina varios catálogos del sistema y analiza los argumentos de llamada y el tipo de dato a regresar. El código fuente de la función reside en el campo `prosrc` de la tabla `pg_proc`.

Debido a esto último, y en contraste con las funciones en C, las funciones en lenguajes procedurales pueden ser sobrecargadas al igual que las escritas en SQL. Se pueden tener varias funciones con el mismo nombre, siempre y cuando los argumentos con que son llamadas difieran en número o en tipo.

10.2 Usando PL/pgSQL

Antes de hablar del lenguaje en sí, hay que dar crédito al autor del lenguaje: Jan Wieck.

Las metas del diseño de PL/pgSQL como lenguaje procedural son:

- que pueda ser utilizado para crear funciones y *triggers*;
- añadir estructuras de control al lenguaje SQL;
- que sea capáz de realizar cómputos complejos;
- que herede todas las definiciones de usuario como tipos, funciones y operadores;
- que sea confiable para correr dentro del DBMS;
- de fácil empleo.

El manejador de PL/pgSQL analiza el código fuente de las funciones y produce un árbol de instrucciones en código binario interno la primera vez que es llamado por el DBMS. El código binario producido es identificado dentro del manejador por el `OID` de la función. Esto garantiza que al cambiar la función con una secuencia `DROP/CREATE`[1], el cambio tendrá efecto sin necesidad de realizar una nueva conexión al servidor.

Para todas las expresiones y secuencias SQL utilizadas en la función, el intérprete de código PL/pgSQL binario crea y prepara un plan de ejecución usando los controladores `SPI`[2] `SPI_prepare()` y `SPI_saveplan()`. Esto se hace la primera vez que cada secuencia individual es procesada en la función PL/pgSQL. Así, una función con código condicional que contiene varias secuencias para las cuales un plan de ejecución será requerido, sólo prepararán y salvarán aquellos planes que realmente serán utilizados durante la vida de la conexión al DBMS.

Incluso es posible crear funciones de cómputo condicional complejo y utilizarlas posteriormente para definir operadores o utilizarlas en índices funcionales. Sin embargo, para conversiones de entrada o salida o cálculos sobre tipos de datos definidos por el usuario, no es posible realizarlos en PL/pgSQL.

10.3 Estructura de PL/pgSQL

El lenguaje PL/pgSQL es no hace diferencia entre mayúsculas y minúsculas. Todas las palabras reservadas e identificadores pueden aparecer en una mezcla de mayúsculas y minúsculas.

PL/pgSQL es un lenguaje por bloques. Se define un bloque como:

[1] Actualmente no se cuenta con un mecanismo como `ALTER FUNCTION`.

[2] SPI: *Server Programming Interface*. Enlace para programar del lado del servidor del DBMS.

```
[<<etiqueta>>]
[DECLARE
    declaraciones]
BEGIN
    aserciones
END;
```

Puede haber cualquier número de bloques anidados en la sección de aserciones de un bloque. Los bloques anidados pueden ser utilizados para ocultar variables de la parte exterior del bloque de aserciones. Las variables declaradas en la sección de declaraciones que precede a un bloque son inicializadas al valor por omisión cada vez que el control pasa al interior del bloque, no solamente cada vez que la función es invocada.

Es importante no confundir la estructura de control `BEGIN` . . . `END` para agrupar aserciones de PL/pgSQL con los comandos del DBMS para control de transacciones. Las funciones y los *triggers* no pueden comenzar o hacer `commit` de transacciones y PostgreSQL no tiene transacciones anidadas.

10.3.1 Comentarios en PL/pgSQL

Existen dos tipos de comentarios en PL/pgSQL. Un doble guión `--` comineza un comentario que termina con el fin de la línea donde aparece. La secuencia `/*` comienza un bloque de comentario que termina con la aparición de la secuencia `*/`. Los bloques de comentarios no pueden aparecer anidados, pero un doble guión puede aparecer dentro de un bloque de comentario y a su vez, el doble guión puede ocultar los delimitadores de bloque de comentarios `/*` y `*/`.

10.3.2 Bloque de declaraciones

Todas las variables, renglones y registros utilizados en un bloque o sus bloques anidados deberán ser declarados en la sección de declaraciones del bloque excepto por las variables de control de ciclos de un `FOR` que itera sobre un rango de valores enteros. Los parámetros dados a una función PL/pgSQL son automáticamente declarados con lo identificadores usuales `$n`, donde `n` toma los valores desde 1 hasta el número de argumentos pasados a la función. Las declaraciones tienen la siguiente sintáxis:

```
name [ CONSTANT ] type [ NOT NULL ] [ DEFAULT | := value ];
```

Declara una variable del tipo base especificado. Si la variable es declarada como `CONSTANT`, el valor no podrá ser alterado dentro de la función. Si el modificador `NOT NULL` es especificado, la asignación de un valor `NULL` resultará en un error de ejecución. Como el valor por omisión de todas las variables en SQL es `NULL`, todas aquellas que sean declaradas como `NOT NULL` deberán tener un valor por omisión especificado.

El valor por omisión es evaluado cada vez que la función es llamada. Así, asignando el valor `'now'` a una variable de tipo `datetime` causará que la variable contenga la fecha y hora al momento de ser invocada, no el de cuando la función fué precompilada en código binario.

```
name class %ROWTYPE;
```

Declara un renglón con la estructura de registros de la clase dada. La clase debe de ser una tabla existente o una vista en la base de datos. Los campos del rengón son accedidos en notación punto[3]. Los parámetros a una función pueden ser tipos compuestos (renglones completos de una tabla). En este caso, el identificador correspondiente `$n` será de tipo renglón, pero deberá utilizarse por medio de un alias utilizando la instrucción `ALIAS` descrita más adelante. Sólo los atributos de usuario del renglón de una tabla son accesibles en el renglón, no pueden obtenerse las entradas de `OID` ni de otros atributos del sistema, dado que el renglón puede provenir de una vista y las vistas no tienen atributos del sistema utilizables.

Los campos de tipo renglón heredan los tamaños de campos y precisiones numéricas de los tipos definidos en las tablas.

[3] Es decir, el campo `nombre` de la tabla `personas` será referenciado como `personas.nombre`.

```
name RECORD;
```

Los registros son similares a los tipos renglón, pero no tienen una estructura predefinida. Son utilizados en las selecciones y en los ciclos `FOR` para contener un renglón de una operación `SELECT`. El mismo registro puede ser utilizado varias veces. Acceder a un registro o intentar asignar un valor a un campo de un registro cuando no existe un renglón actual resulta en un error de ejecución.

Los renglones `NEW` y `OLD` en un *trigger* son pasados al procedimiento como registros automáticamente. Esto es necesario porque en PostgreSQL un único *trigger* puede manejar eventos para diferentes tablas.

```
name ALIAS FOR $n;
```

Para mejorar la lectura del código, es posible definir nombres alternos para un parámetro posicional a una función.

Este método es requerido para tipos compuestos que son pasados como argumentos a funciones. La notación punto `$1.salario` empleada en SQL no es permitida en PL/pgSQL.

```
RENAME oldname TO newname;
```

Permite cambiar el nombre de una variable, registro o renglón. Es útil si `NEW` u `OLD` deben de ser refernciados por otro nombre dentro de un procedimiento *trigger*.

10.3.3 Tipos de datos

El tipo de una variable puede ser cualquiera de los tipos base existentes en la base de datos. `type` en la sección de declaraciones anterior se define como:

- Tipo base de PostgreSQL
- `variable%TYPE`
- `clase.campo%TYPE`

`variable` es el nombre de una variable previamente declarada en la misma función y que es visible en ese punto. `clase` es el nombre de una tabla existente o una vista donde `campo` es el nombre de un atributo válido.

Utilizar el tipo `clase.campo%TYPE` causa que PL/pgSQL busque las definiciones de atributos en la primera invocación de la función, durante la existencia de la conexión con servidor. Supongamos que se tiene una tabla con un atributo de tipo `char(20)` y que algunas funciones PL/pgSQL que trabajan con el contenido en variables locales. Luego, se decide que `char(20)` no es suficiente para almacenar la información, se hace un `dump`, se destruye la tabla y se recrea con la nueva definición para el atributo como `char(40)` y restaura la información. En este caso, olvidamos las funciones definidas. Los cálculos dentro de ellas truncarán el valor a veinte carácteres. Pero si el campo fué declarado como `clase.campo%TYPE`, entoces las funciones automáticamente trabajarán con el cambio de tamaño, incluso si la nueva declaración del campo es del tipo `text`.

10.3.4 Expresiones

Todas las expresiones empleadas en PL/pgSQL son procesadas utilizando el ejecutor del servidor remoto. Incluso las expresiones que aparentemente contienen constantes deberán ser evaluadas en tiempo de ejecución (por ejemplo el caso 'now' que se menciona en la subsección 10.3.2), de tal manera que es imposible para el reconocedor sintáctico de PL/pgSQL identificar valores constantes distintos a `NULL`. Todas las expresiones son evaluadas internamente ejecutando la aserción `SELECT expresion` utilizando el controlador SPI. En la expresión, las ocurrencias de identificadores de variables son substituidos por parámetros y los valores actuales

de las variables son pasadas al ejecutor en el arreglo de parámetros. Todas las expresiones empleadas en las funciones de PL/pgSQL son preparadas y salvadas sólo una vez.

La comparación de tipos echa por el reconocedor sintáctico de PostgreSQL tiene algunos efectos colaterales para la interpretación de valores constantes. En particular hay una diferencia entre lo que las dos funciones

```
CREATE FUNCTION logfunc1 (text) RETURNS datetime AS '
    DECLARE
        logtxt ALIAS FOR $1;
    BEGIN
        INSERT INTO logtable VALUES (logtxt, ''now'');
        RETURN ''now'';
    END;
' LANGUAGE 'plpgsql';
```

y

```
CREATE FUNCTION logfunc2 (text) RETURNS datetime AS '
    DECLARE
        logtxt ALIAS FOR $1;
        curtime datetime;
    BEGIN
        curtime := ''now'';
        INSERT INTO logtable VALUES (logtxt, curtime);
        RETURN curtime;
    END;
' LANGUAGE 'plpgsql';
```

hacen. En el caso de la función `logfunc1()` el reconocedor sintáctico sabe cuando prepara el plan para el `INSERT`, que la cadena `'now'` debe de ser interpretada como del tipo `datetime` porque el campo destino de `logtable` es de ese tipo. De esta manera, lo almacenerá como una constante en ese instante y este valor constante será empleado en las posteriores invocaciones, durante el tiempo de la conexión. No es necesario aclarar que esto no es lo que el programador tenía en mente al definir la función.

En el caso de la función `logfunc2()`, el reconocedor sintáctico no conoce el tipo al que `'now'` deberá de ser convertido y por lo tanto regresa un tipo de datos `text` conteniendo la cadena `'now'`. Durante la asignación a la variable `curtime` al momendo de ejecución, el intérprete de PL/pgSQL convierte esta cadena al tipo `datetime` invocando a las funciones `text_out()` y `datetime_in()` para la conversión.

Esta verificación de tipos hecha por el reconocedor sintáctico fué implementado después de que PL/pgSQL fuese integrado por completo en PostgreSQL. Es una diferencia entre las versiones 6.3 y 6.4 y afecta a todas las funciones que usan la preparación de planes de ejecución realizada por el controlador SPI. Utilizando una variable local en la forma previamente descrita es la única manera en que PL/pgSQL pueda interpretar este tipo de valores correctamente.

Si los registros de campos son empleados en expresiones o aserciones, el tipo de los datos de los campos no deberá de cambiar durante las invocaciones a la misma expresión. Es indispensable mantener esto en mente al escribir *triggers* que manejen eventos para más de una tabla.

10.3.5 Aserciones

Cualquier cosa que no sea entendida por el reconocedor sintáctico de PL/pgSQL como se especifica más adelante, será colocado en una consulta enviada al DBMS para su ejecución. La consulta resultante no deberá de regresar ningún dato.

Asignación.

La asignación de un valor a una variable, renglón o campo de un registro, se escribe como

```
identifier := expression;
```

Si el tipo de dato resultante de la expresión no equivale al tipo de dato de la variable, o si la variable tiene un tamaño distinto, el valor resultante será ajustado implícitamente por el compilador de PL/pgSQL utilizando el tipo resultante y el tipo definido para la variable. Nótese que esto podría potencialmente generar errores en tiempo de ejecución por parte de las funciones de conversión.

Una asignación de una selección completa en un registro o renglón puede ser hecho de la siguiente manera

```
SELECT expresiones INTO destino FROM ...;
```

`destino` puede ser un registro, una variable reglón o una lista separada por comas de variables y campos de registro o renglón.

Si un renglón o lista de variables es empleada como destino, los valores deberán coincidir exactamente con la estructura del destino o un error en tiempo de ejecución ocurrirá. La palabra reservada `FROM` puede ser seguida por cualquier calificación válida, agrupamiento, ordenamiento, etc., que sea válida en una aserción `SELECT`.

Existe una variable especial llamada `FOUND` de tipo booleano que puede ser empleada inmediatamente después de un `SELECT INTO` para verificar que la asignación tuvo éxito.

```
SELECT * INTO myrec FROM EMP WHERE empname = myname;
IF NOT FOUND THEN
    RAISE EXCEPTION ''El empleado % no fue encontrado'', myname;
END IF;
```

Si la selección regresa varios renglones, sólo el primero será asignado a los campos destino, el resto será silenciosamente ignorado.

Invocando a otra función.

Todas las funciones definidas en PostgreSQL regresan un valor. De tal forma que, la forma natural de llamar a una función es ejecutando una consulta `SELECT` o realizando una asignación (que resulte en un `SELECT` interno a PL/pgSQL). Pero hay casos en que alguien no esté interesado en el resultado de la función.

```
PERFORM consulta
```

ejecuta un `SELECT consulta` sobre el controlador SPI y descarta el valor. Sin embargo, de todas formas los identificadores como variables locales son reemplazados por parámetros.

Regresando de la función.

```
RETURN expresion
```

La función termina y el valor de `expresion` será retornado como valor al invocante. El valor no puede ser indefinido. Si el control llega al final de la función sin regresar explícitamente un valor por medio de `RETURN`, se generará un error en tiempo de ejecución.

Las expresiones resultantes serán automáticamente convertidas al tipo de dato de la función, tal como se describe en la asignación que la invoca.

Abortando y mensajes.

Como se pudo observar en los ejemplos anteriores, existe una aserción `RAISE` que se utiliza para escribir mensajes en el mecanismo `elog` de PostgreSQL.

```
RAISE nivel ''formato'' [, identificador [...]];
```

En la parte de formato, % se utiliza como contenedor para la serie de identificadores subsiguientes, separados por comas. Los niveles posibles son `DEBUG` (que sólo actúan en versiones de depuración del DBMS), `NOTICE` (escritas en la bitácora del DBMS y enviadas también a la aplicación del cliente) y `EXCEPTION` (escritas en la bitácora del DBMS y abortan la aplicación).

Condicionales.

```
IF expresion THEN
    asercion
[ELSE
    asercion]
END IF;
```

La expresión debe de regresar un valor que pueda ser convertido en tipo booleano.

Ciclos.

Existen varios tipos de ciclos:

```
[<<etiqueta>>]
LOOP
    aserciones
END LOOP;
```

Es un ciclo incondicional que deberá ser terminado explícitamente con una aserción `EXIT`. La etiqueta opcional puede ser usada por las aserciones `EXIT` de ciclos anidados para especificar que nivel de anidamiento debe ser terminado.

```
[<<etiqueta>>]
WHILE expresion LOOP
    aserciones
END LOOP;
```

Es un ciclo condicional que es ejecutado mientras que la evaluación de la expresión sea verdadera.

```
[<<etiqueta>>]
FOR nombre IN [ REVERSE ] expresion .. expresion LOOP
    aserciones
END LOOP;
```

Es un ciclo que itera sobre un rango de valores enteros. La variable `nombre` es creada automáticamente con tipo entero y existe sólo al interior del ciclo. Las dos expresiones dan los límites inferior y superior del rango en que habrá de ser evaluada la variable y sólo se calculan al iniciar el ciclo. El paso de iteración siempre es uno.

```
[<<etiqueta>>]
FOR registro | renglon IN clausula_de_seleccion LOOP
    aserciones
END LOOP;
```

Al registro o renglón le será asignado cada uno de los renglones resultantes de la cláusula de selección. Si el ciclo termina con una aserción `EXIT`, el último renglón asignado es aún accesible al terminar el ciclo.

```
EXIT [ etiqueta ] [ WHEN expresion ];
```

Si no se dá una etiqueta, el ciclo interior será terminado y la acción a continuación de la secuencia `END LOOP` será ejecutada a continuación. Si la etiqueta es dada, deberá ser la etiqueta del nivel actual o de un nivel exterior de anidamiento de bloques de ciclo. En este caso el ciclo invocado por la etiqueta termina y el control es pasado a la siguiente aserción a continuación del final de ciclo o bloque indicado por la etiqueta.

Capítulo 11

Triggers

Lo pescaron como al Trigger de Santa Julia...

Esto de los *trigger* es de lo más simpático. Resulta que uno puede ir y automatizar acciones que usualmente se ejecutan antes o después de una consulta, de tal manera que al llevar a cabo la consulta se "dispare" la ejecución de determinadas acciones, sin intervención humana.

Los *trigger* se pueden escribir en cualquiera de los siguientes lenguajes: C, PL/pgSQL y PL/Tcl. Sin embargo, examinando la tabla `pg_language` se puede observar que existe la referencia a Lisp:

```
mancha=> select * from pg_language;
lanname |lanispl|lanpltrusted|lanplcallfoid|lancompiler
--------+-------+------------+-------------+--------------
internal|f      |f           |            0|n/a
lisp    |f      |f           |            0|/usr/ucb/liszt
C       |f      |f           |            0|/bin/cc
sql     |f      |f           |            0|postgres
plpgsql |t      |t           |     10979662|PL/pgSQL
(5 rows)
```

pero no existe documentación acerca de cómo se emplea.

Los *triggers* siempre deben de ser declarados como funciones sin argumentos y de tipo `OPAQUE`.

PostgreSQL tiene varias particularidades respecto a *triggers* que deben de ser tomadas en cuenta. La primera de ellas es la creación automática de ciertas variables que son visibles a la función:

`NEW` Variable de tipo `RECORD` y que contiene el registro nuevo en el caso de un *trigger* disparado con `INSERT` o `UPDATE` a nivel de renglones.

`OLD` Variable de tipo `RECORD` y que contiene el registro anterior en el caso de un *trigger* disparado con `DELETE` o `UPDATE` a nivel de renglones.

`TG_NAME` Variable de tipo `name` que contiene el nombre del *trigger* disparado.

`TG_WHEN` Variable de tipo `text` que contiene la cadena '`BEFORE`' o la cadena '`AFTER`' dependiendo de la definición del *trigger*.

`TG_LEVEL` Variable de tipo `text` que contiene la cadena '`ROW`' o la cadena '`STATEMENT`' dependiendo de la definición del *trigger*.

`TG_OP` Variable de tipo `text` que contiene alguna de las cadenas '`INSERT`', '`UPDATE`' o '`DELETE`' indicando que operación disparó al *trigger*.

`TG_RELID` Variable de tipo `oid` que contiene `el OID` de la tabla que disparó el *trigger*.

`TG_RELNAME` Variable de tipo `name` que contiene el nombre de la tabla que disparó el *trigger*.

`TG_NARGS` Variable de tipo `integer` que contiene el número de argumentos dados al *trigger* durante la creación del mismo.

`TG_ARGV[]` Variable de tipo arreglo de `text` que contiene los argumentos dados al momento de crear el *trigger*. El índice comienza en cero y puede ser indicado como una expresión. Índices no válidos ($< 0 || \geq tg\_nargs$) regresan `NULL`. Este es el único mecanismo para pasar argumentos a un *trigger*, ya que como se recordará la función activada no puede tener argumentos.

En segundo lugar, deben de regresar o `NULL` o un registro o renglón conteniendo exactamente la misma estructura de la tabla con la que fué disparado el *trigger*. Los *triggers* disparados por una acción `AFTER` siempre deberán de regresar el valor `NULL` sin ningún efecto. Los *triggers* disparados por una acción `BEFORE` sólo deberán de regresar el valor `NULL` cuando se deba de invalidar la acción para el renglón que lo dispara. De otra manera, el renglón o registro regresado reemplaza al insertado o actualizado por la operación. Es posible reemplazar valores en particular directamente en `NEW` y regresar éste o construir un renglón o registro completamente nuevo que suplante al original.

11.1 Manejo de excepciones

PostgreSQLno tiene un modelo de manejo inteligente de excepciones. En cualquier momento que el reconocedor sintáctico, el planeador, el optimizador o el ejecutor deciden que una aserción no puede ser procesada, la transacción completa se aborta y el sistema regresa al estado de espera de consultas por parte del cliente.

Sin embargo, es posible engancharse al mecanismo de manejo de errores para notar cuando ocurre esto. Pero actualmente es imposible decir qué fue lo que realmente causó el aborto de la operación (error de conversión en la entrada o la salida, error de punto flotante, error del reconocedor sintáctico, etc.). Es también posible que el DBMS se encuentre en un estado inconsistente en este punto, de tal manera que regresar al ejecutor o ingresar otros comandos puede corromper severamente la base de datos. De tal forma que si luego de informar al cliente que la transacción fué abortada, lo más sensato es no continuar con la operación.

De esta forma, lo único que PL/pgSQL hace actualmente cuando se encuentra ante un aborto de operación durante la ejecución de un *trigger*, es escribir en la bitácora alguna información describiendo el número de línea y el tipo de aserción que estaba siendo ejecutada.

Un detalle a tomar en cuenta es que al escribir las funciones en PL/pgSQL es el manejo de los apóstrofes. El texto de las funciones dentro de `CREATE FUNCTION` va entre apóstrofes, así que si estos ocurren dentro del cuerpo de la función deberán ir dobles o escapados con una diagonal: `''` o `\'`.

11.2 Algunos ejemplos simples de funciones en PL/pgSQL

Las siguientes dos funciones en PL/pgSQL son realmente simples y se muestran su ejecución para mayor claridad del mecanismo.

```
CREATE FUNCTION add_one (int4) RETURNS int4 AS '
    BEGIN
        RETURN $1 + 1;
    END;
' LANGUAGE 'plpgsql';
```

```
mancha=> select add_one(5);
add_one
-------
      6
(1 row)
```

```
CREATE FUNCTION concat_text (text, text) RETURNS text AS '
    BEGIN
        RETURN $1 || $2;
    END;
' LANGUAGE 'plpgsql';
```

```
mancha=> select concat_text ('Hola ', 'Lola') as periquita;
periquita
---------
Hola Lola
(1 row)
```

11.3 Funciones PL/pgSQL en tipos compuestos

Como mencionamos, las funciones PL/pgSQL también pueden actuar sobre tipos compuestos.

Si tenemos la siguiente tabla:

```
mancha=> select nombre,salario from emp;
nombre       |salario
-------------+-------
Bill Clinton |   1000
Boris Yeltsin|   2000
(2 rows)
```

Y definimos la siguiente función:

```
CREATE FUNCTION BienPagao (EMP, int4) RETURNS bool AS '
    DECLARE
        emprec ALIAS FOR $1;
        sallim ALIAS FOR $2;
    BEGIN
        IF emprec.salario ISNULL THEN
            RETURN ''f'';
        END IF;
        RETURN emprec.salario > sallim;
    END;
' LANGUAGE 'plpgsql';
```

Podemos hacer consultas para averiguar quienes son aquellos empleados que ganan más de una determinada cantidad.

```
mancha=> select nombre,BienPagao (emp, 500) as SobrePagado from emp;
nombre       |sobrepagado
-------------+-----------
Bill Clinton |t
Boris Yeltsin|t
(2 rows)
```

En efecto, creo que ambos ganan más de lo que merecen.

11.4 Ejemplos de Trigger

Tenemos una tabla de archivos (llamada `archivos`), en la que uno de los campos es un MD5 sobre el contenido del archivo (campo llamado `dmd5`). En una tabla anexa (llamada `aux`), necesitamos llevar la cuenta de cuántos registros tenemos con el mismo MD5, para esto tenemos dos campos: `dmd5` y `cont`. Así, cada vez que insertamos un registro en `archivos` debemos de actualizar el contador correspondiente en `aux`. De igual manera, cuando borramos un registro de `archivos`, deberemos de decrementar el contador correspondiente en `aux` y en caso de que sea el único, eliminar el registro.

Obviamente esto se puede hacer desde el mismo programa de actualización, pero resulta, y es este caso en particular lo que motivó el empleo de *triggers*, que son varios programas los que actualizan esta tabla, así que mantener cada uno de ellos se vuelve un tanto cuanto engorroso. La mejor solución es emplear *triggers*. Cabe señalar que el ejemplo que se muestra a continuación funciona sólo de la versión de PostgreSQL 6.5.3 en adelante, dado que emplea el *Procedure Language* PL/pgSQL y algunas particularidades integradas a partir de esa versión.

Bueno, el código para actualizar los valores es el siguiente:

```
DROP FUNCTION inc_aux ();
CREATE FUNCTION inc_aux () RETURNS OPAQUE AS '
DECLARE
        myrec  record;
BEGIN
    SELECT * INTO myrec FROM aux WHERE aux.dmd5 = NEW.dmd5;
    IF NOT FOUND THEN
        INSERT INTO aux VALUES (NEW.dmd5, 1);
    ELSE
        UPDATE aux SET cont=cont+1 WHERE dmd5 = NEW.dmd5;
    END IF;
    RETURN NEW;
END;
' LANGUAGE 'plpgsql';
DROP TRIGGER ins_arc ON archivos;
CREATE TRIGGER ins_arc BEFORE INSERT ON archivos FOR EACH ROW
 EXECUTE PROCEDURE inc_aux();
```

Los `DROP`s antes de crear la función y el *trigger*, son para garantizar que las funciones no existen previamente y sobre todo, porque tuve que hacer un demonial de pruebas antes de que la cosa jalara.

Recordemos que los *triggers* no pueden recibir argumentos y *siempre* tienen que regresar un valor *opaco*.

El `select` se hace para averiguar si existe un registro con el MD5 y en caso de no existir lo insertamos con el contador en uno y en caso de que ya exista, incrementamos el contador en uno. Muy simple. Sólo es de notar que tuvimos que declarar un registro para hacer la consulta y saber si el registro existe en la tabla anexa para actualizarlo (`update`) o si será necesario crear uno nuevo (`insert`). El registro `NEW` se refiere al registro con que el que fué disparado el *trigger*.

Eliminamos el *trigger* para, en caso de que ya exista uno con el mismo nombre, garantizar que se ejecutará el que vamos a declarar. Como podemos ver, los *triggers* están asociados a tablas y por eso se debe de indicar de que tabla lo eliminamos al momento de efectuar el `drop`.

Ahora veamos el caso en que se eliminan registros. Por supuesto, en este caso estamos eliminando un registro que está en la base, así que no necesitamos ver primero si existe. El único considerando a tomar en cuenta es el caso en que el MD5 es único, en cuyo caso habremos de eliminar de la tabla `dmd5` el registro.

```
DROP FUNCTION dec_aux ();
CREATE FUNCTION dec_aux () RETURNS OPAQUE AS '
BEGIN
```

```
    UPDATE aux SET cont=cont-1 WHERE dmd5 = OLD.dmd5;
    DELETE FROM aux where cont < 1;
    RETURN NULL;
END;
' LANGUAGE 'plpgsql';
DROP TRIGGER del_arc ON archivos;
CREATE TRIGGER del_arc AFTER DELETE ON archivos
FOR EACH ROW EXECUTE PROCEDURE dec_aux();
```

Una manera de probarlo, es la siguiente:

```
mancha=> SELECT * FROM aux WHERE dmd5='12345678901234567890123456789012';
dmd5|cont
----+----
(0 rows)
mancha=> INSERT INTO archivos VALUES ('BORRAME','/',12345,'31-12-1999','23:59:07',
'12345678901234567890123456789012','12345678901234567890123456789012','Caserola');
INSERT 10981731 1
mancha=> SELECT * FROM aux WHERE dmd5='12345678901234567890123456789012';
                            dmd5|cont
--------------------------------+----
12345678901234567890123456789012|   1
(1 row)
mancha=> delete from archivos where arc='BORRAME';
DELETE 1
mancha=> SELECT * FROM aux WHERE dmd5='12345678901234567890123456789012';
dmd5|cont
----+----
(0 rows)
```

Ahora supongamos que tenemos un pequeño sistema de nómina y en el módulo de ABC[1] queremos tener la garantía mínima de que no se insertará un registro sin nombre o con salario negativo. Además, queremos llevar nota de quién y cuando modificó los registros. Sea la tabla:

```
CREATE TABLE emp (
    nombre text,
    salario int4,
    last_date datetime,
    last_user name);
```

Definimos la función que "estampe" los cambios:

```
CREATE FUNCTION emp_stamp () RETURNS OPAQUE AS '
    BEGIN
        -- Verifica que el nombre y el salario se hallan dado
        IF NEW.nombre ISNULL THEN
            RAISE EXCEPTION ''nombre no puede ser NULL'';
        END IF;
        IF NEW.salario ISNULL THEN
            RAISE EXCEPTION ''% no puede tener un salario NULL'', NEW.nombre;
        END IF;
```

[1] Altas, Bajas y Cambios.

```
        -- Que no tenga salario negativo (puede ser cero)
        IF NEW.salario < 0 THEN
            RAISE EXCEPTION ''% no puede tener un salario negativo'', NEW.nombre;
        END IF;
        -- Ahora estampamos quién y cuándo hizo los cambios
        NEW.last_date := ''now'';
        NEW.last_user := getpgusername();
        RETURN NEW;
    END;
' LANGUAGE 'plpgsql';
```

Creamos el *trigger*:

```
CREATE TRIGGER emp_stamp BEFORE INSERT OR UPDATE ON emp
    FOR EACH ROW EXECUTE PROCEDURE emp_stamp();
```

Y vemos un ejemplo:

```
mancha=> insert into emp values ('Meiga Pintita', 2500);
INSERT 11635171 1
mancha=> insert into  emp values ('Misha Negrita', 2500);
INSERT 11635172 1
mancha=> select * from emp;
nombre       |salario|last_date                   |last_user
-------------+-------+---------------------------+---------
Bill Clinton |   1000|Mon 03 Jan 12:28:12 2000 CST|mancha
Boris Yeltsin|   2000|Mon 03 Jan 12:28:36 2000 CST|mancha
Meiga Pintita|   2500|Wed 05 Jan 13:07:12 2000 CST|mancha
Misha Negrita|   2500|Wed 05 Jan 13:08:37 2000 CST|mancha
(4 rows)
mancha=> insert into  emp values ('Perico Cerillo', -1000000);
ERROR:  Perico Cerillo no puede tener un salario negativo
mancha=> insert into  emp values (NULL , 1000000);
ERROR:  nombre no puede ser NULL
mancha=> insert into  emp values ('Perico Cerillo', NULL);
ERROR:  Perico Cerillo no puede tener un salario NULL
```

Sin embargo, se nos olvidó hacer una verificación básica:

```
mancha=> insert into  emp values ('', 1000000);
INSERT 11635173 1
mancha=> select * from emp;
nombre       |salario|last_date                   |last_user
-------------+-------+---------------------------+---------
Bill Clinton |   1000|Mon 03 Jan 12:28:12 2000 CST|mancha
Boris Yeltsin|   2000|Mon 03 Jan 12:28:36 2000 CST|mancha
Meiga Pintita|   2500|Wed 05 Jan 13:07:12 2000 CST|mancha
Misha Negrita|   2500|Wed 05 Jan 13:08:37 2000 CST|mancha
             |1000000|Wed 05 Jan 13:09:50 2000 CST|mancha
(5 rows)
```

Queda como ejercicio verificar que no se inserten nombres en blanco.

Capítulo 12

Sistema de reglas de PostgreSQL

El sistema de reglas en PostgreSQL consiste en modificar las consultas de acuerdo a reglas almacenadas como parte de la base de datos. Dichas consultas modificadas son pasadas, en orden, al optimizador, planificador y finalmente al ejecutor. Esto es a diferencia de otros DBMS que simplemente implementan los sistemas de reglas como procedimientos y *triggers* almacenados. Este sistema es muy poderoso y puede ser empleado para procedimientos, vistas y versiones.

Para comprender cómo funciona el sistema de reglas, es necesario saber cuando es invocado y cuáles son las entradas que recibe y los resultados que arroja.

El sistema de reglas se haya entre el reconocedor sintáctico de consultas y el optimizador. Toma la salida del reconocedor, un árbol de consultas, y reescribe las reglas de acuerdo al catálogo `pg_rewrite`, donde se encuentran las reglas ya en formato de árbol de consulta con alguna información extra, y produce cero o más arboles de consulta como resultado. De esta manera, tanto la entrada como la salida es fácilmente representable como una aserción SQL.

Los arboles de consultas son una representación interna de una aserción SQL donde las partes individuales que lo componen son almacenadas independientemente. Estos arboles de consulta son visibles desde `psql` al iniciar el servidor de PostgreSQL con la bandera de `debug` en nivel cuatro. Las reglas en `pg_rewrite` están escritas en el mismo formato, aún cuando no son presentadas como la salida de depuración descrita.

Leer e interpretar un árbol de consultas requiere experiencia y conocimientos que van más allá de la intención de este documento. Sin embargo en la documentación que se distribuye con PostgreSQL se encuentra toda la información necesaria para comprender y modificar el sistema de reglas. Nuestra intención llega sólo hasta el punto donde el lector pueda utilizar de manera eficiente las reglas.

12.1 Vistas y el sistema de reglas

Una de las aplicaciones inmediatas del sistema de reglas es la implementación de vistas en PostgreSQL, que es realizado por el propio sistema. Es decir, las vistas en PostgreSQL están implementadas con el sistema de reglas. De hecho, las dos aserciones no son diferentes para PostgreSQL:

```
CREATE VIEW mivista AS SELECT * FROM mitabla;
```

expresión válida en PostgreSQL, pero que se convierte en

```
CREATE TABLE mivista (misma lista de atributos que para mitabla);
CREATE RULE "_RETmivista" AS ON SELECT TO mivista DO INSTEAD
    SELECT * FROM mitabla;
```

porque eso es precisamente lo que internamente hace el comando `CREATE VIEW`. Sin embargo, esto presenta algunos efectos colaterales. Uno de ellos es que la información acerca de la vista en los catalogos del sistema

PostgreSQL es exactamente la misma que para la tabla. Así, para el reconocedor sintáctico de consultas, no hay ninguna diferencia entre una tabla y una vista, ambas son relaciones.

12.2 Cómo funcionan las reglas `ON SELECT`

Las reglas **`ON SELECT`** son aplicadas a todas las consultas como el último paso, incluso si el comando dado es **`INSERT`**, **`UPDATE`** o **`DELETE`**. Contienen una semántica diferente a las demás en el sentido de que modifican el árbol de reconocimiento en lugar de crear uno nuevo. Por lo cual son descritas primero.

En realidad sólo podría existir una acción y esta sería una acción **`SELECT`** que es **`INSTEAD`**. Esta restricción fue hecha para hacer el sistema de reglas lo suficientemente seguro para el usuario y restringe las reglas **`ON SELECT`** a las reglas de vistas.

Como ejemplo utilizaremos dos vistas unidas que realizan algunos cálculos y otras vistas que las usan.

Capítulo 13

Herramientas

La distribución de PostgreSQL incluye dos herramientas para trabajar directamente con las bases de datos, y una utilería para hacer respaldos de las mismas. Además se pueden conseguir otras herramientas gráficas para una interacción un poco más cómoda.

13.1 psql

La herramienta canónica para trabajar en modo línea de comandos con PostgreSQL es `psql`. En este modo tenemos una herramienta completa para poder manipular las bases de datos. Este programa cuenta con ayuda en línea, por medio de la instrucción `\?`, para los comandos a `psql` y `\h` para examinar la sintáxis de las instrucciones de SQL.

También acepta opciones en la línea de comandos al momento de ejecución:

```
Usage: psql [options] [dbname]
    -a authsvc     set authentication service
    -A             turn off alignment when printing out attributes
    -c query       run single query (slash commands too)
    -d dbName      specify database name
    -e             echo the query sent to the backend
    -E             echo all queries sent to the backend
    -f filename    use file as a source of queries
    -F sep         set the field separator (default is '|')
    -h host        set database server host
    -H             turn on html3.0 table output
    -l             list available databases
    -n             don't use readline library
    -o filename    send output to filename or (|pipe)
    -p port        set port number
    -q             run quietly (no messages, no prompts)
    -s             single step mode (prompts for each query)
    -S             single line mode (i.e. query terminated by newline)
    -t             turn off printing of headings and row count
    -T html        set html3.0 table command options (cf. -H)
    -u             ask for a username and password for authentication
    -x             turn on expanded output (field names on left)
```

Entre las opciones más útiles, están `-h servidor` (conexión al *host* `servidor`) y `-d base` (conexión a la

base de datos `base`), en cuyo caso también podemos indicarle la opción `-u` para que nos solicite el usuario y la contraseña[1], en caso de que nos conectemos como un usuario distinto al de la sesión actual:

```
mancha@caserola:~$ psql -h breogan -d agn -u
Username: fml
Password:

Welcome to the POSTGRESQL interactive sql monitor:
  Please read the file COPYRIGHT for copyright terms of POSTGRESQL
[PostgreSQL 6.5.3 on i586-pc-linux-gnu, compiled by gcc 2.7.2.3]
```

Otra opción muy útil es `-l` para listar todas las bases existentes:

```
mancha@caserola:~$ psql -h breogan -l -u
Username: fml
Password:

datname     |datdba|encoding|datpath
------------+------+--------+-----------
template1   |   100|       0|template1
postgres    |   100|       0|postgres
agn         |   500|       0|agn
root        |     0|       0|root
plpgsql_test|     0|       0|plpgsql_test
dbarc       |   500|       0|dbarc
zapatos     |   500|       0|zapatos
pruebas     |   500|       0|pruebas
fml         |   534|       0|fml
mancha      |   500|       0|mancha
antiguedades|   500|       0|antiguedades
discos      |   500|       0|discos
discuas     |   500|       0|discuas
hotpizza    |   500|       0|hotpizza
videos      |   500|       0|videos
poblacion   |   500|       0|poblacion
licencias   |   500|       0|licencias
(17 rows)
```

Una combinación realmente útil estas instrucciones es la siguiente:

```
psql -Htc 'select autor,titulo,ndis,medio from discos \
order by autor,titulo' discos \
| sed -e's/<table /<table border=2/' > tabla.de.discos.html
```

equivalente a:

```
psql -Htc 'select autor,titulo,ndis,medio from discos \
order by autor,titulo' -T 'border=2' -o tabla.de.discos.html discos
```

¿Puede el lector indicar qué hacen ambas acciones?

Cabe mencionar que `psql` utiliza la biblioteca `readline` de GNU, por lo cuál cuenta con edición en la línea de comandos.

[1] Opcionalmente podríamos emplear también la opción `-p` para especificar otro puerto, en caso de que el servidor PostgreSQL de la otra máquina no esté escuchando el puerto 5432, lo cuál es una opción al momento de iniciar el servidor.

13.2 pgaccess

Es un programa de utilería escrito por Constantin Teodorescu, en `Tcl/Tk`, y aún cuando se distribuye junto con PostgreSQL, la versión más reciente se puede obtener en
`html://www.flex.ro/pgaccess`

La parte más interesante de `pgaccess` no es la parte de explotación de la información, ya que al estar escrito en `Tcl/Tk` consume demasiada memoria y no es capáz de manejarla en forma eficiente. Sin embargo, lo que hace realmente interesante a este sistema son las posibilidades que una interfáz gráfica tiene para manejar la creación de vistas.

13.3 Respaldos

Las herramientas `pg_dump` y `pg_dumpall` nos ayudan tanto como para respaldar las tablas, como para copiarlas de un sistema a otro en un formato transportable. En algunos casos al actualizar la versión de PostgreSQL será necesario primero respaldarlas con estas herramientas y para posteriormente volverlas a cargar.

`pg_dump` se emplea para respaldar una base de datos o una tabla en particular, mientras que `pg_dumpall` respalda todas las bases de datos en el sistema.

La forma más general de emplear `pg_dump` para respaldar sólo una tabla, es la siguiente:

```
pg\_dump -t mitabla mibase > mibase.mitabla.dump
```

con lo cual en el archivo `mibase.mitabla.dump` tenemos un archivo con los datos y el esquema de la tabla para poder recuperar toda la información en caso de falla del medio físico[2] o llevarlo a otro sistema.

Por ejemplo:

```
mancha@caserola:/Extras/Mancha/Musica$ pg_dump -t genero pruebas
\connect - mancha
CREATE TABLE "genero" (
        "genid" int4 DEFAULT nextval('"genero_genid_seq"') NOT NULL,
        "genero" character varying(32));
REVOKE ALL on "genero" from PUBLIC;
GRANT SELECT on "genero" to "nobody";
COPY "genero" FROM stdin;
1       Blues
2       Jazz
3       Reggae
4       Rock
5       Country
6       Folk
7       Newage
8       Soundtrack
9       Misc
10      Data
11      Sin Clasificar
12      Bazofia
\.
CREATE UNIQUE INDEX "genero_genid_key" on "genero"
    using btree ( "genid" "int4_ops" );
mancha@caserola:/Extras/Mancha/Musica$
```

[2] Que sería la única causa probable de problemas, dado que tanto PostgreSQL como Linux son lo suficientemente confiables.

Produce la salida anterior, donde vemos que lo primero que hace es fijar (o tratar de fijar) al usuario creador de la tabla, luego crea la tabla —que en caso de existir no da problemas—, incluyendo el serie asociado a uno de los campos, fija los permisos existentes, y a continuación copia los datos y finalmente crea el índice que la tabla tenía asociado.

Por supuesto, es mejor comprimir los archivos para ocupar el mínimo espacio posible. Por ejemplo, podemos hacerlo así:

```
pg_dump -t mitabla mibase | gzip -9c > mibase.mitabla.dump.gz
pg_dump -t mitabla mibase | bzip2 -c > mibase.mitabla.dump.bz
```

Ambos programas tienen una serie grande de opciones y se recomienda revisarlas con detenimiento. Ambas pueden ser consultadas con el sistema `man`.

Para recuperar la información en la base de datos —o a otra base, como mencionábamos— podemos emplear la siguiente instrucción en la línea de comandos:

```
gunzip -c mibase.mitabla.dump.gz | psql -e labase
bunzip2 -c mibase.mitabla.dump.bz | psql -e labase
```

y como vimos en el ejemplo, `psql` tiene toda la información requerida para restaurar la tabla o base completa.

Capítulo 14

Interfases

PostgreSQL se distribuye con la librería `libpq` la cuál se puede ligar con cualquier lenguaje de programación disponible en Linux como C, Ada, Pascal, Modula, fortran, etc. y también existen librerías para ser ligado con `Tcl/Tk`, `Python` y `Perl`, que es el único de estos que trataremos en este curso.

14.1 Perl

Existen dos maneras de trabajar con PostgreSQL desde `Perl`, una por medio de la interfase dada por los autores llamada `Pg` y la otra por medio de la interfase genérica `DBI/DBD`. La primera tiene acceso a todas las extensiones que PostgreSQL provee, mientras que la segunda está pensada para trabajar de la misma manera con cualquier base de datos, sistema de indexado o incluso archivos planos o datos en memoria, con la ventaja de que con un cambio mínimo en el fuente se puede pasar de un sistema a otro.

Sin embargo, si la base de datos que se desea usar permanentemente es PostgreSQL, es preferible usar la interfase `Pg`.

14.1.1 Pg

Como mencionamos antes, `Pg` nos dá acceso a todas las funciones de biblioteca de `libpq`. La interfase está implementada orientada a objetos, lo que facilita el uso, ya que no es necesario liberar la conexión ni la memoria empleada, ya que `Perl` invoca a los destructores en cuanto la última referencia a un objeto desaparece.

Para iniciar el trabajo, es necesario incluir la interfase `Pg` y luego iniciar la conexión:

```
use Pg;
$coneccion = Pg::connectdb("dbname=mibase " .
             "dbhost=maquina.remota.com port=5432 user=lola" .
             "password=paquita");
$estatus=$conn->status;
if ($estatus != PGRES_CONNECTION_OK) {
    print STDERR "Error fatal en la conexion ($estatus), terminamos\n";
    exit (1);
}
$resultado = $conn->exec("create database prueba");
$estatus   = $resultado->resultStatus;
if ($estatus != PGRES_COMMAND_OK) {
    print STDERR
        "Error fatal en la creacion de la base ($estatus), terminamos\n";
```

Parámetro	Variable de ambiente	Valor por omisión
host	PGHOST	localhost
port	PGPORT	5432
options	PGOPTIONS	
tty	PGTTY	
dbname	PGDATABASE	usuario actual
user	PGUSER	usuario actual
password	PGPASSWORD	

Tabla 14.1: Variable de ambiente y valores por omisión en `Pg::connectdb`

```
    exit (1);
}
```

Por supuesto, las opciones dadas en `Pg::connectdb` son opcionales y tienen los valores por omisión y correspondientes variables de ambiente que se describen en la tabla 14.1.

Es importante cotejar todas las acciones con los posibles mensajes de error.

Para obtener datos de la base es necesario hacer una consulta y luego traer los resultados, bien de un golpe o uno por uno:

```
$resultado = $conn->exec("select * from archivos group by dmd5");
$estatus   = $result->resultStatus;
if ($result->resultStatus != PGRES_COMMAND_OK) {
    print STDERR "Error al sacar el dmd5\n";
}
while (@renglon = $result->fetchrow) {
    push(@la_lista, $renglon[1]);
}
```

como en este caso. El método `fetchrow` trae las tuplas una por una y en el caso de los tributos nulos, tienen el valor `undef`. Para traer toda la tabla en una sola instrucción:

```
Pg::doQuery($conn, "select nombre, apellido from tabla", \@arreglo);
foreach my $i (0 .. $#arreglo) {
    foreach my $j (0 .. $#{$arreglo[$i]}) {
        print "$arreglo[$i][$j]\t";
    }
    print "\n";
}
```

Métodos de `Pg`

Además de las dos maneras de interacción presentadas, estos son los métodos que proveé `Pg` y que nos permiten tener acceso a partes internas:

`$dbname = $conn->db` Regresa el nombre de la base de datos.

`$pguser = $conn->user` Regresa el nombre del usuario que hizo la conexión.

`$pguser = $conn->pass` Regresa la contraseña de PostgreSQL utilizada en la conexión.

`$pghost = $conn->host` Regresa el nombre canónico de la computadora a donde se hizo la conexión.

`$pgport = $conn->port` Regresa el puerto por el que se hizo la conexión.

`$pgtty = $conn->tty` Regresa la pseudo terminal por la que se hizo la conexión.

`$pgoptions = $conn->options` Regresa las opciones utilizadas en la conexión.

`$status = $conn->status` Regresa el estado de la conexión, los valores a comparar son: `PGRES_CONNECTION_OK` y `PGRES_CONNECTION_BAD`.

`$errorMessage = $conn->errorMessage` Regresa el último mensaje de error asociado con la conexión.

`$fd = $conn->socket` Obtiene el descriptor de archivo asociado al *socket* de la conexión. Un valor de -1 indica que no hay conexión actual.

`$pid = $conn->backendPID` Regresa el *Process ID*[1] del proceso que está respondiendo del lado del servidor.

`$conn->trace(debug_port)` Regresa todos los mensajes entre el servidor y el cliente utilizando el *stream*[2] `debug_port`.

`$conn->untrace` Detiene el proceso de guardar todos los mensajes entre el servidor y el cliente.

`$result = $conn->exec($query)` Envía una consulta al servidor. El valor de regreso es un apuntador a la estructura `PGresult` que contiene toda la información regresada por el servidor. Antes de examinar `$result` es necesario llamar el método `resultStatus` para saber si la operación terminó bien y si el contenido de `$result` es válido.

`($table, $pid) = $conn->notifies` Revisa si se han hecho notificaciones asíncronas. La lista que regresa se compone de la tabla *escuchada* y el PID del servidor.

`$ret = $conn->sendQuery($string, $query)` Envía una consulta al servidor sin esperar por los resultados. Para obtener los resultados, es necesario invocar `getResult` varias veces hasta que regrese un `NULL`, indicando que terminó el proceso.

`$result = $conn->getResult` Espera el siguiente resultado obtenido a partir de `sendQuery` y lo regresa. Regresa `NULL` cuando la consulta ha finalizado y no hay más resultados que mostrar. Este método se bloquea sólo si existe una consulta activa y los datos de respuesta no han sido leidos por `consumeInput`.

`$ret = $conn->isBusy` Regresa `TRUE` si una consulta está en proceso, es decir si `getResult` se bloquearía esperando por el resultado. Un valor `FALSE` indica que `getResult` puede ser llamado sin que bloquee.

`$result = $conn->consumeInput` Si hay resultados esperando a ser leidos, los toma. Luego de ser llamado `consumeInput`, la aplicación puede checar con los métodos `isBusy` o `notifies` para saber si el estado ha cambiado.

`$ret = $conn->getline($cadena, $longitud)` Lee una cadena de longitud `$longitud - 1` del servidor. Regresa `EOF` en caso de que no haya más que leer, `0` si leyó una línea completa y `1` en caso de que el *buffer* esté lleno. Si la línea consiste de los dos carácteres "`\.`", el servidor ha terminado de enviar resultados.

`$ret = $conn->putline($cadena)` Envía una cadena al servidor. La aplicación debe de enviar explícitamente los carácteres "`\.`" para indicarle al servidor que ha terminado de enviar datos.

[1] El identificador del proceso actual.
[2] Mecanismo propio de Unix para la comunicación entre procesos

`$ret = $conn->getlineAsync($buffer, $bufsize)` Versión del método `getline` que no bloquea. Lee hasta `$bufsize` carácteres del servidor. Regresa `-1` si la marca de fin de copia (`\.`) es encontrada, `0` si no hay datos y un valor mayor que `0` indicando el número de bytes leídos.

`$ret = $conn->putnbytes($buffer, $nbytes)` Envía los `$nbytes` contenidos en `$buffer` al servidor. Regresa `0` si no hay error, `EOF` en caso contrario.

`$ret = $conn->endcopy` Este método espera hasta que el servidor haya terminado la copia de datos. Debe de ser empleado cuando la última cadena ha sido enviada con `putline` o cuando la última cadena ha sido recibida con `getline`. Regresa `0` cuando hay éxito, `1` en caso contrario.

`$result = $conn->makeEmptyPGresult($status);` Regresa una variable del tipo `PGresult`, limpia e inicializada con el estado dado en `$status`.

La estructura `PGresult`

Como mencionamos, la estructura `PGresult` contiene toda la información referente al resultado regresado por el servidor luego de una consulta. El método para obtener esta información es:

`$result_status = $result->resultStatus` Regresa el estado del resultado. Para comparar el estado se usan las siguientes constantes, dependiendo de la instrucción enviada:

- `PGRES_EMPTY_QUERY`
- `PGRES_COMMAND_OK`
- `PGRES_TUPLES_OK`
- `PGRES_COPY_OUT`
- `PGRES_COPY_IN`
- `PGRES_BAD_RESPONSE`
- `PGRES_NONFATAL_ERROR`
- `PGRES_FATAL_ERROR`

`Pg` proveé de algunos métodos para examinar la estructura resultante:

`$ntuples = $result->ntuples` Regresa el número de tuplas resultantes de la consulta.

`$nfields = $result->nfields` Regresa el número de campos.

`$ret = $result->binaryTuples` Regresa `1` si las tuplas son binarias.

`$fname = $result->fname($field_num)` Regresa el nombre del campo asociado al número de campo `$field_num`.

`$fnumber = $result->fnumber($field_name)` Regresa el número de campo asociado al nombre de campo `$field_name`.

`$ftype = $result->ftype($field_num)` Regresa el OID del tipo asociado al número de campo dado.[3]

`$fsize = $result->fsize($field_num)` Regresa el tamaño en bytes del tipo asociado al número de campo dado. Si regresa `-1` entonces el campo es de tamaño variable.

[3] Ver sección 9.1.

`$fmod = $result->fmod($field_num)` Regresa los datos de modificación para el tipo específico asociado con el índice de campo dado. Los indices de campo comienzan en el `0`.

`$cmdStatus = $result->cmdStatus` Regresa el estado de la última instrucción dada. En el caso de `DELETE` también regresa el número de tuplas borradas. En el caso de `INSERT` regresa el OID de la tupla insertada, seguido de `1`, el número de tuplas afectadas por la operación.

`$oid = $result->oidStatus` En caso de que la última consulta haya sido un `INSERT` regresa el OID de la tupla insertada.

`$oid = $result->cmdTuples` En caso de que la última instrucción haya sido un `INSERT` o `DELETE` regresa el número de tuplas afectadas.

`$value = $result->getvalue($tup_num, $field_num)` Regresa el valor de la tupla y campo dados. Es una cadena terminada, como en `C`, con `0`. Los cursores binarios no utilizan este método.

`$length = $result->getlength($tup_num, $field_num)` Regresa la longitud del valor para un campo y una tupla dados.

`$null_status = $result->getisnull($tup_num, $field_num)` Regresa el estado respecto a `NULL` para un campo y una tupla dados.

`$res->fetchrow` Trae la siguiente tupla del servidor y regresa `NULL` cuando todas han sido procesadas. Los atributos que tengan valores `NULL`, serán fijados al valor `undef` de `Perl`.

`$result->print($fout, $header, $align, $standard, $html3, $expanded, $pager, $fieldSep, $tableOpt, $caption, ...)` Esta es la función de impresión empleada por `psql`. Imprime todas las tuplas de una manera presentable. La explicación de los parámetros es como sigue:

- `$fout` Descriptor de archivo, donde escribe.
- `$header` Booleano, si emplea un encabezado y contador de tuplas.
- `$align` Booleano, si alínea las columnas.
- `$standard` Booleano, formato en desuso.
- `$html3` Booleano, tablas formateadas bajo el estándar de HTML versión 3.
- `$expanded` Booleano, si usa el modo extendido de impresión.
- `$pager` Booleano, si numera páginas.
- `$fieldSep` Cadena, separador de columnas.
- `$tableOpt` Cadena, opciones para las tablas en HTML.
- `$caption` Cadena, pie de tabla.

  Si se le pasan valores adicionales, serán tomados como nombres de las columnas.

El siguiente es un ejemplo que toma la salida de una consulta al servidor de CDDB y lo inserta en una base de datos.

```
#!/usr/bin/perl

use Pg;
use Getopt::Std;

getopts('ig:');

@args = @ARGV;

$ini    = (defined $opt_i) ? 1 : 0;
$genero = (defined $opt_g) ? $opt_g : 'Sin Clasificar';
$DBUG   = 10000;
$|      = 1;

&inicializa();
&inidb();
foreach $a (@ARGV) {
    &jalacddb($a);
}

# # xmcd CD database file
# # Copyright (C) 1993-1998 CDDB, Inc.
# #
# # Track frame offsets:
# #     200
# #     36531
# #     55138
# #     98442
# #     112786
# #     165902
# #     185123
# #     235011
# #
# # Disc length: 3225 seconds
# #
# # Revision: 6
# # Processed by: cddbd v1.4.1b10PL0 Copyright (c) 1996-1998 CDDB Inc.
# # Submitted via: NotifyCDPlayer(CDDB) 1.51.3
# #
# DISCID=530c9708,540c9c08,560c9508,570c9508,590c9708,590c9c08
# DTITLE=Genesis / Selling England by the Pound
# TTITLE0=Dancing with the Moonlight Knight
# TTITLE1=I know what I like (in your Wardrobe)
# TTITLE2=Firth of Fifth
# TTITLE3=More fool Me
# TTITLE4=The Battle of Epping Forest
# TTITLE5=After the Ordeal
# TTITLE6=The Cinema Show
# TTITLE7=Aisle of Plenty
# EXTD=Phil Collins : Drums, Percussion, Vocal\nMichael Rutherford : 12-String
# EXTD=, Bass, Electric Sitar\nStephen Hackett : Elec Guitar, Nylon Guitar\nTo
# EXTD=ny Banks : Keyboards, 12-String\nPeter Gabriel : Vocals, Flute, Percuss
# EXTD=ion, Oboe\n\nProduced by John Burns & Genesis\nCASCD 1074 (P) 1973 Virg
# EXTD=in Records LTD (C) 1985 Charisma Records LTD\n\nSubmitted by Stephane G
# EXTD=rienenberger, 17/12/1998
# EXTT0=
# EXTT1=
# EXTT2=
# EXTT3=Vocals : Phil
# EXTT4=
# EXTT5=
# EXTT6=
# EXTT7=
# PLAYORDER=
```

```
# .

sub jalacddb {
# Esta funcion es mas grande de lo que puedo tolerar. Y todavia falta
# aniadirle una tonelada de chequeos.
    local ($arc) = @_;
    print "Insertando $arc\n" if ($DBUG <= 2);
    open (DATA, $arc) or die "No puedo leer de $arc\n";
    while (<DATA>) {
        chomp;
        if (m/Track frame offsets:/) {
            foreach my $i (0..100) {
                $disc[$TROLA][$i] = "";
            }
            $ind         = 0;
            $max_ind     = 0;
            $extd        = "";
            $disc[$EXTD] = "";
            $auxtit      = "";
            print "Machea Track frame:[$_]\n" if ($DBUG <= 1);
            $nt = 0;
            while (<DATA>) {
                next if (m/^\#\s+$/);
                chomp;
                if (m/Disc length:/) {
                    print "Machea Disc length:[$_]\n" if ($DBUG <= 2);
                    $aux  = $_;
                    $aux =~ s/(Disc)(\s)(length:)(\s)([0-9]+)/$5/;
                    $disc[$DURS][$nt] = $5 + 0;
                    $disc[$NROL]      = $nt - 1;
                    print "Tenemos $disc[$NROL] rolas: " if ($DBUG <= 2);
                    foreach my $i (0..$disc[$NROL]) {
                        print "$disc[$DURS][$i]:" if ($DBUG <= 2);
                        $disc[$TREX][$i] = "";
                    }
                    print "$disc[$DURS][$disc[$NROL]+1]\n" if ($DBUG <= 2);
                    last;
                } else {
                    @a = split (' ', $_);
                    $disc[$DURS][$nt++] = $a[1] + 0;
                }
            }
        }
        if (m/^DISCID/) {
            print "Machea DISCID:[$_]\n" if ($DBUG <= 1);
            @a = split (/=/, $_);
            $disc[$DISCID][0] = $a[1];
        }
        if (m/^DTITLE/) {
            # Recuerda que a veces vienen en dos lineas:
            # DTITLE=Fischer-Dieskau/Moore / Schubert, Franz: Die schone Mull
            # DTITLE=erin, D795
            print "Machea DTITLE:[$_]\n" if ($DBUG <= 1);
            @a = split (/=/, $_);
            $auxtit .= $a[1];
        }
        if (m/^EXTD/) {
            print "Machea EXTD:[$_]\n" if ($DBUG <= 1);
            @a = split (/=/, $_);
            $disc[$EXTD] .= $a[1];
        }
        if (m/^TTITLE/) { # Hay que guardar tambien el numero de rola...
            print "Machea TTITLE:[$_]\n" if ($DBUG <= 1);
            $aux = $_;
```

```
    @a = split (/=/, $_);
    $a[0] =~ s/(TITLE)([0-9]+)/$2/;
    $ind  = $2 + 0;
    print "$aux->$ind\n" if ($DBUG <= 1);
    $disc[$TROLA][$ind] .= $a[1];
    $max_ind = $ind;
}
if (m/^EXTT/) {
    print "Machea EXTT:[$_]\n" if ($DBUG <= 1);
    $aux = $_;
    @a = split (/=/, $_);
    $a[0] =~ s/(EXTT)([0-9]+)/$2/;
    $ind  = $2 + 0;
    print "indice extension: $aux->$ind:[$a[1]]\n" if ($DBUG <= 10);
    $disc[$TREX][$ind] .= $a[1];
}
if (m/^PLAYORDER/) { # Usualmente el archivo termina con un `.',
                     # pero en la practica, muchos terminan con el PLAYORDER
    print "max_ind ==> $max_ind\n" if ($DBUG <= 3);
    if ($max_ind <= 0) {
        print ILOG "$arc: con ", $max_ind+1, " rolas\n";
    }
    foreach my $i (0..$max_ind) {
        if ($i == $max_ind) {
            $frames = $disc[$DURS][$i+1] * 75 - $disc[$DURS][$i];
        } else {
            $frames    = $disc[$DURS][$i+1] - $disc[$DURS][$i];
        }
        if ($frames >= 270000) {
            print STDERR "\n\n\t\t!!!PELIGRO!!!\n\nframes:$frames\n";
            print ILOG "$arc:frames absurdos $frames\n";
        }
        $framesres = $frames % 75;
        $j         = ($frames - $framesres) / 75;
        $segundos  = $j % 60;
        $j         = $j - $segundos;
        $minutos   = ($j / 60) % 60;
        $horas     = int (int ($j / 60) / 60);
        print ">$i:$disc[$DURS][$i+1]:$disc[$DURS][$i]:$frames=>"
            ."f$framesres:s$segundos:m$minutos:h$horas<\n" if ($DBUG <= 3);
        $disc[$TTIM][$i][$TFRM] = $framesres;
        $disc[$TTIM][$i][$TSEG] = $segundos;
        $disc[$TTIM][$i][$TMIN] = $minutos;
        $disc[$TTIM][$i][$THOR] = $horas;
        $disc[$MSDURS][$i] = sprintf ("%02d:%02d:%02d.%02d",
                                      $horas, $minutos,
                                      $segundos, $framesres);
    }
    print "auxtit:$auxtit\n" if ($DBUG <= 3);
    $disc[$TTOT][$TFRM] = 0;
    $disc[$TTOT][$TSEG] = 0;
    $disc[$TTOT][$TMIN] = 0;
    $disc[$TTOT][$THOR] = 0;
    $tfrm               = 0;
    $tseg               = 0;
    $tmin               = 0;
    $thor               = 0;
    if ($auxtit =~ m/\//) {
        @a = split (/\//, $auxtit, 2);
    } elsif ($auxtit =~ m/\-/) {
        @a = split (/\-/, $auxtit, 2);
    } elsif ($auxtit =~ m/:/) {
        @a = split (/:/, $auxtit, 2);
    } else {
```

```
                $a[0] = $auxtit;
                $a[1] = $auxtit;
            }
            $a[0]               =~ s/^\s+//;
            $a[0]               =~ s/\s+$//;
            $a[1]               =~ s/^\s+//;
            $a[1]               =~ s/\s+$//;
            $disc[$AUTOR]       = $a[0];
            $disc[$TITULO]      = $a[1];
            print "ID:    $disc[$DISCID][0]\n" if ($DBUG <= 4);
            print "Autor: $disc[$AUTOR]\n" if ($DBUG <= 4);
            print "Titulo:$disc[$TITULO]\n" if ($DBUG <= 4);
            print "Ext.:  $disc[$EXTD]\n" if ($DBUG <= 4);
            print "Tracks:\n" if ($DBUG <= 4);
            foreach my $i (0..$max_ind) {
                print "\t$disc[$TROLA][$i]:$disc[$TREX][$i]:$disc[$DURS][$i]"
                    . "($disc[$MSDURS][$i])\n" if ($DBUG <= 4);
                $tfrm += $disc[$TTIM][$i][$TFRM];
                $tseg += $disc[$TTIM][$i][$TSEG];
                $tmin += $disc[$TTIM][$i][$TMIN];
                $thor += $disc[$TTIM][$i][$THOR];
                $disc[$TTOT][$TFRM] = $tfrm % 75;
                $carry = ($tfrm  - $disc[$TTOT][$TFRM]) / 75;
                $disc[$TTOT][$TSEG] = ($tseg + $carry) % 60;
                $carry = ($tseg + $carry - $disc[$TTOT][$TSEG]) / 60;
                $disc[$TTOT][$TMIN] = ($tmin + $carry) % 60;
                print "....................................."
                    . "tmin:$tmin:carry:$carry\n" if ($DBUG <= 4);
                $carry = ($tmin + $carry - $disc[$TTOT][$TMIN]) / 60;
                $disc[$TTOT][$THOR] = $thor + $carry;
            }
            $disc[$TTOT][$TTTO] = sprintf ("%02d:%02d:%02d.%02d",
                    $disc[$TTOT][$THOR], $disc[$TTOT][$TMIN],
                    $disc[$TTOT][$TSEG], $disc[$TTOT][$TFRM]);
            &mete_disco();
            print "=" x 78, "\n";
        }
    }
    close (DATA);
}

sub dame_seq {
    $manda = "select nextval(\'seqdis\')";
    $result = $conn->exec($manda);
    $stat   = $result->resultStatus;
    cmp_eq("jalando sequencia", PGRES_TUPLES_OK, $result->resultStatus);
    cmp_ne("OID status",0, $result->oidStatus);
    $leseq = $result->fetchrow;
    if ($GranError == 1) {
        $GranError = 0;
        print ELOG "$manda ($.)\n" if ($LOGGING);
        print ILOG "dame_seq:$manda --> [$leseq]\n";
    }
    return $leseq;
}

sub inserta_id {
    local ($id, $seq) = @_;
    print STDERR "Insertando: $id\n";
    $manda = "insert into ids values (\'$id\', $seq)";
    $result = $conn->exec($manda);
    $stat   = $result->resultStatus;
    # print LOG "$stat\n" if ($LOGGING);
    cmp_eq("insertando $id:$seq",PGRES_COMMAND_OK, $result->resultStatus);
```

```
    cmp_ne("OID status",0, $result->oidStatus);
    if ($GranError == 1) {
        $GranError = 0;
        print ELOG "$manda ($.)\n" if ($LOGGING);
        print ILOG "inserta_id:$manda\n";
    }
}

sub busca_id {
    local ($id) = @_;
    $manda = "select discid from ids where discid = '$id'";
    $result = $conn->exec($manda);
    $stat   = $result->resultStatus;
    cmp_eq("Buscando $id", PGRES_COMMAND_OK, $result->resultStatus);
    cmp_ne("OID status", 0, $result->oidStatus);
    if ($GranError == 1) {
        $GranError = 0;
        print ELOG "$manda ($.)\n" if ($LOGGING);
    }
    return $result->ntuples;
}

sub mete_disco {
    $seq = &dame_seq ();
    @dd = split /,/, $disc[$DISCID][0];
    foreach my $id (@dd) {
        &inserta_id ($id, $seq);
    }
    $numrolas = $max_ind + 1;
    $disc[$AUTOR]  =~ s/\'/\\\'/g;
    $disc[$AUTOR]  =~ s/\"/\\\"/g;
    $disc[$TITULO] =~ s/\'/\\\'/g;
    $disc[$TITULO] =~ s/\"/\\\"/g;
    $disc[$EXTD]   =~ s/\\/\\\\/g;
    $disc[$EXTD]   =~ s/\'/\\\'/g;
    $disc[$EXTD]   =~ s/\"/\\\"/g;
    foreach my $i (0..$max_ind) {
        $disc[$TROLA][$i] =~ s/\'/\\\'/g;
        $disc[$TROLA][$i] =~ s/\"/\\\"/g;
        $disc[$TREX][$i]  =~ s/\\/\\\\/g;
        $disc[$TREX][$i]  =~ s/\'/\\\'/g;
        $disc[$TREX][$i]  =~ s/\"/\\\"/g;
    }
    $values = "$seq, \'$disc[$TITULO]\', \'$disc[$AUTOR]\', \'$disc[$EXTD]\'"
        . ", $numrolas, \'$disc[$TTOT][$TTTO]\', $legenero";
    $manda = "insert into ndis values ($values)";
    $result = $conn->exec($manda);
    $stat   = $result->resultStatus;
    cmp_eq("insertando $disc[$AUTOR]:$disc[$TITULO]", PGRES_COMMAND_OK,
           $result->resultStatus);
    cmp_ne("OID status",0, $result->oidStatus);
    if ($GranError == 1) {
        $GranError = 0;
        print ELOG "$manda ($.)\n" if ($LOGGING);
        print ILOG "mete_disco:$manda\n";
    }
    foreach my $i (0..$max_ind) {
        &inserta_rola ($seq, ($i+1), $disc[$TROLA][$i], $disc[$MSDURS][$i],
                       $disc[$TREX][$i]);
    }
}

sub inserta_rola {
    local ($seq, $nrola, $tit, $dur, $ext) = @_;
```

```
    $values = "$seq, $nrola, \'$tit\', \'$dur\', \'$ext\'";
    $manda = "insert into rolas values ($values)";
    $result = $conn->exec($manda);
    $stat   = $result->resultStatus;
    cmp_eq("insertando $seq:$tit",PGRES_COMMAND_OK, $result->resultStatus);
    cmp_ne("OID status",0, $result->oidStatus);
    if ($GranError == 1) {
        $GranError = 0;
        print ELOG "$manda ($.)\n" if ($LOGGING);
        print ILOG "inserta_rola:$manda\n";
    }
}

sub inicializa {
    $dbmain    =  'mancha';
    $dbname    =  'pruebas';
    $tabname   =  'discos';
    $dbhost    =  'caserola';
    $trace     =  '/tmp/pgtrace.out';
    $cnt       =  2;
    $DEBUG     =  1; # set this to 1 for traces
    $errorlog  =  'log.inscddb.log';
    $ellog     =  '/tmp/inscddb.log';
    $logirrec  =  'irrecuperables.inscddb.log';
    $LOGGING   =  1;
    $GranError =  0;
    $DURS      =  0;
    $DISCID    =  1;
    $AUTOR     =  2;
    $TITULO    =  3;
    $NROL      =  4;
    $TROLA     =  5;
    $TREX      =  6;
    $EXTD      =  7;
    $MSDURS    =  8;
    $TTIM      =  9;
    $TTOT      = 10;
    $THOR      =  0;
    $TMIN      =  1;
    $TSEG      =  2;
    $TFRM      =  3;
    $TTTO      =  4;
    $SIG{PIPE} = sub { print "Se rompio la canieria\n!Llamen al plomero!\n\n" };
    open (LOG,  ">>$ellog")    || die "No puedo crear $ellog\n";
    open (ELOG, ">>$errorlog") || die "No puedo crear $errorlog\n";
    open (ILOG, ">>$logirrec") || die "No puedo crear $logirrec\n";
}

sub inidb {
    $conn = Pg::connectdb("dbname=$dbname host=$dbhost");
    cmp_eq("probando conexion", PGRES_CONNECTION_OK, ($estatus=$conn->status));
    if ($estatus != PGRES_CONNECTION_OK) {
        print STDERR "Error fatal en la conexion ($estatus), terminamos\n";
        exit (1);
    }
    if ($DEBUG) {
        open(TRACE, ">$trace") || die "can not open $trace: $!";
        $conn->trace(TRACE);
    }
    $db = $conn->db;
    cmp_eq("comparando bases", $dbname, $db);
    $user = $conn->user;
    cmp_ne("comparando usuarios", "", $user);
    $host = $conn->host;
```

```
    cmp_ne("comparando hosts", "", $host);
    $port = $conn->port;
    cmp_ne("comparando puertos", "", $port);
    $result = $conn->exec("SET DateStyle to 'ISO'");
    cmp_eq("fijando la fecha", $result->resultStatus, $result->resultStatus);
    if ($ini == 1) {
        &limpia_base ();
        &crea_base ();
    }
    $legenero = &busca_genero ($genero);
    print "El id para genero $genero es $legenero\n";
}

sub busca_genero {
    local $legenero;
    $manda = "select genid from genero where genero='$genero'";
    $result = $conn->exec($manda);
    $stat   = $result->resultStatus;
    cmp_eq("obteniendo $genero", PGRES_COMMAND_OK, $result->resultStatus);
    cmp_ne("OID status", 0, $result->oidStatus);
    if ($result->ntuples) { # ya existe, obtenlo
        $legenero = $result->fetchrow;
    } elsif ($GranError == 1) { # no existe, insertalo
        $GranError = 0;
        print ELOG "$manda ($.)\n" if ($LOGGING);
        print STDERR "Error al sacar el genero\n";
        $manda = "insert into genero (genero) values ('$genero')";
        $result = $conn->exec($manda);
        $stat   = $result->resultStatus;
        cmp_eq("insertando $genero",PGRES_COMMAND_OK, $result->resultStatus);
        cmp_ne("OID status",0, $result->oidStatus);
        $manda = "select genid from genero where genero='$genero'";
        $result = $conn->exec($manda);
        $stat   = $result->resultStatus;
        cmp_eq("obteniendo $genero",PGRES_COMMAND_OK, $result->resultStatus);
        cmp_ne("OID status",0, $result->oidStatus);
        $legenero = $result->fetchrow;
    }
    return $legenero;
}

sub cmp_eq {
    my $mns = shift;
    my $cmp = shift;
    my $ret = shift;
    my $msg;

    print ELOG "$mns: ";
    if ("$cmp" eq "$ret") {
        print ELOG "ok $cnt\n";
    } else {
        $msg = $conn->errorMessage;
        print ELOG "error $cnt: $cmp, $ret\n$msg\n";
        $GranError = 1;
    }
    $cnt++;
}

sub cmp_ne {
    my $mns = shift;
    my $cmp = shift;
    my $ret = shift;
    my $msg;
```

```
    print ELOG "$mns: ";
    if ("$cmp" ne "$ret") {
        print ELOG "ok $cnt\n";
    } else {
        $msg = $conn->errorMessage;
        print ELOG "error $cnt: $cmp, $ret\n$msg\n";
        $GranError = 1;
    }
    $cnt++;
}

sub limpia_base {
    $des[0]    = "DROP TABLE ids";
    $des[1]    = "DROP TABLE ndis";
    $des[2]    = "DROP TABLE rolas";
    $des[3]    = "DROP INDEX xautor";
    $des[4]    = "DROP INDEX xtitulo";
    $des[5]    = "DROP SEQUENCE seqdis";
    $des[6]    = "DROP SEQUENCE genero_genid_seq";
    $des[7]    = "DROP INDEX genero_genid_key";
    $des[8]    = "DROP TABLE genero";
    $Max_Tab   = 8;
    foreach my $i (0..$Max_Tab) {
        print "Actuando $i:";
        print "\tDestruir: $des[$i]\n";
            if (defined $des[$i]) {
                $result = $conn->exec($des[$i]);
                cmp_eq("destruyendo la tabla",PGRES_COMMAND_OK, $result->resultStatus);
                cmp_eq("redestruyendo la tabla","DROP", $result->cmdStatus);
            }
    }
}

sub crea_base {
    $tab[0]    = "CREATE TABLE ids (discid CHAR(8) NOT NULL, discseq INT4 NOT NULL)";
    $tab[1]    = "CREATE TABLE ndis (discseq INT4 NOT NULL, titulo TEXT, "
        . "autor TEXT, extdisc TEXT, numrolas INT2, durdisc TIME, genid INT4 NOT NULL)";
    $tab[2]    = "CREATE TABLE rolas (discseq INT4 NOT NULL, nrola int4, "
        . "rolatit TEXT, roladur TIME, rolaext TEXT)";
    $tab[3]    = "CREATE INDEX xautor on ndis using btree (autor text_ops)";
    $tab[4]    = "CREATE INDEX xtitulo on ndis using btree (titulo text_ops)";
    $tab[5]    = "CREATE SEQUENCE seqdis";
    $tab[6]    = "CREATE TABLE genero (genid serial, genero varchar(32))";
    $Max_Tab   = 6;
    foreach my $i (0..$Max_Tab) {
        print "Actuando $i:";
        print "\tConstruir: $tab[$i]\n";
        if (defined $tab[$i]) {
            $result = $conn->exec("$tab[$i]");
            cmp_eq("creando la tabla",PGRES_COMMAND_OK, $result->resultStatus);
            cmp_eq("recreando la tabla","CREATE", $result->cmdStatus);
        }
    }
}
```

En este otro ejemplo, hacemos un CGI que realiza consultas sobre una base de datos en PostgreSQL.

```
#!/usr/bin/perl

use Pg;
use CGI;

$DBUG=1001;
```

```
$cgi = new CGI;
&inicializa ();
&forma ();
&consulta ();

sub forma {
    print $cgi->header, "\n";
    print $cgi->start_html(-title => 'Consulta a la base de discos',
                           -author => 'mancha@caserola.mancha.baras.net',
                           -base => 'true',
                           -expires => 'now'),"\n";
    print $cgi->h1('Consulta a la base de datos de discos');
    print $cgi->startform;
    print "Seleccione algún campo de la forma";
    print "<P>Artista: ";
    print $cgi->scrolling_list(-name => 'autor',
                               -values => ['Indiferente', @lesartistes],
                               -size => 5,
                               -default => 'Indiferente');
    print "<P>Título: ";
    print $cgi->scrolling_list(-name => 'titulo',
                               -values => ['Indiferente', @lestitres],
                               -size => 5,
                               -default => 'Indiferente');
    print "<P>";
    print $cgi->submit('Action','Consultar');
    print $cgi->end_form;
    print $cgi->hr, "\n";
    print $cgi->end_html;
}

sub by_number {
    $a <=> $b;
}

sub consulta {
    foreach my $key ($cgi->param) {
        print "[[$key]=>" if ($DBUG == 23);
        @valores = $cgi->param($key);
        print "[", join(' ', @valores), "]]<BR>" if ($DBUG == 23);
        $cosa = join (' ', @valores);
        $consulta{$key} = $cosa
            if (length $cosa && $cosa ne 'Indiferente' && $key ne 'Action');
    }
    print "Buscaremos por:<BR>" if ($DBUG == 23);
    foreach my $k (keys %consulta) {
        print "$k ==> [$consulta{$k}]<BR>" if ($DBUG == 23);
        push (@cbus, "$k ~~ \'%$consulta{$k}%\'");
    }
    $manda  = 'SELECT *,oid FROM discos WHERE ' . join (' AND ', @cbus);
    $manda .=  ' order by autor,titulo';
    print STDERR "query: [$manda]\n";
    print "query: $manda<BR>" if ($DBUG == 23);
    print LOG "$manda\n";
    &presenta ($manda);
}

sub presenta {
    local ($query) = @_;
    print "en presenta enviaremos: $query<BR>" if ($DBUG == 23);
    $result = $conn->exec($query);
    cmp_eq("consultando $tabname", PGRES_COMMAND_OK, $result->resultStatus);
    cmp_eq("Status $tabname", PGRES_TUPLES_OK, $result->cmdStatus);
    my $i = 0;
```

```
    while (@renglon = $result->fetchrow) {
        @{$renglones[$i]} = @renglon;
        $i++;
    }
    $tot   = --$i;
    $total = $tot + 1;
    print $cgi->hr;
    print "<H1>Recibimos un total de $total registro",
        ($total == 1) ? "" : "s","</H1>";
    print $cgi->hr;
    print "Recibimos un total de $tot registros<BR>" if ($DBUG == 23);
    if ($forma1) {
        foreach my $i (0..$tot) {
            print "<TABLE BORDER=2>";
            print "<TR><TD><B>Autor</B></TD>";
            print "<TD>@{$renglones[$i]}[0]</TD></TR>\n";
            print "<TR><TD><B>Titulo</B></TD>";
            print "<TD>@{$renglones[$i]}[1]</TD></TR>\n";
            print "<TR><TD><B>NDis</B></TD>";
            print "<TD>@{$renglones[$i]}[2]</TD></TR>\n";
            print "<TR><TD><B>Medio</B></TD>";
            print "<TD>@{$renglones[$i]}[3]</TD></TR>\n";
            print "</TABLE>";
            print $cgi->hr;
        }
    } else {
        print "<TABLE BORDER=2><TR>";
        print "<TD><B>Autor</B></TD>";
        print "<TD><B>Título</B></TD>";
        print "<TD><B>NDis</B></TD>";
        print "<TD><B>Medio</B></TD></TR>";
        foreach my $i (0..$tot) {
            print "<TR><TD>@{$renglones[$i]}[0]</TD>";
            print "<TD>@{$renglones[$i]}[1]</TD>";
            print "<TD>@{$renglones[$i]}[2]</TD>";
            print "<TD>@{$renglones[$i]}[3]</TD></TR>";
        }
        print "</TABLE>";
    }
    close (LOG);
}

sub inicializa {
    $dbname  = 'discos';
    $dbhost  = 'caserola';
    $tabname = 'discos';
    $ellog   = '/tmp/consultas.discos';
    chomp ($soy = `whoami`);
    chomp ($estoy = `hostname`);
    $cnt     = 1;
    print "Yo soy: [$soy]\ny estoy en la maquina: [$estoy]<P>"
        if ($DBUG == 23);
    $conn = Pg::connectdb("dbname=$dbname host=$dbhost");
    cmp_eq("<P>probando conexion", PGRES_CONNECTION_OK, $conn->status);
    $db = $conn->db;
    cmp_eq("<P>comparando bases", $dbname, $db);
    $user = $conn->user;
    cmp_ne("<P>comparando usuarios", "", $user);
    $host = $conn->host;
    cmp_ne("<P>comparando hosts", "", $host);
    $port = $conn->port;
    cmp_ne("<P>comparando puertos", "", $port);
    $result = $conn->exec("SET DateStyle to \'Postgres\'");
    cmp_eq("<P>fijando la fecha", $result->resultStatus,
```

```
            $result->resultStatus);
    $manda = "select distinct on autor autor from discos";
    print "<P>enviaremos: $manda\n" if ($DBUG > 0);
    $result = $conn->exec($manda);
    cmp_eq("<P>consultando $tabname", PGRES_COMMAND_OK,
            $result->resultStatus);
    cmp_eq("<P>Status $tabname", PGRES_TUPLES_OK, $result->cmdStatus);
    $cuantos = $result->ntuples;
    print "<P>obtuvimos $cuantos autores\n" if ($DBUG > 0);
    while ($aux = $result->fetchrow) {
        print "<P>($aux)" if ($DBUG > 101);
        push (@autores, $aux);
    }
    @lesartistes = sort @autores;
    $manda = "select distinct on titulo titulo from discos";
    print "<P>enviaremos: $manda\n" if ($DBUG > 0);
    $result = $conn->exec($manda);
    cmp_eq("<P>consultando $tabname", PGRES_COMMAND_OK,
            $result->resultStatus);
    cmp_eq("<P>Status $tabname", PGRES_TUPLES_OK, $result->cmdStatus);
    $cuantos = $result->ntuples;
    print "<P>obtuvimos $cuantos titulos\n" if ($DBUG > 0);
    while ($aux = $result->fetchrow) {
        push (@titulos,$aux);
    }
    @lestitres = sort @titulos;
    open(LOG, ">>$ellog") || die "No puedo escribir en $ellog";
}

sub cmp_eq {
    my $mns = shift;
    my $cmp = shift;
    my $ret = shift;
    my $msg;

    print "<P>$mns: " if ($DBUG > 23);
    if ("$cmp" eq "$ret") {
        print "<P>ok $cnt\n" if ($DBUG > 23);
    } else {
        $msg = $conn->errorMessage;
        print "<P>error $cnt: $cmp, $ret\n$msg\n" if ($DBUG > 23);
        $GranError = 1;
    }
    $cnt++;
}

sub cmp_ne {
    my $mns = shift;
    my $cmp = shift;
    my $ret = shift;
    my $msg;

    print "<P>$mns: " if ($DBUG > 23);
    if ("$cmp" ne "$ret") {
        print "<P>ok $cnt\n" if ($DBUG > 23);
    } else {
        $msg = $conn->errorMessage;
        print "<P>error $cnt: $cmp, $ret\n$msg\n" if ($DBUG > 23);
        $GranError = 1;
    }
    $cnt++;
}
```

Objetos grandes

En PostgreSQL, el tamaño máximo de una tupla coincide con el del tamaño de página, de 8K, y dado que una tupla no puede ser almacenada en más de una página, la única manera de almacenar objetos más grandes que este tamaño es por medio de la intarfase de *objetos grandes*.

Dicho de manera escueta, los objetos grandes son almacenados por partes en tuplas (en el sentido interno de PostgreSQL) y se emplean indices B-tree para acelerar el acceso a cada una de las partes que conforman un objeto grande.

Internamente PostgreSQL emplea una interfase semejante a la del manejo de archivos de Unix. Hacia el programador, por medio de `libpq`, provee de funciones de alto nivel y la implementación para `Perl`, proveé de métodos adecuados. Estos métodos proveen acceso orientado por archivo para objetos de gran tamaño, con analogías a las funciones de sistema `open`, `close`, `read`, `write`, `lseek` y `tell`. Para ser consistentes, se les añade el prefijo `lo_` al ser llamadas.

Para usar la interfase de objetos grandes, es necesario encerrar las operaciones en un bloque de transacciones[4].

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

`$lobj_fd = $conn->lo_open($lobjId, $mode)` Abre un objeto grande y regresa un OID. El modo de apertura se fija con la variable `$mode` igual que en el método `lo_create`. Regresa `-1` en caso de error.

`$ret = $conn->lo_close($lobj_fd)` Cierra un objeto grande. Regresa `0` en caso de éxito, `-1` en caso de error.

`$nbytes = $conn->lo_read($lobj_fd, $buf, $len)` Lee `$len` carácteres en la variable `$buf` del objeto grande definido por `$lobj_fd`. Regresa el número de bytes leidos o `-1` en caso de error.

`$nbytes = $conn->lo_write($lobj_fd, $buf, $len)` Escribe `$len` bytes de la variable `$buf` en el objeto grande definido por `$lobj_fd`. Regresa el número de bytes escritos o `-1` en caso de error.

`$ret = $conn->lo_lseek($lobj_fd, $offset, $whence)` Cambia de posición el punto de lectura o escritura dentro del objeto grande definido por `$obj_id`. Actualmente, `$whence` sólo puede valer 0 (L_SET).

`$lobjId = $conn->lo_creat($mode)` Crea un objeto grande. `$mode` es una máscara de bits que describe los diferentes atributos del objeto. Utiliza las siguientes constantes:

- PGRES_INV_WRITE Para escritura
- PGRES_INV_READ Para lectura
- PGRES_INV_SMGRMASK
- PGRES_INV_ARCHIVE Para ser almacenada en un histórico

  Estas banderas pueden ser combinadas con `OR`, como en el caso de abrir para lectura y escritura. En caso de error, regresa el valor definido por la constante `PGRES_InvalidOid`.

`$location = $conn->lo_tell($lobj_fd)` Regresa la posición de lectura o escritura en el objeto grande definido por `$lobj_fd`.

`$ret = $conn->lo_unlink($lobjId)` Borra un objeto grande. Regresa `-1` en caso de error.

[4] Ver la sección 8.11.

`$lobjId = $conn->lo_import($filename)` Importa un archivo Unix como un objeto grande y regresa el OID del nuevo objeto.

`$ret = $conn->lo_export($lobjId, $filename)` Exporta un objeto grande a un archivo Unix. Regresa `-1` en caso de error, `1` en caso de éxito.

14.1.2 DBI/DBD

DBI son las siglas en inglés de **D***ata***b***ase* **i***ndependent interface for Perl*. El concepto de funcionamiento es que mientras DBI es una interfase independiente del soporte de un manejador de base de datos en particular, los DBD **D***ata***b***ase* **D***rivers* le dan la funcionalidad para cada manejador o subsistema —dado que incluye soporte para archivos planos y *hashes* en memoria o en disco—.

Enumeramos a continuación los métodos empleados por este sistema:

`$dbh = DBI->connect("dbi:Pg:dbname=$dbname ;host= $host;port= $port;options= $options; tty= $tty", " $username", "$password");` Se conecta a la base indicada por `$dbname` en la computadora `$host`, por el puerto `$port`, con las opciones `$options`, con las características de terminal `$tty`, como el usuario `$username` con contraseña `$password`. Todos los parámetros señalados son opcionales y reemplazados como se indican en la tabla 14.1.

Cabe recordar que si se indica el nombre de una computadora, el servidor en esa máquina debe de previamente haber sido inicializado con la opción `-i`.

Entre las opciones posibles, está la de indicarle al servidor cuántos *buffers* (de ocho kilobytes cada uno) se desean en la copia del servidor que nos estará atendiendo. Se corresponde con la opción `-B` descrita en la página 15.1.

`@driver_names = DBI->available_drivers;` Regresa una lista con los controladores que la instalación de DBI conoce.

`@data_sources = DBI->data_sources('Pg');` Es una manera de preguntar por cuáles bases de datos están disponibles en el manejador. Se necesita acceso a la tabla `template1` sin necesidad de autentificarse como usuario.

`DBI->trace($trace_level, $trace_file)` Recolecta información al momento de ejecución en el archivo descrito por `$trace_file`, con el nivel de granularidad `$trace_level`.

`$rv = $h->err;` Regresa el resultado de invocar el método `status`, en caso de invocar una conexión, o `resultStatus` en los demás casos.

`$str = $h->errstr;` Regresa el resultado de invocar al método `errorMessage`.

`$h->trace_msg($message_text);` Envía el mensaje `$message_text` utilizando el método `trace`.

`$sth = $dbh->prepare($statement, /%attr);` Prepara la consulta para ser enviada al servidor. Éste método es genérico en `DBI/DBD` y en realidad PostgreSQL no necesita preparar las consultas.

`@row_ary = $dbh->selectrow_array($statement, /%attr, @bind_values);` Luego de preparar la consulta se extrae la información con éste método dejando la información en el arreglo `@row_ary`.

`$ary_ref = $dbh->selectall_arrayref($statement, /%attr, @bind_values);` Recibe una tupla.

`$rv = $dbh->do($statement, /%attr, @bind_values);` Ejecuta la consulta dada.

`$rc = $dbh->commit;` Realiza un `commit`.

`$rc = $dbh->rollback;` Realiza un `rollback`.

Nombre del tipo	SQL92
bool	BOOL
text	el mismo
bpchar	CHAR(n)
varchar	VARCHAR(n)
int2	SMALLINT
int4	INT
int8	el mismo
money	el mismo
float4	FLOAT(p) p<7=float4, p<16=float8
float8	REAL
abstime	el mismo
reltime	el mismo
tinterval	el mismo
date	el mismo
time	el mismo
datetime	el mismo
timespan	TINTERVAL
timestamp	TIMESTAMP

Tabla 14.2: Correspondencia de tipos de datos entre PostgreSQL y los definidos por el estándar SQL92

`$rc = $dbh->disconnect;` Desconecta del servidor y termina la sesión.

`$rc = $dbh->ping;` Verifica que la conexión esté viva.

`$sth = $dbh->table_info;` Regresa todas las tablas y vistas pertenecientes al usuario que realizó la conexión. No regresa indices, secuencias ni tablas del sistema.

`@names = $dbh->tables;` Idéntico al anterior.

`$sth = $dbh->DBD::Pg::db::attributes($table);` Regresa el `oid`, nombre, tipo y longitud de los atributos de la tabla especificada.

`$type_info_all = $dbh->type_info_all;` Regresa los tipos de acuerdo al mapa especificado en la tabla 14.2 haciendo la conversión entre PostgreSQL y SQL92.

`@type_info = $dbh->type_info($data_type);` Regresa la información del tipo especificado.

`$sql = $dbh->quote($value, $data_type);` Convierte el valor a una cadena segura con todos los carácteres mapeados de tal manera que no haya carácteres inválidos.

`AutoCommit (boolean)` Si se utiliza, todas las transacciones terminarán con un `commit` automáticamente.

`pg_auto_escape (boolean)` Si se utiliza, todas las cadenas que contengan secuencias de escape serán automáticamente convertidas.

`$rv = $sth->execute(@bind_values);` Ejecuta la consulta.

`$rv = $sth->rows;` Regresa el número de tuplas afectadas por la última consulta.

`$blob = $sth->blob_read($id, $offset, $len);` Lee un objeto grande. Si el *offset* y la longitud son cero, regresa el objeto grande completo.

`pg_size (array-ref, read-only)` Regresa el tamaño en bytes de cada columna.

`pg_type (hash-ref, read-only)` Regresa una referencia a un arreglo para cada columna.

`pg_oid_status (integer, read-only)` Regresa el `oid` de la última consulta.

`pg_cmd_status (integer, read-only)` Regresa el tipo de la última consulta (`INSERT`, `DELETE`, `UPDATE`, `SELECT`).

Ejemplo de DBI/DBD

```
$dbname = 'mitabla';
$host = 'miservidor.com';
$options = '-B 2048';
$username = 'lola';
$password = 'lolita';
$dbh = DBI->connect("dbi:Pg:dbname=$dbname;host=$host;options=$options",
    "$username","$password");
$sth = $dbh->prepare("insert into table(foo,bar,baz) values (?,?,?)");
while(<CSV>) {
    chop;
    my ($foo,$bar,$baz) = split /,/;
    $sth->execute($foo, $bar, $baz);
}
```

14.2 PHP

PHP Versión 3.0 es un lenguaje de *script* incrustado en HTML y que corre *en* el servidor del protocolo http. Su sintáxis está basada en C, Java y Perl con algunas novedades propias. La meta del lenguaje es permitir a los desarrolladores de páginas de WEB escribir páginas dinámicas con rapidéz y simplicidad.

El siguiente es un ejemplo escueto, ya que el soporte de PHP para PostgreSQL es extenso y motivo de otro manual.

```
<HTML>
<HEAD>
  <TITLE>Pruebas de PHP3 y PostgreSQL</TITLE>
</HEAD>

<BODY>

<H1>Probando PHP3</H1>

<P>Hola, la fecha es: <?echo date("D M d, Y H:i:s", time())?>

<P>El epoch de ésta máquina es: <?echo time()?>

<P>Las funciones de fechas son:

<CODE>
<UL>
  <LI>Y - Year eg. <?echo date("Y")?>
  <LI>y - Year eg. <?echo date("y")?>
  <LI>M - Month eg. <?echo date("M")?>
  <LI>m - Month eg. <?echo date("m")?>
  <LI>D - Day eg. <?echo date("D")?>
  <LI>d - Day eg. <?echo date("d")?>
  <LI>z - Day of the year eg. <?echo date("z")?>
```

```
  <LI>H - Hours in 24 hour format eg. <?echo date("H")?>
  <LI>h - Hours in 12 hour format eg. <?echo date("h")?>
  <LI>i - Minutes eg. <?echo date("i")?>
  <LI>s - Seconds eg. <?echo date("s")?>
  <LI>U - Seconds since epoch eg. <?echo date("U")?>
</UL>
</CODE>

<P>El info de esta conexion es:
<?php
    echo phpinfo();
?>

<P>Ahora probaremos las funciones de PostgreSQL:

<?php
    $database = pg_connect ("localhost", "5432", "", "", "discos");
    if (!$database) {
        echo "No me puedo conectar\n";
        exit;
    }
    $result = pg_exec ($database, "select * from discos order by autor,titulo");
    if (!$result) {
        echo "No pude ejecutar el exec\n<BR>";
        echo "y la razon es: [";
        echo pg_errormessage($database);
        echo "]\n<BR>";
        exit;
    }
    $nr = pg_numrows ($result);
    echo "Tenemos $nr renglones de resultado\n<BR>";
    $row = 0;
    while ($data = pg_fetch_object ($result, $row)) {
        echo "$row:\t";
        echo $data->autor  . " ==> ";
        echo $data->titulo . " (";
        echo $data->ndis   . " ";
        echo $data->medio  . ")\n<BR>";
        $row++;
    }

?>

<P>Un ejemplo con objetos grandes:

<?php
    $database = pg_Connect ("mimaquina.com", "5432", "", "", "imagenes");
    pg_exec ($database, "begin");
    $oid = pg_locreate ($database);
    echo ("$oid\n");
    $handle = pg_loopen ($database, $oid, "w");
    echo ("$handle\n");
    pg_lowrite ($handle, "gaga");
    pg_loclose ($handle);
    pg_exec ($database, "commit");
    pg_exec ($database, "end");
?>

</BODY>
</HTML>
```

Capítulo 15

Detalles de instalación, puesta a punto del servidor y errores comunes y cómo solucionarlos

15.1 Inicio del servidor

Una opción realmente útil al momento de levantar el servidor es la opción `-F` la cuál le indica al servidor que no sincronice después de cada escritura. Esto es, que sin esta bandera, el servidor hará una escritura física luego de cada escritura lógica a la base de datos para evitar pérdida de datos. Sin embargo, al emplear PostgreSQL en el sistema operativo Linux, podemos confiarle al mismo sistema operativo que éste lleve a cabo la tarea de sincronizar lo que está en memoria con lo que está en el disco duro de una manera eficiente. El empleo de esta bandera acelera la operación del servidor, dado que no es él quién tiene que preocuparse de mantener la sincronía.

Para utilizarla, al levantar el servidor, se le añade: `-o -F`, siendo la opción `-o` para indicarle que la siguiente bandera es para los servidores hijos.

Otra opción realmente útil es emplear la opción `-B` la cuál se refiere a cúantas áreas temporales de memoria utilizará para cada consulta. Estas áreas son de ocho kilobytes y se recomienda hacer diversas pruebas, ya que dependiendo el número de conexiones simultáneas junto con el tamaño de las consultas y, por supuesto, la memoria total disponible, se puede mejorar en mucho el comportamiento. En nuestra experiencia, un servidor con mucha carga de consultas, cada una de ellas muy pesada, se mejora con `-B 2048`.

La opción `-S` desasocia el inicio del servidor de la consola donde se ejecuta. Se recomienda utilizar siempre esta opción, salvo cuando sea necesario obtener información extra del servidor, en cuyo caso, deberá iniciarse con la opción `-d`.

Una opción indispensable para operar con conexiones desde otras máquinas es la opción `-i`. No es el caso de la operación donde sólo será consultado localmente, lo cual incluye consultas desde el servidor de `httpd`, dado que este es un proceso que corre en la misma máquina.

Existen también las opciones `-D`, `-a` y `-b`, las cuales se emplean para indicar dónde se encuentran los datos, el método de autenticación de usuarios y la trayectoria hasta el servidor que habrá de ser ejecutado por cada conexión, respectivamente. En una instalación normal, no hace falta utilizar ninguna de estas banderas.

Algo realmente útil es añadir al `script` de inicialización la siguiente línea, justo antes de arrancar al servidor:

```
rm -f /tmp/.s.PGSQL.5432 /var/lock/subsys/postgresql
```

esto es debido a que habrá ocasiones en que o bien la computadora donde corre el servidor fue indebidamente reinicializada o el servidor mismo fué terminado de una forma inapropiada. Con esta línea, previa la reinicialización del servidor, borramos el archivo `pipe` utilizado por este —en el puerto 5432, el usual— y el archivo que se emplea como candado cuando se tiene a un servidor activo.

No es recomendable terminar el servidor con un `SIGKILL`, en todo caso siempre debe ser terminado con `SIGHUP`, `SIGINT` o `SIGTERM` para garantizar que libera todos los recursos que tiene asignados antes de terminar.

15.2 Autentificación de usuarios

En el archivo `pg_hba.conf` —localizado usualmente en el directorio `/var/lib/pgsql` en la instalación de RedHat— se pueden dar los métodos de autentificación de usuarios a nivel de máquinas, subredes y redes. En el caso más común, sólo deseamos que los usuarios locales, es decir de la misma computadora y los de la red local sean usuarios autentificados del servidor. En este caso basta con tener las siguientes líneas en dicho archivo:

```
local        all                                      trust
host         all         127.0.0.1     255.255.255.255   trust
host         all         192.168.0.0   255.255.255.0     ident     sameuser
```

En la primera línea indicamos que todos los usuarios locales tienen acceso a las bases de datos por medio de `UNIX domain sockets`, es decir únicamente de forma local. Con la segunda línea especificamos que los mismos usuarios locales tienen acceso a las bases de datos por medio de `Internet domain sockets`, es decir, que pueden iniciar una conexión por medio de los servicios de red, que es precisamente el método más empleado. Finalmente, con la tercera línea indicamos que todas las máquinas en la red `192.168.0.0` tienen acceso a todas las bases de datos, siempre y cuando, el protocolo `ident` sea capáz de autentificarlos a un usuario local.

Por supuesto, estos usuarios remotos deben de tener acceso a la base de datos en sí.

Supongamos que queremos que el usuario `fulgano` de una máquina en otra red tenga acceso sólo a una base de datos de nuestro sistema, con las restricciones que fijemos para él en dicha base. Primero debemos de darle acceso a conectarse con el servidor por medio de la línea:

```
host         labase      200.43.51.17  255.255.255.0     crypt
```

con lo cual cualquier usuario de la máquina con la dirección en particular que dimos puede conectarse, siempre y cuando tenga un `password` que sea validado en el sistema local para el usuario que inicia la conexión. El `password` viaja encriptado y es comparado contra el `password` encriptado que se encuentra localmente. Otros métodos seguros son `krb4` y `krb5` que utilizan las versiones 4 y 5 de Kerberos. Un método altamente inseguro, y por ende no recomendado, es `password` donde la contraseña viaja sin encriptar por la red.

Un error muy común es crear una base de datos para ser alimentada por un usuario o un proceso y consultada por medio de `scripts` o `CGIs`, sin tener en cuenta los permisos de acceso. Si iniciamos una sesión en la base de datos a emplear con el cliente `psql`, podemos ver los permisos de cada tabla con el comando `\z`:

```
discos=> \z
Database    = discos
 +-------------+--------------------------+
 | Relation    | Grant/Revoke Permissions |
 +-------------+--------------------------+
```

```
 | discos      |                          |
 | ids         |                          |
 | lesdiscs    |                          |
 | lola        |                          |
 | ndis        |                          |
 | pepa        |                          |
 | pp          |                          |
 | rolas       |                          |
 | seqdis      |                          |
 +-------------+--------------------------+
discos=>
```

y en este caso observamos que no hay ningún tipo de permisos para cada una de las tablas. Esto quiere decir que sólo el usuario dueño de la base puede hacer consultas y actualizar la información. Véase la sección 8.10 para ver cómo habrán de darse los permisos adecuados. Obviamente, en este caso lo que deseamos es que el usuario encargado de mantener la información pueda insertar, leer, actualizar y borrar registros, mientras que los clientes que se conectan por medio del `CGI` sólo pueden hacer consultas. En tal caso, podemos dar todos los permisos al usuario `encargado` y al usuario `nobody`, que por lo general es el usuario con el que el servidor de `httpd` ejecuta los `CGIs` sólo tendrá los permisos de lectura:

```
discos=> grant all on discos,ids,rolas to encargado;
CHANGE
discos=> grant select on discos,ids,rolas to nobody;
CHANGE
discos=> \z
Database    = discos
 +-------------+-----------------------------------+
 | Relation    | Grant/Revoke Permissions          |
 +-------------+-----------------------------------+
 | discos      | {"=","encargado=arwR","nobody=r"} |
 | ids         | {"=","encargado=arwR","nobody=r"} |
 | lesdiscs    |                                   |
 | lola        |                                   |
 | ndis        |                                   |
 | pepa        |                                   |
 | pp          |                                   |
 | rolas       | {"=","encargado=arwR","nobody=r"} |
 | seqdis      |                                   |
 +-------------+-----------------------------------+
discos=>
```

Mientras que a las tablas `lesdiscs`, `lola`, `ndis`, `pepa`, `seqdis` y `pp` sólo el dueño de la base de datos tiene acceso.

Bibliografía

[Ullman1999] **Ullman, Jeffrey** y **Widom, Jennifer**, *Introducción a los Sistemas de Bases de Datos.* Editorial Prentice Hall, México 1999, ISBN: 970-17-0256-5.

[Date1993] **Date, C. J.**, *Introducción a los Sistemas de Bases de Datos, Volúmen I*, quinta edición, Editorial Addisson Wesley Longman, México 1998, ISBN: 968-444-220-3.

[Silberschatz] **Silberschatz, Abraham, Korth, Henry F.**. y **Sudarshan, S.**, *Fundamentos de Bases de Datos*, tercera edición, Editorial McGraw Hill, España 1998, ISBN: 0-07-044756-X.

Índice de Materias