Instituto Tecnológico Superior de Jerez.

Jerez de García Salinas a 20 de Marzo del 2020.

Cristofer Casas Murillo.

cristofer32513@gmail.com

S17070157.

INGENIERÍA EN SISTEMAS COMPUTACIONALES.

Administración de Bases de Datos.

6to. Semestre.

Cuestionario y Mapa Conceptual (Particionamiento).

ISC. Salvador Acevedo Sandoval.

MySQL.

1. Asignación de espacio en disco para base de datos.

En general, el almacenamiento de la base de datos se hace a través de estructuras lógicas de almacenamiento que permiten que el espacio en disco se asigne de forma dinámica cuando la base de datos crece de tamaño más de lo previsto.

2. Asignación de espacio en disco para tablas.

Formado por uno o más datafiles, cada datafile solo puede pertenecer a un determinado tablespace, básicamente consiste en páginas de base de datos con un tamaño por defecto de 16KB. Las páginas se agrupan en áreas de 64 páginas consecutivas.

3. Asignación de espacio en disco para usuarios.

De funciona de manera similar a la de las tablas.

4. Particionamiento de tablas

a) ¿Qué es y para qué se utiliza?

Es una técnica que se usa para reducir la cantidad de lecturas físicas a la base de datos cuando ejecutamos consultas, es decir, es el proceso donde tablas muy grandes son divididas en múltiples partes más pequeñas.

Se utiliza para separar una tabla grande en tablas individuales más pequeñas, de este modo, las consultas que acceden pueden ejecutarse más rápido debido a que sólo a una fracción de los datos.

También ayuda con el mantenimiento y la reducción del tiempo de respuesta en general.

b) Tipos.

Según quien realice la gestión del particionado, podemos distinguir dos tipos de particionado:

- **Manual:** El particionado lo podríamos realizar nosotros en nuestra lógica de procesos de carga ETL. El problema es que se tendrá que gestionar este particionado para saber de qué tabla se tienen que leer los datos que le estemos pidiendo.
- **Automático:** La gestión la realiza de forma automática el motor de base de datos tanto a la hora de insertar registros como a la hora de leerlos, esto según nos permita el SGBDR que estemos utilizando.

MySQL implementa el particionado horizontal:

- **RANGE:** La asignación de los registros de la tabla a las diferentes particiones se realiza según un rango de valores definido sobre una determinada columna de la tabla o expresión. Es decir, nosotros indicaremos el número de particiones a crear, y para cada partición, el rango de valores que serán la condición para insertar en ella.

- **LIST:** La asignación de los registros de la tabla a las diferentes particiones se realiza según una lista de valores definida sobre una determinada columna de la tabla o expresión. Es decir, nosotros indicaremos el número de particiones a crear, y para cada partición, la lista de valores que serán la condición para insertar en ella.

- **HASH:** Está pensado para repartir de forma equitativa los registros de la tabla entre las diferentes particiones. Es decir, es MySQL quien hace ese trabajo. Para definir este tipo de particionado, deberemos de indicarle una columna del tipo integer o una función de usuario que devuelve un integer.

- **KEY:** Similar al HASH, pero la función para el particionado la proporciona MySQL automáticamente. Se pueden indicar los campos para el particionado, pero siempre han de ser de la clave primaria de la tabla o de un índice único.

Además, debemos tener en cuenta que la definición de particiones no es estática. Es decir, MySQL tiene herramientas para poder cambiar la

configuración del particionado, en otras palabras, permite añadir o suprimir particiones existentes, fusionar particiones en otras, dividir una partición en varias, etc.

c) Limitaciones/restricciones.

El particionado tiene sus limitaciones y sus restricciones, pues no se puede realizar sobre cualquier tipo de columna o expresión, tenemos un límite de particiones a definir y habrá que tener en cuenta algunas cosas para mejorar el rendimiento de las consultas y evitar que estas se recorran todas las particiones de una.

- **Construcciones prohibidas:**
  - Procedimientos almacenados, funciones almacenadas, UDF o complementos.
  - Variables declaradas o variables de usuario.

- **Operadores aritméticos y lógicos.**
  - El uso de los operadores aritméticos +, - y * se permite en la partición de expresiones. Sin embargo, el resultado debe ser un valor entero o NULL (excepto en el caso de [LINEAR]).
  - El operador DIV también es compatible; El operador "/"no está permitido.
  - Los operadores de bits |, &, ^, <<, >>, y ~ no se permiten.

- **Número máximo de particiones:** El número máximo posible de particiones para una tabla dada que no usa el motor de almacenamiento NDB es de 8192 (este número incluye subparticiones).

- **Claves foráneas no admitidas:** Las tablas particionadas que usan el motor de almacenamiento InnoDB no admiten claves foráneas.

d) Instrucciones de ejemplo para cada tipo.

- **Crear partición:**
  - **Range.**

```
CREATE TABLE `salaries_partitions_salary` (
  `emp_no` int NOT NULL,
  `salary` int NOT NULL,
  `from_date` date NOT NULL,
  `to_date` date NOT NULL)

PARTITION BY RANGE(salary)(
  PARTITION part1 VALUES LESS THAN(30000),
  PARTITION part2 VALUES LESS THAN(60000),
  PARTITION part3 VALUES LESS THAN(90000),
  PARTITION part4 VALUES LESS THAN(120000),
  PARTITION part5 VALUES LESS THAN(150000),
  PARTITION part_default VALUES LESS THAN MAXVALUE
);
```

- **List.**

```
CREATE TABLE `category_partitions_category_id` (
  `category_id` tinyint unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(25) NOT NULL,
  `last_update` timestamp NOT NULL DEFAULT
    CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`category_id`))
PARTITION BY LIST(category_id)(
  PARTITION accion VALUES IN(1, 2, 3, 4),
  PARTITION terror VALUES IN(5, 6, 7, 8),
  PARTITION ciencia_ficcion VALUES IN(9, 10, 11, 12),
  PARTITION romance VALUES IN(13, 14, 15, 16)
);
```

- **Hash**

```
CREATE TABLE `employees_partition_emp_no` (
  `emp_no` int NOT NULL,
  `birth_date` date NOT NULL,
  `first_name` varchar(14) NOT NULL,
  `last_name` varchar(16) NOT NULL,
```

```
`gender` enum('M','F') NOT NULL,
`hire_date` date NOT NULL,
PRIMARY KEY (`emp_no`))
PARTITION BY HASH(emp_no) PARTITIONS 5;
```

- **Key**

```
CREATE TABLE `departments_partition_by_dept_no` (
`dept_no` char(4) NOT NULL,
`dept_name` varchar(40) NOT NULL,
PRIMARY KEY (`dept_no`))
PARTITION BY KEY() PARTITIONS 3;
```

- **Borrar partición:**

  ALTER TABLE [nombre_tabla] DROP PARTITION [nombre_particion];

- **Añadir particiones:**
  - Si se tiene una partición default:
    - ALTER TABLE [nombre_tabla] REORGANIZE PARTITION [nombre_particion_default] INTO (
    - PARTITION [nombre_particion] VALUES LESS THAN (TO_DAYS("2012-07-01"),
    - PARTITION [nombre_particion_default] VALUES LESS THAN MAXVALUE));
  - Sin partición default:
    - ALTER TABLE [nombre_tabla] ADD PARTITION (
    - PARTITION [nombre_particion] VALUES LESS THAN (TO_DAYS("2012-07-01")));

- **Consultar partición:**

```
SELECT PARTITION_NAME,TABLE_ROWS FROM
information_schema.PARTITIONS WHERE
TABLE_NAME='[nombre_tabla]';
```

PostgreSQL.

1. Asignación de espacio en disco para base de datos.

Se realiza dinámicamente.

2. Asignación de espacio en disco para tablas.

Puede configurarse manualmente o de forma dinámica sin exceder el límite.

3. Asignación de espacio en disco para usuarios.

De manera similar al de las tablas

4. Particionamiento de tablas

a) ¿Qué es y para qué se utiliza?

En PostgreSQL el tipo de particionamiento soportado se denomina particionado mediante herencia de tablas. Cada partición puede ser creada como una tabla hija de una única tabla padre.

Sirve para reducir la cantidad de datos a recorrer en cada consulta SQL y aumentar el rendimiento.

b) Tipos.

Existen dos tipos de particionado más comunes en PostgreSQL, que son:

- **Particionamiento por rangos:** La tabla es particionada mediante rangos definidos en base a la columna de llave primaria o cualquier columna que no se solape entre los rangos de valores asignados a diferentes tablas hijas.
- **Particionamiento por lista:** La tabla es particionada listando los valores de cada una de las llaves en cada partición.

c) Limitaciones/restricciones.

El tamaño máximo de una tabla puede ser de hasta 32 de TB siendo esto más que suficiente, sin embargo, sin el mantenimiento adecuado y sin una estrategia adecuada de particionamiento, este tipo de tablas serán casi inutilizables después de que pese 1TB.

d) Instrucciones de ejemplo para cada tipo.

Para la creación de particiones se tomó en cuenta un ejemplo.

- **Crear las tablas hijas.**

  test=# CREATE TABLE prueba_1 (CHECK (col >0 AND col <1000001)) INHERITS (prueba);

- **Crear reglas adicionales.**

  test=# CREATE RULE prueba_3_rule AS ON INSERT TO prueba WHERE (col >2000000 AND col <3000001) DO INSTEAD INSERT INTO prueba_3 VALUES (NEW.*);

- **Insertar valores.**

  test=# INSERT INTO prueba_6 SELECT * FROM prueba WHERE (col >5000000 AND col <6000001);

- **Eliminar datos de la tabla padre sin usar la cláusula ONLY.**

  test=# DELETE FROM prueba;
  DELETE 12000000
  test=# SELECT count(*) FROM prueba;
  count
  ——-
  0
  (1 row)

- **Eliminar datos de la tabla padre usando la cláusula ONLY.**

  test=# DELETE FROM ONLY prueba;
  DELETE 6000000
  test=# SELECT count(*) FROM ONLY prueba;
  count

_____

0

(1 row)

Podemos observar que, al realizar una consulta sin usar dicha cláusula, el resultado es la suma de las filas de las tablas hijas:

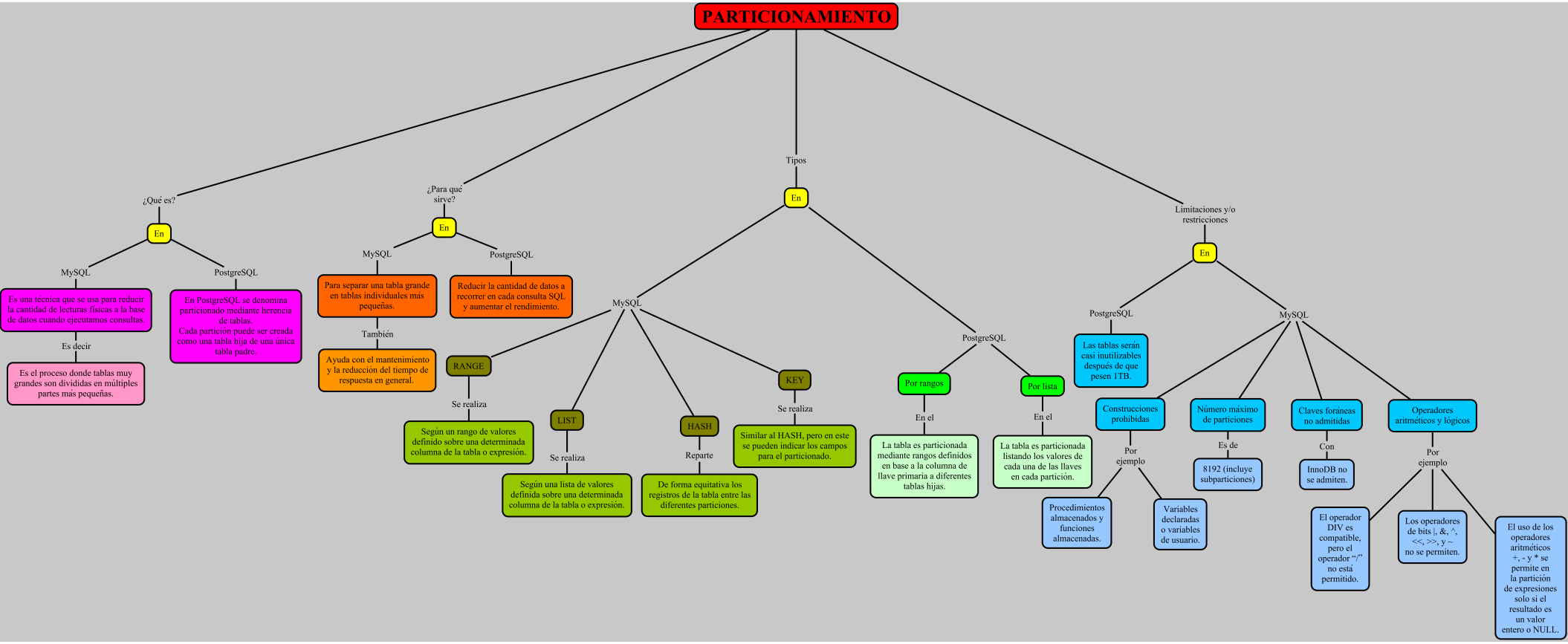
test=# SELECT count(*) FROM prueba;

count

_____

6000000

(1 row)


PARTICIONAMIENTO
¿Qué es?
En
MySQL
Es una técnica que se usa para reducir la cantidad de lecturas físicas a la base de datos cuando ejecutamos consultas.
Es decir
Es el proceso donde tablas muy grandes son divididas en múltiples partes más pequeñas.
PostgreSQL
En PostgreSQL se denomina particionado mediante herencia de tablas.
Cada partición puede ser creada como una tabla hija de una única tabla padre.
¿Para qué sirve?
En
MySQL
Para separar una tabla grande en tablas individuales más pequeñas.
También
Ayuda con el mantenimiento y la reducción del tiempo de respuesta en general.
PostgreSQL
Reducir la cantidad de datos a recorrer en cada consulta SQL y aumentar el rendimiento.
Tipos
En
MySQL
RANGE
Se realiza
Según un rango de valores definido sobre una determinada columna de la tabla o expresión.
LIST
Se realiza
Según una lista de valores definida sobre una determinada columna de la tabla o expresión.
HASH
Reparte
De forma equitativa los registros de la tabla entre las diferentes particiones.
KEY
Se realiza
Similar al HASH, pero en este se pueden indicar los campos para el particionado.
PostgreSQL
Por rangos
En el
La tabla es particionada mediante rangos definidos en base a la columna de llave primaria a diferentes tablas hijas.
Por lista
En el
La tabla es particionada listando los valores de cada una de las llaves en cada partición.
Limitaciones y/o restricciones
En
PostgreSQL
Las tablas serán casi inutilizables después de que pesen 1TB.
MySQL
Construcciones prohibidas
Por ejemplo
Procedimientos almacenados y funciones almacenadas.
Variables declaradas o variables de usuario.
Número máximo de particiones
Es de
8192 (incluye subparticiones)
Claves foráneas no admitidas
Con
InnoDB no se admiten.
Operadores aritméticos y lógicos
Por ejemplo
El operador DIV es compatible, pero el operador "/" no está permitido.
Los operadores de bits |, &, ^, <<, >>, y ~ no se permiten.
El uso de los operadores aritméticos +, - y * se permite en la partición de expresiones solo si el resultado es un valor entero o NULL.

Referencias

- NE. (NE). Particiones de base de datos. 2020, de ibm.com Sitio web: https://www.ibm.com/support/knowledgecenter/es/SSH2TE_1.0.0/com.ibm.7700.r2.common.doc/doc/c00000092.html
- Leonardo De Seta. (2009). Particionado de tablas para aumentar el rendimiento. 2020, de dosideas.com Sitio web: https://dosideas.com/noticias/base-de-datos/576-incrementar-rendimiento-de-bbdd-con-qparticionado-de-tablasq
- NE. (NE). Chapter 23 Partitioning. 2020, de mysql.com Sitio web: https://dev.mysql.com/doc/refman/8.0/en/partitioning.html
- NE. (NE). Tabla Hechos Venta. Particionado en MySql.. 2020, de dataprix.com Sitio web: https://www.dataprix.com/es/blog-it/respinosamilla/tabla-hechos-venta-particionado-mysql
- Daniel Santana. (NE). Particiones en MySQL y Oracle. 2020, de blogspot.com Sitio web: http://dan1456bd.blogspot.com/p/particiones-en-mysql-y-oracle.html
- NE. (2019). FRAGMENTACIÓN HORIZONTAL EN MYSQL. 2020, de blogprog.gonzalolopez.es Sitio web: https://blogprog.gonzalolopez.es/articulos/fragmentacion-horizontal-en-mysql.html
- PostgreSQL. (27 de noviembre de 2009). Obtenido de PostgreSQL: https://beastieux.com/2009/11/27/postgresql-particionamiento-de-tablas/
- Rozvodb. (18 de mayo de 2018). Obtenido de Rozvodb: http://rozvodb.com/node/8
- NE. (NE). 15.14.3. Gestión del espacio de ficheros. 2020, de download.nust.na Sitio web: http://download.nust.na/pub6/mysql/doc/refman/5.0/es/innodb-file-space.html
- NE. (NE). Unidad 3: Configuración y Administración del Espacio en Disco. 2020, de sites.google.com Sitio web: https://sites.google.com/site/itjabd23/home/asignatura/plan-de-estudios/unidad-3-configuracion-y-administracion-del-espacio-en-disco
- NE. (2017). ¿Cuánto espacio ocupan las tablas y bases de datos de un servidor PostgreSQL?. 2020, de linuxito.com Sitio web: https://www.linuxito.com/gnu-linux/nivel-medio/841-cuanto-espacio-ocupan-las-tablas-y-bases-de-datos-de-un-servidor-postgresql