Titulación: Grado en Ingeniería Informática, Ingeniería en Sistemas de Información e InfoAde
Curso: 2025-2026. Convocatoria Ordinaria de Mayo
Asignatura: Bases de Datos Avanzadas – Laboratorio

Practica 1: Arquitectura PostgreSQL y almacenamiento físico

ALUMNO 1:

Nombre y Apellidos: Lucas Marcos Vaquerizo

DNI: 78120941S

ALUMNO 2:

Nombre y Apellidos: Juan Gil Aragonés

DNI: _____

Fecha: _____

Profesor Responsable: _____

Mediante la entrega de este fichero los alumnos aseguran que cumplen con la normativa de autoría de trabajos de la Universidad de Alcalá, y declaran éste como un trabajo original y propio.

En caso de ser detectada copia, se calificará la asignatura como Suspensa – Cero.

Es obligatorio proporcionar una explicación a lo que está ocurriendo en PostgreSQL cuando así se indica en la cuestión. No solo vale poner un pantallazo. La ausencia de una explicación hará que sea invalidada esa cuestión.

Introducción

En esta primera práctica se introduce el sistema gestor de bases de datos PostgreSQL (18.1 la última). Está compuesto básicamente de un motor servidor y de una serie de clientes que acceden al servidor y de otras herramientas externas. En esta primera práctica se entrará a fondo en la arquitectura de PostgreSQL, sobre todo en el almacenamiento físico de los datos y del acceso a los mismos. Antes de comenzar es obligatorio configurar lo que se comenta en la cuestión. Hay que resolver la práctica con consultas SQL.

Cuestión 0. Configurar el fichero de Error Reporting and Logging de PostgreSQL para que aparezcan recogidas las sentencias SQL DDL (Lenguaje de Definición de Datos) + DML (Lenguaje de Manipulación de Datos) generadas en dicho fichero. No se pide activar todas las sentencias. No activar la duración de la consulta. También se debe de configurar el log para que en el comienzo de la línea de registro de la información del log ("line prefix") aparezcan vuestros DNI's. ¿Cómo se ha realizado la configuración?

Para encontrar el archivo de la configuración de Postgres, desde la terminal del propio proyecto en DataGrip hemos escrito la sentencia "SHOW config_file;" que nos da la dirección, una vez dentro hemos visto que la mayor parte de la configuración ya está hecha, solo hemos tenido que modificar la parte de "log_line_prefix" para añadir nuestro DNI al comienzo de cada registro y el apartado que dice "log_statement", que se ha puesto en "mod" para que no se tenga un registro de todo.
Apartados como "log_duration" o "log_min_duration_statmen" ya estaban como queríamos.
Tras los cambios y comprobaciones se ha reiniciado el servidor Postgres desde el gestor de servicios del ordenador, para poder comprobar que los cambios se han aplicado abrimos Datagrip y hacemos un CREATE TABLE y varios INSERT, UPDATE, DELETE FROM para comprobar que los logs se hacen correctamente y poniendo al inicio de cada registro el DNI de la persona que los ha hecho.

Organización de Archivos en PostgreSQL

Cuestión 1. Crear una nueva Base de Datos que se llame **PL1**. Después crear una tabla **estudiantes** con los siguientes campos:

- estudiante_id: que debe ser un serial empezando por 1.
- nombre: guarda el nombre del estudiante.
- codigo_carrera: guarda el código de carrera del alumno.
- edad: guarda la edad del estudiante.
- indice: guarda el número de créditos superados por el alumno.

Crear un programa que permita generar 30 millones de registros en un fichero de texto que pueda ser cargado posteriormente en la tabla (preferiblemente en Python) con las siguientes propiedades para los siguientes campos, cuyos valores se deben generar aleatoriamente.

- Código de carrera: deben ser valores aleatorios entre 0 y 100.
- edad: deben ser valores aleatorios entre 18 y 40 años (incluidos).
- índice: deben ser valores aleatorios entre 0 y 10000

Cargar los datos en la tabla y localizar los ficheros relacionados con la tabla. ¿cómo se localizan? ¿Cuánto ocupan? ¿por qué?

Los ficheros se localizan accediendo al OID de la tabla, este se obtiene aplicando el comando "SELECT oid, relname
FROM pg_class
WHERE relname = 'estudiantes';".
Esto nos da el número por el que se guarda la tabla dentro del directorio de Postgres (dentro de data/base), una vez aquí se puede ver cuanto ocupan todos los archivos relacionados a la tabla, utilizamos el comando "SELECT pg_size_pretty(pg_total_relation_size('estudiantes'));" para que nos de el espacio total de este tabla (unos 2597 MB), el gran tamaño se debe a la gran cantidad de datos y a que se separan en varias páginas que también guardan metadatos, haciendo que el tamaño total se eleve (el tamaño del .csv que contiene los datos era originalmente de 899MB).

Cuestión 2. Calcular teóricamente el tamaño en bloques que ocupa la relación **estudiantes** tal y como se realiza en clase de teoría. ¿Concuerda con el tamaño en bloques que nos proporciona PostgreSQL? ¿Cuál es el factor de bloque medio real de la tabla **estudiantes**? ¿Por qué? Realizar una consulta SQL que obtenga ese valor y comparar con el factor de bloque teórico.

Para calcular la cantidad de bloques de manera teórica primero tenemos que ver cuanto ocupa
El tamaño de cada registro en nuestra base de datos será la suma de cada campo (estudiante_i
El número de registros por bloque por lo tanto será de (6553/46) 142 registros por cada bloqu
Usando el comando:
"

```
SELECT
    pg_database.datname,
    pg_database_size(pg_database.datname) / 8192 AS bloques_usados
FROM pg_database
WHERE datname = current_database();
```

"
El propio DataGrip nos dice que el número de bloques es 333.373 (mucho más parecido al que

Cuestión 3. Realizar una consulta que muestre los estudiantes que tengan un índice de 500. ¿Cuántas tuplas se obtienen y cuántos bloques se leen del disco por Postgres? ¿Por qué? Obtener las estadísticas de la tabla y sus campos, para comparar con los resultados obtenidos al aplicar el método visto en teoría. (usar funciones de "The Cumulative Statistics System")

Al realizar la consulta correspondiente (COUNT(*) de los elementos de la tabla con esas caracte
Desde la consola de DataGrip vemos que se están leyendo 248.694 bloques, tiene sentido que
Para ver realmente cuantos bloques se leen y cuantas tuplas se estiman que hay hacemos:
"SELECT
    relname,
    reltuples AS tuplas_estimadas,
    relpages AS bloques
FROM pg_class
WHERE relname = 'estudiantes';"
Lo que nos da números muy parecidos a los que ya teníamos.
Para las estadísticas del campo "Índice" obtenemos que la cardinalidad es de 9999.

Cuestión 4. Volver a realizar la consulta de la cuestión 3 de nuevo. ¿Cuántas tuplas se obtienen y cuántos bloques se leen del disco por Postgres ahora? ¿Por qué? Comparar con la cuestión anterior

Al realizar la consulta de nuevo se obtienen las mismas tuplas (lógico pues la base de datos no ha cambiado), al mismo tiempo también se leen los mismos bloques que en la cuestión anterior, si la base de datos fuera menor el buffer podría almacenar en memoria los datos de la consulta anterior para no tener que volver a leer casi 250.000 bloques, pero si hacemos la consulta "SHOW shared_buffers" vemos que la capacidad es de 128MB, por lo que es muy probable que no se puedan almacenar todos los datos de la consulta anterior, forzando al sistema a volver a pasar por todos los registros secuencialmente (igual que en el caso anterior).

Cuestión 5. Crear una tabla **estudiantes2** cuyas tuplas estén ordenadas físicamente por el campo índice de menor a mayor y que tenga la misma información. Cargar el mismo fichero de datos creado en la cuestión1. Indicar el proceso de generación de dicha tabla ordenada. ¿Cuántos bloques ocupa la tabla ahora? ¿Hay algún cambio? ¿Por qué?

Generamos la tabla "estudiantes2" cargando previamente los datos en una tabla intermedia (es
Por último hacemos "DROP TABLE estudiantes_tmp" para limpiar la tabla temporal (ya no nos h

Cuestión 6. Repetir la cuestión 3 sobre la tabla **estudiantes2** y comparar los resultados obtenidos indicando las conclusiones obtenidas. Relacionarlo con lo visto en teoría.

Si analizamos los resultados de buscar todos los datos cuyo índice es 500 vemos que apenas h

Cuestión 7. Borrar 5000000 tuplas de la tabla **estudiantes** de manera aleatoria usando el valor del campo estudiante_id. ¿Qué es lo que ocurre físicamente en la base de datos? ¿Se observa algún cambio en el tamaño de la tabla y estructuras asociadas a ella? ¿Por qué? Adjuntar el código de borrado.

Cuestión 8. Insertar un nuevo estudiante en la tabla **estudiantes**. ¿dónde se inserta físicamente la nueva tupla? ¿Por qué? (Mirar apartado “System Columns”)

Cuestión 9. En la situación anterior, ¿Qué operaciones se pueden aplicar a la base de datos **PL1** para optimizar el rendimiento de esta? Aplicarlas de tal manera que se recupere el mayor espacio posible. Comentar cuál es el resultado final y qué es lo que ocurre físicamente. (Mirar "Routine Database Maintenance Tasks")

Cuestión 10. Crear una nueva tabla denominada **estudiantes3** con los mismos campos que la cuestión 1 y que esté particionada por el campo código de carrera por medio de la función h = código_carrera mod 20. Insertar los datos del fichero de datos generado en la cuestión 1. Explicar el proceso seguido y comentar qué es lo que ha ocurrido físicamente en la base de datos. ¿Cuándo será útil el particionamiento? ¿Cuántos bloques ocupa cada una de las particiones? ¿Por qué? Comparar con el número bloques que se obtendría teóricamente utilizando el procedimiento visto en teoría. (Ver comando "CREATE TABLE")

Cuestión 11. Repetir la cuestión 3 sobre la tabla **estudiantes3** y comparar los resultados obtenidos con lo visto anteriormente en las tablas **estudiantes** y **estudiantes2** obteniendo conclusiones sobre el método de partición.

Indexación de PostgreSQL

PostgreSQL soporta indexación definida por el usuario para ayudar a acelerar ciertas consultas. Entre otros tipos de índices soporta árboles y hash. En este apartado se va a trabajar sobre ambos tipos de índices, pudiendo observar cómo se organizan internamente y su funcionamiento. (Mirar módulos ”pgstatuple” y “pageinspect”)

Cuestión 12. Borrar todas las tablas **estudiantes, estudiantes2** y **estudiantes3**. Crear una nueva tabla que se llama **estudiantes2** como en la cuestión 5 (prdenado por el campo **índice**) y que tenga cargados todos los datos del fichero de texto generado.

Cuestión 13. Crear un índice de tipo árbol para **estudiante_id**. ¿Dónde se almacena físicamente ese índice? ¿Qué tamaño tiene? ¿Cuántos bloques tiene? ¿Cuántos niveles tiene? ¿Cuántos bloques tiene por nivel? ¿Cuántas tuplas tiene un bloque de cada nivel? Indicar el procedimiento seguido e incluir el código SQL utilizado.

Cuestión 14. Determinar el tamaño de bloques y el número de niveles que teóricamente tendría de acuerdo con lo visto en teoría. Comparar los resultados obtenidos teóricamente con los resultados obtenidos en la cuestión 13.

Cuestión 15. Crear un índice de tipo hash para el campo **estudiante_id**. ¿Dónde se almacena físicamente ese índice? ¿Qué tamaño tiene? ¿Cuántos bloques tiene? ¿Cuántos cajones tiene? ¿Cuántas tuplas tiene de media un cajón? Indicar el procedimiento seguido e incluir el código SQL utilizado.

Cuestión 16. Determinar el tamaño de bloques y el número de cajones que teóricamente tendría de acuerdo con lo visto en teoría. Comparar los resultados obtenidos teóricamente con los resultados obtenidos en la cuestión 15.

Cuestión 17. Crear un índice de tipo árbol para el campo **índice**. ¿Dónde se almacena físicamente ese índice? ¿Qué tamaño tiene? ¿Cuántos bloques tiene? ¿Cuántos niveles tiene? ¿Cuántos bloques tiene por nivel? ¿Cuántas tuplas tiene un bloque de cada nivel? Indicar el procedimiento seguido e incluir el código SQL utilizado.

Cuestión 18. Determinar el tamaño de bloques y número de niveles que teóricamente tendría de acuerdo con lo visto en teoría. Comparar los resultados obtenidos teóricamente con los resultados obtenidos en la cuestión 17.

Cuestión 19. Crear un índice de tipo hash para el campo **índice**. ¿Dónde se almacena físicamente ese índice? ¿Qué tamaño tiene? ¿Cuántos bloques tiene? ¿Cuántos cajones tiene? ¿Cuántas tuplas tiene de media un cajón? Indicar el procedimiento seguido e incluir el código SQL utilizado.

Cuestión 20. Determinar el tamaño de bloques y número de cajones que teóricamente tendría de acuerdo con lo visto en teoría. Comparar los resultados obtenidos teóricamente con los resultados obtenidos en la cuestión 17.

Cuestión 21. ¿Qué conclusiones se puede obtener de la gestión y organización de PostgreSQL sobre los dos índices árbol y hash que se han creado y han sido analizados? ¿Por qué? Comparar con lo visto en teoría.

Monitorización de la actividad de la base de datos

En este último apartado se mostrará el acceso a los datos con una serie de consultas sobre la tabla original. En este apartado se pretende mostrar cómo es el acceso a los datos para diferentes tipos de consultas.

PostgreSQL suministra varias vistas estadísticas que se pueden usar para monitorizar los bloques leídos (tipo statio de la sección The Cumulative Statistics System) de cada una de las estructuras creadas en la base de datos. En este apartado se deben usar esas vistas y está prohibido el uso de otro comando para este fin (Table 28.2).

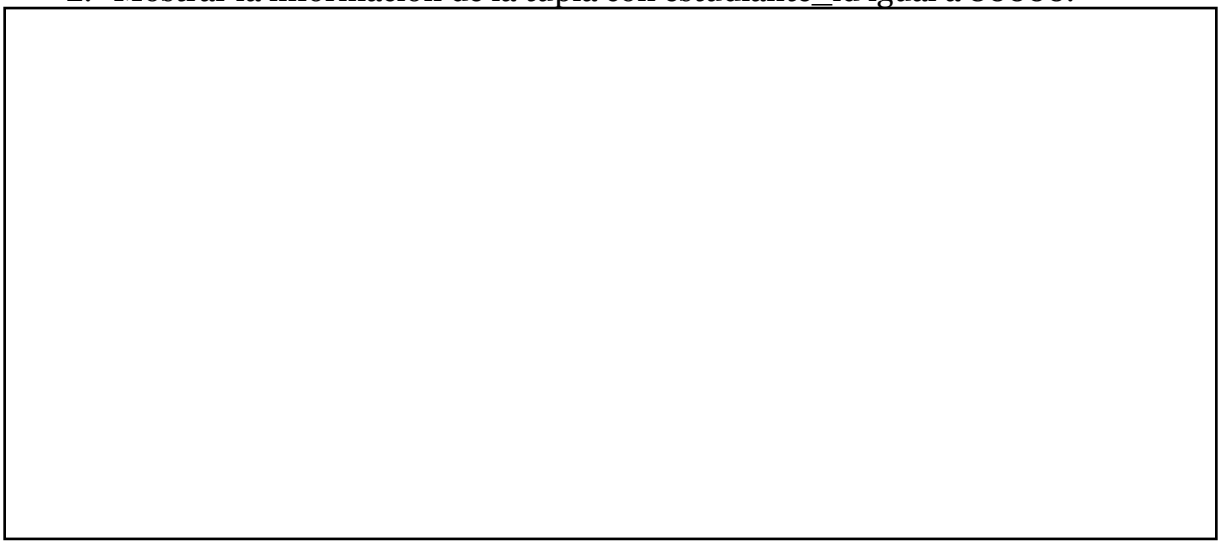
Para ello, borrar todas las tablas creadas y volver a crear la tabla **estudiantes** como en la cuestión 1. Cargar los datos que se encuentran originalmente en el fichero generado en la cuestión 1.

Cuestión 22. Crear un índice primario tipo árbol sobre el campo **índice**, y otro sobre **código_carrera**. También crear un índice hash sobre el campo **estudiante_id** y otro sobre **índice**. ¿Cuál ha sido el proceso seguido para crear cada tipo de índice? Incluir el código SQL utilizado para ello.

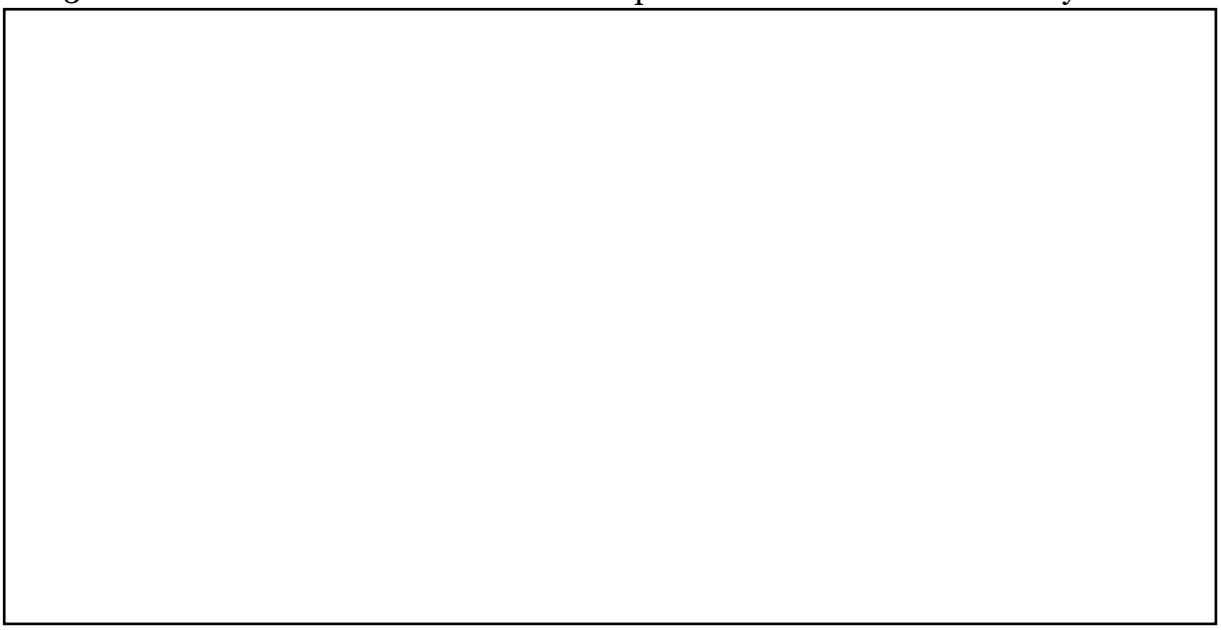
Cuestión 23. Para cada una de las consultas que se muestran a continuación, ¿Qué información se puede obtener de los datos monitorizados por la base de datos al realizar la consulta? Comentar cómo se ha realizado la resolución de la consulta. ¿Cuántas tuplas se han obtenido? ¿Cuántos bloques se han leído de cada estructura? ¿Por qué? Comparar con lo visto en teoría. Importante, reinicializar los datos recolectados de la actividad de la base de datos antes de lanzar cada consulta. Se recuerda que solo se pueden usar vistas sobre las estadísticas de la base de datos (usar funciones de "The Cumulative Statistics System").

1. Mostrar la información de las tuplas con código de carrera igual 50.

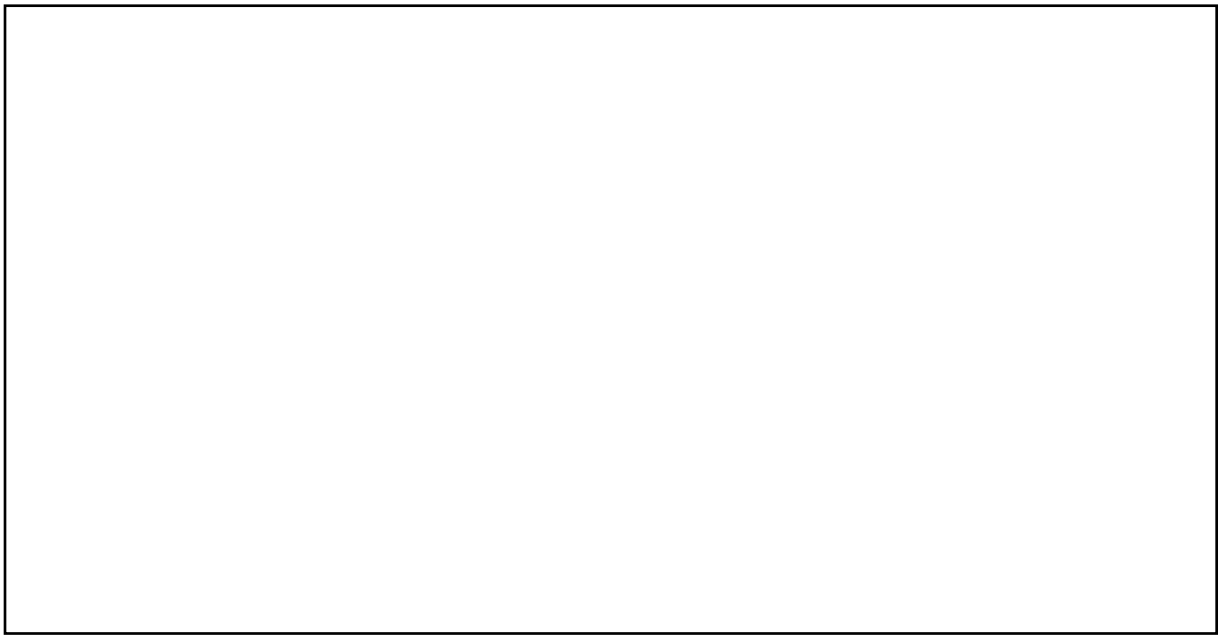
2. Mostrar la información de la tupla con estudiante_id igual a 80000.



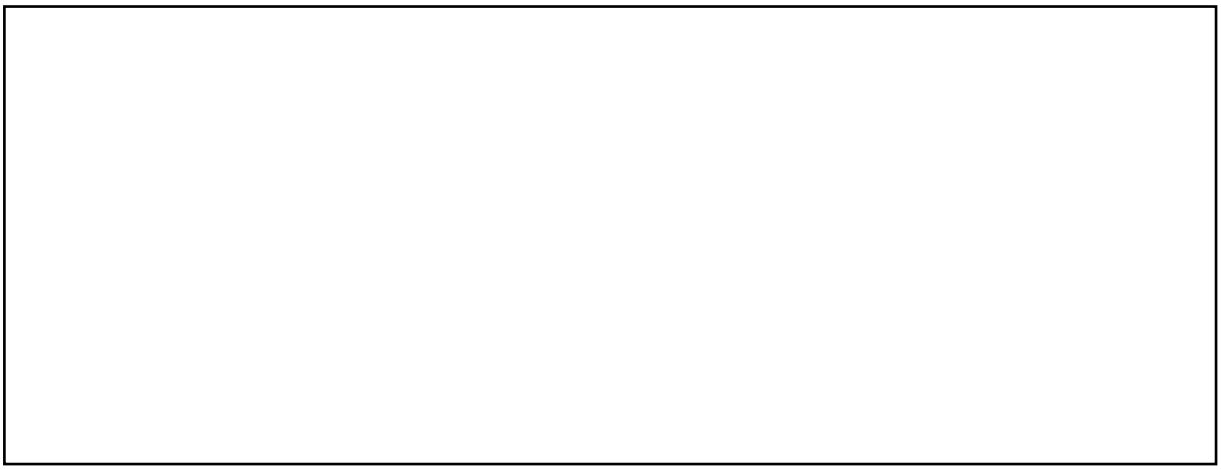
3. Mostrar los datos de los estudiantes que tienen un índice entre 100 y 200.



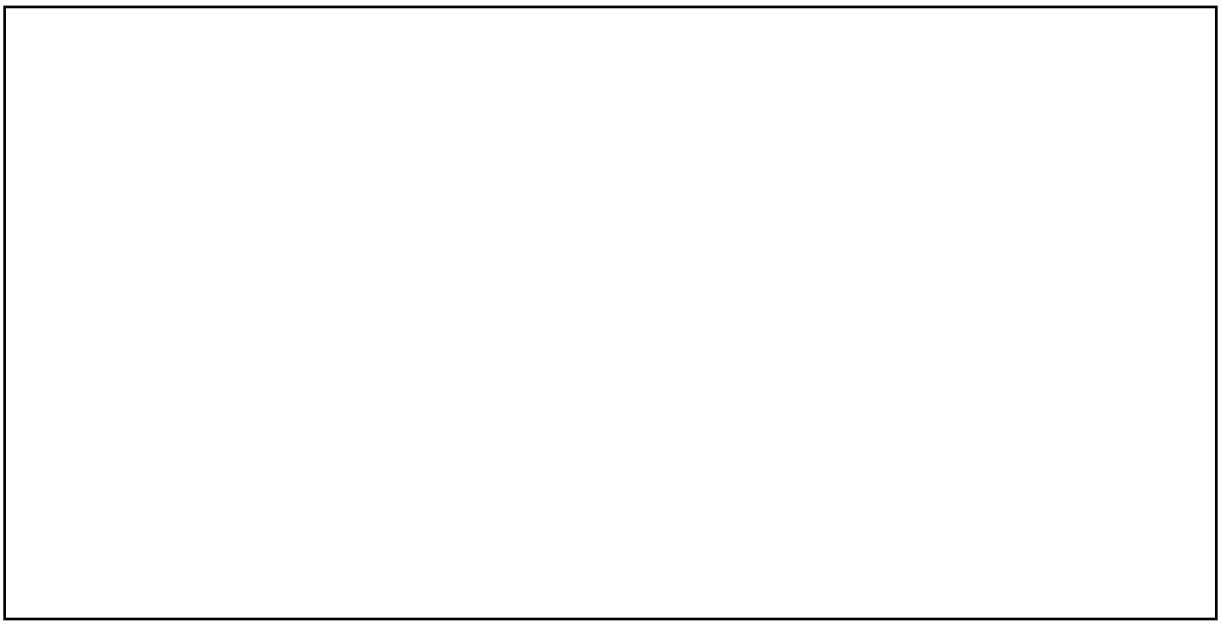
4. Contar el número de estudiantes que tienen una edad de 20 años.



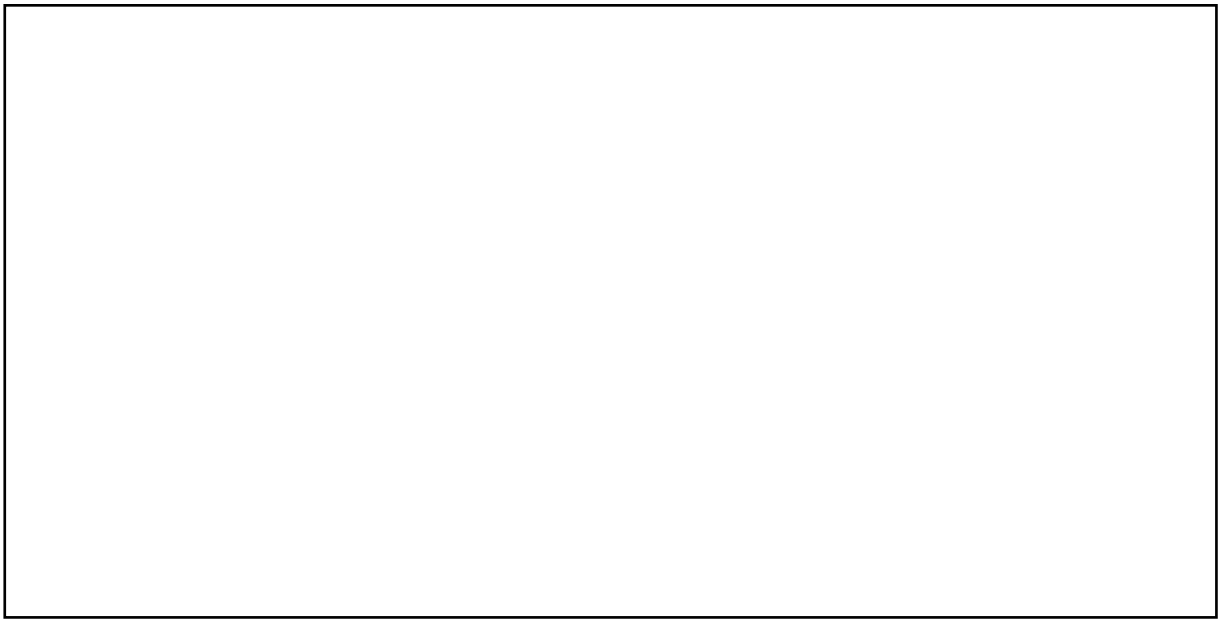
5. Mostrar el número total de estudiantes que tienen el mismo índice.



6. Obtener la media de edad de los estudiantes que tienen la misma carrera.



7. Insertar un nuevo estudiante con un índice de 100.



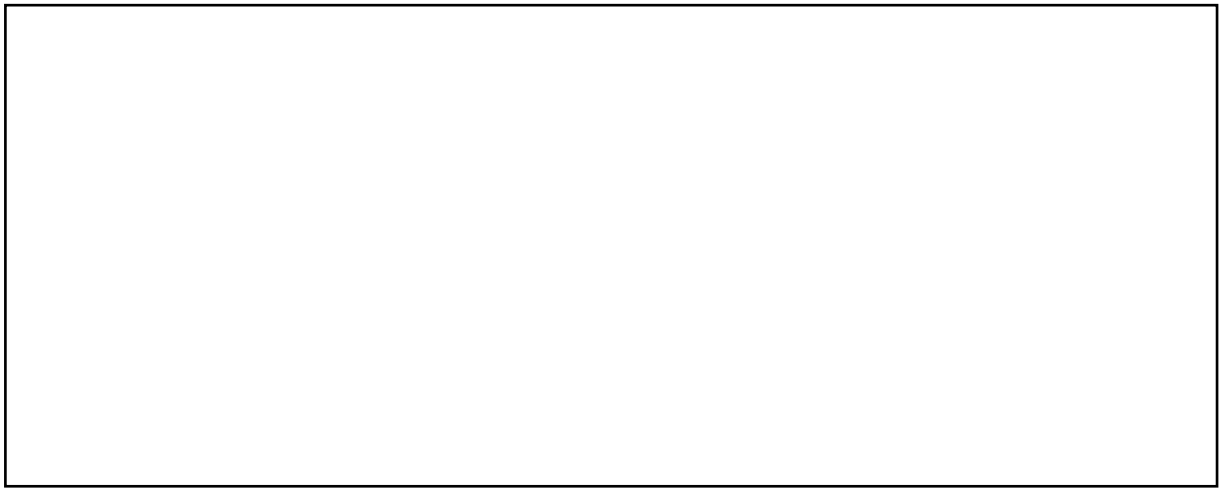
Cuestión 24. Borrar los índices creados en la cuestión 20. Crear un índice multiclave tipo árbol sobre los campos **código_carrera e índice**. Incluir el código SQL utilizado para ello.

Cuestión 25. Para cada una de las consultas que se muestran a continuación, ¿Qué información se puede obtener de los datos monitorizados por la base de datos al realizar la consulta? Comentar cómo se ha realizado la resolución de la consulta. ¿Cuántos bloques se han leído de cada estructura? ¿Por qué? Importante, reinicializar los datos recolectados de la actividad de la base de datos antes de lanzar cada consulta:

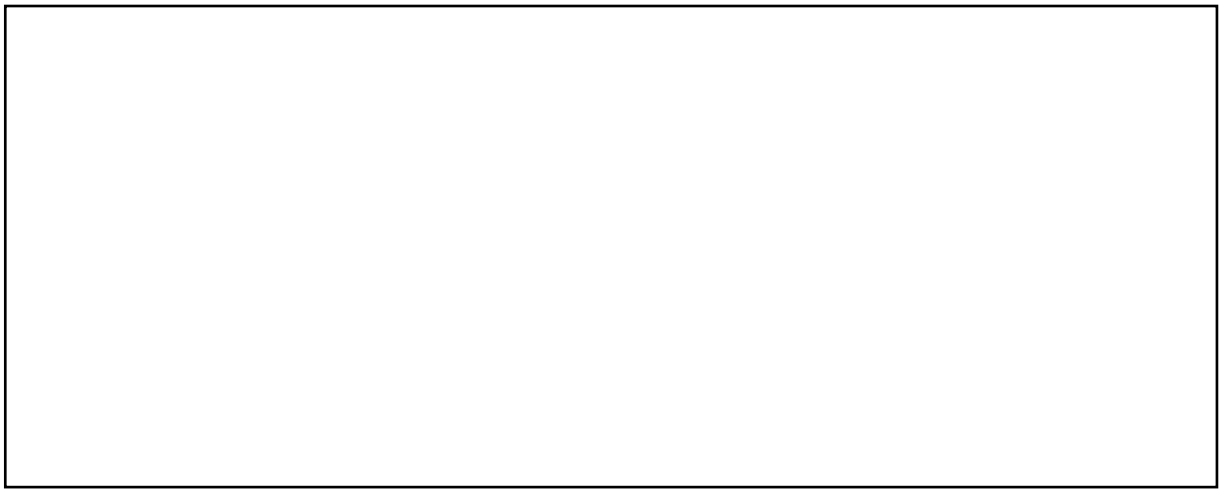
1. Mostrar los estudiantes que tienen un código de carrera de 20 y un índice de 500.

2. Mostrar los estudiantes que tienen un código de carrera de 50 o un índice de 900.

3. Mostrar los estudiantes que tienen un índice de 300.

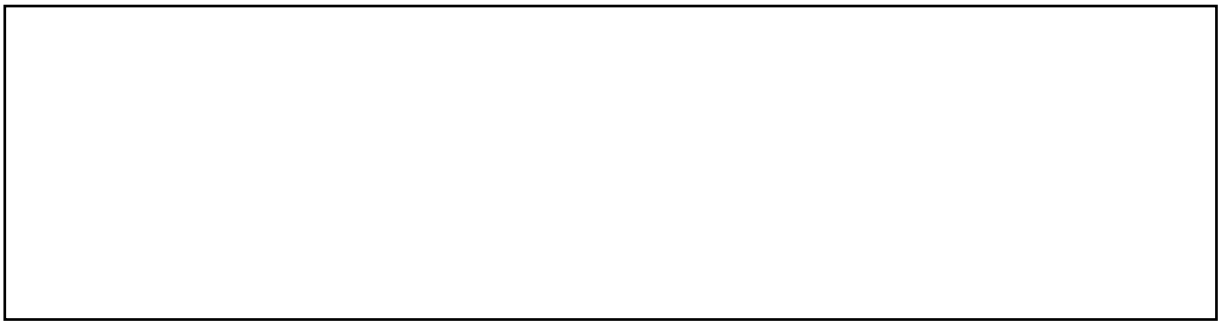


4. Mostrar los estudiantes que tienen un código de carrera de 80.



Cuestión 26. Crear la tabla **estudiantes3** particionada por el campo **edad** de tal manera que los estudiantes que tengan la misma edad estén juntos en la misma partición. Para cada una de las consultas que se muestran a continuación, ¿Qué información se puede obtener de los datos monitorizados por la base de datos al realizar la consulta? Comentar cómo se ha realizado la resolución de la consulta. ¿Cuántos bloques se han leído de cada estructura? ¿Por qué? Comparar con la teoría. Importante, reinicializar los datos recolectados de la actividad de la base de datos antes de lanzar cada consulta.

1. Mostrar el número de estudiantes con una edad de 25.



2. Mostrar los estudiantes que tienen una edad de 20 o 25 o 30 años.

3. Mostrar los estudiantes que tienen una edad entre 25 y 30 años (incluidos).

4. Mostrar los estudiantes que tienen una edad mayor de 50 años.

Cuestión 27. A la vista de los resultados obtenidos de este apartado, comentar las conclusiones que se pueden obtener del acceso de PostgreSQL a los datos almacenados en disco.

Bibliografía (PostgreSQL 18.1)

- Capítulo 1: Getting Started.
- Capítulo 5: 5.6 System Columns.
- Capítulo 5: 5.12 Table Partitioning.
- Capítulo 11: Indexes.
- Capítulo 19: Server Configuration.
- Capítulo 24: Routine Database Maintenance Tasks.
- Capítulo 27: Monitoring Database Activity. The statistics Collector
- Capítulo 27.6: Monitoring Disk Usage. Determining Disk Usage
- Capítulo 52: System Catalogs
- Capítulo 65.1: B-Tree Indexes
- Capítulo 65.6: Hash Indexes
- Capítulo 67: Database Physical Storage
- Capítulo VI.II: PostgresSQL Client Applications.
- Capítulo VI.III: PostgresSQL Server Applications.
- Apéndice F: Additional Supplied Modules. Pageinspect, pgstatutuple
- Apéndice G: Additional Supplied Programs. Oid2name