

**Titulación:** Grado en Ingeniería Informática e Ingeniería en  
Sistemas de Información  
**Curso:** 2021-2022. Convocatoria Ordinaria de Junio  
**Asignatura:** Bases de Datos Avanzadas – Laboratorio

## **Practica 1: Arquitectura PostgreSQL y almacenamiento físico**

**ALUMNO 1:**

**Nombre y Apellidos:** \_\_\_\_\_

**DNI:** \_\_\_\_\_

**ALUMNO 2:**

**Nombre y Apellidos:** \_\_\_\_\_

**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.

### **Plazos**

Trabajo de Laboratorio: Semana 7 Febrero, 14 Febrero, 21 Febrero, 28 Febrero y 7 de Marzo.

Entrega de práctica: Día 14 de Marzo. Aula Virtual

Documento a entregar: Un **fichero con formato ZIP** con las respuestas a las cuestiones planteadas, así como los ficheros de log de postgresql relacionados con la resolución de la práctica y el fichero history del cliente pgcli. El fichero se deberá llamar: **DNIdelosAlumnos\_PL1.zip**

**AMBOS ALUMNOS DEBEN ENTREGAR EL FICHERO EN LA PLATAFORMA.**

## Introducción

En esta primera práctica se introduce el sistema gestor de bases de datos **PostgreSQL versión 14**. 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 0. En esta práctica **no se permite el uso de pgAdmin u otra herramienta administrativa gráfica**. Hay que resolver la práctica con el cliente pgcli de Python.

**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 y el nombre del host con su puerto. ¿Cómo se ha realizado la configuración?

## Organización de Archivos en PostgreSQL

**Cuestión 1.** Crear una nueva Base de Datos que se llame **PL1**. Después crear una tabla **alumno** que tenga un campo denominado carnet de tipo integer, un campo nombre de tipo text, un campo denominado código\_de\_carrera de tipo integer, un campo edad de tipo integer y un campo denominado puntuación de tipo integer. Notar que no hay primary key en la tabla por el momento. Localizar los ficheros relacionados con la tabla. ¿cómo se localizan? ¿Cuánto ocupan y por qué?

**Cuestión 2.** Insertar los datos que se entregan en el fichero de texto denominado **datos\_alumno.txt**. ¿Cuánto ocupa la información original a insertar? ¿Cuánto ocupa la tabla ahora? ¿Por qué? ¿Cuánto tarda en cargar? Calcular teóricamente el tamaño en bloques que ocupa la relación **Alumno** tal y como se realiza en teoría. ¿Concuerda con el tamaño en bloques que nos proporciona PostgreSQL? ¿Por qué?

**Cuestión 3.** ¿Cuál es el factor de bloque medio real de la tabla **Alumno**? Realizar una consulta SQL que obtenga ese valor y comparar con el factor de bloque teórico siguiendo el procedimiento visto en teoría.

**Cuestión 4.** Realizar una consulta que muestre los datos del alumno que tiene el carnet 24567890. ¿Cuántas tuplas se obtienen y cuántos bloques se leen por Postgres? ¿Por qué? Comparar con los resultados obtenidos al aplicar el método visto en teoría.

**Cuestión 5.** Realizar una consulta que muestre el número de alumnos que tienen una puntuación de 525. ¿Cuántas tuplas se obtienen y cuántos bloques se leen por Postgres? ¿Por qué? Comparar con los resultados obtenidos al aplicar el método visto en teoría.

**Cuestión 6.** Crear una tabla **Alumno2** cuyas tuplas estén ordenadas físicamente por el campo puntuación que tenga la misma información que el fichero **datos\_alumno.txt**. Indicar el proceso de generación de dicha tabla ordenada.

Cuestión 7. Repetir las cuestiones 4 y 5 sobre la tabla **Alumno2** y comparar los resultados obtenidos indicando las conclusiones obtenidas.

Cuestión 8. Borrar 2.000.000 de tuplas de la tabla **Alumno** de manera aleatoria usando el valor del campo carnet. ¿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 9.** En la situación anterior, ¿Qué operaciones se pueden aplicar a la base de datos **PL1** para optimizar el rendimiento de esta? Aplicarla a la base de datos **Alumno** de tal manera que se recupere el mayor espacio posible. Comentar cuál es el resultado final y qué es lo que ocurre físicamente.

**Cuestión 10.** Crear una nueva tabla denominada **Alumno3** con los mismos campos que la cuestión 1 y que esté particionada por medio de una función HASH que devuelva 30 valores sobre el campo carnet. Insertar los datos del fichero **datos\_alumno.txt** 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.

Cuestión 11. Repetir las cuestiones 4 y 5 sobre la tabla **Alumno3** y comparar los resultados obtenidos con lo visto anteriormente en las tablas **Alumno** y **Alumno2** 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.

Cuestión 12. Borrar todas las tablas **Alumno** y **Alumno2**. Crear una nueva tabla que se llama **Alumno** con los mismos campos que la cuestión 2 pero el campo carnet será la Primary Key (PK). ¿Qué ocurre físicamente? ¿Qué ficheros se crean y por qué? ¿Por qué tienen ese tamaño?

**Cuestión 13.** Cargar el fichero de datos **datos\_alumno.txt**. ¿Cuánto tarda ahora en cargar? Comparar con la cuestión 2.

**Cuestión 14.** Crear un índice de tipo árbol para el campo carnet. ¿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 15.** Determinar el tamaño de bloques que teóricamente tendría de acuerdo con lo visto en teoría y el número de niveles. Comparar los resultados obtenidos teóricamente con los resultados obtenidos en la cuestión 14.



Cuestión 16. Utilizando el procedimiento comentado en teoría para buscar sobre índices de tipo árbol, determinar el “**número de bloque leído**” de cada estructura hasta encontrar el registro de datos que tiene el carnet 26935691. Mostrar el código SQL utilizado. Se suministra el procedimiento almacenado **hex\_to\_int** que permite convertir el valor hexadecimal del ítem en un valor entero.

Cuestión 17. Crear un índice de tipo árbol para el campo puntuación. ¿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 que teóricamente tendría de acuerdo con lo visto en teoría y el número de niveles. Comparar los resultados obtenidos teóricamente con los resultados obtenidos en la cuestión 17.

**Cuestión 19.** Utilizando el procedimiento comentado en teoría para buscar sobre índices de tipo árbol visto, determinar el número de bloque leído de cada estructura hasta encontrar el registro de datos que tiene una puntuación de 456. Mostrar el código SQL utilizado

**Cuestión 20.** Crear un índice de tipo hash para el campo carnet. ¿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 21.** Determinar el tamaño de bloques que teóricamente tendría de acuerdo con lo visto en teoría. Comparar los resultados obtenidos teóricamente con los resultados obtenidos en la cuestión 20.

**Cuestión 22.** Crear un índice de tipo hash para el campo puntuación. ¿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 23.** Determinar el tamaño de bloques que teóricamente tendría de acuerdo con lo visto en teoría. Comparar los resultados obtenidos teóricamente con los resultados obtenidos en la cuestión 22.

Cuestión 24. ¿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 puede usar para monitorizar los bloques leídos (tipo statio) 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.

Para ello, borrar todas las tablas creadas y volver a crear la tabla **Alumno** como en la cuestión 12. Cargar los datos que se encuentran originalmente en el fichero datos\_empleado.txt

**Cuestión 25.** Crear un índice hash sobre el campo carnet y otro sobre puntuación. También crear un índice primario tipo árbol sobre el campo puntuación. ¿Cuál ha sido el proceso seguido para crear cada tipo de índice? Incluir el código SQL utilizado para ello.

**Cuestión 26.** 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.** Se recuerda que solo se pueden usar vistas sobre las estadísticas de la base de datos.

1. Mostar la información de las tuplas con carnet 23234566.

2. Mostar la información de las tuplas con puntuación de 3550.

3. Contar el número de alumnos que tienen una puntuación de menos de 300.

4. Mostrar los alumnos que tienen en el nombre los caracteres 135678.

5. Actualizar el nombre del empleado que tiene el carnet 23567890

6. Mostrar la información de los alumnos que tienen un carnet igual a 23567890 o 21557891

7. Insertar un nuevo alumno con código de carrera 20.

**Cuestión 27.** Borrar los índices creados sobre la tabla **alumno** y crear un índice multiclave tipo árbol sobre los campos carnet y puntuación. Incluir el código SQL utilizado para ello.



**Cuestión 28.** 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 alumnos cuyo carnet es 13933295 o su nombre es alumno25

2. Mostrar las tuplas cuyo carnet es 19886862 y su puntuación es 50

3. Mostrar las tuplas cuyo carnet es 19886862 o su puntuación es 50

4. Mostrar los alumnos cuya puntuación es 60.

**Cuestión 29.** Crear la tabla **alumno3** como en la cuestión 10 pero sobre el campo código de carrera. 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 las tuplas cuyo código de carrera es mayor que 12
2. Mostrar las tuplas cuyo código de carrera es 6 o 12
3. Mostrar las tuplas cuyo código de carrera es mayor que 10 y menor que 15.

**Cuestión 30.** 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 14)**

- Capítulo 1: Getting Started.
- Capítulo 5: 5.5 System Columns.
- Capítulo 5: 5.11 Table Partitioning.
- Capítulo 11: Indexes.
- Capítulo 20: Server Configuration.
- Capítulo 25: Routine Database Maintenance Tasks.
- Capítulo 28: Monitoring Database Activity. The statistics Collector
- Capítulo 29: Monitoring Disk Usage. Determining Disk Usage
- Capítulo VI.II: PostgreSQL Client Applications.
- Capítulo VI.III: PostgreSQL Server Applications.
- Capítulo 52: System Catalogs.
- Capítulo 64: B-Tree Indexes.
- Capítulo 70: Database Physical Storage.
- Apéndice F: Additional Supplied Modules. Pageinspect, pgstatutuple
- Apéndice G: Additional Supplied Programs. Oid2name