

**TRABAJO PRÁCTICO INTEGRADOR**

**Asignatura:** Base de Datos Aplicadas.

**Tema:** Gestión de Proyectos Informáticos.

**Grupo Nº:** 2 (Dos)

**Integrantes:** Imfeld, Facundo Nicolas

Lopez, William Juan Jose

Nasir, Khalil Abdul

Ojeda Nuñez, Pablo Nicolas

Troncoso, Clarise

**Profesores:** Ing. Fantin, Andres Pablo

Ing. Fernandez, Juan Carlos

**Año:** 2020

**Primer etapa: Tareas sobre BT1**

**Consignas generales para la primer etapa  
Actividades a desarrollar en la Primer Etapa**

Administración del gestor de bases de datos (SGBD): Realizar una instalación nueva del SGBD sobre una máquina virtual destinada específicamente para tal fin, diferente a la utilizada para los trabajos de clases (por ej. se puede comenzar con una mv desde cero, o a partir de una instantánea o una clonación previa al desarrollo de las guías de trabajos prácticos).

**Responder a las siguientes consignas**:

**Responder los siguiente, relacionado al SGBD implementado:**

* + Indicar cantidad de memoria RAM mínima y recomendada.
  + Ídem para el espacio en disco.
  + ¿Puede su SGBD instalarse en cualquier SO, sin limitación de arquitectura, del lenguaje o la localización del mismo?
  + Indicar cuáles son las alternativas para la instalación.
  + ¿Tiene soporte para discos sin formato (dispositivos en bruto o raw)? ¿da soporte parcial a esta característica? Enumerar las restricciones y ventajas expuestas por el fabricante si se da soporte a ésto.
  + Cuando la base de datos está vacía ¿Cuánto mide el espacio de tablas?
  + Cuando la base de datos tiene datos equivalentes a 1MB ¿Cuánto mide el espacio de tablas? ¿Por qué?
  + Indicar la estructura de carpetas de instalación de los programas y ejecutables y archivos de configuración propios del SGBD.
  + Ídem anterior para los datos de tablas, y distintos registros de logs, ubicación predeterminada.
  + Describir brevemente la aplicación utilizada para administrar el SGBD (si está basada en Java, si es una aplicación web, nativa, etc.).
  + Ingresar a la aplicación y describir las opciones que se observan. Investigar con la ayuda del motor: ¿Qué es una tabla?, ¿qué es un índice? ¿qué es una vista? ¿qué es un trigger?
  + Enumerar las capacidades de su SGBD:
  + Tamaño máximo de espacio de tablas.
  + Cantidad máxima de tablas, índices, stored procedures, vistas.
  + Cantidad máxima de columnas por tabla.
  + Longitud máxima de fila.
  + Tamaños máximos de los tipos de datos que soporta.
  + Explicar si el sistema operativo tiene incidencia sobre los tamaños máximos permitidos de espacios de tabla, catálogo u objetos grandes o sobre los nombres de los mismos.
  + Enumerar los procesos del motor ejecutándose en el sistema operativo, sus nombres de imágenes, tamaño en memoria y funciones de cada uno. Indicar a su parecer los dos más importantes.
  + Indicar cuántos nodos se muestran en el programa administrador del SGBD, y la función de cada uno. No incluya nodos relacionados con los datos de bases de datos de usuario, sólo los nodos que corresponden a funcionalidad predeterminada.
  + ¿Cuántas bases de datos de sistema tiene su SGBD? Enumerar y explicar sucintamente cuál es la función de cada una.
  + Indicar cuántos tipos de objetos distintos puede contener una base de datos típica en su gestor SGBD (tabla, índice, etc.).
  + Realice una síntesis sobre principales diferencias, ventajas y desventajas frente a otros SGBDs (elija al menos dos, puede tomar como referencia <https://db-engines.com/en/>).

**Realizar las siguientes tareas de administración:**

* + Agregar los usuarios “ADMIN” y “OPERADOR”.
  + Permitir que ADMIN sea administrador de sistema. A partir de éste momento todas las tareas administrativas deberán realizarse con esta cuenta (no con *root*).
  + Crear una base de datos para desarrollar el trabajo práctico integrador. Como único requisito debe llamarse BDA-TPI. Escoger el resto de los parámetros usando su criterio, documentar cada paso con una copia de pantalla del asistente de creación o el comando utilizado.
  + Asignar los privilegios necesarios al usuario OPERADOR para que tenga acceso a todas las tareas de administración sobre la base de datos *BDA-TPI* excepto la gestión de usuarios y permisos.

**Análisis del escenario y modelado:**

El escenario sobre el que cada grupo deberá trabajar está basado en una organización cuya descripción se presenta como ANEXO II de esta guía. A efectos de simplificar y unificar criterios respecto al diseño de la base de datos resultante, se presenta como ANEXO III una propuesta de solución para el modelado con un diagrama entidad-relación (DER). Se adjunta también un archivo con la "Notación usada en los DERs de las soluciones propuestas".

**Para el desarrollo del trabajo práctico deberán ejecutarse las siguientes consignas:**

* + Leer el escenario en general en forma individual.
  + Analizar el escenario en grupo, y el DER resultante. Deberán registrarse los ítems que no resulten claros o que se considere que no fueron capturados por el DER con respecto a los requerimientos planteados en el escenario y que no se hayan contemplado en supuestos adicionales.

**Esquema relacional:**

Tomando como entrada el DER y las consideraciones hechas al mismo, se deberá realizar el esquema de la base de datos en el modelo relacional. Este esquema puede ser gráfico o bien un esquema relacional textual.

* + Convertir las entidades y relaciones del DER a esquemas de relación, según la metodología practicada en clases.
  + Tener en cuenta:
    - Nombres dados a las tablas/relaciones.
    - Dominio de los atributos.
    - Atributos de claves primarias y claves foráneas.
    - Los distintos tipos de restricciones (dominio, integridad, participación, etc.).
  + El resultado de esta actividad es un archivo que será parte de la documentación del TPI.

**Esquema físico:**

A partir de la actividad anterior, obtener el script sql para implementar las tablas y relaciones resultantes del modelo relacional en el SGBD instalado.

* + El resultado de esta etapa debe ser un script sql que deberá ejecutarse en el SGBD.

***Recomendaciones sobre los esquemas***

* + Podrá utilizarse en estas instancias algún software o herramienta CASE para facilitar el modelado (p.e. MySQL Workbench o ER-Studio).
  + En el esquema físico tener en cuenta los nombres de tablas y campos, por ejemplo, el domicilio de un cliente no conviene llamarlo “Domicilio Cliente” sino “DomCli” (o similar). También prestar especial atención a los tipos de datos y sus características, justificando adecuadamente cada elección.

**Base de Datos:**

Una vez creada la base de datos con todos los objetos planteados en el esquema físico, realizar las siguientes consignas:

* + Cargar datos en las tablas para que todas tengan al menos una fila. Al menos dos tablas deben tener más de 10 filas.
  + Utilizando alguna herramienta para generación de datos, importar masivamente filas a las tablas indicadas. Tener en cuenta que se intenta trabajar con volúmenes importantes similares a un sistema real, por lo que deberá asegurarse que al menos una tabla cuente con más de 100000 registros.
  + Una vez cargadas las tablas con datos, realizar, a criterio del grupo, 5 (cinco) consultas sql distintas para borrado de filas, y otras 5 para modificación de datos. Las consultas pueden ser ejecutadas sobre una misma tabla o distintas.

**Consultas SQL:**

Deberán resolverse los requerimientos sobre el escenario, entregados como ANEXO IV, mediante consultas efectuadas en el lenguaje SQL.

* + Previo a la ejecución de las consultas deberá cargarse la base de datos con cierta cantidad de datos que aseguren que éstas tengan un resultado visible.

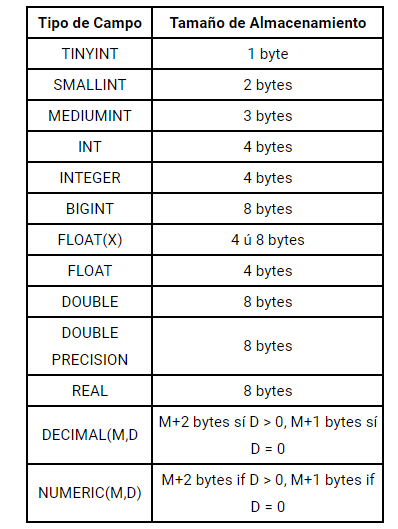
----------------------------------------------------------------------------------------------------------------

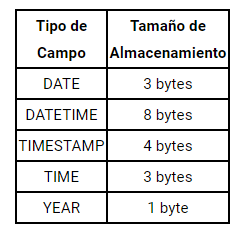
**Preguntas relacionadas al SGBD implementado:**

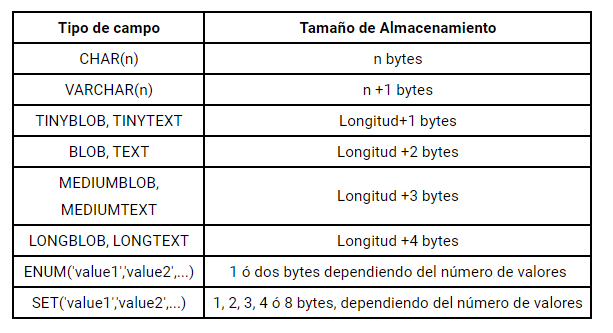
* + Indicar cantidad de memoria RAM mínima y recomendada
    - RAM mínima y recomendada: 512Mb en ordenadores y 1024Mb en Máquinas Virtuales.
  + Ídem para el espacio en disco.
    - Espacio mínimo en disco duro: 1GB.
  + ¿Puede su SGBD instalarse en cualquier SO, sin limitación de arquitectura, del lenguaje o la localización del mismo?
    - Si, MySql es compatible tanto con Windows, Linux y iOS, para arquitecturas de 32 y 64 bits.
  + Indicar cuáles son las alternativas para la instalación.
    - En Windows simplemente debemos descargar el paquete de instalación y ejecutarlo, y seguir los pasos de instalación. En Linux sin embargo podemos instalar a través de los repositorios oficiales o bajando el paquete de instalación.
  + ¿Tiene soporte para discos sin formato (dispositivos en bruto o raw)? ¿da soporte parcial a esta característica? Enumerar las restricciones y ventajas expuestas por el fabricante si se da soporte a esto.
    - Se pueden usar particiones de dispositivos en bruto como ficheros de datos del espacio de tablas. Utilizando un dispositivo en bruto, se puede llevar a cabo operaciones de E/S en windows y algunas versiones de Unix sin que se utilicen el bufer y sin la sobrecarga producida por el sistema de ficheros, lo cual incrementa el rendimiento. Cuando se crea un nuevo fichero de datos se debe colocar la palabra clave newraw inmediatamente a continuación del tamaño del fichero. La partición deberá ser al menos tan grande como el tamaño que se haya especificado.
  + Cuando la base de datos está vacía ¿Cuánto mide el espacio de tablas?
    - Cuando la base de datos está vacía el espacio de tabla es de cero bytes.
  + Cuando la base de datos tiene datos equivalentes a 1MB ¿Cuánto mide el espacio de tablas? ¿Por qué?
    - Cuando el volumen de datos es equivalente a 1MB el espacio de tabla es de 1.048.576bytes. Esto se debe a que 1MB en mysql significa 1024\*1024 bytes. En tanto en las especificaciones de disco generalmente significa 1.000.000bytes.
  + [Indicar la estructura de carpetas de instalación de los programas y ejecutables y archivos de configuración propios del SGBD.](https://wiki.cifprodolfoucha.es/index.php?title=Mysql_Instalaci%C3%B3n)
    - Toda la información la tenemos en el archivo **/etc/mysql/mysql.conf.d/mysqld.cnf**
    - **/etc/mysql** tenemos todos los archivos de configuración globales de la instancia del motor.
    - **/var/run/mysqld** está el servicio del motor de datos.
  + Ídem anterior para los datos de tablas, y distintos registros de logs, ubicación predeterminada.
    - **/var/lib/mysql** tenemos el datadir. Lugar en donde se almacenan los datos físicos, bases de datos, etc.
    - **/var/log/mysql** tenemos todos los archivos de logs de la instancia.
  + Describir brevemente la aplicación utilizada para administrar el SGBD (si está basada en Java, si es una aplicación web, nativa, etc.).
    - MySQL Workbench es una herramienta visual de diseño de bases de datos que integra desarrollo de software, administración de bases de datos, diseño de bases de datos, creación y mantenimiento para el sistema de base de datos MySQL. MySQL Workbench permite diseñar visualmente, modelar, generar y administrar bases de datos.
  + Ingresar a la aplicación y describir las opciones que se observan. Investigar con la ayuda del motor: ¿Qué es una tabla?, ¿qué es un índice? ¿Qué es una vista? ¿Qué es un trigger?
    - Al abrir MySQL Workbench podemos ver un asistente para crear una nueva conexión a nuestro servidor, podemos configurar esta a medida, una vez completamos la conexión podemos ver todas las bases de datos que tenemos en la columna derecha así como también sus tablas, triggers, funciones y SP. Podemos un archivo sql o bien generar uno nuevo, esto a simple vista, también cabe mencionar que incluye todo lo que necesita un modelador de datos para crear modelos complejos de ER, ingeniería directa e inversa y también ofrece características clave para realizar tareas difíciles de gestión de cambios y documentación que normalmente requieren mucho tiempo y esfuerzo.
      1. Tabla: en las bases de datos, se refiere al tipo de modelado de datos, donde se guardan los datos recogidos por un programa. Su estructura general se asemeja a la vista general de un programa de hoja de cálculo. Una tabla es utilizada para organizar y presentar información. Las tablas se componen de filas y columnas.
      2. Índice: es un puntero a una fila de una determinada tabla de nuestra base de datos. Los índices mejoran el tiempo de recuperación de los datos en las consultas realizadas contra nuestra base de datos.
      3. Vistas: son tablas virtuales. Es decir, tablas que no guardan ningún dato propiamente dentro de ellas. Solo muestran los datos que están almacenados en otras tablas. Siendo así, crear vistas en MySQL significa mostrar información de una fuente de origen sin necesidad de mostrar ni exponer a la fuente en sí.
      4. Trigger: son una serie de reglas predefinidas que están asociadas a una tabla. Estas reglas permiten la ejecución de una serie de instrucciones cuando se producen ciertos eventos como pueden ser la inserción de un nuevo registro, la actualización o el borrado de los datos de una tabla.
  + Enumerar las capacidades de su SGBD:
  + Tamaño máximo de espacio de tablas.

El tamaño máximo de un tablespace depende del tamaño de la InnoDb page. A continuación se deja una tabla con los valores máximos según el Innodb page size:

|  |  |
| --- | --- |
| **InnoDB Page Size** | **Maximum Tablespace Size** |
| 4KB | 16TB |
| 8KB | 32TB |
| 16KB | 64TB |
| 32KB | 128TB |
| 64KB | 256TB |

* Cantidad máxima de tablas, índices, stored procedures, vistas.
  + - MySQL no tiene límite en el número de tablas. El sistema de archivos subyacente puede tener un límite en la cantidad de archivos que representan tablas. Los motores de almacenamiento individuales pueden imponer restricciones específicas del motor. InnoDB permite hasta 4 mil millones de tablas. Por otra parte, también se puede decir que tanto tablas como índices, stored procedures y vistas ocupan espacios de tablas, por lo que se podría decir que puede haber tanta cantidad como el tamaño máximo de espacio de tabla lo permita.
  + Cantidad máxima de columnas por tabla.
    - MySQL tiene un límite estricto de 4096 columnas por tabla, pero el máximo efectivo puede ser menor para una tabla determinada.
  + Longitud máxima de fila.
    - La representación interna de una tabla MySQL tiene un límite de tamaño de fila máximo de 65.535 bytes, incluso si el motor de almacenamiento es capaz de admitir filas más grandes.
  + Tamaños máximos de los tipos de datos que soporta.
    - Tipos Numéricos
    - Tipos Fechas



* + - Tipos de Cadenas
  + Explicar si el sistema operativo tiene incidencia sobre los tamaños máximos permitidos de espacios de tabla, catálogo u objetos grandes o sobre los nombres de los mismos.
    - El sistema operativo incide directamente sobre los tamaños máximos permitidos, ya que la instancia utiliza el espacio de almacenamiento físico del sistema operativo en donde está alojado.
  + Enumerar los procesos del motor ejecutándose en el sistema operativo, sus nombres de imágenes, tamaño en memoria y funciones de cada uno. Indicar a su parecer los dos más importantes.
    - Los procesos del motor que se ejecutan en el sistema operativo (CPU) son 3 en total. El proceso de DDL, DML y el optimizador y el proceso de lenguaje de consultas para las peticiones DML no planeadas. A nuestro parecer las dos más importantes son las del procesador de DDL y DML.
  + Indicar cuántos nodos se muestran en el programa administrador del SGBD, y la función de cada uno. No incluya nodos relacionados con los datos de bases de datos de usuario, sólo los nodos que corresponden a funcionalidad predeterminada.
    - El nodo que viene por default en MySql Workbench es el de sys que contiene toda la información del sistema del gestor que a su vez tiene nodos de tablas, vistas, procedimientos almacenados y funciones del gestor de base de datos que se ejecutan cuando se requieran.
  + ¿Cuántas bases de datos de sistema tiene su SGBD? Enumerar y explicar sucintamente cuál es la función de cada una.
    - El SGBD tiene 4 bases de datos de sistema.
      1. [information\_schema.](https://dev.mysql.com/doc/refman/8.0/en/information-schema-introduction.html) Almacena información sobre los metadatos de la base de datos. Información sobre el servidor mysql como puede ser las bases de datos, tablas, tipos de datos de una columna, privilegios, etc.
      2. mysql contiene información de la instancia del motor, como por ejemplo, plugins instalados, usuarios y sus datos, logs, etc.
      3. [perfomance\_schema](https://dev.mysql.com/doc/refman/8.0/en/performance-schema.html) almacena información que permite al DBA monitorear la ejecución del servicio de mysql a un low level.
      4. [sys](https://dev.mysql.com/doc/refman/8.0/en/sys-schema.html). Contiene un conjunto de objeto que permite interpretar la información almacenada por Perfomance schema
  + Indicar cuántos tipos de objetos distintos puede contener una base de datos típica en su gestor SGBD (tabla, índice, etc.).
    - Los tipos de objetos típicos de una base de datos son, tablas, índices, triggers, registros, procedimiento de almacenado, columnas.
  + Realice una síntesis sobre principales diferencias, ventajas y desventajas frente a otros SGBDs (elija al menos dos, puede tomar como referencia <https://db-engines.com/en/>).
    - **MySql frente a Oracle**
      1. Estructuras de las Base de Datos
         * **MySQL:** MySQL es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS). Al igual que otras bases de datos relacionales, MySQL usa tablas, restricciones, disparadores, roles, procedimientos almacenados y vistas como los componentes centrales con los que trabaja. Una tabla consta de filas y cada fila contiene datos para cada columna. MySQL usa claves primarias para identificar de forma única cada fila (también conocido como registro ) en una tabla, y claves externas para asegurar la integridad referencial entre dos tablas relacionadas.
         * **Oracle:** Oracle es una base de datos de varios modelos con un único back-end integrado. Esto significa que puede admitir varios modelos de datos, como documentos, gráficos, relacionales y valores clave dentro de la base de datos.
      2. Índices
         * **MySQL:** la mayoría de los índices MySQL (PRIMARY KEY, UNIQUE, INDEX y FULLTEXT) se almacenan en árboles B. Las excepciones incluyen índices sobre tipos de datos espaciales que utilizan árboles R. MySQL también admite índices hash y el motor InnoDB usa listas invertidas para índices FULLTEXT.
         * **Oracle:** Oracle admite la creación de un índice en:

Una o más columnas de una tabla, una tabla particionada, una tabla organizada por índices o un clúster.

Uno o más atributos de objeto de tipo escalar de una tabla o un clúster.

Una tabla de almacenamiento de tabla anidada para indexar una columna de tabla anidada.

* + - 1. ¿Qué tipos de replicación / agrupación están disponibles?
         * **MySQL:** la replicación en MySQL es una replicación asíncrona unidireccional en la que un servidor actúa como maestro y otros como esclavos. Puede replicar todas las bases de datos, bases de datos seleccionadas o incluso tablas seleccionadas dentro de una base de datos.
         * **Oracle:** Oracle Streams es una función incorporada de la base de datos Oracle que permite la replicación e integración de datos. Su infraestructura flexible satisface una amplia variedad de necesidades de intercambio de información. Oracle Streams permite la propagación de datos, transacciones y eventos en un flujo de datos dentro de una base de datos o de una base de datos a otra.

* + - 1. ¿Dónde se implementan estas bases de datos?
         * **MySQL:** MySQL fue escrito en C y C ++. Tiene soporte para los siguientes lenguajes de programación: C, C ++, Delphi, Perl, Java, Lua, .NET, Node.js, Python, PHP, Lisp, Go, R, D, Erlang.
         * **Oracle:** Oracle fue escrito en lenguaje ensamblador, C y C ++. Tiene soporte para los siguientes lenguajes de programación: Java, .NET, C, C ++, Node.js, Python, PHP, Go, R, Ruby, Ruby on Rails, Perl, Erlang, Rust, COBOL, FORTRAN.

Ambas bases de datos están disponibles en varios sistemas operativos, pero Oracle domina el mundo de las bases de datos en parte porque se ejecuta en más de 60 plataformas, desde mainframes hasta Apple Macs.

* + - **MySql frente a PostgreSQL**
      1. Estructuras de las Base de Datos
         * **MySQL:** MySQL es un sistema de gestión de bases de datos relacionales de código abierto (RDBMS). Al igual que PostgreSQL, y todas las demás bases de datos relacionales para el caso, MySQL usa tablas como componente central y tiene más-menos el mismo conjunto de características que PostgreSQL.
         * **PostgreSQL:** PostgreSQL es un sistema de administración de bases de datos relacionales de objetos (ORDBMS) con énfasis en la extensibilidad y el cumplimiento de estándares. PostgreSQL es compatible con ACID, transaccional, tiene vistas, disparadores y claves externas actualizables y materializadas. También admite funciones y procedimientos almacenados. PostgreSQL usa tablas, restricciones, disparadores, roles, procedimientos almacenados y vistas como los componentes centrales con los que trabaja. Una tabla consta de filas y cada fila contiene el mismo conjunto de columnas. PostgreSQL usa claves primarias para identificar de forma única cada fila (también conocido como registro ) en una tabla, y claves externas para asegurar la integridad referencial entre dos tablas relacionadas.
      2. Índices
         * **PostgreSQL:** PostgreSQL incluye soporte integrado para índices hash y de árbol B regulares. Los índices en PostgreSQL también admiten las siguientes funciones:

**Índices de expresión:** se pueden crear con un índice del resultado de una expresión o función, en lugar de simplemente el valor de una columna.

**Índices parciales:**  indexa solo una parte de una tabla.

* + - * + **MySQL:** la mayoría de los índices MySQL (PRIMARY KEY, UNIQUE, INDEX y FULLTEXT) se almacenan en árboles B. Las excepciones incluyen los índices sobre tipos de datos espaciales que utilizan árboles R. MySQL también admite índices hash y el motor InnoDB usa listas invertidas para índices FULLTEXT.
      1. ¿Qué tipos de replicación / agrupación están disponibles?
         * **PostgreSQL:** PostgreSQL tiene replicación síncrona (llamada replicación 2-safe ), que utiliza dos instancias de base de datos que se ejecutan simultáneamente donde su base de datos maestra está sincronizada con una base de datos esclava. A menos que ambas bases de datos se bloqueen simultáneamente, los datos no se perderán. Con la replicación síncrona, cada escritura espera hasta que se recibe la confirmación tanto del maestro como del esclavo.
         * **MySQL:** la replicación de MySQL es una replicación asíncrona unidireccional en la que un servidor actúa como maestro y otros como esclavos. Puede replicar todas las bases de datos, bases de datos seleccionadas o incluso tablas seleccionadas dentro de una base de datos.
      2. ¿Dónde se implementan estas bases de datos?
         * **PostgreSQL:** PostgreSQL fue escrito en C. Tiene soporte para los siguientes lenguajes de programación: C ++, Delphi, Perl, Java, Lua, .NET, Node.js, Python, PHP, Lisp, Go, R, D, Erlang. Las instalaciones típicas se encuentran en varios servidores Linux, basados ​​en la nube o en las instalaciones. Los principales proveedores de nube tienen soporte PostgreSQL en sus plataformas.
         * **MySQL:** MySQL fue escrito en C y C ++. Tiene soporte para los siguientes lenguajes de programación: C, C ++, Delphi, Perl, Java, Lua, .NET, Node.js, Python, PHP, Lisp, Go, R, D, Erlang. Nuevamente, la implementación basada en la nube y el soporte de plataforma están disponibles. Los clientes pueden optar por utilizar Oracle Corporation o la compilación de código abierto.

**Tareas de Administración**

* Crear los usuarios ADMIN y OPERADOR

create user “ADMIN”@”%” IDENTIFIED BY “Admin” PASSWORD EXPIRE;

* Después de iniciar por primera vez

SET PASSWORD = ‘Admin2020’;

create user ‘OPERADOR’@’%’ IDENTIFIED BY ‘Operador’ PASSWORD EXPIRE;

* Después de iniciar por primera vez

SET PASSWORD = ‘Operador2020’;

* Permitir que ADMIN sea administrador del sistema. A partir de este momento, todas las tareas se tienen que realizar con esta cuenta

GRANT ALL PRIVILEGES ON \*.\* TO ‘ADMIN’@’%’ WITH GRANT OPTION;

* Crear una base de datos para desarrollar el trabajo práctico integrador. Como único requisito debe llamarse BDA-TPI. Escoger el resto de los parámetros usando su criterio, documentar cada paso con una copia de pantalla del asistente de creación o el comando utilizado.

CREATE DATABASE BDA-TPI;

* Asignar los privilegios necesarios al usuario OPERADOR para que tenga acceso a todas las tareas de administración sobre la base de datos *BDA-TPI* excepto la gestión de usuarios y permisos.

GRANT ALL PRIVILEGES ON \*.\* TO ‘OPERADOR’@’%’

**Consultas de Actualización**

-- Consulta 1

UPDATE empleados

SET titulacion= ‘Universitario’

WHERE dni= 35082214;

-- Consulta 2

UPDATE productos

SET tipo= ‘Software’

WHERE nombre = ‘QC Report’;

-- Consulta 3

UPDATE productos

SET nombre = ‘E.R.S.’

WHERE tipo= ‘Informe Tecnico’;

-- Consulta 4

UPDATE productos

SET descripcion= 'Modulo de Facturacion’'

WHERE nombre = ‘Gestor de ventas’;

-- Consulta 5

UPDATE lenguajes

SET nombre = 'Python'

WHERE cod\_lenguaje = 1;

**Consultas de Eliminado**

-- Consulta 1

DELETE

FROM empleados

WHERE tipo = NULL;

-- Consulta 2

DELETE

FROM productos

WHERE estado = 'No';

-- Consulta 3

DELETE

FROM productos

WHERE nombre = 'Gestor de ventas V1';

-- Consulta 4

DELETE

FROM empleados

WHERE titulacion = 'Primario';

-- Consulta 5

DELETE

FROM lenguajes

WHERE nombre = 'Cobol';

**/\* 1.1. Nombre de los proyectos que produjeron al menos 10 productos distintos \*/**

WITH productos\_por\_proyectos AS(

SELECT DISTINCT f.cod\_proyecto, COUNT(p.cod\_producto) AS cantidad

FROM fases f INNER JOIN genera g INNER JOIN productos p

ON f.cod\_proyecto = g.cod\_proyecto AND g.cod\_producto = p.cod\_producto

GROUP BY f.cod\_proyecto

HAVING cantidad >= 10

)

SELECT cod\_proyecto

FROM productos\_por\_proyectos;



**/\* 1.2. Listar los pares (Producto 1, Producto 2) tales que Producto 1 fue producida por una fase que también produjo Producto 2 \*/**

WITH id\_productos AS (

SELECT g.cod\_producto, f.cod\_fase

FROM fases f INNER JOIN genera g

ON f.cod\_proyecto = g.cod\_proyecto

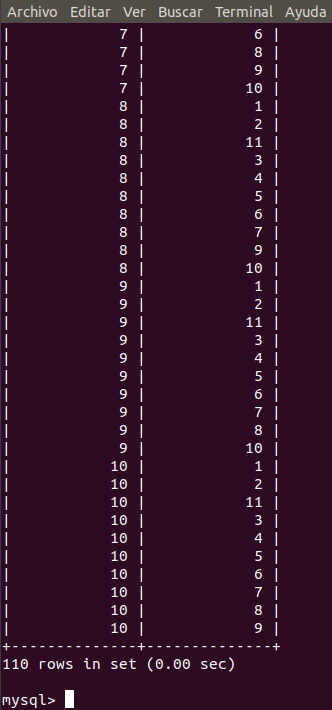
)

SELECT DISTINCT p1.cod\_producto, p2.cod\_producto

FROM id\_productos p1 INNER JOIN id\_productos p2

ON p1.cod\_producto != p2.cod\_producto

WHERE p1.cod\_fase = p2.cod\_fase;



**/\* 1.3 Listar los nombres de los empleados involucrados en la fase que se haya asignado ningún recurso \*/**

Por interpretación del Modelo Entidad Relación y analizando las cardinalidades entre las entidades, concluimos que no se puede realizar esta consulta dado que una fase siempre va a tener asignada al menos un recurso.

;

**/\*1.4 Listar los nombres de los proyectos privadas que no lanzaron ningún producto del tipo prototipo \*/**

SELECT DISTINCT P.nombre FROM proyectos P

INNER JOIN fases F ON P.cod\_proyecto = F.cod\_proyecto

INNER JOIN genera G ON F.cod\_fase = G.cod\_fase

WHERE G.cod\_producto NOT IN (

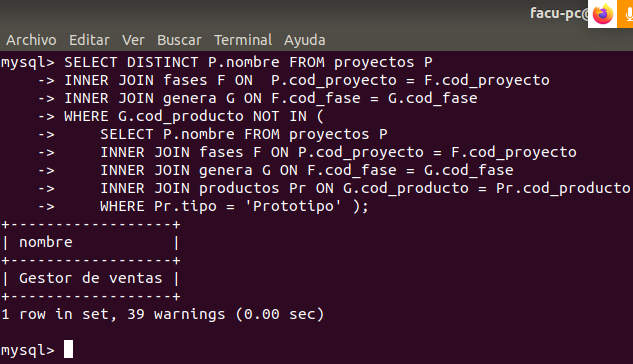
SELECT P.nombre FROM proyectos P

INNER JOIN fases F ON P.cod\_proyecto = F.cod\_proyecto

INNER JOIN genera G ON F.cod\_fase = G.cod\_fase

INNER JOIN productos Pr ON G.cod\_producto = Pr.cod\_producto

WHERE Pr.tipo = 'Prototipo' );



**/\*1.5. Listar los empleados (Informáticos) que hayan trabajado en todos los proyectos.**

**\*/**

SELECT e.cod\_empleado

FROM empleados e, proyectos p, gastos g, informaticos i

WHERE e.cod\_gasto = g.cod\_gasto AND e.cod\_gasto = p.cod\_gasto AND g.cod\_gasto = p.cod\_gasto AND i.cod\_e = e.cod\_empleado

GROUP BY e.cod\_empleado

HAVING count(\*) = (SELECT count(\*) FROM proyectos);

**2. Consultas con modificación de ER**

**La empresa necesita identificar recursos dados de baja y el costo y fecha de adquisición de estos reflejando que un recurso puede ser a su vez parte de otro y que una parte puede ser también parte de varias al mismo tiempo.**

**Inserte datos para reflejar la situación siguiente y luego resuelva.**

**#1 esta formado por 2,3,4 y 16**

**#5 esta formado por 6 y 7**

**#6 esta formado por 8**

**#7 esta formado por 15**

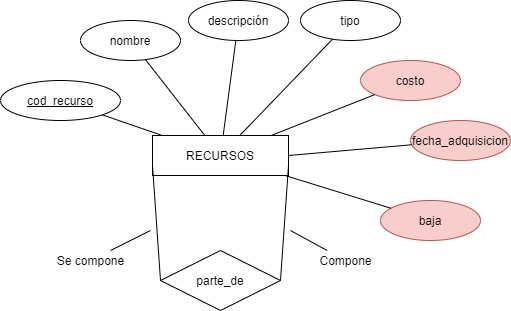
**#8 esta formado por 14**

**#9 esta formado por 10 y 11**

**#12 esta formado por 11 y 13**

**#13 esta formado por 11**

**Se propone el siguiente cambio para reflejar lo que se solicita en el item anterior**

****

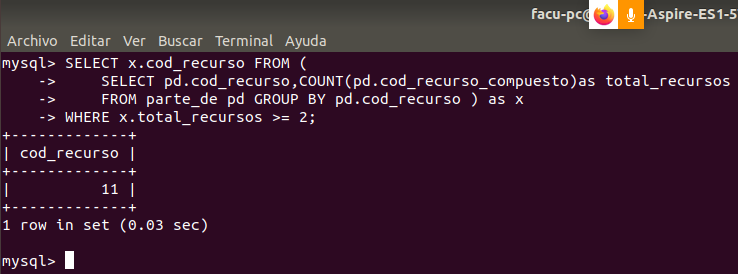
**--2.1. Nombre de los recursos que forman parte de al menos dos recursos distintos.**

SELECT x.cod\_recurso FROM (

SELECT pd.cod\_recurso,COUNT(pd.cod\_recurso\_compuesto)as total\_recursos

FROM parte\_de pd GROUP BY pd.cod\_recurso ) as x

WHERE x.total\_recursos >= 2;



**--2.2. ídem 2.1 pero listando de que recursos forman parte**

WITH recursos\_R1 AS (

SELECT x.cod\_recurso FROM (

SELECT pd.cod\_recurso,COUNT(pd.cod\_recurso\_compuesto)as total\_recursos

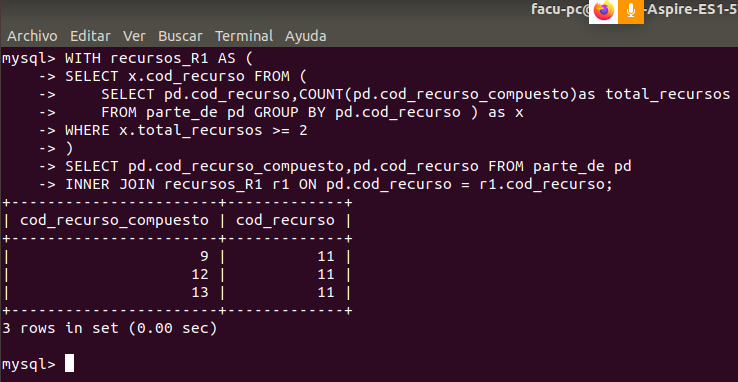
FROM parte\_de pd GROUP BY pd.cod\_recurso ) as x

WHERE x.total\_recursos >= 2

)

SELECT pd.cod\_recurso\_compuesto,pd.cod\_recurso FROM parte\_de pd

INNER JOIN recursos\_R1 r1 ON pd.cod\_recurso = r1.cod\_recurso;



**--2.5. Listar los recursos que no forman parte de ningun otro recurso (forma directa)**

SELECT DISTINCT pd.cod\_recurso\_compuesto FROM parte\_de pd

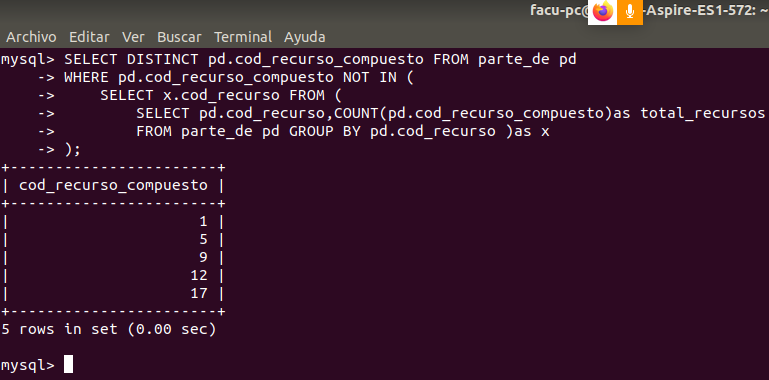
WHERE pd.cod\_recurso\_compuesto NOT IN (

SELECT x.cod\_recurso FROM (

SELECT pd.cod\_recurso,COUNT(pd.cod\_recurso\_compuesto)as total\_recursos

FROM parte\_de pd GROUP BY pd.cod\_recurso )as x

);

****

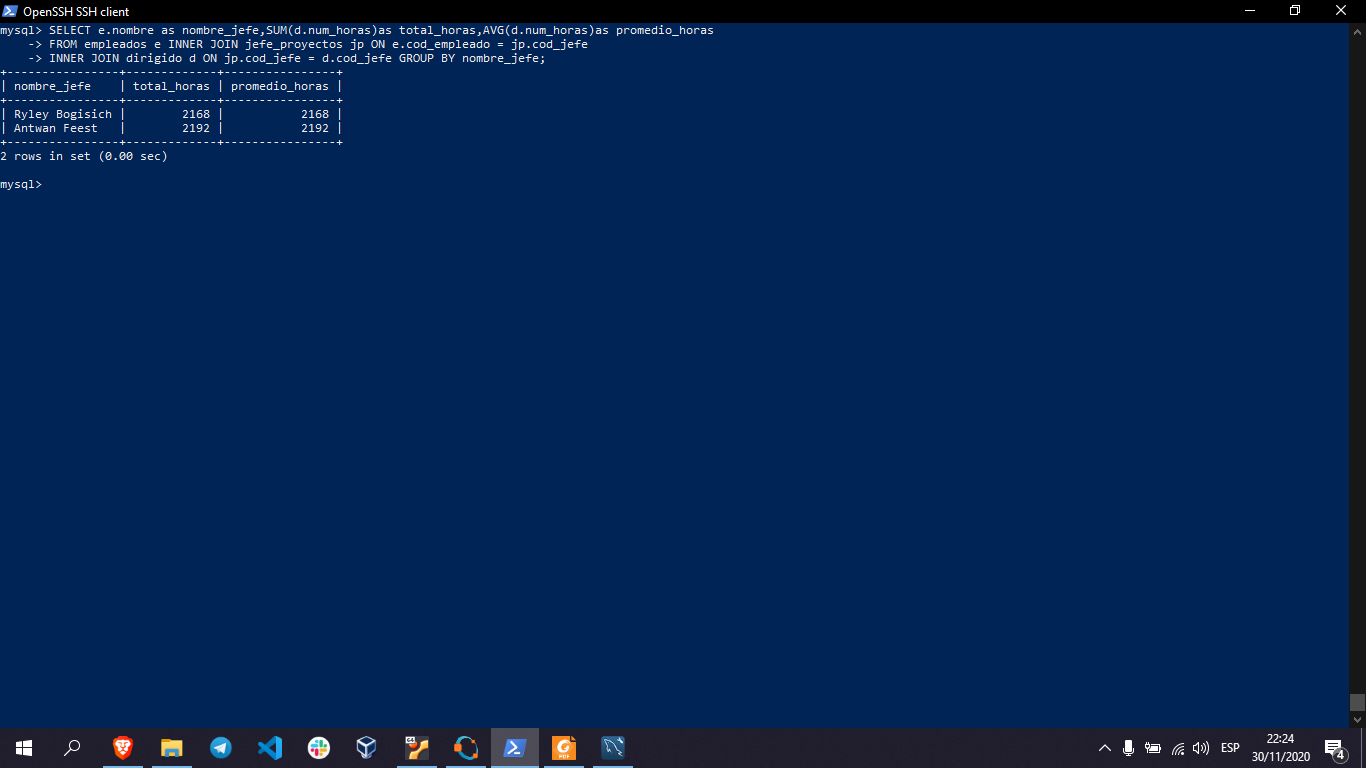
**--3.1 Listar los jefes de proyectos, el total de sus horas dedicadas y el promedio de**

**--horas por proyectos**

SELECT e.nombre as nombre\_jefe,SUM(d.num\_horas)as total\_horas,AVG(d.num\_horas)as promedio\_horas

FROM empleados e INNER JOIN jefe\_proyectos jp ON e.cod\_empleado = jp.cod\_jefe

INNER JOIN dirigido d ON jp.cod\_jefe = d.cod\_jefe GROUP BY nombre\_jefe;



**/\*3.2 Listar los 3 jefes de proyectos que obtuvieron el mayor gasto acumulado\*/**

WITH tabla as (

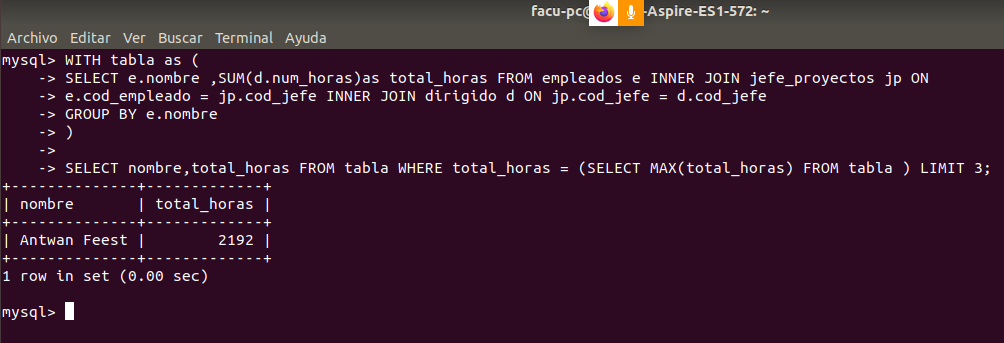
SELECT e.nombre ,SUM(d.num\_horas)as total\_horas FROM empleados e INNER JOIN jefe\_proyectos jp ON

e.cod\_empleado = jp.cod\_jefe INNER JOIN dirigido d ON jp.cod\_jefe = d.cod\_jefe

GROUP BY e.nombre

)

SELECT nombre,total\_horas FROM tabla WHERE total\_horas = (SELECT MAX(total\_horas) FROM tabla ) LIMIT 3;



**--3.3 El Jefe de proyecto cubre el 3% del total del los gastos generados en el proyecto. Obtener el listado de cuanto deberá pagar por jefe de proyecto y por proyecto**

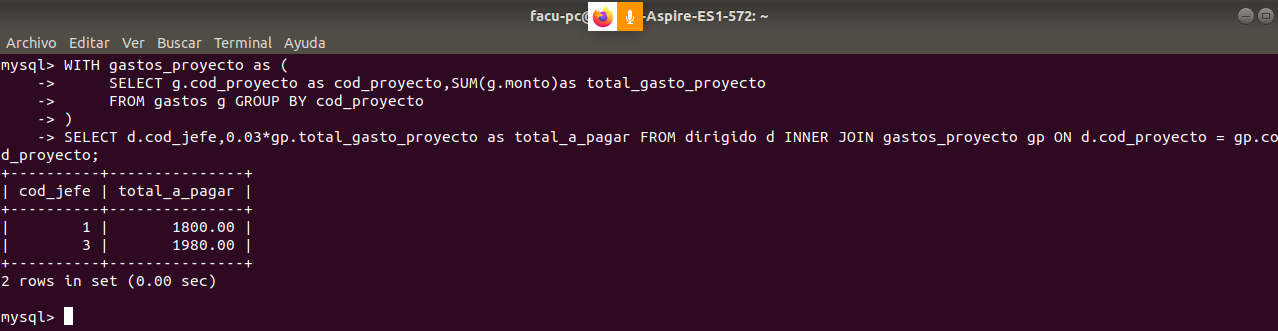
WITH gastos\_proyecto as (

SELECT g.cod\_proyecto as cod\_proyecto,SUM(g.monto)as total\_gasto\_proyecto

FROM gastos g GROUP BY cod\_proyecto

)

SELECT d.cod\_jefe,0.03\*gp.total\_gasto\_proyecto as total\_a\_pagar FROM dirigido d INNER JOIN gastos\_proyecto gp ON d.cod\_proyecto = gp.cod\_proyecto;

****

**--3.4 Listar el gasto total del Proyecto,**

**-- número de productos generados y suma de las cantidades de fases.**

WITH tabla1 as (

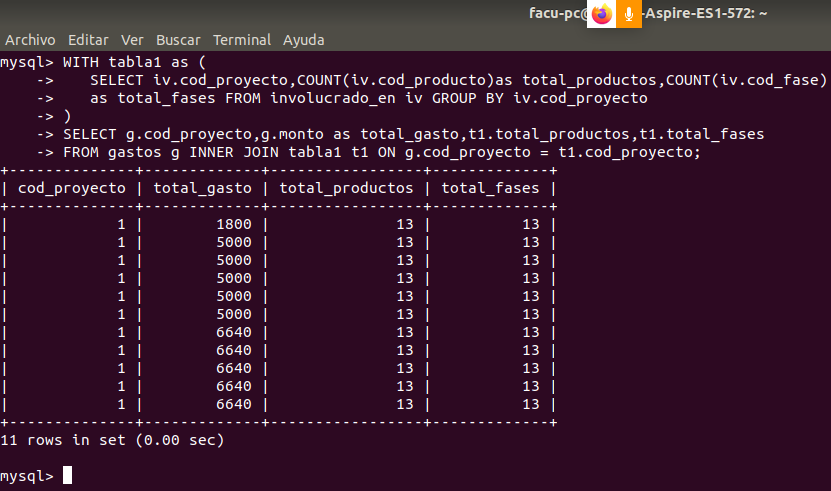
SELECT iv.cod\_proyecto,COUNT(iv.cod\_producto)as total\_productos,COUNT(iv.cod\_fase)

as total\_fases FROM involucrado\_en iv GROUP BY iv.cod\_proyecto

)

SELECT g.cod\_proyecto,g.monto as total\_gasto,t1.total\_productos,t1.total\_fases

FROM gastos g INNER JOIN tabla1 t1 ON g.cod\_proyecto = t1.cod\_proyecto;

****

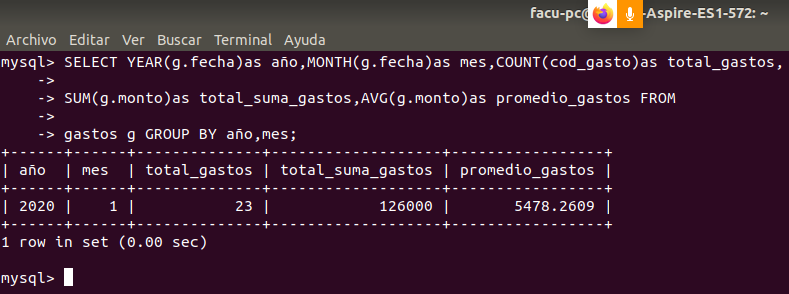
**--3.5 Para cada mes y año, número total de gastos,**

**-- costo total y monto promedio por proyecto.**

SELECT YEAR(g.fecha)as año,MONTH(g.fecha)as mes,COUNT(cod\_gasto)as total\_gastos,

SUM(g.monto)as total\_suma\_gastos,AVG(g.monto)as promedio\_gastos FROM

gastos g GROUP BY año,mes;



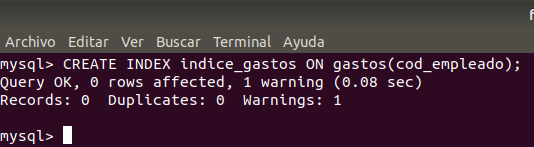
**4. Indices**

**Antes de realizar lo siguiente, Registre el tamaño de la BD y Ejecute la consulta A) registe el tiempo que lleva ejecutala.**

****

**--4.1 Crear un índice sobre cod\_empleados en gastos**

CREATE INDEX indice\_gastos ON gastos(cod\_empleado)



**4.2. Registre el tamaño de la BD y Ejecute la consulta A) registe el tiempo que lleva ejecutarla. ¿que cambios hubo y porque?**

**4.3. Ejecutar el explain anterior. Verificar que ahora se define un index scan. Es decir, se usa el índice. Ejecutar la consulta y proponer e implementar formas de mejorar los resultados obtenidos.**