4.

**i.Explique cuál es el propósito de la rol PÚBLICO , qué privilegios tiene y qué usuarios tienen esta rol.**

El rol público es un rol especial que cada cuenta de usuario de la base de datos tiene automáticamente cuando se crea la cuenta. Por defecto, no tiene privilegios otorgados, pero tiene numerosas concesiones, principalmente a objetos Java. No puede abandonar el rol público y una concesión o revocación manual de este rol no tiene sentido, porque la cuenta de usuario siempre asumirá este rol. Como todas las cuentas de usuario de la base de datos asumen el rol público, no aparece en el DBA\_ROLES y SESSION\_ROLES vistas del diccionario de datos.

ii. **Explique la diferencia entre estas vistas: DBA\_SYS\_PRIVS , DBA\_TAB\_PRIVS y DBA\_ROLE\_PRIVS**

R/ La vista DBA\_SYS\_PRIVS describe los privilegios del sistema otorgados a usuarios y roles, la vista DBA\_TAB\_PRIVS Lista de todos los privilegios de todos los objetos de la base de datos y la vista DBA\_ROLE\_PRIVS describe los roles otorgados a todos los usuarios y roles en la base de datos.

**iii.Cuando hablamos de privilegios en Oracle, encontramos tres *categorías* principales de privilegios, que son: privilegios de SISTEMA , privilegios de OBJETO y jerarquía de privilegios , definen cuál es el propósito de cada categoría y proporcionan algunos ejemplos de privilegios que pertenecen a cada uno de ellos.**

**Privilegios de sistema**

Un privilegio del sistema es el derecho a realizar una acción particular o realizar una acción sobre cualquier objeto de un tipo particular. Los objetos incluyen tablas, vistas, vistas materializadas, sinónimos, índices, secuencias, grupos de caché, esquemas de replicación y funciones, procedimientos y paquetes PL / SQL. Solo el administrador de la instancia o un usuario con ADMINprivilegios puede otorgar o

revocar privilegios del sistema.

Ejemplo 1:



Ejemplo 2:



**Privilegios de Objeto**

Un privilegio de objeto es el derecho a realizar una acción particular en un objeto o acceder al objeto de otro usuario. Los objetos incluyen tablas, vistas, vistas materializadas, índices, sinónimos, secuencias, grupos de caché, esquemas de replicación y funciones, procedimientos y paquetes PL / SQL.

Ejemplo 1:



Ejemplo 2:



**Jerarquías de privilegios**

Algunos privilegios confieren otros privilegios. Por ejemplo, el ADMINprivilegio confiere todos los demás privilegios. El CREATE ANY TABLEprivilegio del sistema confiere el CREATE TABLEprivilegio de objeto.

**9.**Lecturas

**Principales Casos De Uso De Redis Por Tipos De Estructura De Datos Principales**

* Diferencias entre Redis y otras bases de datos sql.

Redis es un sistema de base de datos NoSQL.Esto implica que, a diferencia de los sistemas de bases de datos controlados por SQL (lenguaje de consulta estructurado) como MySQL, PostgreSQL y Oracle, Redis no almacena datos en esquemas de bases de datos bien definidos que constituyen tablas, filas y columnas. En cambio, Redis almacena datos en estructuras de datos, lo que lo hace muy flexible de usar.

* **Clientes preferidos para usar Redis**

* + Redes sociales
  + Plataformas SaaS
  + Tiendas en línea
  + Aplicaciones Juegos en Línea
  + Sectores de viaje y hospitalidad
  + Foros Comunitarios
  + Comercio electronico

* Diferentes tipos de datos y sus usos
  + **Redis Strings:** puede contener cualquier tipo de dato y una longitud máxima de 512 MB.

* + Casos de uso de cadenas Redis
    - **Caché de sesión**:los datos se almacenan temporalmente en la RAM acelerando la experiencia de un sitio web
    - **Caché de sesión**:los datos se almacenan temporalmente en la RAM acelerando la experiencia de un sitio web.
    - **Colas:** puede ayudarlo a administrar el tamaño de la cola según la tasa de llegada y salida para la distribución de recursos.
    - **Uso y facturación medida:** es la medición en tiempo real para los modelos de precios basados en el consumo .

* + **Listas De Redis:** son cadenas que se ordenan por su orden de inserción.puede agregar elementos al principio o al final de las listas, lo cual es muy útil para los trabajos en cola.
  + Lista de casos de uso de Redis
    - **Sitios de redes sociales:** las plataformas sociales lo usan para personalizar su feeds o historias de tendencias
    - **Fuentes RSS:** cree fuentes de noticias a partir de fuentes personalizadas donde puede obtener las últimas actualizaciones

y permitir que los seguidores interesados ​​se suscriban a su fuente RSS.

* + - **Tablas de clasificación:** foros y plataformas de gestión para agregar artículos y tabla de calificación y ordenar
  + **Conjuntos Redis:** son tipos de datos potentes que admiten operaciones potentes como intersecciones y uniones, se usan cuando desea

realizar una auditoría. los conjuntos no permiten claves duplicadas

* + Casos de uso de conjuntos Redis
    - **Análisis de las ventas de comercio electrónico:** analizar el comportamiento de los clientes, como búsquedas o compras de una categoría o subcategoría de producto específica.
    - **Seguimiento de direcciones IP:** una gran herramienta para los desarrolladores que desean analizar todas las direcciones IP que visitaron una página
    - **Filtrado de contenido inapropiado:** para recopilar información del usuario
  + **Conjuntos Ordenados:** son una colección de cadenas que asignan un orden a sus elementos y son una de las estructuras de datos más avanzados.
  + Casos de uso de conjuntos ordenados de Redis
    - **Plataformas de preguntas y respuestas :** los conjuntos ordenados de Redis para clasificar las respuestas más votadas para cada pregunta propuesta y garantizar que el contenido de mejor calidad aparezca en la parte superior de la página.
    - **Marcadores de aplicaciones de juegos :**  para mantener sus listas de puntajes altos, ya que los puntajes pueden repetirse, pero las cadenas que contienen los detalles únicos del usuario no pueden.
    - **Servicio de programación de tareas:** puede asociar una puntuación para clasificar la prioridad de una tarea en su cola. Para cualquier tarea que no tenga un puntaje anotado, puede usar la opción PESOS con un valor predeterminado de 1.
    - **Geo Hashing:** la API de indexación geográfica permite indexar ubicaciones según la latitud y la longitud, convirtiendo los datos multidimensionales en datos lineales.
  + **Redis Hashes :** son mapas entre campos de cadena y valores de cadena,necesita crear esencialmente un contenedor de campos únicos y sus valores para representar objetos.
  + Casos de uso de Redis Hashes
    - **Perfiles de usuario:** pueden usar un solo hash para todos los campos de usuario.
    - **Publicaciones de usuarios:** para asignar todas las fotos o publicaciones archivadas a un solo usuario.
    - **Almacenamiento de métricas de múltiples inquilinos:** para registrar y almacenar sus métricas de productos y ventas de una manera que garantice una separación sólida entre cada inquilino

* **compañías que confían en Redis.**
  + Pinterest
  + Uber
  + Slack
  + Airbnb
  + Twitter
  + Reddit
  + Stack Overflow y Quora
  + instagram

**Fallas graves en la seguridad de tarjetas de crédito y credenciales en Rappi .**

* **Por qué Rappi tenía problemas con la información de los usuarios de tarjetas de crédito**

R/

* + No estaba protegiendo de manera segura la información de las tarjetas ya que se podrían conocer información parcial de las tarjetas de crédito y datos del dueño de la tarjeta de crédito (nombre,dirección, email) y no seguía los mecanismos o estándares recomendados para la recolección de la información.
  + Utilizaban los mecanismos correctos para los servidores protegidos pero tomaban la información desde el navegador del cliente.
  + Las credenciales de inicio de session no estaban protegidas
* **cómo almacenar datos de tarjetas de crédito de manera efectiva**
  + la información viaje cifrada y se almacene de forma segura
  + Usar proveedores que tengan los altos estándares de seguridad
  + usar el estándar PCI
* **Qué es PCI DSS**

PCI DSS es la normativa internacional de seguridad para todas las entidades que almacenan, procesan o transmiten datos de titulares de tarjeta o datos sensibles de autentificación.

* **cómo debe manejar la información de las tarjetas de crédito**
  + transmitir de manera segura los datos de las tarjetas de crédito de los consumidores reunión y transmitiendo los datos sensibles de manera segura.
  + guardar los datos de manera segura mediante cifrado, vigilancia continua y verificación de seguridad.
  + validar periódicamente (anual) que los controles de vigilancia y verificación de seguridad no tengan vulnerabilidades.
* **lecciones aprendidas para sus proyectos futuros y lo que no debe hacer para proteger a sus usuarios.**
  + se debe contactar a una empresa especializada para la auditoria y revisión de el estandar PCI.
  + si era un caso de 0-day la empresa debió tener de parte de la alta gerencia un apoyo más oportuno no sólo en respuesta del incidente sino el talento humano y en pruebas.
  + manejar el sistema transaccional con empresas especializadas en el tema para evitar los bug de seguridad.
  + periódicamente estar haciendo pruebas de seguridad para que sean identificadas por el área de seguridad y no esperar un ataque informático.
  + hacer una revisión de las plataformas tecnológicas para validar si son las correctas de acuerdo a la necesidad de la empresa.

**10 Reasons To Consider A Multi-Model Database**

es una necesidad generadas por las aplicaciones modernas por que las compañías se enfrentan a de base de datos en su infraestructura. Las compañías pueden implementar las base de datos NoSQL para disminuir un poco los sistemas, licencias y desarrolladores que necesiten.

* **Consolidación:** la base de datos NoSQL permite a los ingenieros desarrollar y modelar de manera más fácil aunque en la mayoría de los casos se complican por que las base de datos de los sistemas tienen múltiples niveles.
* **Escala De Rendimiento:** Por lo general el uso de las aplicaciones aumenta su necesidad de base de datos por lo que se debe contactar al equipo encargado para aumentar su capacidad y lo recomendable es tenerlo en una sola máquina.
* **Complejidad Operacional:** integrar las múltiples base de datos a los diferentes desarrollos se puede volver un tema muy complejo por el tem operacional y adicional mantener la integridad de los datos se vuelve complicado y la tolerancia a los fallos casi imposible
* **Flexibilidad:** debido que es complicado tener todos los datos es un solo modelo,este enfoque proporciona un modelo de datos sin complejidad.
* **Fiabilidad:** la fiabilidad es un tema difícil de manejar y más cuando se tienen diferentes diferentes base de datos debido a que alguna puede fallar y se tiene que tener soporte para toda las base de datos. los tiempos de inactividad pueden ser muy costoso para las compañías.
* **Consistencia De Datos:** se debe tener la funcionalidad de transacción de nivel superior integrada en la aplicación para que haya coherencia en los modelos y los datos.
* **Tolerancia A Fallos:** tener múltiples plataformas conectadas es una tarea que necesita de mucho soporte y carga operativa por lo que se tiene que tener un equipo capacitado para atender los diferentes fallos en las diferentes plataformas y lenguajes
* **Costo:** el uso de los sistemas de base de datos van aumentando el costo en el tiempo con actualizaciones, parches,correcciones,etc.
* **Transacciones:** generalmente se implementan en una sola máquina brindan garantías en las transacciones pero cuando están en diferentes máquinas no se garantiza con datos que se almacenen por su diseño arquitectónicos.
* **Mejores Aplicaciones:** Alimentar un aplicativo de múltiples base de datos se puede convertir en una pesadilla, pero con las bases de datos multimodelo no tenemos ese problema, adicional aumenta el rendimiento. con todos los beneficios anteriormente informados el mercado se dirige hacia este modelo por que cumpliran mejor los requerimientos existentes