PRÁCTICA P3. Definición y Modificación de Datos en SQL

Objetivos

- Construir, depurar y ejecutar sentencias SQL de modificación de información almacenada en una base de datos: inserción, actualización y borrado (LMD).
- Construir, depurar y ejecutar sentencias SQL de definición de datos (LDD): creación, eliminación y alteración de la estructura de los elementos que componen un esquema de base de datos.

Contenidos

Esta práctica consiste en la elaboración de sentencias SQL para **introducir**, **eliminar** y **actualizar** datos almacenados en tablas de un esquema de bases de datos (**LMD**, lenguaje de manipulación de datos), así como para **crear**, **destruir** y **alterar** la estructura de tablas o vistas (**LDD**, lenguaje de definición de datos).

Normas de realización

- ❑ Es necesario **contestar** los ejercicios **en el orden indicado en este enunciado**. Es posible que haya ejercicios que dependan de otros previos.
- ❑ Es posible que algún apartado necesite **más de una sentencia** SQL para ser resuelto; en tal caso hay que escribirlas en el orden en que deben ser ejecutadas para que funcionen correctamente.
- ❑ Se puede proponer **distintas soluciones** de un ejercicio cuando haya lugar a ello. Es habitual que exista una solución que sólo use sentencias LMD y otra que también use sentencias LDD[1].
- ❑ Es imprescindible que **cualquier restricción de integridad** (referencial (clave ajena), de columna, de tabla, etc.) existente antes de un determinado ejercicio, **siga existiendo después** de ejecutar el conjunto de sentencias que dan solución a dicho ejercicio.
- ❑ Debe **justificarse** adecuadamente toda **decisión importante** tomada al realizar cada apartado.

Esquema relacional de partida

Cada estudiante (o equipo de prácticas) trabajará con el **esquema** que haya **creado durante** la realización de **la práctica P1. Diseño lógico** de bases de datos (esquema **KEPASSA**).
Un objetivo adicional de las prácticas de la asignatura es tomar conciencia de que ***no basta con transformar el esquema conceptual en el esquema lógico*** –lo que garantiza que el esquema lógico específico se ajuste adecuadamente a los requisitos de datos– sino que también ***es necesario que dicho esquema permita realizar las operaciones*** identificadas en los requisitos de procesamiento (las que se realizarán en esta práctica).

Así, es posible que se deba modificar (alterar) la estructura del esquema lógico específico creado en la práctica P1. Diseño lógico…, ya sea para **corregir** los errores u omisiones cometidos, o porque el esquema no permite resolver de forma apropiada los ejercicios planteados en esta práctica.

[1] ¿Ya **tienes claro** qué significa LDD y LMD? ¿y las **diferencias** entre las sentencias SQL de cada tipo? Repasa las diapositivas… ;-)

Ejercicios

0. Corregir y Completar el script de creación de las tablas del esquema KEPASSA.

←El resultado de este ejercicio será un *script* de creación de las tablas. Debes ponerle el nombre **bdxxyy-p3ej0-esquema.sql**. Puedes usar la plantilla disponible en el Aula Virtual (Recursos y Contenidos). Más información en el apartado 'Documentación que se debe entregar'.

a) Aplica las correcciones realizadas por tu profesor/a de prácticas en el informe de corrección de la ***sesión 4*** de la práctica ***P1. Diseño Lógico***.

- Recuerda que para resolver el conflicto de nombres con la tabla SERIES.USUARIO de la práctica P2, la tabla del esquema KEPASSA (práctica P1 y P3) llamada **USUARIO debe renombrarse a MIUSUARIO**. Cámbialo ahora en tu script de creación de tablas, si no lo hiciste antes.

- Revisa que las referencias mediante claves ajenas sean correctas, también las claves primarias y únicas.

- Es imprescindible que exista un ciclo referencial entre MENSAJE y CHAT_GRUPO, así como una autorreferencia en MENSAJE (columna msj_original).

- Asegúrate de que los tipos de datos y longitudes sean adecuados. Esto sobre todo lo harás en el ejercicio 1, pero por ahora ya te avanzamos algunas cuestiones que deben incluir tus tablas:

  * La columna mensaje_id de MENSAJE debe ser CHAR(10).

  * La columna diahora de MENSAJE debe ser DATE, puesto que en realidad almacenará fechas completas.

- Una vez corregido, **ejecútalo** para que todas tus tablas queden creadas correctamente.

b) Si tu esquema ya incluye el **CREATE TABLE participacion (...)**, pasa al siguiente ejercicio.

Si no, entonces **abre el script** llamado **bd-p3ej0-esquema-mastablas.sql** y **copia el código** que permite crear la tabla PARTICIPACION, que nos hace falta para completar el esquema KEPASSA. **Pega** el código **en** el lugar adecuado de **tu script de creación de tablas**. Debes solucionar los problemas que puedan surgir al ejecutar el script completo.

1. Insertar filas en las tablas del esquema**.**

←El resultado de este ejercicio será un *script* de inserción de datos en las tablas. Debes llamarlo **bdxxyy-p3ej1-insert.sql**. Tienes una plantilla disponible en el Aula Virtual (Recursos y Contenidos). Más información en el apartado 'Documentación que se debe entregar'.

Importante: con el fin de que todos introduzcáis un conjunto de filas de **calidad** y no ocupéis tiempo en "inventar" datos, ponemos a vuestra disposición el script **bdxxyy-p3ej1-insert.sql** con las sentencias INSERT de un amplio y variado conjunto de **filas**. Además, al final incluye un comando UPDATE que calcula e introduce los valores adecuados en una **columna calculad**a.

Lógicamente, es muy probable que tengas que **modificar tus propias tablas** en mayor o menor medida, para que el script con los INSERT pueda ejecutarse sin errores (cambiar nombres de tablas y/o columnas, valores que se ajusten a los tipos de datos y los tamaños de las columnas, etc.).

Echa un vistazo a los tipos de los datos que se insertan en las columnas de las tablas en el script **bdxxyy-p3ej1-insert.sql** y cambia en tus CREATE TABLE lo que sea necesario.

Insistimos, aunque tendrás la tentación de modificar los INSERT para que se adapten a tus tablas (cambiando nombres de tabla y columna, es mejor idea **modificar tus CREATEs** (cambia nombres de tabla, columna, tipos de datos, etc.) para que los INSERT funcionen.

En cualquier caso, debes entregar el script con los CREATE (definitivos) como parte de la documentación de esta práctica (ver apartado 'Documentación que hay que entregar').

En el script facilitado, muchos de los INSERT aparecen en el orden correcto para que no surjan errores de integridad referencial (claves ajenas). Por ejemplo, cuando hay tablas (por ejemplo, CONTACTO) que referencian a otras (en este caso, MIUSUARIO), es necesario insertar primero las filas en la tabla referenciada (MIUSUARIO) y después en la que referencia (CONTACTO).

Pero si ejecutas el script[2] tal y como está, generará múltiples errores.

a) Es imprescindible **solucionar** el problema del **ciclo referencial** entre MENSAJE y CHAT_GRUPO.

b) Es ineludible **reordenar** las sentencias INSERT sobre MENSAJE (que contiene una **auto-referencia**) para que cuando se ejecuten no provoquen errores de integridad referencial (claves ajenas).

c) Debes solucionar cualquier otro problema debido a la integridad referencial (claves ajenas), recolocando las INSERT adecuadamente.

d) Cuando todas las INSERT funcionen sin errores, confirma los datos insertados (ejecuta la sentencia COMMIT).

[2] Ejecución **en secuencia de todas las** sentencias **INSERT** en el orden en el que aparecen en el fichero .sql.

←La resolución de los **ejercicios del 2 al último** debe estar incluida en un mismo *script*. Debes llamarlo **bdxxyy-p3ej2-9.sql**. Tienes una plantilla disponible en el Aula Virtual (Recursos y Contenidos). Más información en el apartado 'Documentación que se debe entregar'.

---

2. Añadir y Eliminar columnas.

a) Añade una **columna** llamada texto en la tabla MENSAJE, de tipo cadena de caracteres de longitud variable con tamaño máximo 35, y que admita nulos.

b) Añade una columna llamada numero_mensajes en la tabla MIUSUARIO, de tipo numérico con 3 dígitos, valor por defecto 0 y que no admita nulos.

c) Elimina la columna de tabla MIUSUARIO que contiene la descripción.

---

3. Modificar valores de una columna.

a) Redacta una sola **sentencia UPDATE** que, para todo usuario, calcule el número de mensajes que ha redactado cada uno, y lo introduzca en la columna numero_mensajes de MIUSUARIO.

- Ejecuta una sentencia COMMIT para **confirmar** los cambios realizados.

b) Modifica el mensaje anclado en los chats de grupo cuyo último mensaje ha sido publicado antes del 03/04/2024, de forma que el **texto del mensaje** contenga esto: 'CHAT ANTIGUO: VALORA SU BORRADO'. Resuelve este ejercicio con **una sola** sentencia UPDATE que realice todas las modificaciones.

c) Muestra mediante una SELECT los mensajes anclados de los chats indicados en b) y comprueba que el cambio es correcto.

d) Deshaz la modificación realizada en el paso **b)**, ejecutando la **sentencia ROLLBACK**;

- Muestra de nuevo los mensajes anclados y comprueba que los datos están como al principio.

---

4. Modificar el valor de una clave primaria de manera ordenada.

Este ejercicio consiste en implementar "a mano" una cadena de ON UPDATE CASCADE.

a) Cambia el teléfono del usuario 600000004 por el nuevo valor **610000004**. Hay que tener en cuenta que dicho usuario puede estar referenciado desde otras tablas. Te vendría bien hacerte un diagrama referencial del esquema.

Soluciona los problemas que puedan surgir con las restricciones de integridad referencial (claves ajenas), realizando las operaciones adecuadas y en el orden correcto.

Se permite desactivar/reactivar restricciones de integridad (las estrictamente necesarias).

b) Comprueba que el cambio ha quedado correcta y completamente realizado en todas las tablas afectadas.

- Asegúrate de que los cambios realizados quedan **permanentes**. Si es necesario, ejecuta una sentencia COMMIT.

- Asegúrate de que todas las restricciones de integridad están activas al finalizar el ejercicio.

5. Borrar algunas filas de una tabla.

a) Elimina mensajes cuya fecha es anterior al 10/02/2025, que no están anclados en ningún chat, que no sean la respuesta a ningún otro, que están publicados en chats de grupo con más de 3 miembros, y que han sido escritos por un usuario que pertenece a menos de 4 chats.

Te ayudará redactar previamente una SELECT que muestre las filas que deben ser eliminadas. Por cierto, son 5 filas las que se deben borrar.

- Ejecuta una sentencia COMMIT para **confirmar** los cambios realizados.

6. Borrar algunas filas de varias tablas.

> *Importante*: Hay que asegurarse de que todas las claves ajenas implicadas tienen **comentadas** las cláusulas **ON DELETE**, de forma que *Oracle* entienda **NO ACTION**.

Este ejercicio consiste en implementar 'a mano' una cadena de ON DELETE CASCADE.

a) Elimina todos los datos referentes al chat de grupo con código 'C004'.

Es decir, se desea **eliminar toda la información** que pueda existir en la base de datos **relacionada con ese chat de grupo** particular.

Antes de hacerlo, ten en cuenta que, puesto que ese chat de grupo puede referenciar a filas de otras tablas y también ser referenciado desde otras tablas, es importante que encuentres el modo más adecuado de eliminar toda la información afectada. Así que... **1º)** Establece un orden de borrado que minimice los problemas con las claves ajenas (referencias), y **2º)** Soluciona los problemas que puedan surgir debido a claves ajenas y que no se pueden resolver con un orden de eliminación adecuado. Se permite desactivar/reactivar restricciones de integridad (las estrictamente necesarias).

- Asegúrate de que todas las restricciones de integridad están activas al finalizar el ejercicio.

- Asegúrate de que los cambios realizados quedan **permanentes**. Si es necesario, ejecuta una sentencia COMMIT.

7. Crear y manipular una vista.

a) Define una vista llamada INTERACCION_ADMIN que, para cada usuario administrador de cada chat de grupo muestre la siguiente información:

- el teléfono del usuario administrador,
- el nombre del usuario,
- la fecha de registro del usuario,
- el nombre del chat de grupo en el que es administrador,
- el número de mensajes que el usuario ha publicado en el chat en el que es administrador.

Columnas de la vista: (tel_admin, nom_admin, f_registro, nom_chat, n_mensajes)

b) Muestra (con SELECT *) el contenido completo de la vista, ordenado por nom_admin y nom_chat.

c) Modifica la definición de la **vista** para que:

1) desaparezca la columna f_registro.

2) aparezca una columna total_mensajes que muestre cuántos mensajes en total contiene el chat de grupo.

Nuevas columnas de la vista: (tel_admin, nom_admin, nom_chat, n_mensajes, total_mensajes)

- Vuelve a **mostrar** la vista (paso **b**) y comprueba que el paso **c)** se ha realizado correctamente.

d) Inserta dos nuevos mensajes

d.1) redactado por el usuario '600000008' para el chat de grupo 'C007', con el identificador 'MSJ00701', la fecha actual (SYSDATE), que no sea reenviado, ni respuesta a ningún otro mensaje, y con el texto 'Q deberes puso la de mates?'.

d.2) redactado por el usuario '600000011' para el mismo chat de grupo 'C007', con el identificador 'MSJ00702', la fecha actual (SYSDATE), que no sea reenviado, que sea respuesta al mensaje d.1), y con el texto 'Toda la pagina 36 del libro'.

e) Visualiza el contenido de la vista (repite el paso **b**), y **contesta**: ¿se aplica el cambio? ¿Por qué? Responde dentro de un comentario.

- Ejecuta una sentencia COMMIT para **confirmar** los datos insertados.

8. Restricciones de integridad: Asertos.

- Redacta en código **ANSI SQL** dos **asertos** que aseguren el cumplimiento de estas dos reglas de integridad generales:

a) Aserto 1: "Todo chat de grupo debe tener al menos un mensaje".

b) Aserto 2: "El mensaje anclado de todo chat de grupo debe ser uno de los mensajes publicados en ese mismo chat"

Nota: para redactar cada aserto correctamente, piensa en qué tendría que pasar para que una fila lo incumpliera.

Importante:

- Redacta cada aserto utilizando la notación vista en clases de **teoría**:

  CREATE ASSERTION *nombre_aserto*

  CHECK(NOT EXISTS <*select que obtiene las filas que incumplirían el aserto*>);

- Hay que **recordar** que *Oracle SQL* no implementa la sentencia CREATE ASSERTION, pero siempre se puede ejecutar la SELECT que incluye, y que por tanto mostraría las filas que incumplieran la restricción de integridad correspondiente, y así asegurarse de que se está programando correctamente el aserto. Si el resultado de la SELECT devuelve una tabla vacía, significa que se cumple el aserto.

9. Creación y uso de índices.

- La siguiente consulta obtiene el identificador de los mensajes que son respuestas a otros dentro de chats de grupo con algún participante que tiene más de 5 contactos. La SELECT está intencionadamente redactada para incumplir claramente muchas de las Buenas Prácticas que hemos visto en clases de teoría.

```
SELECT DISTINCT M.mensaje_id
FROM mensaje M
     JOIN chat_grupo G    ON M.chat_grupo = G.codigo
     JOIN participacion P ON P.chat_grupo = G.codigo
WHERE M.msj_original IS NOT NULL
  AND EXISTS (SELECT *
              FROM miusuario U
              WHERE U.telefono = P.usuario
              AND U.telefono IN (SELECT usuario
                                 FROM contacto
                                 GROUP BY usuario
                                 HAVING COUNT(*) > 5));
```

a) Ejecuta esta consulta en el SQL Developer y visualiza el **plan de ejecución** mediante el cual el SGBD Oracle la ha realizado. Para ello pincha en el botón "Explicación del plan..." o pulsa la tecla F10. Anota el valor de la columna "COSTE" que aparece en la primera fila del plan, que da una medida de lo que le ha costado al SGBD ejecutar la sentencia.

Indica dentro de un comentario cuál es el valor de COSTE.

b) Aunque lo ideal para mejorar su rendimiento sería redactar la consulta siguiendo las buenas prácticas, esta vez vamos a hacerlo mediante la creación de un índice.

Escribe una sentencia para crear un índice que consiga que la sentencia SELECT anterior se ejecute con menor coste. Fíjate en que contiene una correlación entre la SELECT anidada que accede a MIUSUARIO y la SELECT exterior y que además se utiliza esa misma columna para comparar, la columna MIUSUARIO.telefono con CONTACTO.usuario mediante otro anidamiento. Eso es lo que hay que acelerar el acceso a la tabla CONTACTO vía su columna usuario.

c) Vuelve a ejecutar la misma SELECT y visualiza de **nuevo** el **plan de ejecución**. Mira si ha utilizado el índice que has creado y observa si ha **cambiado el coste** de ejecutar la consulta.

Si creas un índice y el coste no varía, y/o el SGBD no lo utiliza, trata de crear otro que sí lo consiga.

Una vez que lo hayas conseguido, escribe dentro de un comentario las respuestas a estas 2 preguntas: 1) ¿Cuál es el nuevo valor de COSTE?, y 2) ¿Ha mejorado respecto de la ejecución previa a la existencia del índice?

Documentación que se debe entregar

- La entrega se realizará **mediante el Aula Virtual** antes de la fecha límite indicada en la **Tarea** correspondiente.

Basta con que sólo **uno de los miembros de cada equipo de prácticas realice la entrega**.

- Se debe entregar un **fichero comprimido** con el nombre **bdxxyy-p3**(**.zip** o **.rar**), previa sustitución de bdxxyy por el nombre del usuario Oracle de prácticas.

Dicho fichero comprimido contendrá **3 ficheros script:**

* En *Recursos* y *Contenidos* del Aula Virtual hay **plantillas** para los 3 ficheros, que puedes reutilizar.

1) bdxxyy-p3ej0-esquema.sql: script SQL *Oracle* con las sentencias **CREATE** que definen las tablas MIUSUARIO, CONTACTO, EMAIL_CONTACTO, CHAT_GRUPO, MENSAJE, PARTICIPACION del esquema utilizado en la práctica (quizá revisado y mejorado una vez corregida la práctica P1). Ha de ejecutarse sin errores y dejar las tablas bien creadas.

2) bdxxyy-p3ej1-insert.sql: script SQL *Oracle* con las sentencias **INSERT** que introducen los datos en las tablas del esquema. Ha de ejecutarse sin errores y dejar en las tablas los mismos datos que los profesores han facilitado.

3) bdxxyy-p3ej2-9.sql: script SQL *Oracle* con los ejercicios **del 2 al último** de esta práctica
Para cada uno de los ejercicios del 2 al último deberá contener lo siguiente:
- Comentario con *Número* y *título* de ejercicio (breve frase en negrita en su enunciado).
- Opcional: Comentario con aclaraciones que se considere necesarias.
- Las *sentencias* (LDD y/o LMD) necesarias para resolver el ejercicio, en el orden en el que deben ser ejecutadas y fácilmente legible: con las adecuadas indentaciones (márgenes, sangrías).
Es posible presentar *varias soluciones alternativas* del mismo ejercicio. Se puede indicar (comentario) la alternativa que se considera más adecuada.

* Todos los scripts entregados deben contener (como un comentario) una **cabecera** con información acerca de la práctica y sus autores/as:

- Asignatura (**Bases de Datos**)
- Curso académico (**20XX/XX**) y convocatoria (**junio**).
- Identificador (**P3**) y nombre de la **práctica** (**Definición y modificación de datos en SQL**).
- Nombre del **usuario** de prácticas (**bdxxyy**), y **nombre** y **apellidos** de cada componente del equipo, si no se entrega individualmente.

Criterios de evaluación

Es obligatorio entregar **todos** los *scripts* SQL.

Si el fichero comprimido o los scripts no se denominan como se indica, se penalizará convenientemente.

Se tendrá en cuenta el uso de las Buenas Prácticas de la programación SQL que hemos estudiado en la parte de teoría

Además de la corrección de las respuestas a los ejercicios, se valorará convenientemente el orden, la estructura, la claridad y la legibilidad de la documentación presentada (scripts).

Es imprescindible respetar estrictamente las normas y el formato de presentación del informe de la práctica, detallados en este documento