Test Fin Prueba entregada de datos relacionales

- Fecha de entrega No hay fecha de entrega
- Puntos 10
- Preguntas 20
- Disponible 9 de sep a las 0:00 - 29 de jun de 2025 a las 23:59
- Límite de tiempo Ninguno
- Intentos permitidos Ilimitado

Instrucciones

Este test incluye un total de **20 preguntas** sobre los diferentes contenidos aprendidos en esta unidad.

¡Adelante!

Volver a hacer la prueba

Historial de intentos

	Intento	Hora	Puntuación
MÁS RECIENTE	Intento 1	4 minutos	10 de 10

Puntuación para este intento: 10 de 10
Entregado 19 de nov a las 12:38
Este intento ha tenido una duración de 4 minutos.

Pregunta 1
0,5 / 0,5 puntos
Una transacción es...

- Un listado completo de una tabla de la base de datos.
- Una consulta de selección que implica la consulta en varias tablas de la base de datos.

¡Correcto!

- Un conjunto de operaciones de actualización de la base de datos que deben realizarse todas o ninguna.

Una transacción es el conjunto de operaciones necesarias para llevar a cabo una determinada tarea que implique modificaciones en la base de datos. Dichas operaciones deben tratarse como una única unidad de trabajo, de modo que se efectúen todas o ninguna.

Pregunta 2

0,5 / 0,5 punto

Prueba entregada

DriverManager...

¡Correcto!

Actúa como proveedor de objetos *Connection* siempre y cuando hayamos cargado en memoria el *driver* que corresponda con el proveedor de base de datos requerido.

DriverManager requiere que se hayan cargado en memoria previamente los *drivers* de los proveedores de base de datos con los que vayamos a trabajar.

- Actúa como proveedor de objetos *Connection* y tiene incorporados muchos *drivers* de forma predeterminada.
- Representa una conexión con una base de datos.

Pregunta 3

0,5 / 0,5 puntos

Cuando trabajamos con *JDBC* utilizamos *Class.forName* para:

- Cargar en memoria el *JDBC*.
- Cargar en memoria todas las librerías de *java.sql*.

¡Correcto!

- Cargar en memoria los *drivers*.

Class.forName(…) se utiliza para cargar en memoria un software específico. En el contexto de acceso a bases de datos con *JDBC,* lo usamos para cargar en memoria los *drivers* del proveedor de base de datos.

Pregunta 4

0,5 / 0,5 puntos

Indica cuáles de las siguientes son propiedades ACID.

¡Correcto!

- [x] Atomicidad.
- [ ] Atención.
- [ ] Asistencia.

¡Correcto!

- [x] Consistencia.
- [ ] Seguridad.

¡Correcto!

- [x] Aislamiento.

¡Correcto!

- [x] Durabilidad.

Un SGBD que quiera llamarse transaccional debe cumplir las cuatro propiedades ACID: atomicidad, consistencia, aislamiento y durabilidad.

Pregunta 5

0,5 / 0,5 punto

Prueba entregada

Un objeto de la clase *ResultSet*...

- Sirve para ejecutar una sentencia SQL de tipo *SELECT*.
- Representa una conexión con una base de datos que permite ejecutar una sentencia *SELECT*.

¡Correcto!

- Se crea como resultado de la ejecución de una sentencia *SELECT* y contiene el conjunto de registros resultado de su ejecución.

El método *executeQuery(…)* de las clases *Statement* o *PreparedStatement* ejecuta una sentencia SQL de tipo *SELECT* y devuelve un objeto *ResultSet* con el conjunto de registros resultante.

Pregunta 6

0,5 / 0,5 puntos

Indica cuáles de las siguientes palabras corresponden a estados en los que puede encontrarse una transacción.

¡Correcto!

- [x] Activa.
- [ ] Abierta.
- [ ] Cerrada.

¡Correcto!

- [x] Parcialmente comprometida.

¡Correcto!

- [x] Fallida.
- [ ] Cancelada.
- [ ] Borrada.

¡Correcto!

- [x] Abortada.

¡Correcto!

- [x] Comprometida.

Una transacción puede encontrarse en uno de los siguientes estados: activa, parcialmente comprometida, fallida, abortada y comprometida.

Pregunta 7

0,5 / 0,5 puntos

Marca las afirmaciones correctas sobre la clave principal de una tabla.

¡Correcto!

- [x] No puede tener valores duplicados.

¡Correcto!

Sirve para es

Puede tener

Prueba entregada

¡Correcto!

Identifica inequívocamente a cada uno de los registros de la tabla.

Sirve exclusivamente para establecer interrelaciones con otras tablas. Para la identificación de cada registro se utiliza otro tipo de clave.

Cada tabla suele tener una clave principal. La clave principal no podrá tener valores duplicados, ya que sirve para identificar inequívocamente a cada uno de los registros. Además, sirve para establecer interrelaciones con otras tablas de la BD.

Pregunta 8

0,5 / 0,5 puntos

Indica cuáles de estos atributos serían idóneos como clave principal

Nombre

Telefono

Edad

¡Correcto!

Nif

¡Correcto!

CodigoArticulo

¡Correcto!

NumeroFactura

FechaFactura

Pregunta 9

0,5 / 0,5 puntos

Indica cuáles de estas sentencias SQL pertenecen al *DML* (*Data Management Language*).

¡Correcto!

INSERT

DROP

¡Correcto!

DELETE

ALTER

¡Correcto!

UPDATE

CREATE

¡Correcto!

- [x] *SELECT*

Prueba entregada

El DML está formado por el conjunto de sentencias SQL que tienen que ver con la manipulación de los datos (añadir registros, borrar registros, modificar registros, recuperar registros). Las sentencias que forman parte de este subconjunto son: *INSERT, SELECT, UPDATE* y *DELETE*.

Pregunta 10

0,5 / 0,5 puntos

¿Cuál de las siguientes sentencias SQL tiene que ver con la gestión de las transacciones?

- [ ] INSERT
- [ ] UPDATE

¡Correcto!

- [x] ROLLBACK
- [ ] DROP

¡Correcto!

- [x] COMMIT
- [ ] CREATE
- [ ] SELECT

¡Correcto!

- [x] BEGIN

La gestión de transacciones en MySQL se realiza a partir de estas tres sentencias: BEGIN, COMMIT y ROLLBACK.

Pregunta 11

0,5 / 0,5 puntos

El dominio de un campo o atributo es:

- [ ] El entorno que delimita dónde puede utilizarse dicho campo.

¡Correcto!

- [x] El conjunto de valores posibles que puede tomar dicho campo.

El dominio de un campo es el conjunto de valores posibles que puede tomar dicho campo, y viene determinado por el tipo de datos y las restricciones.

- [ ] La tabla donde está dicho campo.

Pregunta 12

0,5 / 0,5 puntos

Dado el siguiente código SQL:

```
USE FERRET

BEGIN;
   INSERT INTO CLIENTE VALUES
('12345678A','LUIS LOPEZ LOPEZ', 'C/ SOL, 3','639-639-639', 'MADRID');
   INSERT INTO FACTURA VALUES(5446,'2017-12-15',0,'12345678A');
   INSERT INTO DETALLE VALUES(5446,'MAR1',1,13);
   INSERT INTO DETALLE VALUES(5446,'TOR7',2,0.9);
   UPDATE PRODUCTO SET STOCK=STOCK-1 WHERE CODIGO='MAR1';
   UPDATE PRODUCTO SET STOCK=STOCK-2 WHERE CODIGO='TOR7';
ROLLBACK;
```

Prueba entregada

¿Qué ocurre con la base de datos tras su ejecución?

Que se ha añadido un cliente, una factura con sus líneas de detalle, y se ha actualizado el stock.

¡Correcto!

Aunque se ha añadido un cliente, una factura con sus líneas de detalle y se ha actualizado el stock, finalmente no ha ocurrido nada con la base de datos porque se han revertido los cambios con la sentencia ROLLBACK.

La sentencia ROLLBACK aborta la transacción, revirtiendo los cambios.

Se ha añadido una nueva factura, pero no se ha podido actualizar el stock.

Pregunta 13

0,5 / 0,5 puntos

Cualquier tarea de manipulación de una base de datos con Java *JDBC* lleva consigo las siguientes tareas:

1: establecer conexión con la base de datos. 2: interactuar con la base de datos. 3: cerrar la conexión con la base de datos.

¡Correcto!

1: cargar el *driver* en memoria. 2: establecer conexión con la base de datos. 3: interactuar con la base de datos. 4: cerrar la conexión con la base de datos.

Cualquier operación con bases de datos utilizando *JDBC* requiere de estas 4 tareas: 1. Cargar el *driver* en memoria. 2. Establecer conexión con la base de datos. 3. Interactuar con la base de datos. 4. Cerrar la conexión con la base de datos.

1: cargar el *driver* en memoria. 2: establecer conexión con la base de datos. 3: interactuar con la base de datos.

Pregunta 14

0,5 / 0,5 puntos

Dada la siguiente línea de código:

*String sql = "SELECT * FROM ARTICULOS WHERE CODIGO = ?";*

Prueba entregada

¡Correcto!

La variable *sql* contiene una sentencia SQL que puede ser precompilada usando un objeto de tipo *PreparedStatement*. La interrogación es un parámetro.

PreparedStatement nos brinda un sistema para precompilar la sentencia SQL y guardarla para ser ejecutada inmediatamente, sin necesidad de analizarla en cada caso. Cada interrogación en la sentencia SQL es un parámetro que habrá que sustituir antes de ejecutar la sentencia.

La variable *sql* contiene una sentencia SQL que puede ser precompilada usando un objeto de tipo *PreparedStatement*. La interrogación es un error, ya que no es un carácter permitido en SQL.

La variable *sql* contiene una sentencia SQL que podría ejecutarse utilizando un objeto de la clase *Statement*.

Pregunta 15

0,5 / 0,5 puntos

Dada la siguiente línea de código, y siendo *sentencia* un objeto de tipo *PreparedStatement*:

sentencia.setString(3, domicilio);

Estamos sustituyendo un parámetro en la sentencia SQL y el 3 hace referencia al cuarto parámetro, ya que el primero es el 0.

¡Correcto!

Estamos sustituyendo un parámetro en la sentencia SQL y el 3 hace referencia al tercer parámetro.

Las sentencias SQL precompiladas por *PreparedStatement* tienen parámetros que se especifican mediante interrogaciones. En la línea de código expuesta, el 3 hace referencia al tercer argumento de la sentencia SQL, comenzando desde 1, y *domicilio* hace referencia a la variable de memoria que contiene el valor de sustitución.

Estamos sustituyendo el campo *domicilio* de la tabla por el valor de la tercera variable que esté definida en el programa Java.

Pregunta 16

0,5 / 0,5 puntos

Dado el siguiente código, y siendo "*con*" un objeto de tipo *Connection*:

```
String sql = "DELETE FROM CLIENTE WHERE NIF=?";
String nif = "513332255L";
PreparedStatement sentencia = con.prepareStatement(sql);
sentencia.setString(1, nif);
int valor = sentencia.executeUpdate();
```

La variable *v* rio.

Prueba entregada

La variable *valor* contendrá el valor 1 si la sentencia SQL se ha ejecutado con éxito y 0 si ha ocurrido algún problema.

¡Correcto!

La variable *valor* contiene el número de filas que se han borrado al ejecutar la sentencia SQL.

El método *excuteUpdate()* devuelve el número de filas que han sido afectadas tras la ejecución de la sentencia SQL.

Pregunta 17

0,5 / 0,5 puntos

Dada la siguiente sentencia:

con = DriverManager.getConnection(cadenaConexion, a, b);

La variable *a* debería contener la contraseña de acceso a la base de datos.

¡Correcto!

La variable *a* debería contener el nombre de usuario que tiene permiso para acceder a la base de datos.

El método *getConnection* tiene la siguiente estructura: *getConnection(cadena_conexión, usuario, contraseña)*;

La variable *a* debería contener el nombre del DBMS.

Pregunta 18

0,5 / 0,5 puntos

Dada la siguiente cadena de conexión:

String cadenaConexion = "jdbc:mysql://Perico:3308/Mercado";

Marca las afirmaciones correctas.

¡Correcto!

- [x] *Jdbc:mysql* es el proveedor.
- [ ] *Mercado* es el nombre del servidor.
- [ ] *3308* es la contraseña del usuario con permisos para uso de la base de datos.

¡Correcto!

- [x] *Perico* es el nombre del servidor.

¡Correcto!

- [x] *Mercado* es el nombre de la base de datos.
- [ ] *Perico* es el nombre de la base de datos.

¡Correcto!

- [x] *3088* es el puerto.

La cadena de conexión especificada en esta pregunta tiene el siguiente formato: *proveedor://servidor:puerto/Nombre_BD*.

Prueba entregada

Pregunta 19

0,5 / 0,5 puntos

Los métodos *setAutoCommit()*, *commit()* y *rollback()* pertenecen a la clase...

- *ResultSet*
- *Statement*

¡Correcto!

- *Connection*

La clase *Connection* dispone de los métodos *setAutoCommit()*, *commit()* y *rollback()* para la gestión de transacciones en la base de datos.

Pregunta 20

0,5 / 0,5 puntos

Dado el siguiente código, y siendo la variable "*con*" un objeto *Connection*:

```
String sql = "UPDATE CLIENTE SET DOMICILIO=?, TLF=? WHERE NIF=?;";
PreparedStatement sentencia;
int afectados;
sentencia = con.prepareStatement(sql);
afectados = sentencia.executeUpdate();
```

¿Qué falta para que funcione correctamente?

- No falta nada, es correcto.

¡Correcto!

Falta asignar valores a los parámetros especificados por las interrogaciones, ejecutando métodos de tipo *sentencia.setString(…)*.

Los objetos *PreparedStatement* definen una sentencia SQL con parámetros que son identificados por caracteres de interrogación y que deben ser sustituidos mediante métodos *getString(…)*, *getInt(…)*, *getFloat(…)*, etc. antes de ejecutar la sentencia SQL.

Hay que corregir la sentencia SQL porque no es correcta, debemos sustituir las interrogaciones por valores reales.

Puntuación de la prueba: 10 de 10